On Improving Heuristics for Maximum Likelihood and Multiple Sequence Alignment in Phylogenetic Reconstruction

Zack Mahdavi
Department of Computer Sciences
The University of Texas at Austin
zkmusa@cs.utexas.edu

Spring 2006

Acknowledgments

I would like to thank my advisor, Tandy Warnow (Computer Sciences, The University of Texas at Austin), for guiding me throughout this research and while I was writing this thesis. It wouldn't have been possible without her. Her classes motivated me to pursue research in the field of phylogenetics.

I am very grateful for working with Ganesh Ganapathy and David Zhao (Computer Sciences, The University of Texas at Austin), who both demonstrated undying patience for my multitude of questions.

Abstract

In the construction of phylogenetic trees, exact methods for maximum likelihood and multiple sequence alignment are very computationally intensive algorithms, even on a fixed tree. Since anything above cubic time is too computationally complex to be practical in phylogenetics, heuristics methods are used to approximate a solution. We attempt to optimize maximum likelihood heuristic methods to improve performance and accuracy by eliminating "fast evolving sites." In addition, we compare various multiple sequence alignment heuristic methods to meet an overall goal of improving generalized tree alignment heuristic methods.

Contents

Chapter 1

Introduction

The construction of phylogenetic trees has become an extremely important problem in biology. Its application has been critical to understanding complex subjects such as bacterial resistance, viral evolution, and even protein evolution.

The theory of evolution, which is the idea that all species descended from a single common ancestor, is a strongly supported theory. The theory states that a common ancestor repeatedly split into new species over the course of time; the species present in today's world are a direct result of this procedure.

Charles Darwin introduced evolution to the world in the book, *On the Origin of Species*, where he cites specific pieces of evidence that support his theory. In this book, Darwin included exactly one figure: a phylogenetic tree.

A *phylogenetic tree* parallels with a typical tree found in Computer Science. A parent node contains child nodes, and a node without children is called a leaf. The set of species (or taxa) being studied are labeled at the leaves of the tree; an internal node of the tree represents a common ancestor of its children nodes. The tree may be binary, but many times it is not. Although it is biologically correct for a phylogenetic tree to contain a root, it is often very hard to find a root, so unrooted trees are generally used in phylogenetic studies. An example of a rooted and unrooted phylogenetic tree is shown in Figure 1.1.

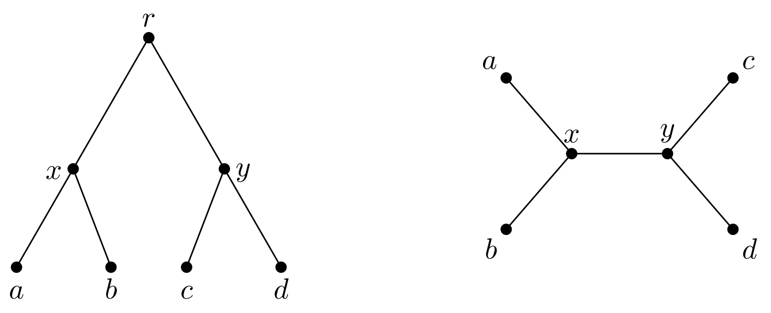


Figure 1.1: Example of rooted and unrooted tree.

In order to build a phylogenetic tree, data from species (or taxa) must be collected. The data that can be used ranges widely. For example, we might record which taxa has wings and which do not; this would translate to a binary collection of data. If a taxon had wings, it would be assigned 1. If a taxon did not have wings, it would be assigned 0. We can also collect sequences of DNA or protein from taxa.

Many methods have been developed to estimate phylogenetic trees, but some are more accurate than others. Also, all of these methods are extremely computationally complex, and several problems in phylogenetics are NP-hard. This directly impacts our ability to accurately construct phylogenetic trees.

Many heuristics are used to "solve" these optimization problems. They function to find only an approximation of the solution with the reward of significant improvements in running time and space. In addition, these heuristics work reasonably well when applied to small datasets, but they tend to not give optimal results on large datasets. Despite the shortcomings, from a practical standpoint, heuristics are chosen over exact methods for any dataset larger than a few sequences.

There is a major downside for applying heuristics to datasets: an exact solution is not computed. Instead, heuristics return an approximation of the exact solution, which may or may not be as accurate as the true solution. However, heuristics can be adjusted to provide better optimized solutions.

This thesis covers two approaches to improving heuristics in phylogeny. Both approaches deal with optimizing the accuracy of the heuristics. One research topic focuses on improving the accuracy of maximum likelihood heuristics by eliminating "fast evolving" sites. The second research topic attempts to optimize the accuracy of multiple sequence alignment heuristics for three sequences. Both of these topics share in common the necessity for improving the accuracy of heuristic methods, regardless of the problem being optimized.

Chapter 2

Maximum Likelihood Optimization Through Manipulation of Fast Evolving Sites

2.1 Introduction

A generally favored method used in phylogenetic tree estimation is *maximum likelihood.* This method selects the tree and the parameters on the tree, so that the probability of the data on the tree is maximized [5]. It can be formally defined as:

- **Input:** Set S of aligned sequences
- **Output:** A tree T along with parameters θ such that $Pr[S|T, \theta]$ is maximized

The maximum likelihood problem, which is NP-hard, has no simple analytical solution, so an iterative solution is required [36]. The method requires a stochastic model of evolution, which serves as a way of describing the evolution of a molecule as a series of point mutations that occur along edges of the tree. In most stochastic models, including the Jukes Cantor model [13], the edge length between two nodes represents the expected number of mutations that will occur between the sequence at the parent node and the sequence at the child node. A "long" edge length indicates a strong likelihood of a mutation to occur at that edge. With the Jukes Cantor model, the next state selected is random. In other words, if a transformation will occur from state `A` to another state, the state at the edge will equally be `C`, `T`, or `G` if DNA sequences are used. Other models, such as the General Time Reversible (GTR) model [29], assign probabilities to the next states chosen, which will alter the chance of a state being selected.

The maximum likelihood method is a two step process. Given a set of sequences, a tree is generated. Next, the parameters (probabilities) along the edges must be chosen to maximize the probability of the data on the tree. This is generally done as a heuristic search, where all parameters are first fixed. Then, each parameter is optimized. Once this process is complete, another tree is generated, and the process repeats. Finally, the tree along with the parameters that generate the maximum probability for that tree is returned. Because there are an exponential number of trees to select from, the search can take a long time to converge.

An established method in phylogenetics is *maximum parsimony*, which is a cladistic method that builds a tree based on observed differences and similarities between the taxa. The tree is leaf-labeled by the taxa, while the internal nodes of the tree are leaf-labeled with sequences such

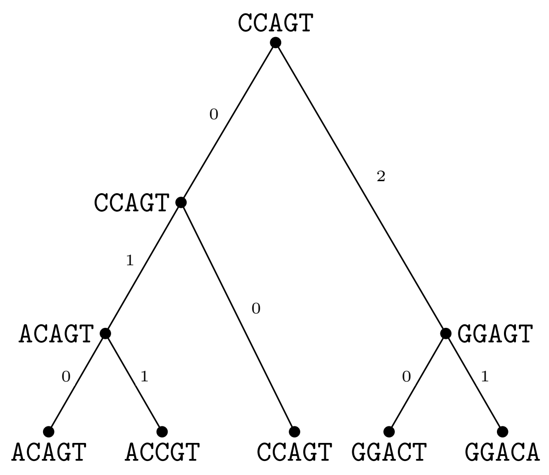


Figure 2.1: Maximum Parsimony applied to a rooted tree.

that the total number of changes along the edges between the sequence-labeled nodes is minimized. An example of maximum parsimony is illustrated in Figure 2.1.

In this example, the edges of the tree are labeled with a tree length, which is the number of substitutions that occur between the sequences at the nodes of the edge. If the sequences at the parent and child nodes are the same, then the tree length is 0. The maximum parsimony tree returned is the tree with the smallest parsimony score (sum of all the edges). The internal sequences are labeled to minimize the parsimony score.

Maximum parsimony is considered problematic, however. Under the Jukes-Cantor model, maximum parsimony is not statistically consistent, so it is not guaranteed to produce the true tree with high probability, given sufficient data. This was shown by Felsenstein in 1978 [4], which led him to develop maximum likelihood, a problem that has been shown to be statistically consistent under the Jukes-Cantor model.

2.2 Problem

Recently, longer and longer DNA sequences are now becoming available for analysis. In addition, many datasets now involve hundreds to thousands of taxa. Since the size of a dataset is equal to the number of species used multiplied by the length of the sequences, dataset sizes are growing rapidly.

Speed in methods like maximum likelihood is becoming a critical factor in phylogeny reconstruction. Current maximum likelihood heuristics are simply too computationally expensive to make maximum likelihood an option for analyzing large datasets. In the field, when datasets become very large, other methods are often used, such as neighbor joining, maximum parsimony, and Mr. Bayes.

This research seeked to develop improved maximum likelihood heuristics. I investigated whether modifications to the input datasets will help speed up the analyses without loss of topological accuracy (conjectures along these lines have been made in the biology community).

Different sites (positions) among a sequence can vary in their rate of evolution [16]. This is due to various biological factors; for example, a site that is part of the coding region for hemoglobin will evolve much slower than a site that doesn't code for anything. Therefore, the DNA region that codes for hemoglobin will be considered constrained. Unconstrained sites generally have a fast rate of evolution. These sites with fast rates of evolution can cause “noise” during phylogeny reconstruction. It is possible that methods such as maximum likelihood spend a lot of time trying to optimize these sites, rather than optimizing the sites that experience slow rates of evolution. There is a conjecture among systemists that eliminating fast evolving sites may lead to faster maximum likelihood analyses without loss of topological accuracy [27].

2.2.1 Heuristic for identifying and eliminating fast evolving sites

In an attempt to enhance the performance and accuracy of the maximum likelihood method, we analyzed a dataset and developed a method to mark the fast evolving sites. This was done by the following process:

1. Perform a maximum parsimony analysis to generate a tree using the full dataset. Such an analysis generates a tree and labels the internal nodes of the tree such that a minimal parsimony score is obtained. The tree is then altered to produce a new tree, and this new tree is scored. If the score of the new tree is less than the score of the previous tree, the new tree is picked as the optimal tree, and the process repeats. Otherwise, the previous tree is again altered, and a new tree is scored again.
2. One site is picked and removed from the dataset. The tree is scored using maximum parsimony using the full dataset minus the selected site. The site removed in step two is placed back into the dataset. This process is repeated for all sites.
3. At this point, we have generated a maximum parsimony score for each site. Next, a mean and standard deviation among all scores is calculated.
4. Two truncated datasets are created. One truncated dataset (sd1) excludes all sites whose maximum parsimony score exceeds mean + one standard deviation. The second dataset (sd2) excludes all sites whose maximum parsimony score exceeds mean + two standard deviations.

Table 2.1 demonstrates which sites will be included in the truncated datasets produced by this heuristic.

Table 2.1: Sites included in the sd1, sd2, and full datasets.
Assume mean = 10; st. dev. = 2

Site	Score	SD1	SD2	Full
1	6	✓	✓	✓
2	13	—	✓	✓
3	12	✓	✓	✓
4	14	—	✓	✓
5	16	—	—	✓
...				
100	4	✓	✓	✓

Fast evolving site identification on real datasets

The site identification heuristic was tested on three real datasets:

- Angio: Includes 228 aligned DNA sequences of angiosperm plants (flowering plants) [34]. Each sequence contains 4811 sites.

- Nem: Consists of 682 aligned small subunit rRNA sequences (with 1808 sites per sequence) from 678 species of *Nematodes* (roundworms). Four outgroups are also included [20].

- rbcL: Contains 500 aligned sequences of the Rubisco gene of a chloroplast for many different plants [1]. Each sequence contains 1398 sites.

The site distributions are shown in Figure 2.2. As expected, a smooth distribution is evident; there are few sites with high score differences, whereas most sites have scores similar to the mean. These are the fast evolving sites. The angio dataset contains about 50% of sites that do not vary at all, whereas only 13% of the Nem sites and approximately 26% of the rbcL sites do not vary. It is clear that the angio dataset contains a much greater percentage of conserved sites compared to the other datasets.

After removal of the fast evolving sites, the sd1 truncated angio dataset contained 89% (4260 sites) of the original dataset, while the sd2 truncated angio dataset contained 94% (4529 sites) of the original dataset. The nem dataset is similar: the sd1 truncated dataset contained 84% (1518 sites) of the original dataset, while the sd2 truncated dataset contained 94% (1698 sites) of the original dataset. Finally, the rbcl sd1 truncated dataset contained 88% (1231 sites) of the original dataset, while the rbcl sd2 truncated dataset contained 94.4% (1320 sites) of the original dataset. These truncated datasets now contain much less fast evolving sites than the original full datasets.

2.2.2 Heuristic for evaluating truncated datasets using maximum likelihood

Once the truncated datasets obtained from the site elimination heuristic have been determined, the following is performed:

1. For each truncated dataset, a set T of ratchet starting trees is generated.

2. Score the trees from T using maximum likelihood. This is done using the program PAUP*. Parameters for each tree will be generated.

3. Perform a 12 hour heuristic search using PAUP* with T as the set of starting trees. The parameters from step 2 are fixed in this step. A final tree is generated.

4. Score the final tree using maximum likelihood with PAUP*.

5. Using the tree from step 4, run PAUP* for 12 hours. This step is similar to step 3, but the original full dataset is used instead.

6. Score the final tree using maximum likelihood with PAUP*.

The above process is also performed using the full dataset instead of the truncated dataset, with one exception: steps 5 and 6 are not performed when a full dataset is used. It would be redundant to perform steps 5 and 6 on the full dataset.

It is also important to note that all stochastic models used in all of the experiments were GTR+gamma+I (general time reversible) models.

