A Minimum Spanning Tree Framework for Inferring Phylogenies

Daniel Giannico Adkins

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-157
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-157.html

December 15, 2010

A Minimum Spanning Tree Framework for Inferring Phylogenies

by

Daniel Giannico Adkins

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Satish B. Rao, Chair
Professor Umesh V. Vazirani
Professor Brent D. Mishler

Fall 2010

A Minimum Spanning Tree Framework for Inferring Phylogenies

Abstract

A Minimum Spanning Tree Framework for Inferring Phylogenies

by

Daniel Giannico Adkins

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Satish B. Rao, Chair

This dissertation discusses how to write efficient, deterministic programs to infer phylogenetic trees. These programs are based on a theoretically optimal polynomial time algorithm. The programs are practical and efficient.

The problem is to infer phylogenetic trees from sequence data at the leaves. Likelihood approaches attempt to solve an NP-complete optimization problem, and are computationally intense.

Our approach is to apply efficient, polynomial time algorithms which are provably accurate to within a constant factor of optimal.

We present a program which is practical and computationally efficient.

Contents

Acknowledgments

I gratefully acknowledge financial support from the Fannie and John Hertz Foundation.

Chapter 1

Introduction

A *phylogeny* is the evolutionary development and history of a group of organisms. A *phylogenetic tree* is a formal representation of the phylogeny, or evolutionary history of a set of species. Since Charles Darwin published "The Origin of the Species," phylogeny has primarily been a taxonomy problem of interest to biologists.

With the relatively recent discovery of DNA and the advent of automatic sequencing of entire genomes, phylogeny has been recast as a mathematical or computational problem. Biologists have proposed mathematical models of evolution which posit that evolution is, by and large, a random process acting on DNA over time. Some models posit that mutations occur independently and randomly to individual DNA characters, provided no external factors such as survival are at stake. Other models look at larger changes, such as the rearrangement of entire genes, by observing the movement of conserved segments of the genome.

Whatever the particular model of evolution, the general idea is that the expected number of evolutionary events that occur in a genome is a function of time. Over time, *speciation events* occur where a single species splits into two distinct species, never to recombine. Of course, species do sometimes reintegrate, but we choose to ignore that in our models. The result of this process is an evolutionary tree with a root and a set of distinct species or organisms at the leaves.

The combinatorial problem is, given a set of data for the species at the leaves and assuming some reasonable model of evolution, to reconstruct