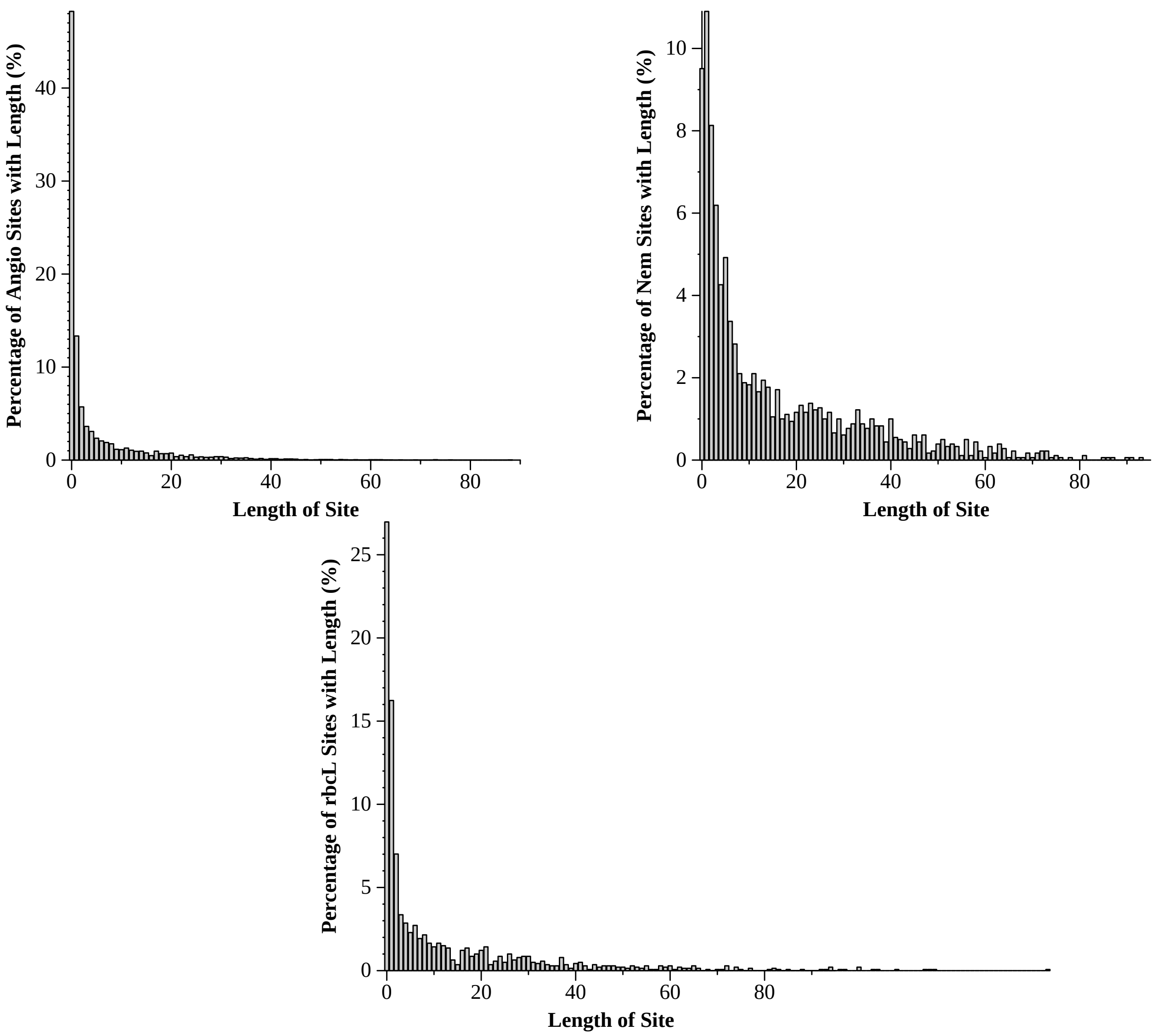


Figure 2.2: Site distributions for angio, nem, and rbcL datasets

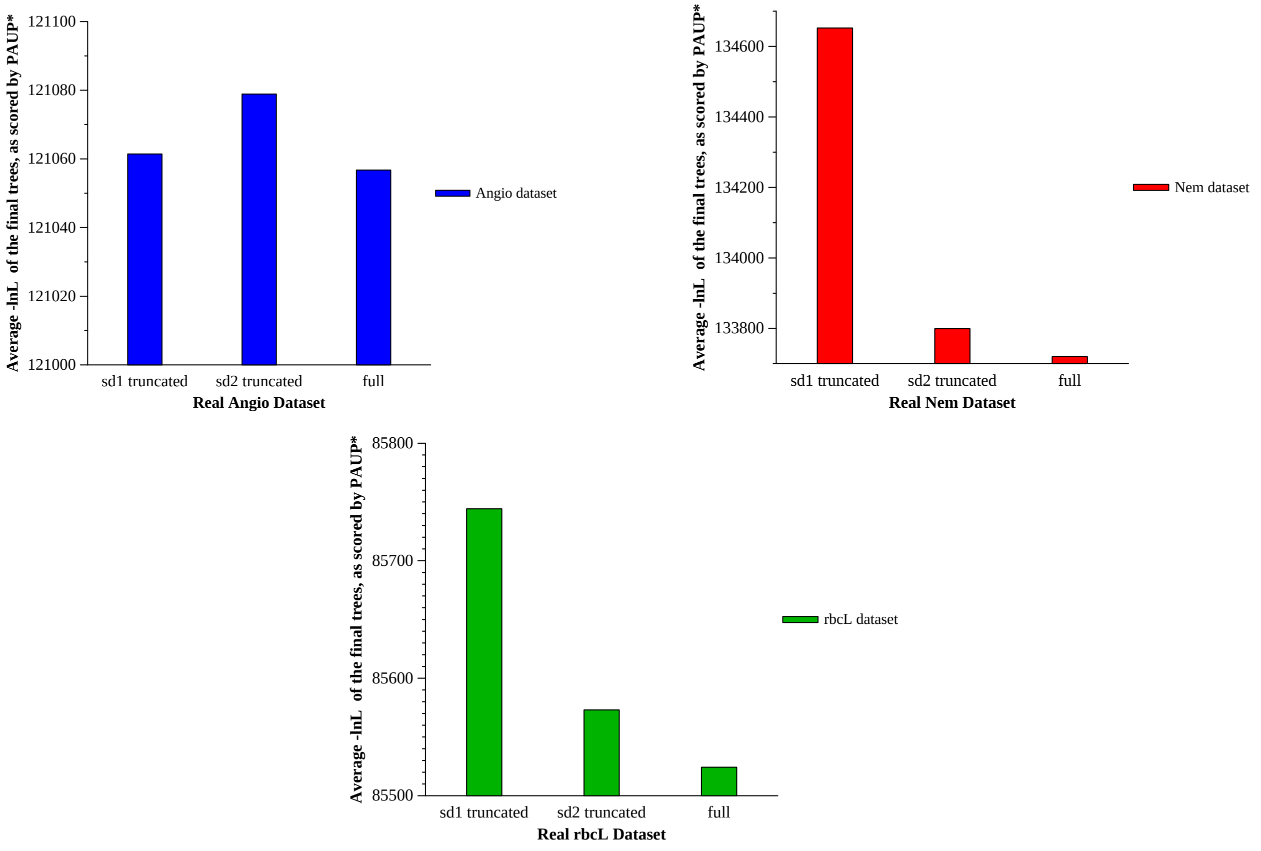


Figure 2.3: Average -lnL of the final trees for both the truncated datasets and the full dataset.

2.2.3 Real dataset manipulation of angio, nem, and rbcl

To evaluate the heuristic, truncated datasets were created from the angio, nem, and rbcl datasets. The alpha shape parameters for the angio, nem, and rbcl datasets were set to 0.67, 0.52, and 0.40, respectively. The results are scored using the log likelihood value (lnL), and the lowest -lnL is chosen as the maximum likelihood tree. Results for this first round of tests is shown in Figure 2.3.

For the angio dataset, the best tree with the lowest -lnL value is produced using the full dataset. The truncated datasets did not produce the best tree, although the difference is not large.

In addition, the heuristic applied to the nem dataset did not result in what we had expected the overall outcome of the heuristic would be. Running the analysis on the sd1 dataset did produce the highest negative log likelihood value, and the full dataset had the lowest negative log likelihood value. However, the difference in the -lnL value is very small, and this data cannot be considered proof that the heuristic is not working.

Finally, the results of the rbcl dataset are similar to that of nem. The final tree produced from the truncated sd1 dataset has the highest -lnL, while the tree produced from the full dataset has the lowest -lnL.

Unfortunately, the heuristic's performance on any of the real datasets was disappointing. The overall performance of the truncated datasets was worse than the full dataset. However, the difference in performance was not large. We conjecture that running the heuristic on simulated datasets might improve the heuristic's performance.

2.2.4 Simulated dataset manipulation of angio

After obtaining disappointing results from applying our heuristic to real datasets, we decided to shift focus to simulated datasets. The real concern that we are trying to address is topological accuracy, and therefore a simulated dataset is necessary to determine if the fast evolving heuristic results in improved topological accuracy. In particular, we were interested in determining if the analysis applied to a subset of sites is significantly better than when the analysis is applied to the full dataset.

Since simulation is a stochastic process, every simulated dataset is different from the others, but all of them are derived from the same distribution. In this case, five simulated datasets were created from the original angio dataset, in which the alpha shape parameter of the gamma distribution was maintained at 0.67. We used *SeqGen* [45] to generate these five datasets. The sequence length of each sequence in the datasets was 4811 sites. The simulated datasets were then subjected to the same heuristic that the real datasets were subjected to in the previous experiment. As a result, a total of 15 datasets were tested: the 5 simulated datasets as well as sd1 and sd2 truncated versions of each simulated dataset.

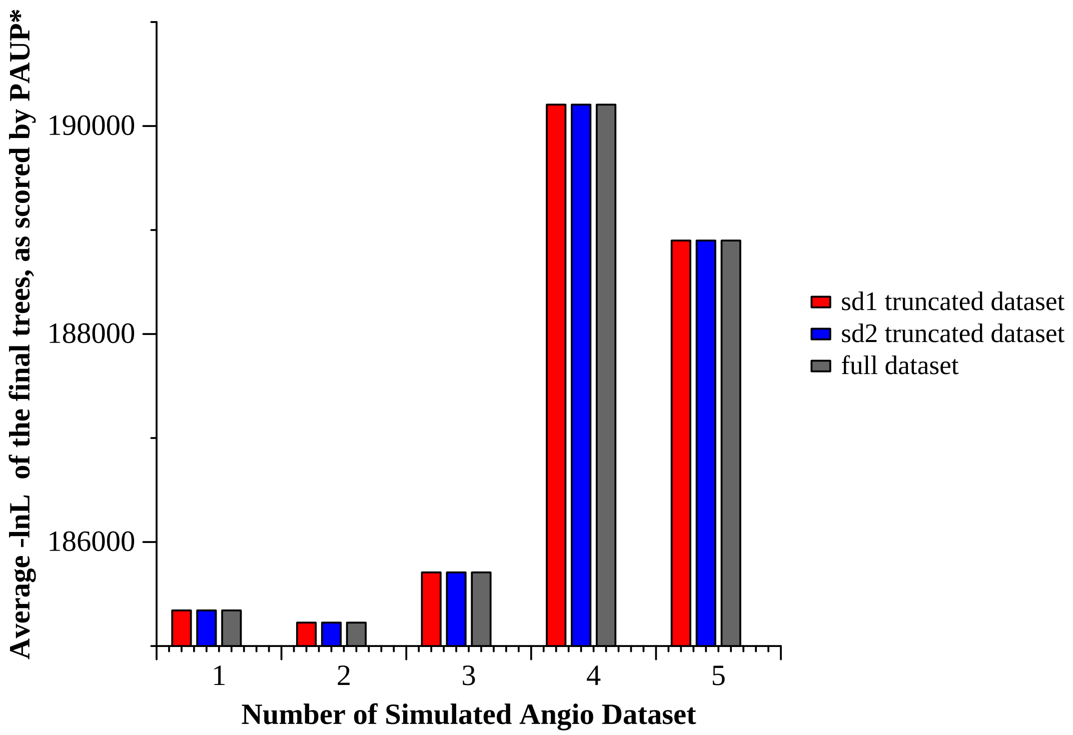


Figure 2.4: Average lnL of the final trees for both the truncated datasets and the full dataset of all five simulated angio datasets.

The results are shown in Figure 2.4 for one of the five simulated datasets. For every simulated dataset, the application of the heuristic resulted in -lnL values with *very little* difference for the sd1, sd2, and full datasets. The values that were varied only differed by about 0.6. In other words, there was no benefit to deleting the fast evolving sites. We conjectured that these simulated datasets were too "easy" for our heuristic. A more complicated dataset was necessary in order to determine the effectiveness of the fast evolving sites heuristic.

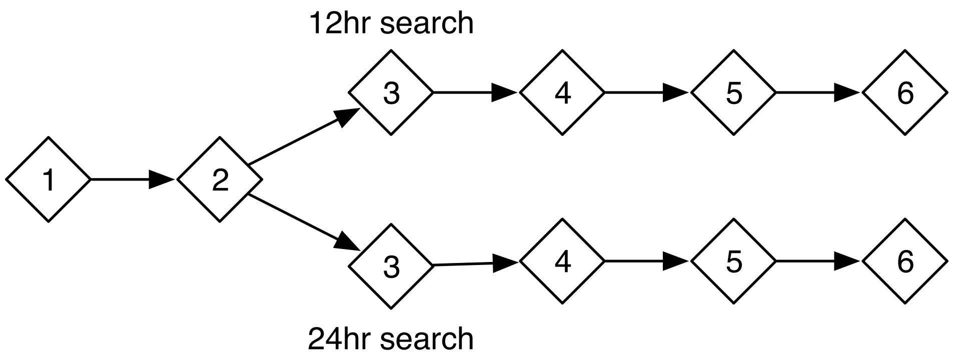


Figure 2.5: Flow chart illustrating modified heuristic used during the gamma experiment. Step 3 is independently performed twice: once for 12 hours, and once for 24 hours. All other steps after step 3 are unchanged.

2.2.5 Modification of gamma distribution on angio, nem, and rbcl datasets

Most stochastic models assume that all sites evolve independently of each other under identical processes. In the real world, sites generally do not evolve at the same rate of evolution. A good example mentioned earlier is the DNA region that encodes hemoglobin. In order to integrate rates into a stochastic model, a rate is taken from a probability distribution, which is usually a gamma distribution. This rate is then maintained across all edges of the tree. This method of solving for rate variation still upholds the rule that stochastic models evolve under independent and identical processes.

The alpha value is the gamma shape parameter, and it is determined from the distribution of the expected number of substitutions. Low alpha values (less than 0.5) create “flat” curves, where lower rates of evolution are preferred. As the number increases, the curve begins to “slope”, resulting in larger probability densities for a given evolutionary rate, specifically the higher evolutionary rates.

To determine the performance of the fast evolving site deletion heuristic, we altered the gamma distributions of three model trees, angio, nem, and rbcL, to produce three new datasets. We set the alpha shape parameter for the ‘gamma2’ dataset to be the same as that used for the real dataset. The ‘gamma1’ dataset uses an alpha shape parameter that is exactly double that of ‘gamma2’. Table 2.2 list the actual values of the parameters used. As a result, a total of nine datasets were then subjected to the fast-evolving site elimination heuristic to produce sd1, sd2, and full datasets. All in all, 27 datasets in total were tested.

Dataset	gamma1	gamma2	gamma3
angio	0.67	0.33	0.17
nem	0.52	0.26	0.69
rbcL	0.40	0.20	0.69

Table 2.2: Alpha shape parameters for gamma distribution simulated datasets.

The analysis on these 27 datasets was slightly modified. Normally, the final ratchet tree was subjected to a 12 hour PAUP* heuristic search. This time, however, the ratchet tree obtained in step 2 was subjected to a 12 hour and a 24 hour heuristic search. These searches were conducted independently of one another. In addition, after each heuristic search is complete, the final trees

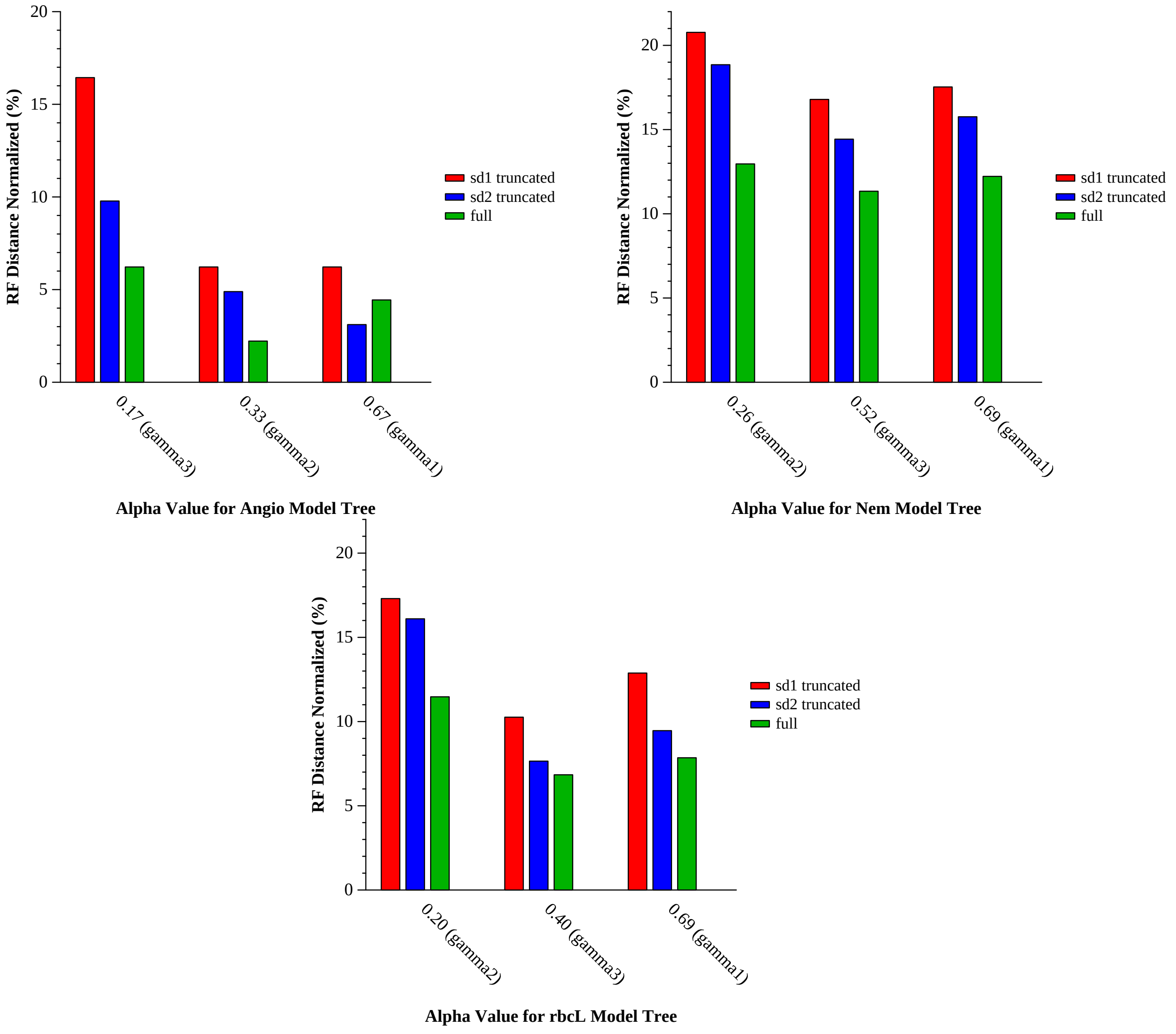


Figure 2.6: Normalized RF distance among angio, nem, and rbcl datasets with varying gamma distributions. Heuristic method ran for 24 hours.

are then scored using the full dataset of whatever is the current dataset among the 27 datasets. All of the other steps remained the same. Figure 2.5 illustrates the modified 6 step process.

Effect of Alpha Value on Difficulty of Dataset

Figure 2.6 examines the effect of varying the alpha shape parameter on Robinson-Foulds distance after the method was executed for 24 hours. Robinson-Foulds (RF) distances were calculated to determine the topological difference between the model tree and the final tree produced by our analysis. When the two trees share a greater number of similar edges, the RF distance will be lower.

It is evident with all three datasets that the lowest alpha value always has the worst topological accuracy (highest RF distance). In addition, the RF distance for the "middle" alpha value is always less than that of the other alpha values.

Almost all of the trees constructed using the full dataset had better topological accuracy than the truncated datasets. There is one exception: the full angio dataset, with alpha set to .67, had *worse* topological accuracy than the sd2 truncated dataset. We conjectured we would see this result with all of the tests, but this clearly wasn't the case. It is unclear why this specific alpha value caused the topological accuracy of the full dataset to be worse than the sd2 truncated dataset.

Effect of 12/24 Hour Heuristic Search

Figures 2.7, 2.8, and 2.9 compare performance and topological accuracy when a 12 or 24 heuristic search is applied to the angio, nem, and rbcL datasets, respectively.

For the most part, the performance and accuracy of the method applied to the angio dataset is independent of the time in which the heuristic is ran. However, when alpha is set to 0.17, the 24 hour heuristic search is more accurate, and the performance of this 24 hour heuristic search is also better than the 12 hour heuristic search.

The nem dataset shows more consistent results. The full dataset always performs better and has a better accuracy regardless of alpha. In addition, these results show a strong correlation between log likelihood values and topological accuracy; for the nem dataset, lower log likelihood values mean better topological accuracy.

Results for the rbcL dataset are very similar to the nem dataset. In almost all cases, a 24 hour search results in better performance and topological accuracy than a 12 hour search. However, with alpha set to .20 or .69, the tree generated from an sd2 truncated dataset has equal topological accuracy regardless of the heuristic search run time. This most likely implies that the sd2 truncated dataset is "easier" than the other 2 datasets; therefore, the extra time is not necessary to find a good final tree.

This evidence is strong support that our proposed fast evolving site elimination heuristic does not provide for improved topological accuracy. In fact, the analysis results in a final tree with worse topological accuracy than the tree that is estimated using the full dataset.

2.2.6 Heuristic performance on scaled model trees

Our final test involved scaling up the model tree. This scales all of the branch lengths in the model tree, which increases the height of this tree. The goal behind performing this procedure is to make the trees more difficult to estimate; we conjectured that if the datasets are harder to analyze, then the heuristic search will have better performance and accuracy when applied to the truncated datasets.

The alpha values chosen were the same as those used for the real dataset: alpha for nem is 0.26, while alpha for rbcL is 0.20. The model trees were scaled up by factors of 2, 4, and 8. Only the simulated nem and rbcl datasets was tested. Also, a 12 hour and 24 hour heuristic search was performed in exactly the same way as in the previous gamma test. In total, nine datasets were tested (sd1, sd2, and full datasets from three scale factors).

An interesting note is that PAUP* would never terminate while attempting to score datasets scaled by factors of 4 and 8. We allowed PAUP* 120 hours (five days) to complete processing, but PAUP* would not complete even in this extended window of time. The PAUP* process stalled after several hours, and no CPU time was used during the stalled period. We attempted to restart PAUP* several times, but PAUP* would never complete the heuristic search. Therefore, this data had to be excluded from the results. As a result, we were unable to compare the effect across scaling the datasets by varying values.

The results are mixed, as seen in Figure 2.10. Regardless of whether the heuristic search ran for 12 or 24 hours, the final tree obtained from the sd1 truncated nem dataset has very similar

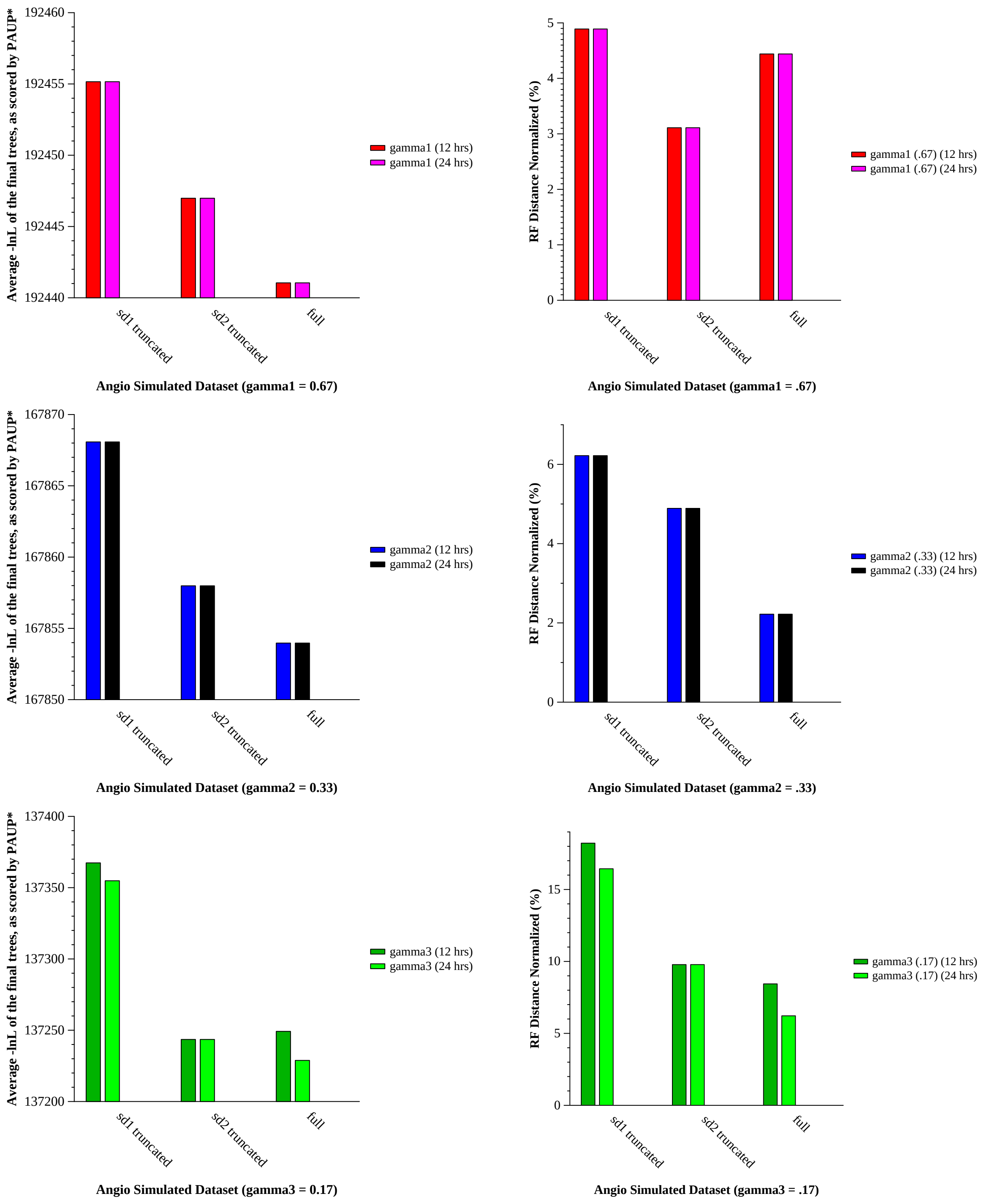


Figure 2.7: Comparison of 12/24 hour heuristic search for Angio dataset. Average -lnL and normalized RF distance are shown with varying gamma distributions.

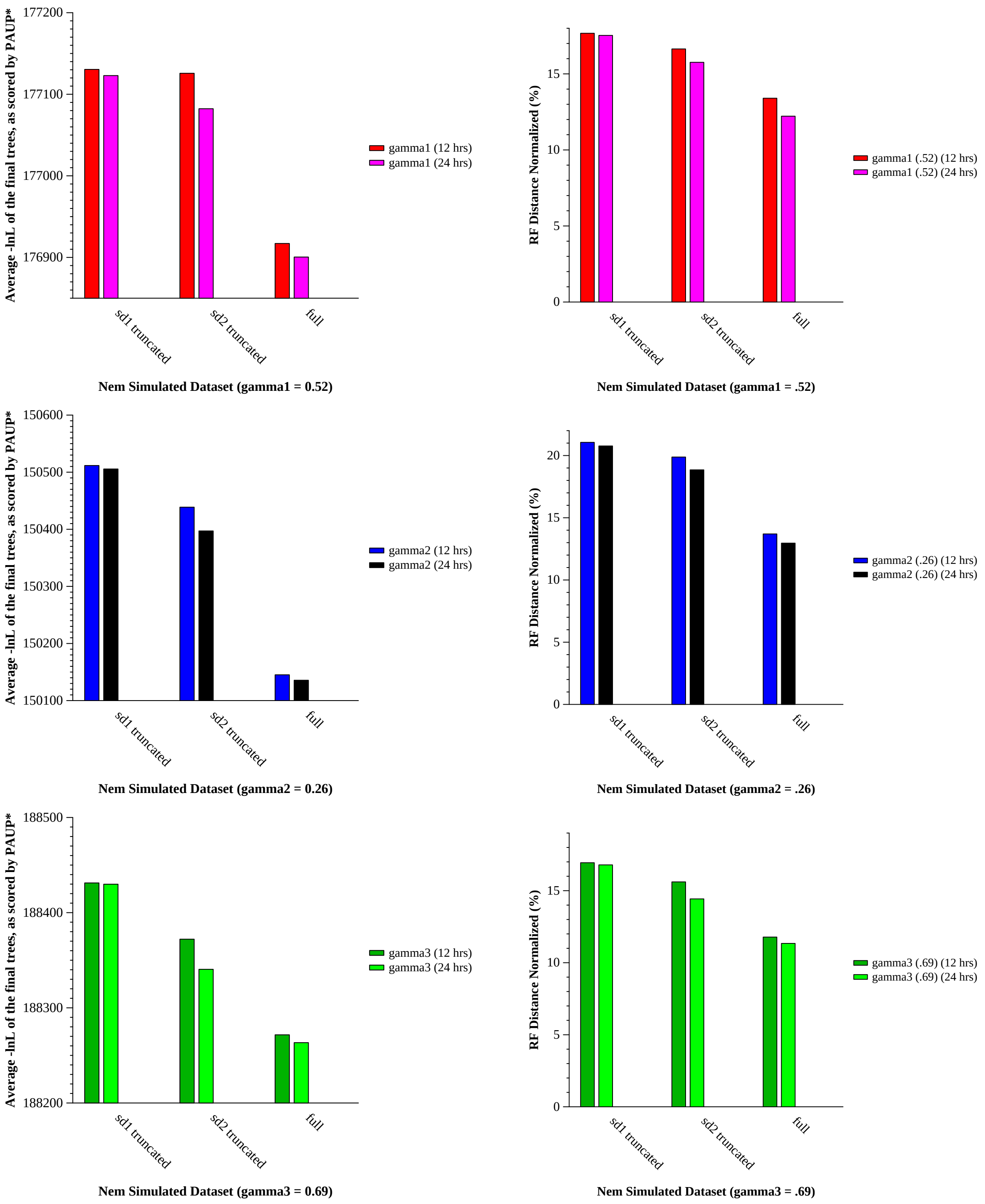


Figure 2.8: Comparison of 12/24 hour heuristic search for Nem dataset. Average -lnL and normalized RF distance are shown with varying gamma distributions.

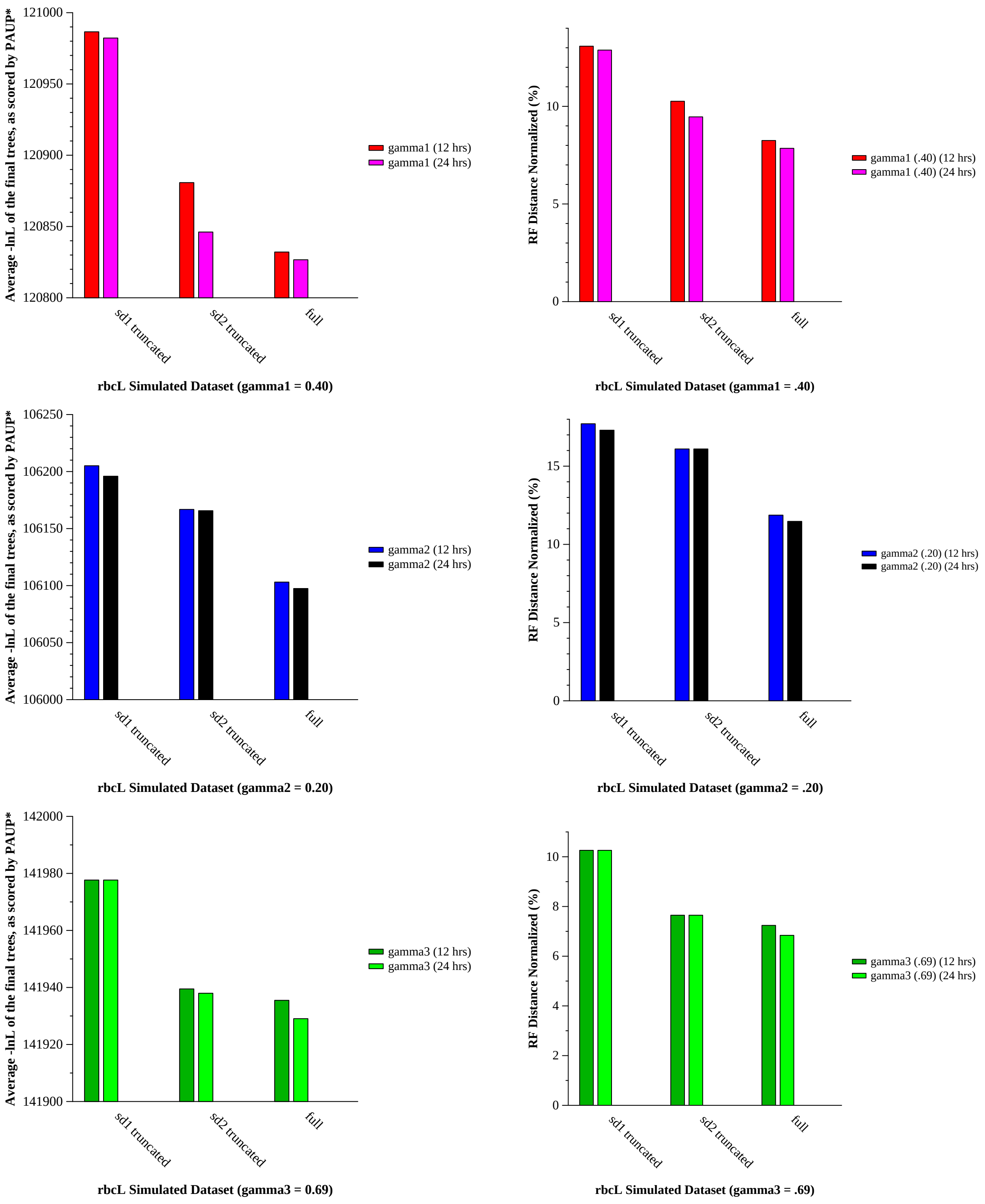


Figure 2.9: Comparison of 12/24 hour heuristic search for rbcL dataset. Average -lnL and normalized RF distance are shown with varying gamma distributions.

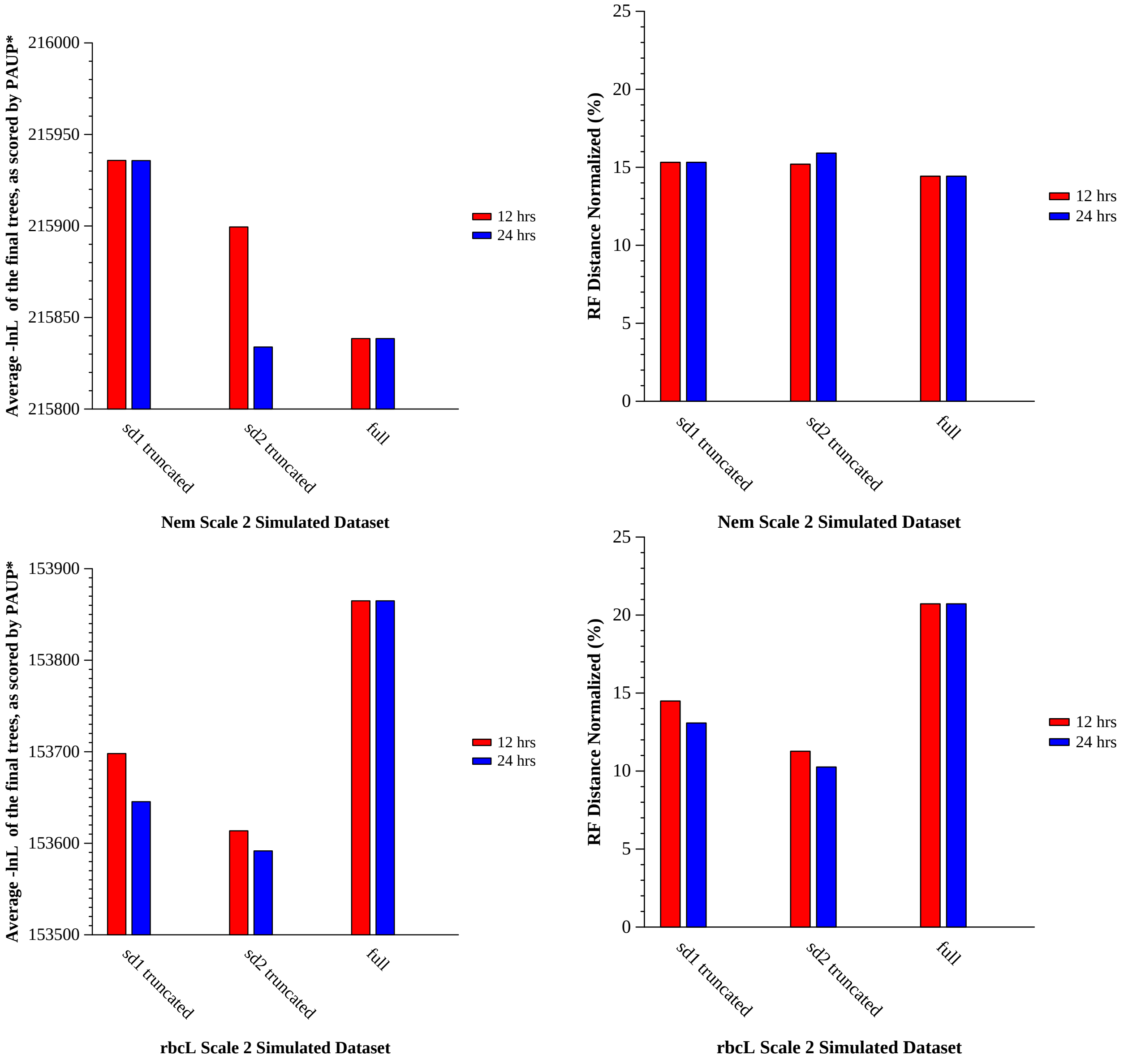


Figure 2.10: Average lnL and RF distance of the final trees for both the truncated datasets and the full dataset on angio, nem, and rbcl datasets. Model trees were scaled by a factor of 2 with an alpha value of .25, and the heuristic was halted after 12 or 24 hours. The alpha shape parameter was set to 0.25.

topological accuracy and performance. The same observation can be made for the nem full dataset. On the other hand, the nem sd2 truncated dataset turned out some unexpected results: although there is an improvement in -lnL when a 24 hour heuristc search is applied over the 12 hour heuristic search, the topological accuracy *reduced* when using the 24 hour heuristic search. This is very unexpected, as this relationship is not seen in any of the other data. Also, the full nem dataset performed better than the truncated datasets, which is not what we expected to see.

The rbcL results were different. We saw improved topological accuracy and performance using a 24 hour heuristic search on the truncated datasets compared to a 12 hour search. The heuristic search run time had no effect on the full dataset, however. In addition, the truncated datasets had better -lnL scores, but topological accuracy was worse than the full dataset.

In future work, we need to determine the reason why PAUP* would not terminate with input datasets scaled by factors of 4 and 8. It might be beneficial to use another ML package as a workaround. Obtaining results using datasets of various scales will aid in comparing the effect of using harder "data".

2.3 Discussion

We hope to confirm conjectured improvements in two areas when the fast evolving site elimination heuristic is applied to a dataset. First, eliminating fast evolving sites should have improved performance of maximum likelihood, which would allow the PAUP* maximum likelihood heuristic to achieve lower negative log likelihood (lnL) scores. Second, when simulated datasets are used, there should have been an improvement in topological accuracy (measured in RF distance) when the fast evolving sites were eliminated. The results were mixed.

2.3.1 Application of Heuristic to Real Datasets

We expected the fast-evolving site heuristic search to produce better results when using truncated, real datasets. This did not turn out to be the case; the heuristic search always returned a better tree using the full dataset rather than the truncated datasets. However, the differences in the negative log likelihood values were not as large as we had expected.

2.3.2 Application of Heuristic to Simulated Datasets

We conjectured that the real datasets were too "easy" to see a major difference among the application of the heuristic search on the different datasets. We used SeqGen to create five simulated datasets from a real angio dataset. There was very little change -lnL values among the different datasets. It is possible that these simulated datasets were not hard enough to see any difference between the truncated and full datasets.

2.3.3 Modification of Alpha Shape Parameters

In order to make the datasets harder, we altered the rate of evolution of the sites by changing the gamma distribution through modification of the alpha shape parameter. Various alpha parameters were chosen for angio, nem, and rbcL datasets. In general, the final tree generated from a full dataset had better topological accuracy than a tree generated from a truncated dataset. Also, a correlation tends to exist among the datasets: trees with lower log likelihood values tend to have better topological accuracy.

We believe it is possible that switching to a model that does implement rate variation across sites might actually improve our heuristic; therefore, removing the fast evolving sites from the dataset would possibly get us closer to the model tree. This will need to be investigated in future work.

2.3.4 Scaling of Model Trees

Our final test involved scaling the model trees, which increases the height of the model trees. This experiment, unfortunately, did not turn out any significant results. In fact, several tests would not complete, even when the maximum likelihood estimation was allowed to run for several days; the reason is unknown.

It is also possible that PAUP* is good enough to estimate trees from the supplied datasets. We propose that switching to another maximum likelihood program, such as RAxML [35], might reveal better results using the fast evolving sites heuristic. This again will need to be studied in future work.

After taking many approaches to eliminate fast evolving sites, we decided to focus our attention on improving tree alignment heuristics using multiple sequence alignment methods.

Chapter 3

Optimizing Multiple Sequence Alignment Heuristics

3.1 Introduction

In most phylogenetic analyses, before a phylogenetic reconstruction method such as maximum likelihood can be applied, the sequences must be placed in a data matrix called a *multiple alignment* [25]. In this matrix, the rows represent the sequences, and the columns represent *positional homology*, meaning that they are assumed to have evolved from a common ancestor. Evolution proceeds not only by substitution events but also insertion and deletion events, called *indels*, so the problem of determining the occurrence of indel events is an important and necessary part of phylogenetic analysis.

An *alignment history* is an evolutionary history that includes gaps to indicate indels. As an example, suppose that we have an ancestral sequence `ACATTG`, which has two children. One child is obtained from the ancestral sequence by the deletion of a `C` and the insertion of a `G` between the two `T`'s; and the second is obtained by the substitution of the terminal `G` with a `C`, and the deletion of the second `A`. Then the multiple sequence alignment may be represented either as a matrix or as a tree as follows:

```
ACAT-TG
A-ATGTG
AC-T-TC
```

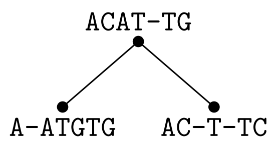


Figure 3.1: A multiple sequence alignment as a matrix and as a tree.

In other words, the full evolutionary history defines the true alignment. Hence, reconstructing a true alignment is necessary for reconstructing an evolutionary history. The goal is to estimate an alignment history from a set of unaligned sequences.

Unfortunately, the exact methods that return a true solution are not practical. These algorithms are not usable for anything larger than a few short sequences. Therefore, heuristic methods are necessary to approximate a solution; although the solution returned by a heuristic method may not be as accurate as that of an exact method, it can be computed very fast with little space.

3.2 Pairwise alignments

In phylogenetic analyses, *pairwise alignment* is the problem of inferring an alignment between two sequences. In most evolutionary models, it can be assumed that one sequence is the parent of the other, and so the pairwise alignment is the problem of determining an evolutionary history explaining the change from a parent to its child. This naturally leads to the formulation of edit distances, which forms the basis of alignment algorithms.

3.2.1 Edit sequences and edit distances

In the evolutionary model we have described, sequences evolve through insertions, deletions and substitutions. An *edit transformation* from a sequence A to a sequence B is a sequence of operations transforming A to B. The cost of a transformation depends on the cost of each substitution and each indel. The *edit distance* between A and B is the cost of an optimal transformation from A to B.

Here, we assume that we have an alphabet Σ of length r. The cost of a substitution is given by an $r \times r$ substitution cost matrix M. We use $M[\alpha, \beta]$ to denote the cost of substituting α with β. A match is a substitution of α with α, the cost of which is usually defined as 0.

Gap costs are assigned on the basis of indel length. There are two approaches to do this. A *linear gap cost* assigns a constant cost to each indel event. Hence, $g(l) = cl$. The second approach is known as the *affine gap cost*; in this method, the cost of a gap is equal to a gap initiation cost (c_0) plus a gap extension cost c_1 for each gap. Hence, $g(l) = c_0 + c_1 l$, where l is the length of the gap.

Clearly, an edit transformation implies an alignment and vice versa. Given two sequences A of length m and B of length n, we can determine the cost of an edit transformation from sequence A to sequence B. Let $c_{(\alpha,\beta)}$ be the number of substitutions of α by β, and let n_l be the number of gaps of length l. The cost of the transformation can then be computed:

$$\sum_{l \in \mathbb{N}^+} (n_l \cdot g(l)) + \sum_{(\alpha,\beta) \in \Sigma^2} (c_{(\alpha,\beta)} \cdot M[\alpha, \beta])$$

As an example, consider the following alignment.

```
ACGA--TTGATT
ACGGAATTC---
```

If the cost of a match is 0, the cost of a substitution is 1 and the cost of an insertion or deletion is 2, then the total cost of the alignment is 12.

Formally, *pairwise alignment* is the following optimization problem.

- Input: An alphabet Σ of size r, a sequence A of length m over Σ, a sequence of B of length n over Σ, an $r \times r$ substitution cost matrix M, and a gap-cost function $g(l)$.

- Output: An edit sequence transforming A into B via substitutions, insertions and deletions which minimizes

$$\sum_{l \in \mathbb{N}^+} n_l \cdot g(l) + \sum_{(\alpha,\beta) \in \Sigma^2} c(\alpha, \beta) \cdot M[\alpha, \beta],$$

  where n_l is the number of gaps of length l, and $c(\alpha, \beta)$ is the number of substitutions of α for β.

3.2.2 Algorithms

Needleman and Wunsch in 1970 describe the first algorithm for pairwise alignment of biological sequences using linear gap costs[24]. It optimized the alignment based on maximum similarity. This was a dynamic programming algorithm with a complexity of $O(mn)$ time and $O(mn)$ space, where m is the length of sequence A, and n is the length of sequence B. A nearly identical algorithm was independently developed by Sankoff [30] and Sellers [33], which aligned the sequences based on minimizing edit distance. Waterman et al., in 1976, created an algorithm with time complexity of $O(m^2n)$; it handled any type of gap [44]. Finally, in 1982, Gotoh constructed an algorithm for pairwise alignment with affine gap sequences in $O(mn)$ time [6].

All of these algorithms are dynamic programming algorithms that require $O(mn)$ space. Hirschberg refined this dynamic programming space requirement to $O(m)$ using a traceback technique[8]. The time requirement increases by a factor of two, however. Myers and Miller applied Hirschberg's technique to pairwise alignment with affine gap costs, obtaining an $O(mn)$ time, $O(m)$ space algorithm [23]. Quadratic time and linear space have from then on become the academic standard for pairwise alignment algorithms.

3.3 Multiple Sequence Alignment

Multiple sequence alignment (MSA) is the general problem of aligning a set S of sequences to determine homology. Several of the optimization problems are based on extensions of pairwise alignments to multiple alignments. This includes tree alignment, where the cost of a tree is the sum of its edge lengths; in addition, sum-of-pairs alignment returns a cost, which is the sum of the induced pairwise alignments.

We discuss generalized tree alignment, tree alignment, median alignment, and sum-of-pairs alignment.

3.4 Generalized Tree Alignment

Generalized tree alignment (GTA) is an extension of pairwise alignment for n sequences. This problem takes in a set S of n sequences and a function $f(\cdot,\cdot)$, which will compute the pairwise alignment cost between two sequences. Generalized tree alignment will return a tree T leaf-labeled by the set S of sequences and with additional sequences labeling the internal nodes of T to minimize

$$\sum_{(v,w)\epsilon E} f(A_{opt}(s_v, s_w))$$

where s_v and s_w are the sequences assigned to nodes v and w, respectively. E is the edge set of T, and $A_opt(s_v, s_w)$ denotes an optimal pairwise alignment on s_v and s_w.

With even simple affine gap penalties, generalized tree alignment is an NP-hard problem [42, 43]. This is because finding the optimal internal nodes for tree T to minimize the total cost is NP-hard.

Generalized tree alignment is a problem with really no satisfactory software. Current exact methods for finding the optimal tree run in $O(2^nk^n)$ time and $O(k^n)$ space for linear gap penalties, where n is the number of leaves in the tree, and k is the maximum sequence length[32]. The running time is greater for affine gap penalties. As a result, it is only possible to run exact GTA methods on very small sequences.

In order to run GTA on longer sequences, heuristic MSA methods must be used. These methods limit the search domain that MSA will operate in. As a result, the methods perform faster than exact MSA methods, but these heuristic methods are not nearly as accurate.

Because exact MSA methods are limited to short sequences, heuristic MSA methods are required to evaluate long sequences. However, although heuristic MSA methods are very fast, these methods are generally not very accurate. On the other hand, exact MSA methods are very accurate, but extremely slow. It would be a significant step if the parameters of the heuristic methods could be optimized so that these methods would be able to deliver the accuracy of the exact MSA methods.

3.4.1 Generalized Tree Alignment Heuristics

Heuristic methods for generalized tree alignment, such as GESTALT [18] and POY [12], exist, but none of them are able to scale well to datasets with hundreds of sequences and thousands of nucleotides in length.

The basic structure of a GTA heuristic can be described as such:

1. Examine a tree and try to label the internal sequences optimally to minimize the length (i.e. score it).
2. Repeat the following until no improvement found.
   (a) Move to a neighboring tree and score it.
   (b) If score improves, set optimal to current tree.

The process of scoring the tree is a computational subproblem known as *tree alignment*.

3.5 Tree Alignment

The purpose of multiple sequence alignment is to produce an alignment given a set of sequences. However, sometimes we want to produce an optimal alignment for a given, fixed tree. Tree alignment is the problem of producing a multiple sequence alignment on a set of sequences over a fixed tree. Formally, tree alignment is the following optimization problem.

- **Input**: A set S of n sequences each of length at most k, a tree T leaf-labeled by S and an edit distance function d between sequences.
- **Output**: A labeling of the internal vertices of T such that $\Sigma_{e \in T} d(e)$ is minimized, where $d(e)$ is the edit distance between the endpoints of e.

Figure 3.2 demonstrates a tree alignment problem. In this instance, the set S of sequences contains 4 taxa at the leaves. The true tree is shown on the left. Reconstructions typically work with unrooted trees. Using the set S of sequences along with a tree T leaf-labeled by S, we are attempting to label the internal nodes so that the resulting total edit distance of the tree is minimized. If the reconstruction successfully returns the tree with the smallest total edit distance, then the internal nodes in both the reconstructed tree and the true tree should be the same.

David Sankoff introduced tree alignment in 1975; he also developed an exact algorithm for linear gap penalties with a time requirement of $O(2^n k^n)$ and space requirement of $O(k^n)$[31]. Knudsen later presented an exact algorithm for affine gap penalties in $O(16.81^n k^n)$ time and $O(7.442^n k^{n-1})$ space[17]. Tree alignment is in fact NP-hard [42].

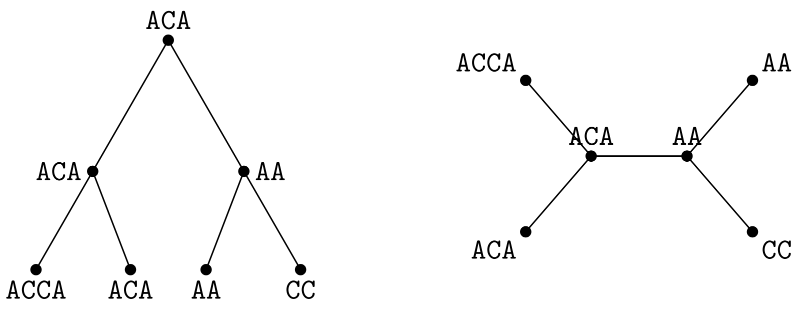
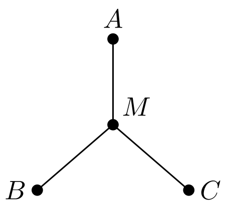


Figure 3.2: The true tree and reconstructing the true tree

3.5.1 Median Alignment

A *median alignment* is a specific type of tree alignment for aligning 3 sequences via a related median on a Steiner or star tree.



This tree is referred to as a *star* or *Steiner tree* since it only contains one internal node.

The cost function for a median alignment is evaluated as the sum of the three pairwise alignments between the three sequences and the median. In other words, median alignment, produces a sequence X, which minimizes:

$$d(X, A) + d(X, B) + d(X, C)$$

where d is a cost function describing the cost of the alignment.

Whereas the number of combinations of substitutions or indels remains small in the pairwise case, even for very complex gap functions, the number of combinations in dealing with medians is much larger, and therefore makes the alignment much harder to provably solve.

3.5.2 The Iterated Heuristic: a Heuristic for Tree Alignment

Considering that a median alignment only optimizes 3 sequences, one might question its usefulness in phylogenetic tree reconstruction. The purpose behind optimizing median alignments is to improve the iterated heuristic, which is a heuristic for tree alignment. The iterated heuristic finds the minimum score of a tree by computing medians for the internal nodes. Figure 3.3 illustrates uniform lifted alignment. With uniform lifted alignment, at each depth of the tree, all internal nodes are labeled with either their left or right child node sequence.

An iterated heuristic can be formally described as such:

1. For a fixed tree T leaf-labeled by a set of sequences S, label all the internal nodes. A *uniform lifted alignment* can be used to label the internal nodes; every depth of the tree will be labeled with either the left or right child. This will guarantee that the tree length as at most twice the optimal tree length. See Figure 3.3 for an example.

2. Pick an internal node g, and perform a median alignment of its two child nodes and its parent node. If the score for the median is less than the score for g, replace g with the median.

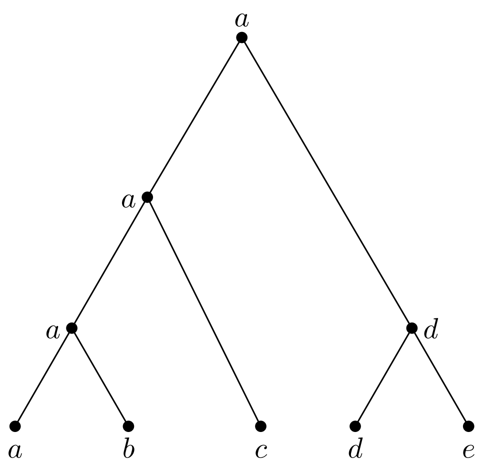


Figure 3.3: Example of uniform lifted alignment with internal nodes labeled by left children

3. Repeat step 2 on all internal nodes until there are no more changes that can be made to all internal nodes.

Performing a median alignment on each internal node will improve the overall tree length. This is why optimizing the median alignment is important.

3.6 Alignment Methods for Three Sequences

A median alignment or sum-of-pairs alignment can be used to align three sequences.

3.6.1 Sum-of-Pairs Alignment

Sum-of-pairs (or SP) alignment is a type of alignment that operates directly on the sequences, without reference to a tree. Formally, SP alignment is the following optimization problem:

- **Input**: A set S of n sequences and an edit distance function d between sequences,
- **Output**: A global alignment A of the sequences in S that minimizes

$$\sum_{k<k'\leq n} d(A_{k,k'}),$$

  where $A_{k,k'}$ is the pairwise alignment between sequences k and k' induced by A.

SP alignment is NP-complete [42].

A simple dynamic programming sum-of-pairs alignment algorithm has exponential time and space complexity, as with tree alignment.

3.6.2 Algorithms

A majority of the alignment algorithms available use sum-of-pairs alignment. However, it's possible to convert a sum-of-pairs alignment to a median alignment, by estimating the median from the sum-of-pairs alignment. In order to compute this median alignment from a sum of pairs alignment, for each site in the alignment, we choose a symbol from $\Sigma \bigcup \{\text{-}\}$ to minimize the tree length on that site. If this is done for all sites, we will have found the optimal solution for that median when using a site-by-site analysis. However, this may not always be the optimal solution.

In 1985, Murata et al. developed an algorithm based on that of Needleman and Wunsch to obtain a sum-of-pairs algorithm for linear gap penalties [21]. Gotoh later developed a sum-of-pairs alignment algorithm for affine gap penalties in $O(n^3)$ time and space, where n is the number of sequences [7]. However, Gotoh's algorithm only allows for combinations of moves that contain at least two matches, and as a result, the entire search space is not traversed. Later, Huang improved Gotoh's algorithm by reducing the $O(n^3)$ time and space requirement to $O(n^2)$; however, this algorithm also suffered from the same setbacks as that of Gotoh's algorithm [10].

An alignment algorithm was independently presented by Ukkonnen [41] and Myers [22] that runs in $O(kd^2)$ time in the worst case, where k is the maximum length of the two sequences, and d is the minimum edit distance. This algorithm also requires $O(d^2)$ space, but this can be reduced further to $O(d)$ space if one of the two conditions occur: only an alignment cost is required, or if the Hirschberg technique is used. The standard pairwise alignment techniques are designed to work with arbitrary substitution and gap cost constants. However, if these costs are limited to small integers, a more efficient pairwise alignment algorithm can be developed.

A tree alignment algorithm for 3 sequences using linear gap penalties was published by Powell et al. [28]. One was a general purpose solver with $O(n^3)$ time and space bounds. The other was a Ukkonen-based solver. Powell's Ukkonnen-based solver computes an alignment for three sequences and also computes the minimum tree length. However, it does not output a median. It is a case-based solver, such that in certain situations, it will vastly outperform an exact solver. However, in other situations, it will vastly underperform an exact solver.

Knudsen's algorithm for tree alignment using affine gap penalties can also be used to estimate a median in $O(16.81^n k^n)$ time and $O(7.442^n k^{n-1})$ [17].

All of the sum-of-pairs alignment algorithms are exact methods. These algorithms are able to run within cubic time and space; therefore, they are not practical, since processing power limits the sequence length to short sequences. As a result, heuristics were developed to improve the performance of median alignment. Lancia and Ravi developed GESTALT, which includes an exact method for pairwise alignment with affine gap penalties, an exact method for median alignment with linear gap penalties, and a heuristic method for median alignment with linear gap penalties[18]. The heuristic first computs all optimal alignments between two sequences and determines the best way to complete the alignment when adding the third sequence. This technique reduces the time required from $O(n^3)$ to $O(n^2)$. It would be interesting if one could extend this technique to affine gap penalties.

3.6.3 An Exact Method for Median Alignment

There is an inherent problem in using sum-of-pairs alignment to estimate a median. It computes a sum-of-pairs alignment, which although the alignment may be an optimal solution when using sum-of-pairs alignment, it may not be an optimal solution when using a median alignment. Also, since MSA doesn't compute a median, the median produced by the reconstruction methods may not agree with sum-of-pairs alignment.

It may be hard to see why a sum-of-pairs alignment and median alignment would be different, given 3 sequences, A, B, and C, and a cost function d. The reason is that the alignments operate differently. A sum-of-pairs alignment would minimize:

$$d(A, B) + d(A, C) + d(B, C)$$

A median alignment, produces a sequence X, which minimizes:

$$d(X, A) + d(X, B) + d(X, C)$$

A median alignment will operate on aligning the sequences with X, whereas a sum of pairs alignment optimizes each of the sequences directly with each other.

In fact, none of the heuristics, excluding GESTALT, is able to compute a median alignment. GESTALT is able to compete a median alignment, but only for linear gap penalties. Since we are working with affine gap penalties, we need a method that will compute a median alignment for affine gap penalties.

Tandy Warnow and David Zhao developed an exact method for median alignment for affine gap penalties. The best known exact median alignment methods, which operate on linear gap penalties, run in $O(n^2)$ space and $O(n^3)$ time. Warnow's and Zhao's exact method for affine gap penalties operates on $O(n^3)$ time and space, so it can run reliably only on sequences of length 200.

3.7 Goals

The overall goal we are trying to achieve is to create an improved heuristic for generalized tree alignment that can scale well to datasets with hundreds of sequences and thousands of sites. We believe that in order to achieve this goal, we need to optimize the scoring component of generalized tree alignment: tree alignment.

To improve tree alignment, we hope that improving median estimators will result in an improved tree alignment iterated heuristic. Current exact median calculators are only limited to around 200 sequences, which is insufficient and impractical for any systematist. We conjecture that using existing heuristic median calculators will be sufficient to improve tree alignment performance, while maintaining accuracy.

A second goal is to evaluate the current GTA heuristics (such as GESTALT) in terms of topological accuracy and compare these heuristics to our two-phase reconstruction method approach.

3.8 The Problem

It is very hard to compute an exact alignment in a reasonable amount of time and space. Approximate solutions are a workaround to this problem, since a heuristic method can return a solution quickly and efficiently. However, heuristic methods are generally not very accurate: the solution that is returned can deviate greatly from the true solution.

The problems associated with a heuristic method that result in less than optimal solutions are:

- Every heuristic method uses a different algorithm to compute an approximation. The algorithms used by these methods will be discussed later. Since no standard exists for comparing heuristic methods, each method is designed and optimized for different kinds of data. As a result, a biologist must be careful to choose the right heuristic method for his or her data.

- The parameters used for a heuristic method may not be tuned to deliver optimal performance. Altering these parameters can result in significant performance improvements.

We first evaluated the computational limitations of exact median calculators for short sequences. We also compared the quality and computational limits of the heuristic methods to that of the exact methods. Finally, we compared a select group of heuristic methods in terms of estimating the true tree length.

Here's a brief overview of the process:

1. A sequence generator known as ROSE is used to generate a set S of 3 sequences. ROSE also generates the true alignment as well as the true tree.

2. Set S is then given as input to the heuristic methods along with a certain set of parameters. Each method returns an alignment.

3. A median is estimated using the alignment generated by the method. This is done using a median consensus algorithm, which estimates a median using Maximum Parsimony. Table 3.1 explains how this algorithm works. We now have a median tree.

4. The median tree can now be compared to the true tree. Accuracy is measured with respect to (a) minimization of tree length, or (b) distance to true median.

Sequence						
A	A	A	A	A	A	A
B	A	A	C	A	C	-
C	A	C	T	-	-	-
Selected Nucleotide	A	A	A/C/T	A	A/C	-

Table 3.1: This table shows the six possible cases required to compute a median, given three aligned sequences, A, B, and C. The median consensus method uses a maximum parsimony analysis to pick the median site. We choose the most common nucleotide. If all 3 nucleotides are different, then the median nucleotide is randomly picked from the 3 nucleotides. We also prefer a nucleotide to a gap, unless two gaps exist: in this case, the gap is picked.

3.9 Simulating

In order to optimize the heuristic multiple sequence alignment algorithms, we needed to generate sets of three phylogenetically related sequences. We employed a software tool known as ROSE [38], which implements a probabilistic model of evolution for DNA, RNA, and protein sequences. A family of evolutionarily related sequences is created based off an evolutionary tree. This family of sequences are derived from a common ancestor by insertion, deletion, and substitution of nucleotides.

ROSE also offers parameters to alter the various properties of the sequences produced. Table 3.2 presents a detailed description of the simulation parameters used in this study. We explored a wide range of parameters and decided that these parameters had the greatest effect on the heuristic methods.

In addition to generating the sequences that will be evaluated, ROSE also outputs the true alignment for the set of sequences. This true alignment is very important, as we will compare this alignment with the alignments of the various MSA methods. This will allow us to benchmark the accuracy of the various MSA methods. In addition, ROSE outputs the true median sequence, or the sequence from which the other sequences were derived.

Name of Parameter	**Description**
Sequence Length	Length of each sequence at the root.
Edge Lengths	Expected number of changes on an edge.
Mean Substitution Rate	Probability of a substitution from the ancestor to the new sequence per unit of edge length.
Indel Threshold	Defines probability that an indel event will occur from the ancestor to the new sequence
Indel Length Distribution	If ROSE decides an indel event should occur (denoted by indel threshold), this distribution denotes how long the indel event should be. We hard coded two distributions: "long", with an average indel of 9.1837 (st. dev: 7.1858), and "short", with an average indel of 1.9981 (st. dev: 1.1624).
Max Indel Length	Maximum allowed length of an indel event.

Table 3.2: These are the ROSE simulation parameters modified in this study.

3.10 Reconstructing

Once the set of sequences were generated by ROSE, we ran various multiple sequence alignment methods on this set. Afterwards, we compared the accuracy of the alignments produced by the various MSA methods, along with other statistical data.

Several MSA methods were used for this study. These include ClustalW [40], MSA [19], DCA [39], T-Coffee [26], MAFFT [15, 14], and MUSCLE [2, 3]. We discuss two classes of algorithms that these methods use: progressive algorithms and tree-less algorithms.

3.10.1 Tree-less Algorithms

It would be ideal if the set of sequences could be aligned simultaneously instead of appending each sequence one by one to an alignment. Tree-less algorithms are able to align the entire set at once, but due to time and space issues, the set can only be limited to up to three sequences with no more than 200 nucleotides. Unlike progressive algorithms, tree-less algorithms do not use a guide tree. MSA is a program developed by Lipman et al [19] and is able to compute an exact multiple alignment on a set of more than 3 sequences by limiting the search space that is traversed. It performs a sum-of-pairs alignment to minimize the total alignment cost. However, if an exact solution is not found within this search space, MSA will terminate and not return an alignment. Even operating within the truncated search space, MSA is very time and space intensive. This was a major problem, and as a result, the MSA program was not widely adopted.

Nearly a decade later, Stoye introduced the divide and conquer algorithm (DCA), which uses MSA as a base method; DCA can take any base method[39]. The DCA algorithm splits up the set S of sequences into small chunks which can be fed into MSA. The aligned chunks have to then be concatenated. DCA is intelligently able to determine how to split up the set into chunks so that concatenation of the results can be straightforward. If a specified chunk does not result in a good alignment, DCA splits up the chunk into more chunks and recomputes an alignment. This process continues until all chunks can be optimally aligned. The result is a method that is much faster than MSA [37]; on a set of 6 sequences with length between 255 and 293 amino acids, MSA computed an alignment in 118 minutes, while DCA computed the same alignment in just 61.8 seconds.

Early methods, such as the MSA method, use sum-of-pairs alignment to find a solution. However, recent methods use guide trees to find a solution.

3.10.2 Progressive Algorithms

Progressive algorithms are widely implemented among heuristic methods. These algorithms, originally described by Hogeweg [9], are very effective in aligning a set of sequences in very little time and space. The basic idea is to compute a pairwise alignment edit distance for all sequences in a set S. A guide tree is created fromm these distances. Next, the two most closely related sequences, as determined by the guide tree, are aligned to create an alignment, which is stored in the parent node of these sequences in the tree. Once all of the closest sequences have been aligned, this process is repeated for the closest alignments among the subtrees. This process continues until the root contains an alignment.

ClustalW is a heuristic method that implements a progressive algorithm. It is a non-iterative, deterministic algorithm that works with affine gap penalties. It works by first computing a dendrogram of sequences based on a pairwise sequence alignment algorithm. These sequences are then added one by one to the multiple alignment based on the order of the sequences in the dendrogram. The obvious disadvantage of this is that once a sequence has been added to the multiple alignment, its alignment cannot be altered at a later point in the process.

T-Coffee is also another heuristic method that implements a progressive algorithm. However, it attempts to avoid the problem that plagues ClustalW's alignment strategy. T-Coffee first creates a record of all the pairwise alignments between the set of sequences. As a result, intermediate alignments are based not only on the sequences to be aligned next, but also on the overall big picture view of how all the sequences align with each other. The overall result is that T-Coffee is more accurate than ClustalW at the expense of modest sacrifices in speed [26].

In 2002, MAFFT was developed, which employs a new algorithm to improve speed and accuracy over CLUSTALW and T-Coffee [15]. The process consists of a progressive alignment in a manner similar to T-Coffee, followed by a second progressive alignment on the new set of aligned sequences. Once this is complete, a third step refines the alignment to produce an optimal score. MAFFT's performance is impressive: using a BAliBASE benchmark [11], MAFFT on average computed an alignment in 227 seconds, whereas ClustalW computed the same alignment in 1657 seconds.

Another method used is known as MUSCLE, and it operates in a similar fashion to MAFFT [3]. The algorithm used in MUSCLE results in improved space and time complexities. Both MAFFT and MUSCLE are designed for proteins.

3.10.3 Parameters for Reconstruction Methods

In an attempt to optimize the reconstruction methods, we altered three main parameters of these methods: substitution cost, gap initialization cost, and gap extension cost. These parameters are defined in Section 3.2. However, these parameters are not available in all of the methods. Table 3.3 details which methods make use of the reconstruction parameters.

3.10.4 Statistical Data

Table 3.4 presents the statistical data gathered. All of the data gathered was averaged over the total number of replicates.

Method	Mismatch Cost	Gap Initialization	Gap Extension
MSA	✓	✓	✓
DCA (MSA)	✓	✓	✓
ClustalW	—	✓	✓
T-Coffee	✓	✓	✓
MAFFT	—	—	—
MUSCLE	—	—	—

Table 3.3: Parameters used by the reconstructive methods.

Statistic	**Description**
Average Tree Length	Defined as the sum of the edit distance for all edges. This is what we are optimizing.
Average Distance to True Median	Distance between True Median and Estimated Mean. It is measure of accuracy.

Table 3.4: Statistical data gathered for each experiment.

3.11 Results

We performed two studies; first, we compared median estimation among multiple sequence alignment heuristic methods to an exact median alignment algorithm. The dataset that was analyzed consisted of three sequences with a starting length of 200 sites/nucleotides. We tested shorter sequences so that the heuristic methods could be compared to an exact method. Secondly, we selected a subset of the heuristic methods tested in the first group and compared them to the true median using three sequences of length 5000. Longer sequences were used to evaluate heuristic methods under more practical conditions.

All of the tests were executed on the CIPRES cluster in the San Diego Supercomputer Center (SDSC). Each machine utilized four processors, and each processor was a dual-core 2.5Ghz AMD Opteron.

50 replicates were carried out for each test, and the data reported is an average of these 50 replicates.

3.11.1 Explanation of Figures

The figures show *Percent above Exact Tree Length* and *Normalized Distance to True Median.* "Percent above Exact Tree Length" compares the tree length of a specific tree created from specific parameters to the tree length of the optimal tree as determined by an exact method under the same optimal conditions. Percentages that approach 0 mean that the reconstructed tree and the exact tree have similar tree length. Since we want the heuristic methods to return trees that closely approximate the optimal median tree, values that are closer to 0% are ideal.

"Normalized Distance to True Median" is a measure of the similarity between the actual alignment as determined by Rose and the reconstructed alignment as determined by the reconstruction method. It is determined by computing an edit distance between the true median sequence and the reconstructed median sequence. The data is normalized with respect to the true tree length. Ideal values are those that approach 0; this indicates that the two alignments are similar.

3.11.2 Comparison of Heuristic Methods to an Exact Method for Sequences of Length 200

Six methods (MSA, DCA (MSA), ClustalW, T-Coffee, MAFFT, and MUSCLE) were applied to four test cases. Three sequences with a length of 200 nucleotides beginning at the root were generated. Since indel events occur, the three sequences may be greater than or less than 200 nucleotides in length. We can execute an exact median alignment method for affine gap penalties using these sequences because the sequences are short. We compared tree length of the estimated median for each method to the optimal tree length as determined by an exact method for affine gap penalties.

We used two different exact median alignment methods under various conditions. For mismatch cost of 1, we employed Powell et al's Ukkonen-based method. This method has a time complexity dependent on the total tree length (sum of the edit distance on three edges) of the alignment. Therefore, this method was too slow to compute an exact tree length when mismatch cost is 2. Therefore, we used an in house exact median solver for mismatch cost of 2. Both methods return the same tree length for three given sequences; the difference is found in running time performance.

A gap initialization cost of 5, and a gap extension cost of 1 was maintained throughout this study. Mismatch cost was varied with a value of either 1 or 2.

Three parameters were studied: indel threshold, substitution rate, and indel length distribution. For each study, one parameter was varied, while the others were fixed.

Indel Threshold

To observe the effect of indel threshold on the various heuristic methods, several parameters were maintained at a constant value. In general, low values were picked for the parameters; lower values tend to be more biologically accurate, and allow the true tree length to approximately follow the exact tree length. Table 3.5 details the parameters, and the results are shown in Figure 3.4 when mismatch cost is 1.

Parameter	Value
Sequence Length	200
Indel Threshold	0.000 to 0.030 (step size: 0.001)
Mean Substitution Rate	0.1
Indel Length Distribution	short
Edge Lengths	Equal (1:1:1)
Mismatch Cost	1 or 2

Table 3.5: Parameters maintained to observe effect of varying indel threshold.

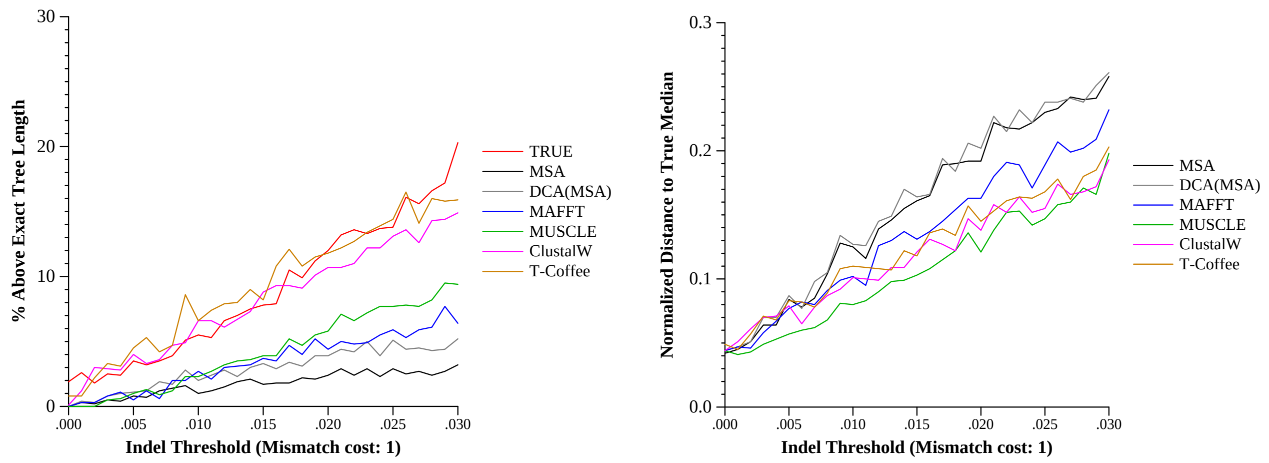


Figure 3.4: Effect of indel threshold on heuristic median estimation when mismatch cost is 1. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

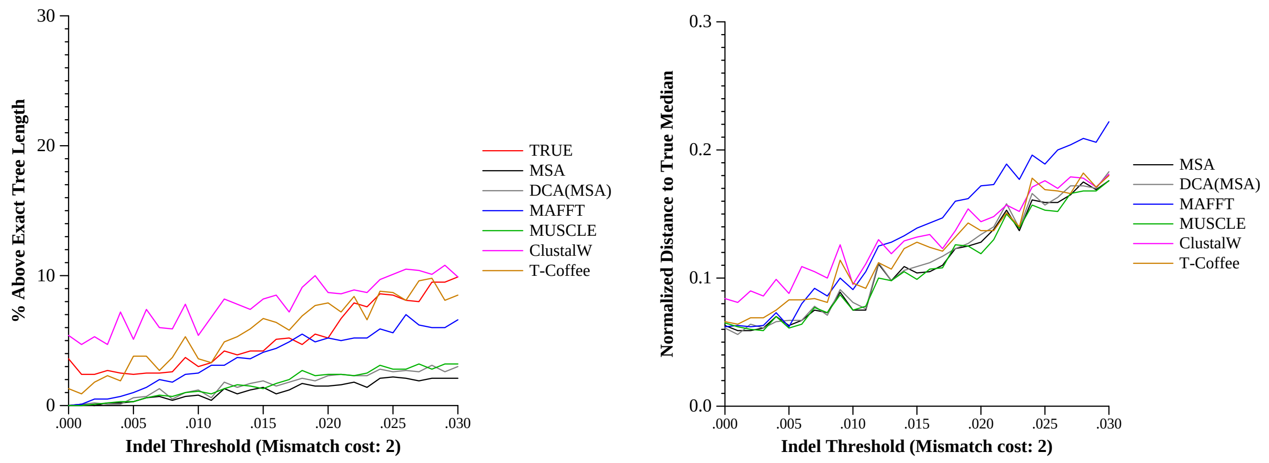


Figure 3.5: Effect of indel threshold on heuristic median estimation when mismatch cost is 2. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

Figure 3.4 indicates that, among all heuristic methods, MSA is able to deliver the most optimal tree, regardless of indel threshold. DCA(MSA) follows closely behind, along with MAFFT and MUSCLE. However, MSA and DCA(MSA) are poor at returning an accurate tree, which is reflected by the high distance to true median. MUSCLE is able to find an optimal tree that is very closely related to the true tree.

On the other hand, ClustalW and T-Coffee do not return close to optimal tree lengths, but the tree length score closely matches that of the true tree; in other words, the tree that the two methods return have a similar tree length as the tree that represents the actual evolutionary history. In addition, ClustalW and T-Coffee return a reconstruction with low distance to true median, which means the accuracy is good.

Figure 3.5 shows the results when mismatch cost is 2 and indel threshold is varied. All of the methods output a tree with tree length within 10% of the optimal tree length. This means that the higher mismatch cost of 2 is causing the methods to return more optimal solutions. In terms of accuracy, MSA and DCA(MSA) greatly improve with a much lower distance to true median. In fact, the distance to true median of the trees generated by MSA and DCA(MSA) is very close to that of MUSCLE.

As the indel threshold increases, the tree length of the heuristic methods increases (which means the results are less optimal) over time. This is understandable, since an increase in the number of indels will make the dataset "harder" to analyze. This is due to the fact that there are more indel events that must be taken into account in order to perform a good alignment.

Substitution Rate

We also varied the substitution rate. The parameters are listed in Table 3.6, and the results are shown in Figure 3.6.

T-Coffee and ClustalW again return median trees with tree lengths similar to that of the true tree; however, in this case, the trees are not accurate when substitution rate is greater than 0.10. In fact, the distance to true median increases with increasing substitution rate for all methods. It is likely that the higher substitution rates (greater than 0.10) are not biologically accurate, and therefore the reconstruction accuracy degrades as these methods are not able to approximate the true tree. Instead, these methods would be attempting to return an optimal tree, which is not similar to the true tree at these higher substitution rates.

Percent above exact tree length scores of the final trees from MSA, DCA(MSA), MUSCLE, and MAFFT are unaffected by the increasing substitution rate. Although this is the case, the accuracy of these reconstructions still degrade as substitution rate is increased.

When mismatch cost is set to 2 , the results are not very different. Figure 3.7 shows the results. However, the true tree more closely correlates with the exact tree. We want the true tree to be as close to the exact tree as possible because heuristic methods are designed to optimize the output to deliver the best optimal tree. In addition, T-Coffee fares better with the higher mismatch cost. Although tree length scores of MSA and DCA(MSA) are unaffected by varying the mismatch cost, these two methods don't have similar accuracy at the higher mismatch cost. MUSCLE, on the other hand, returns trees with low distance to true median scores regardless of mismatch cost.

Parameter	Value
Sequence Length	200
Indel Threshold	0.01
Substitution Rate	0.0 to 0.30 (step size: 0.01)
Indel Length Distribution	short
Edge Lengths	Equal (1:1:1)
Mismatch Cost	1 or 2

Table 3.6: Parameters maintained to observe effect of varying substitution rate.

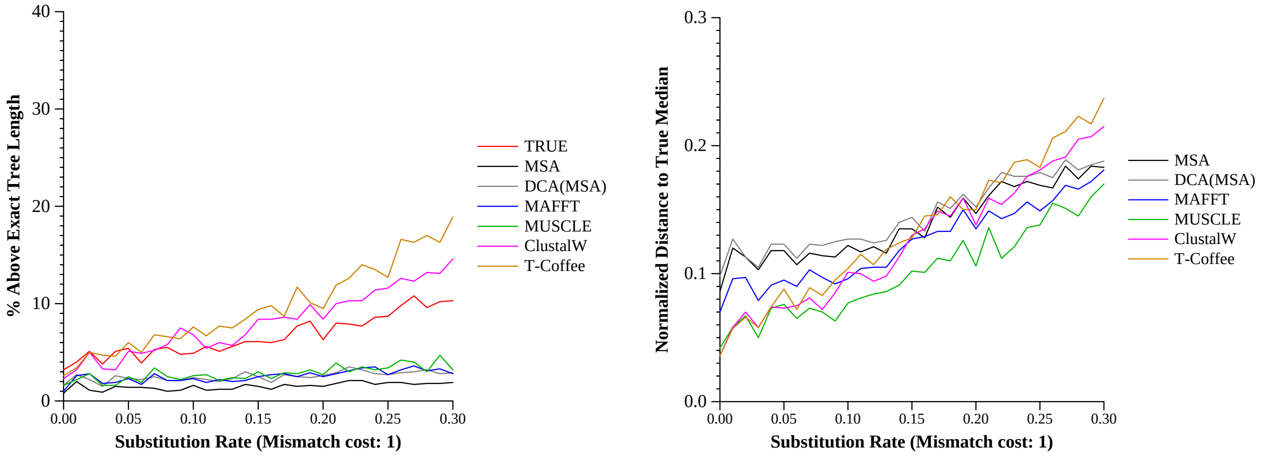


Figure 3.6: Effect of substitution rate on heuristic median estimation when mismatch cost is 1. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

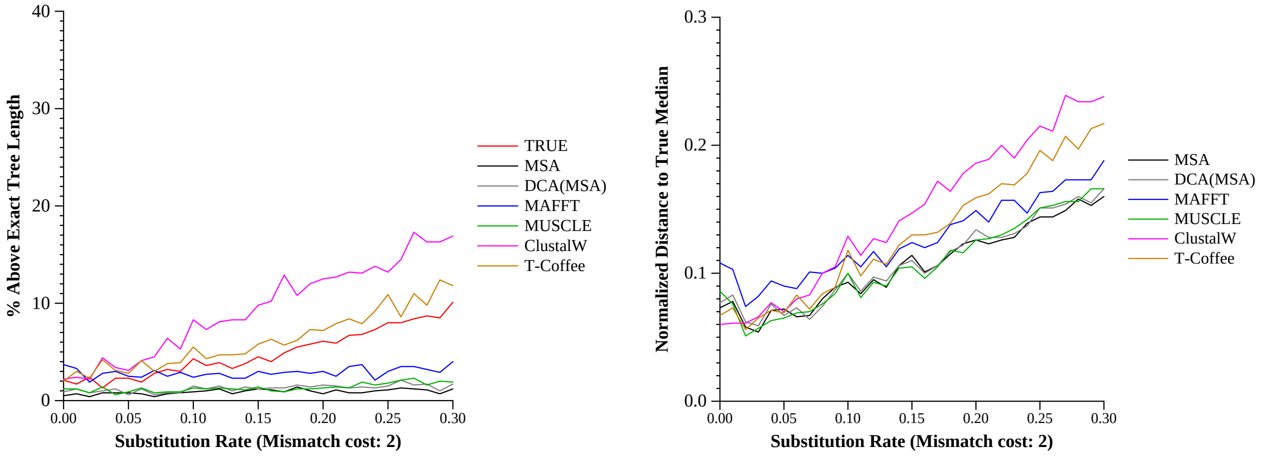


Figure 3.7: Effect of substitution rate on heuristic median estimation when mismatch cost is 2. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

Indel Length Distribution

We also compared the effect of the indel length distribution on the heuristic methods. The indel length distribution is a geometric distribution with a maximum sequence length of 50; a parameter is used (which ranges from 0.1 to 0.5) to define the shape of the distribution. If this parameter is small (i.e. 0.1), then the probability of a long indel will be higher than if the parameter is large (i.e. 0.5). The parameters are listed in Table 3.7, and the results are shown in Figure 3.8 for mismatch cost 1. Results for mismatch cost 2 are shown in Figure 3.9.

MSA, DCA(MSA), MAFFT, and MUSCLE are unaffected by the varying indel length distribution. The % above exact tree length stays constant for these methods over the varying indel distribution values, and there is a modest improvement in accuracy as the indel distribution increases. For the other methods, there is a gradual decease in % above exact tree length as the indel distribution parameter is increased. Since the average indel length decreases as the indel distribution parameter is increased, the dataset gets “easier” with higher indel distribution parameters, so the ability to derive homology from the unaligned sequences is also easier. Therefore, we would expect the methods to improve in performance.

Some of the trends seen when mismatch cost is varied for indel length distribution was also seen when mismatch cost was varied for substitution rate and indel threshold. First, the true tree length closely approximates the optimal tree length for the higher mismatch cost. Second, T-Coffee is able to return a more optimal solution under the higher mismatch cost conditions. Finally, MSA and DCA(MSA) once again return more accurate trees when the mismatch cost is 2.

Parameter	Value
Sequence Length	200
Indel Threshold	0.01
Substitution Rate	0.1
Indel Length Distribution	Geometric with parameter: 0.10 to 0.50 (step size: 0.01)
Edge Lengths	Equal (1:1:1)
Mismatch Cost	1 or 2

Table 3.7: Parameters maintained to observe effect of varying indel distribution.

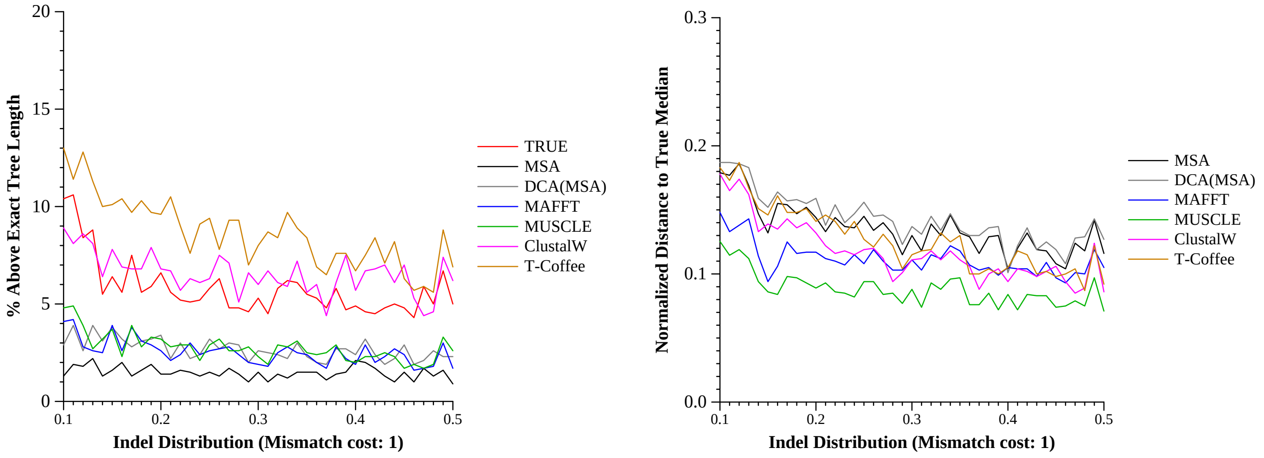


Figure 3.8: Effect of indel distribution on heuristic median estimation when mismatch cost is 1. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

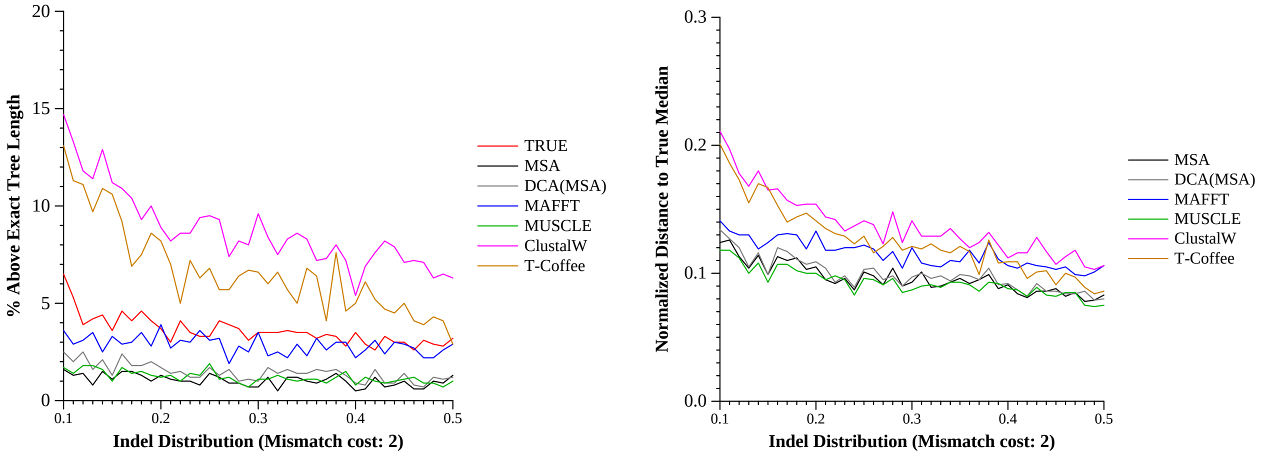


Figure 3.9: Effect of indel distribution on heuristic median estimation when mismatch cost is 2. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

It is clear from the results presented thus far that the exact method produces a median tree with a tree length score lower than can be produced by any of the other heuristic methods. This is due to the fact that it is an exact method for affine gap penalties. However, as shown in Figure 3.10, an exact method increases with cubic time. On the other hand, the heuristic methods, such as MUSCLE, have only slight increases in running time to compute an alignment as the sequence length increases.

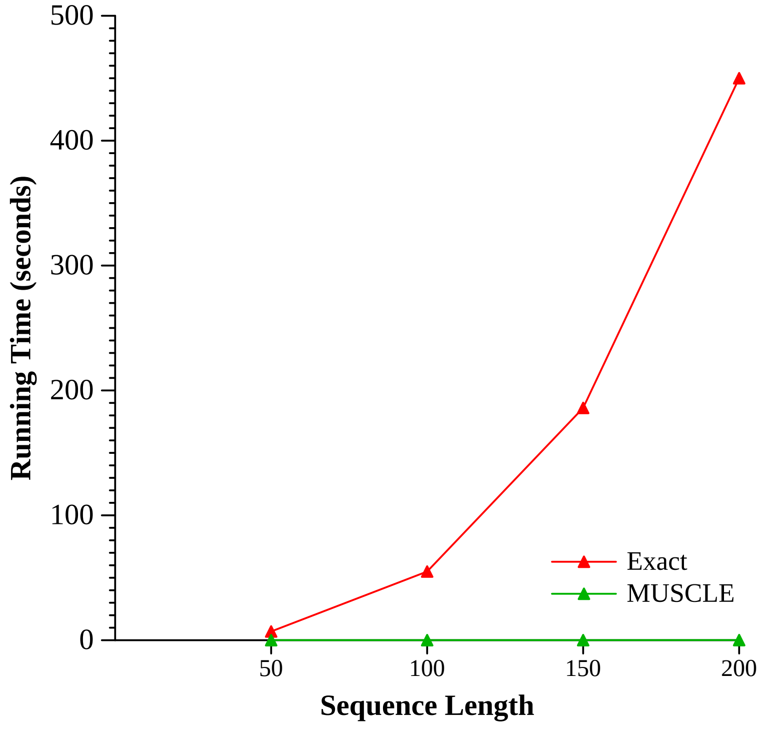


Figure 3.10: Running time (sec) comparison between the exact method and a heuristic method (MUSCLE).

3.11.3 Comparison of Heuristic Methods to an Exact Method for Sequences of Length 5000

In the first study, we estimated medians using an initial ancestral sequence of 200 nucleotides. Although this allowed us to obtain a tree length of an optimal median alignment, small sequences are not actually used by systematists. On average, practical datasets use sequences of 1000 to 5000 nucleotides in length. We compared a subset of the heuristics tested in the first study on a dataset with an initial ancestral sequence of 5000 nucleotides. Therefore, it is not possible to use the exact median estimator, since our exact median estimator will crash with sequences of length greater than 200. As a result, we are only able to compare medians of the heuristic methods to the true tree.

We compared DCA(MSA), MAFFT, and MUSCLE, which produced medians with tree length scores comparable to the exact method and had good running time performance. ClustalW was also analyzed since it is the most popular MSA method in use by systematists. MSA also produced good tree length scores, but the running time of MSA is prohibitively slow to perform on large datasets; therefore, we did not include it in this analysis. T-Coffee also was not included in this analysis because of it's prohibitive running time and mediocre performance.

The parameters tested are similar to those tested in the first study. In addition, the gap initialization cost was always 5, and the gap extension cost was always 1. Also, when the indel length distribution was not the subject of comparison, we used the "short" distribution as the standard.

Indel Threshold

Figure 3.11 illustrates the effect of indel threshold on tree length with respect to the *true* tree length for mismatch cost 1, using parameters specified in Table 3.8. This data was rather unexpected in some regards. When the indel threshold is greater than 0.005, all of the four heuristic methods produce medians with lower tree length than that of the true tree. Since the true evolutionary history (true tree) is not usually the same as the actual tree, we would expect the heuristic methods to produce medians with lower tree lengths than the true tree. This is due to the fact that we are optimizing the methods to return the most optimal tree instead of optimizing the methods to deliver a tree closest to that of the true tree.

Figure 3.11 also shows that the tree length of the heuristic method slightly worsens when the indel threshold is between 0.0 and 0.005. Our best explanation for this is that these heuristic methods are designed to align sequences; when only a few indels are present in the actual alignment, a heuristic method will overcompensate and introduce too many indels into the alignment. Therefore, the tree length of the resulting median tree will increase within this margin of low indel thresholds.

Both of these trends are noticed when mismatch cost is 2, as shown in Figure 3.12. In fact, the trends are more pronounced, although the period where the tree length worsens extends out to 0.01 when mismatch cost is 2.

Regardless of mismatch cost, MUSCLE returns the lowest distance to true median score, which means the tree it returns is the most accurate of the heuristic methods tested. When mismatch cost is 2, DCA(MSA) is able to closely approximate MUSCLE, but DCA(MSA) is least accurate when mismatch cost is 1.

Parameter	Value
Sequence Length	5000
Indel Threshold	0.000 to 0.030 (step size: 0.001)
Mean Substitution Rate	0.1
Indel Length Distribution	short
Edge Lengths	Equal (1:1:1)
Mismatch Cost	1 or 2

Table 3.8: Parameters maintained to observe effect of varying indel threshold.

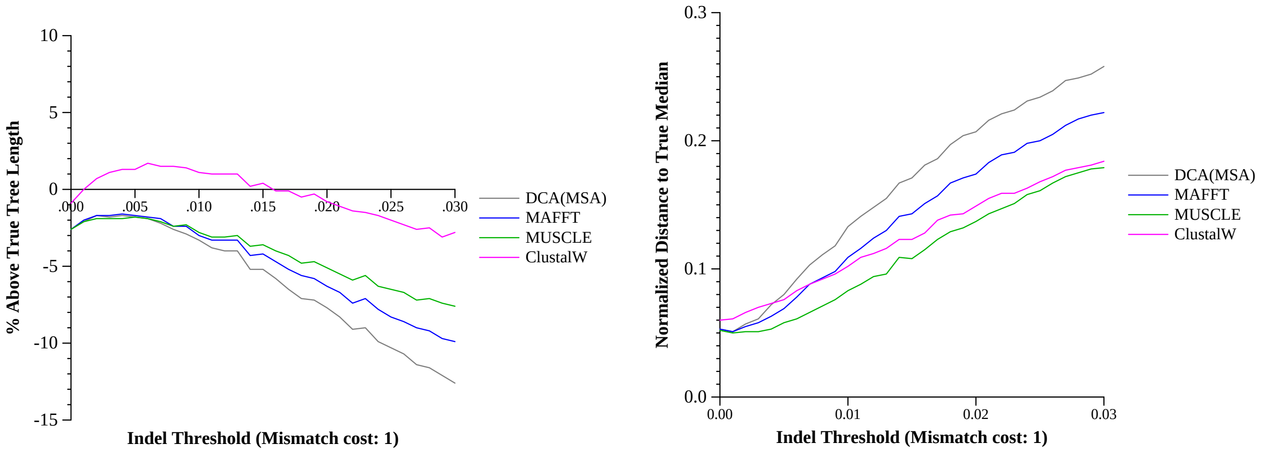


Figure 3.11: Effect of indel threshold on heuristic median estimation when mismatch cost is 1 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

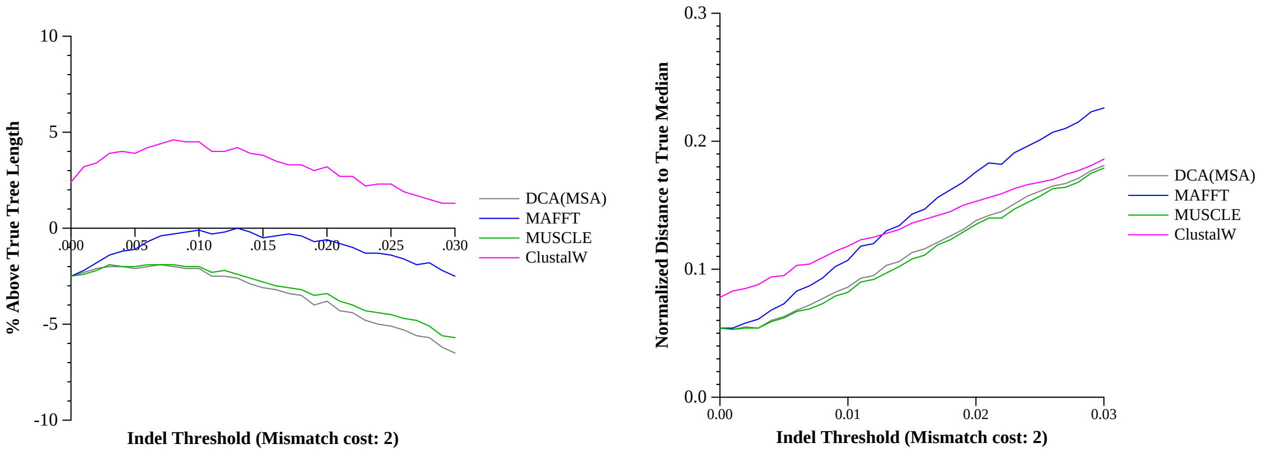


Figure 3.12: Effect of indel threshold on heuristic median estimation when mismatch cost is 2 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

Substitution Rate

Figure 3.13 shows the effect of various substitution rates for mismatch cost 1, with parameters as specified in Table 3.9. Results for mismatch cost set to 2 are shown in Figure 3.14 All of the methods performed better compared to the true tree as the substitution rate increased. We expected this to be the case. Substitution rates greater than 0.3 are much higher than what would be seen biologically; when these large substitution rates are used, the true tree length will not be optimized. Since the heuristic methods are attempting to optimize the median of the three sequences, we would expect to see them perform better (lower % above true tree) than what the actual evolutionary history dictates. Therefore, it would be beneficial to select a low substitution rate; the optimal tree returned by the heuristic methods would be close to the true tree in this case.

The same trends observed when varying indel threshold are apparent when the mismatch cost is increased to 2. DCA(MSA) is much more accurate when under the higher mismatch cost conditions, while MUSCLE, MAFFT, and ClustalW are consistent for both mismatch cost values. Also, the methods better approximate the true tree when the mismatch cost is 2.

Under first glance, it may seem bizarre that % above true tree length increases under the ClustalW heuristic, but decreases when the other methods are used. Compare Figure 3.14 (5000 length sequences with mismatch cost 2) to Figure 3.7 (200 length sequences with mismatch cost 2); for sequences of length 200, we were able to compare all of the methods to the exact method. Notice that the true tree length gradually increases as the substitution rate is increased in Figure 3.7, while MAFFT, MUSCLE, and DCA(MSA) stay constant. In Figure 3.14, we are comparing the methods to the true tree length, not the exact tree length. If Figure 3.7 is rotated such that the true tree length is horizontal, the graph will look like Figure 3.14. ClustalW will be above the true tree length line (x-axis), whereas the other methods will slope downwards. Therefore, ClustalW increases in Figure 3.14 because the tree length score degrades at a more rapid pace than the rate at which the true tree length degrades.

Parameter	Value
Sequence Length	5000
Indel Threshold	0.01
Substitution Rate	0.0 to 0.30 (step size: 0.01)
Indel Length Distribution	short
Edge Lengths	Equal (1:1:1)
Mismatch Cost	1 or 2

Table 3.9: Parameters maintained to observe effect of varying substitution rate.

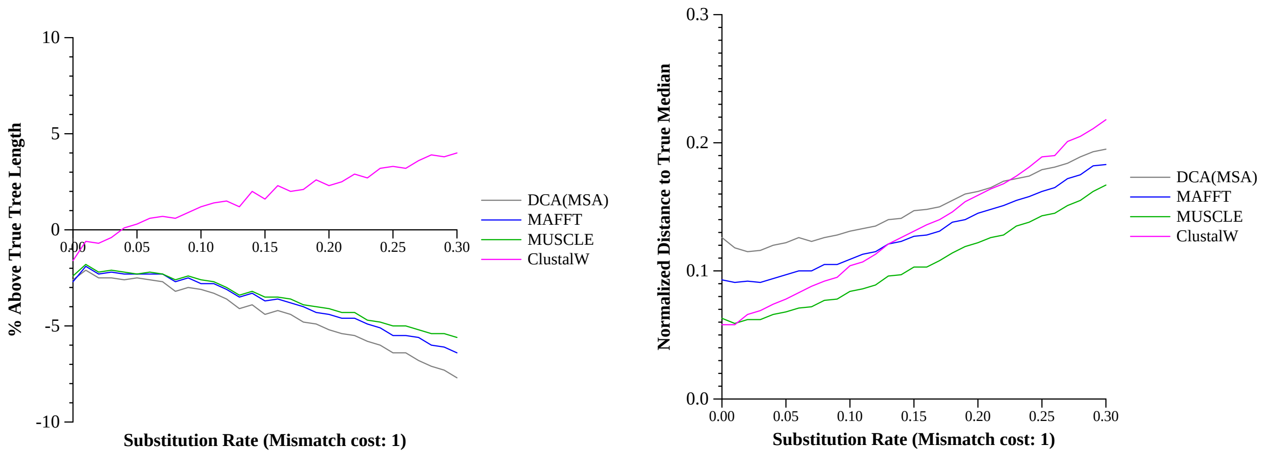


Figure 3.13: Effect of substitution rate on heuristic median estimation when mismatch cost is 1 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

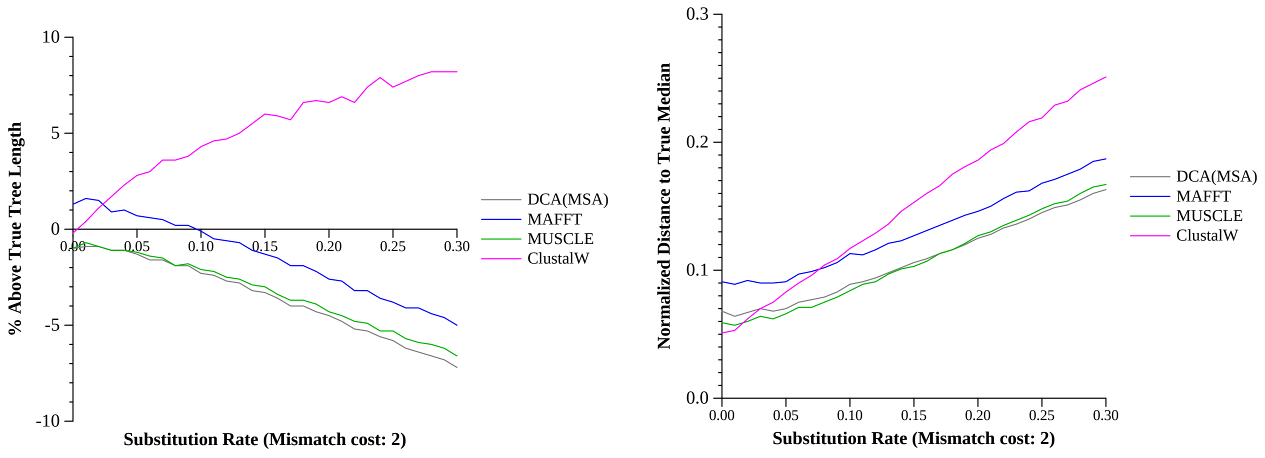


Figure 3.14: Effect of substitution rate on heuristic median estimation when mismatch cost is 2 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

Indel Length Distribution

We also varied the indel length distribution in a similar fashion as was done for sequences of 200 nucleotides in length. Table 3.10 lists the parameters used in this study. The indel length distribution parameter is inversely proportional with the size of an indel; a small indel length distribution parameter value will result in a large indel event. Figures 3.15 and 3.16 are the results when mismatch cost is 1 or 2, respectively.

MAFFT, MUSCLE, and DCA(MSA) are unaffected when the indel length distribution parameter is increased from 0.2 to 0.5. In addition, for both mismatch costs, these three methods return trees with tree lengths that are lower than the true tree length. ClustalW is again an exception since it returns a tree greater than the true tree length.

When the mismatch cost is 2, the final tree length returned by MAFFT is closely inline with the true tree length regardless of indel distribution parameter. However, although the distance to true median is approximately 0.13, other methods are able to give more trees closer to the true median.

DCA(MSA) also has a lower distance to true median when mismatch cost is 2, while MUSCLE consistently returns trees with lower distance to true median scores.

All methods, with the exception of ClustalW, return more optimal trees at low indel length distribution parameters; recall that small indel length distribution parameters result in long gaps. This again is due to the fact that the heuristic methods are attempting to deliver an optimal median tree rather than the true tree. The reconstruction parameters (mismatch cost, gap initialization cost, and gap extension cost) play a role in determining the optimal alignment. Since we set a gap initialization cost of 5 and a gap extension cost of 1, we would expect the methods to favor longer indels over short indels. Therefore, the methods deliver medians with lower tree lengths when a long indel distribution is used.

Parameter	Value
Sequence Length	5000
Indel Threshold	0.01
Mean Substitution Rate	0.1
Indel Length Distribution	Geometric with parameter: 0.10 to 0.50 (step size: 0.01)
Edge Lengths	Equal (1:1:1)
Mismatch Cost	1 or 2

Table 3.10: Parameters maintained to observe effect of varying indel distribution.

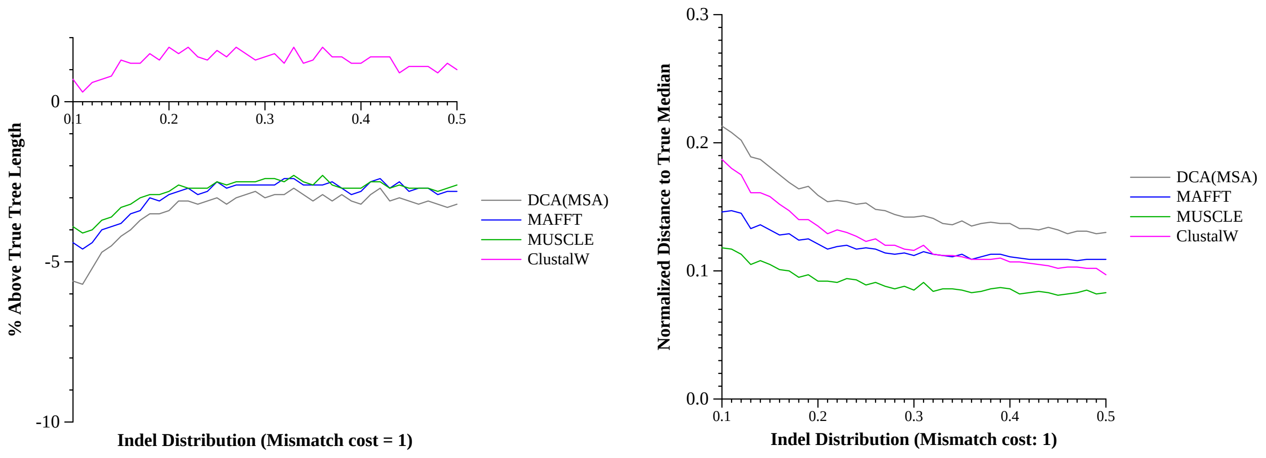


Figure 3.15: Effect of indel distribution on heuristic median estimation when mismatch cost is 1 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

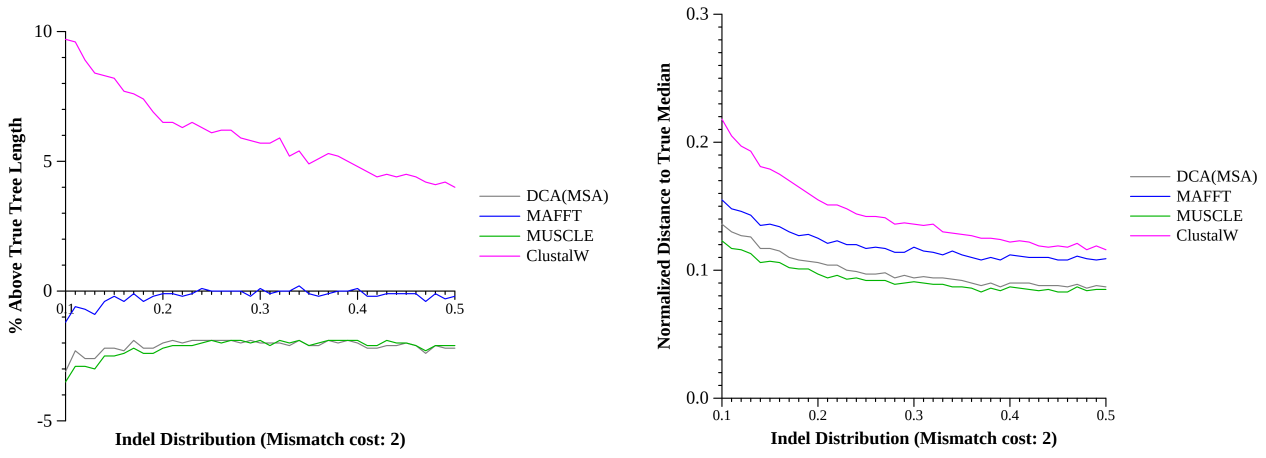


Figure 3.16: Effect of indel distribution on heuristic median estimation when mismatch cost is 2 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

Edge Lengths

Recall that Rose generates three sequences from an ancestral sequence. This is a median tree, where the three sequences are connected directly to the ancestral sequence. We wanted to see if increasing the proportion along the edges had an effect on the heuristic methods. We controlled the parameters specified in Table 3.11.

To adjust the edge lengths, we adjusted the *edge length multiplier*. This adjusts the ratio of the three edges. For example, if the edge length multiplier is 1.5, then the length of three edges will be: 1.0, 1.5 ($1.0 * 1.5^1$), and 2.25 ($1.0 * 1.5^2$). The edge length multiplier times the substitution rate defines the number of substitutions along an edge.

The results are shown in Figure 3.17 for a mismatch cost of 1, and in Figure 3.18 for a mismatch cost of 2.

It is clear that the heuristic methods perform *better* compared to the true tree when edge lengths are uneven. This was expected. Varying the edge lengths means that the dataset generated by Rose will not be equally evolutionarily related. Therefore, the length of the true tree will not be as low compared to the true tree length when edges are equal. On the other hand, the alignment methods are still trying to find the most optimal tree. Therefore, the heuristic method performance will improve compared to the true tree length.

For mismatch cost of 1 and with edge length multipliers greater than 0.2, the trees returned by ClustalW have similar tree lengths that match closely with the true tree length. In addition, ClustalW has good accuracy with a low distance to true median score in this case.

Once again, DCA(MSA) has poor accuracy when mismatch cost is 1, but accuracy closely approximates MUSCLE when mismatch cost is 2.

Parameter	Value
Sequence Length	5000
Indel Threshold	0.01
Substitution Rate	0.1
Indel Length Distribution	short
Edge Length Multiplier	1.0 to 3.0 (step size: 0.1)
Mismatch Cost	1 or 2

Table 3.11: Parameters maintained to observe effect of varying edge lengths.

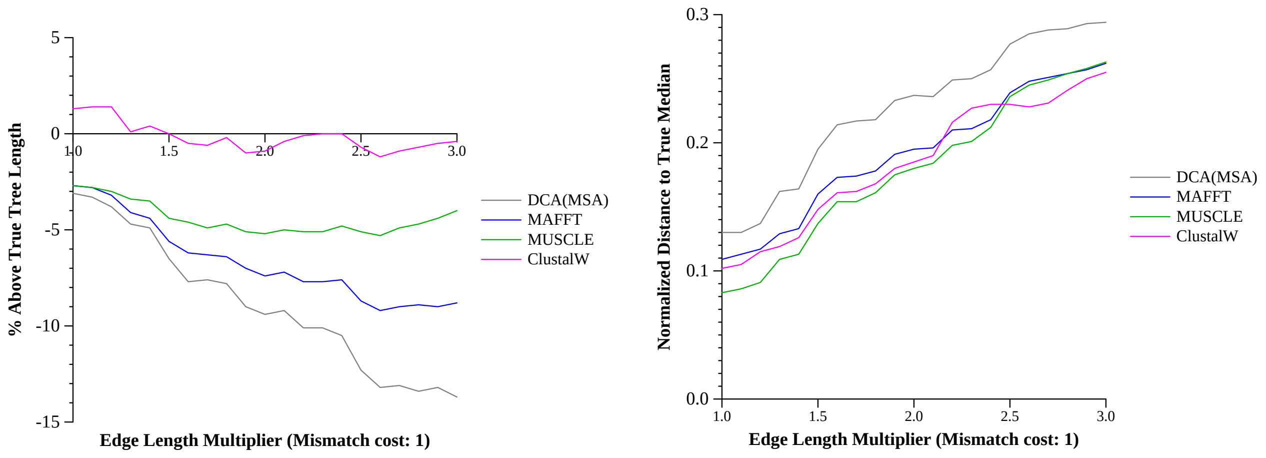


Figure 3.17: Effect of edge lengths on heuristic median estimation when mismatch cost is 1 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

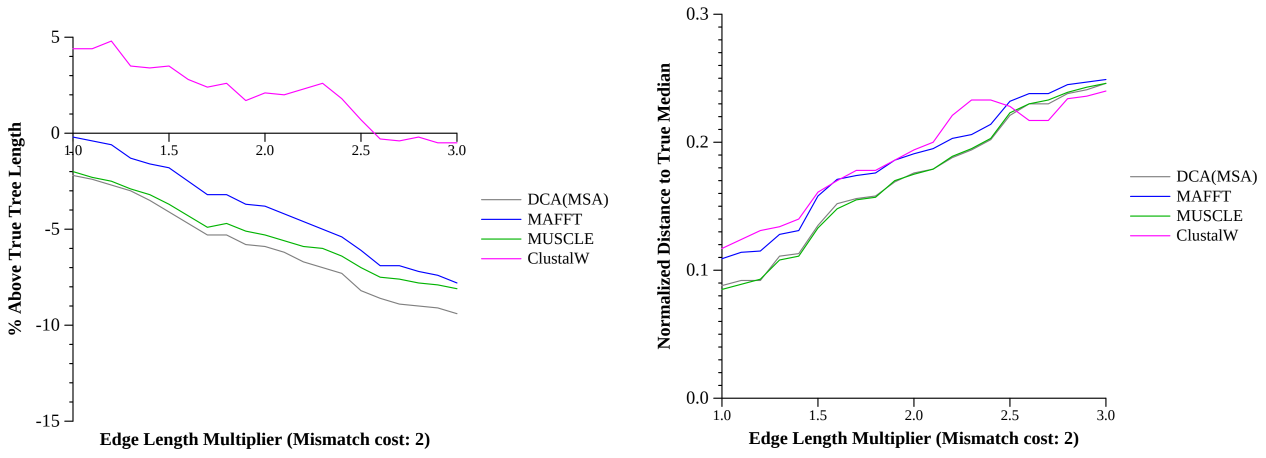


Figure 3.18: Effect of edge lengths on heuristic median estimation when mismatch cost is 2 for sequences of length 5000. Tree length of the estimated median for each method is compared to the optimal tree length as determined by an exact method for affine gap penalties. In addition, distance to true median scores are given (which are a measure of accuracy).

3.12 Discussion

This study focused on measuring heuristic MSA method performance to achieve close to optimal median trees. Although this is the preferred method in computer science, it is not the goal a systematist would strive for. A systematist wants to reconstruct the evolutionary history of a set of sequences rather than find the optimal tree for a set of sequences. In other words, a systematist would want the method that returns a median that closely approximates the true tree rather than the exact tree. In order to achieve this goal, we will need to optimize the parameters such that the exact tree is similar to the true tree.

We varied several variables and measured heuristic performance on datasets of three sequences with 200 or 5000 sites in length.

With an initial sequence of length 200, we varied indel threshold, mean substitution rate, and indel length distribution for mismatch costs of 1 or 2. Percent above exact tree length stays approximately constant as the indel threshold increases for MAFFT, MUSCLE, DCA(MSA), and MSA. However, for all of the methods, distance to true median increases with increasing indel threshold. Varying the substitution rate produces similar results. Therefore, if we wanted to generate true trees similar to an optimal tree, we will need to pick very small substitution rates (i.e. 0.01) and a very small indel threshold (i.e. 0.001).

Also, it appears that higher indel length distribution parameters cause the true tree to better approximate the exact tree, although there is a very large difference in tree length between the true tree and exact tree in this case. Recall that higher indel length distribution parameters results in smaller average indels. This observation is logical since it is easier to determine homology when smaller indel events have occured.

Results are similar when the initial ancestral sequence is 5000 sites in length. Due to the fact that the exact median method will not compute solutions for sequences greater than 200 sequences in length, we compared the heuristic tree length performance to the true tree. ClustalW is unable to return approximately optimal trees, so we conjecture that ClustalW will not perform well if we optimize the parameters such that a true tree is similar to an exact tree. With current parameters, however, it is sometimes able to return a tree with similar tree length to the true tree with good accuracy. On the other hand, MAFFT, MUSCLE, and DCA(MSA) are more consistent in the results that are returned.

A mismatch cost of 2 allows the true tree and exact tree to be closer related in terms of tree length. The true tree and exact tree are not as similar when mismatch cost is 1. Therefore, the mismatch cost of 2 seems to give better results.

Finally, DCA(MSA) experienced poor accuracy (high distance to true median) for mismatch cost 1, but the accuracy was excellent and closely approximated MUSCLE when mismatch cost is 2. This occurred regardless of whether the initial sequence length was 200 sites or 5000 sites. Therefore, to obtain a close correlation between true tree and exact tree, we would want to analyze sequences of length 5000.

In addition, for sequences of length 200, the curves defining the relationship among varying the parameters are very jagged. This outcome was unexpected since this was not the case for sequences of length 5000. It is not known why this occurs.

We conclude that MUSCLE, MAFFT, and DCA(MSA) produce acceptable median estimations with respect to minimizing tree length. ClustalW and T-Coffee seemed to give poorer results under these conditions, although these methods closely approximated the true tree length; although the tree length may be similar to the true tree length, it does not mean that the tree produced by these methods is similar to the true tree.

The running times for most of these methods are good for sequences with a length of 5000 nucleotides. Average processing time is around 30 seconds for 5000 sites. MSA and T-Coffee were exceptions to this observation. T-Coffee, which supposedly performs very well for sets with many sequences, performed very poorly. This can be attributed the long period of time it takes for the program to initialize. MSA was the best method at minimizing tree length, but it was prohibitively slow.

With the exception of T-Coffee and MSA, all of the heuristic methods are fast enough for use in the iterated heuristic.

3.13 Future Work

We plan on incorporating one of the MSA methods, such as MUSCLE, MAFFT, or DCA(MSA) into existing software for generalized tree alignment and tree alignment. It can then be determined if improvements can be obtained with respect to minimizing tree length in reasonable time periods.

In addition, we will create our own heuristic median solver. This median solver can then be compared to the methods analyzed in this study in terms of performance and accuracy.

Also, it will be necessary to compare Generalized Tree Alignment algorithms as well as the standard two-phase phylogeny estimation approach. This will be necessary to determine if standard two-phase approach is sufficient to obtain good tree reconstructions.

Finally, it will be necessary to determine how best to optimize the heuristic methods to approximate the true evolutionary history. This will likely require we adjust parameters such that the

Chapter 4

Conclusions

Although exact methods are useful for obtaining exact solutions, they cannot be applied by biologists on real data collected in the field. They are simply too computationally slow and take up too much space to execute. As a result, heuristic methods are used to get an approximate solution in a fraction of the time and the space. We wanted to improve the accuracy and performance of a maximum likelihood heuristic method, and also apply multiple sequence alignment heuristic methods to optimize median alignment.

To improve maximum likelihood heuristics, we thought that “fast evolving sites” caused the heuristic methods to slow down and lose focus as they try to optimize these sites. We used maximum parsimony to locate these fast evolving sites, and then truncated the sequences to exclude these fast evolving sites. However, the heuristic methods did not run any faster nor perform any better using the truncated datasets when compared to the regular datasets.

We also have an ultimate goal of optimizing generalized tree alignment to handle datasets with hundreds of sequences and thousands of sites. In order for this to be achieved, it was necessary to optimize the scoring component of generalized tree alignment, which is tree alignment. The iterated heuristic can be used to score a tree, such that for each internal node, a median alignment is generated. We conjectured that if we can improve median alignment running time and tree length, we will have improved the iterated heuristic, which will take us a step closer to optimizing generalized tree alignment.

Our study showed that DCA(MSA), MAFFT, and MUSCLE will be good candidates to incorporate into existing tree alignment and generalized tree alignment software. We will then be able to compare the performance of the improved iterated heuristic to the original programs.

Bibliography

[1] M. Chase, D. Soltis, R. Olmstead, D. Morgan, D. Les, B. Mishler, M. Duvall, R. Price, H. Hillis, Y.-L. Qiu, A. Cron, J Retig, E. Conti, J. Palmer, J. Manhart, K. Systma, H. Michaels, W. Kress, K. Karol, W. Clark, M. Hedrn, B. Gaut, R. Jansen, K.-J. Kim, C. Wimpe, J. Smith, G. Furnier, S. Strauss, Q.-Y. Xiang, G. Plunkett, P. Soltis, S. Swenson, S. Williams, P. Gadek, C. Quinn, L. Eguiarte, E. Golenberg, G. Jr, S. Graham, S. Barrett, S. Dayanandan, and V. Albert. Phylogenetics of Seed plants: An Analysis of Nucleotide Sequences from the Plastid Gene *rbcL*. *Annals of the Missouri Botanical Garden*, 80:528–580, 1993. 2.2.1

[2] R. C. Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5(113), 2004. 3.10

[3] R. C. Edgar. MUSCLE: a multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32(5):1792–1797, 2004. 3.10, 3.10.2

[4] J. Felsenstein. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology*, 27(4):401–410, 1978. 2.1

[5] J. Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *J. Mol. Evol.*, 17:368–376, 1981. 2.1

[6] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol*, 162:705–708, 1982. 3.2.2

[7] O. Gotoh. Alignment of three biological sequences with an efficient traceback procedure. *J. Theor. Biol.*, 121:327–337, 1986. 3.6.2

[8] D. S. Hirschberg. A linear-space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18:341–343, 1975. 3.2.2

[9] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the the construction of phylogenetic trees. *J. Mol. Evol.*, 20:175–186, 1984. 3.10.2

[10] X. Q. Huang. Alignment of three sequences in quadratic space. *Applied Computing Review*, 1(2):7–11, 1993. 3.6.2

[11] O. Poch J. D. Thompsan, F. Plewniak. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15:87–88, 1999. 3.10.2

[12] Daniel A. Janies and Ward C. Wheeler. Poy version 3.0, documentation and command summary, phylogeny reconstruction via direct optimization of dna and other data. Technical report, American Museum of Natural History, Nov 2002. 3.4.1

[13] T. H. Jukes and D. R. Cantor. Evolution of protein molecules. In H. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, 1969. 2.1

[14] K. Katoh, K. Kuma, H. Toh, and T. Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.*, 33(2):511–518, 2005. 3.10

[15] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, 30(14):3059–3066, 2002. 3.10, 3.10.2

[16] M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217:624–626, 1968. 2.2

[17] B. Knudsen. Optimal multiple parsimony alignment with affine gap cost using a phylogenetic tree. In G. Benson and R. Page, editors, *WABI 2003, LNBI 2812*, pages 433–446. Springer-Verlag, Berlin, 2003. 3.5, 3.6.2

[18] G. Lancia and R. Ravi. GESTALT: Genomic Steiner alignments. *Lecture Notes in Computer Science*, 1645:101–114, 1999. 3.4.1, 3.6.2

[19] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Natl. Acad. Sci. USA*, 86:4412–4415, 1989. 3.10, 3.10.1

[20] M.J.Brauer, M.T.Holder, L.A.Dries, D.J.Zwickl, P.O.Lewis, and D.M.Hillis. Genetic Algorithms and Parallel Processing in Maximum-Likelihood Inference. *Molecular Biology and Evolution*, 19(10):1717–1726, 2002. 2.2.1

[21] M. Murata, J. S. Richardson, and J. L Sussman. Simultaneous comparison of three protein sequences. *PNAS*, 82(10):3073–3077, 1985. 3.6.2

[22] E. W. Myers. An $O(nd)$ difference algorithm and its variations. *Algorithmica*, 1:251–266, 1986. 3.6.2

[23] E. W. Myers and W. Miller. Optimal alignments in linear space. *Bioinformatics*, 4(1):11–17, 1988. 3.2.2

[24] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970. 3.2.2

[25] C. Notredame. Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, 2001. 3.1

[26] C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, 302:205–217, 2000. 3.10, 3.10.2

[27] D. Pisani. The Systematics Association 5th Biennial Meeting, August 2005. 2.2

[28] D. R. Powell, L. Allison, and T. I. Dix. Fast, optimal alignment of three sequences using linear gap costs. *J. Theor. Biol.*, 207:325–336, 2000. 3.6.2

[29] C. Saccone, C. Lanave, G. Pesole, and G. Preparata. Influence of base composition on quantitative estimates of gene evolution. *Methods Enzymol.*, 183:570–583, 1990. 2.1

[30] D. Sankoff. Matching sequences under deletion/insertion constraints. *PNAS*, 69:4–6, 1972. 3.2.2

[31] D. Sankoff. Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, 28(1):35 – 42, January 1975. 3.5

[32] D. Sankoff and R. J. Cedergren. Simultaneous comparison of three or more sequences related by a tree. In D. Sankoff and J. B. Kruskall, editors, *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison*, pages 253–263. Addison Wesley, New York, 1993. 3.4

[33] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793, 1974. 3.2.2

[34] D. E. Soltis, P. S. Soltis, D. L. Nickrent, L. A. Johnson, W. J. Hahn, S. B. Hoot, J. A. Sweere, R. K. Kuzoff, K. A. Kron, M. W. Chase, S. M. Swenson, E. A. Zimmer, S. M. Chaw, L. J. Gillespie, W. J. Kress, and K. J. Sytsma. Angiosperm Phylogeny Inferred from 18S ribosomal DNA sequences. *Annals of the Missouri Botanical Garden*, 84:1–49, 1997. 2.2.1

[35] A. Stamatakis. RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21:456–463, 2005. 2.3.4

[36] M. Steel. The maximum likelihood point for a phylogenetic tree is not unique. *Society of Systematic Biologists*, 43:560–564, 1994. 2.1

[37] J. Stoye. Multiple sequence alignment with the divide-and-conquer method. *Gene*, 211:GC45–GC56, 1998. 3.10.1

[38] J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163, 1998. 3.9

[39] J. Stoye, V. Moulton, and A. W. Dress. Dca: an efficient implementation of the divide-and-conquer approach to simulataneous multiple sequence alignment. *Comput. Appl. Biosci.*, 13(6):625–626, 1997. 3.10, 3.10.1

[40] J. Thompson, D. Higgins, and T. Gibson. 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(22):4673–4680, 1994. 3.10

[41] E. Ukkonen. On approximate string matching. *Found. Comput. Theory*, 158:487–495, 1983. 3.6.2

[42] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J. Comput. Biol.*, 1(4):337–348, 1994. 3.4, 3.5, 3.6.1

[43] H. T. Wareham. A simplified proof of the np- and max snp-hardness of multiple sequence tree alignment. *J. Comput. Biol.*, 2(4):509–514, 1995. 3.4

[44] M. S. Waterman, T. F. Smith, and W. A. Beyer. Some biological sequence metrics. *Advan. Math.*, 20:367–387, 1976. 3.2.2

[45] Scott Hazelhurst Zsuzsanna. A comparative study of biological distances for est clustering. Technical Report TR-Wits-CS-2003-3, University of the Witwatersrand, May 2003. 2.2.4

Index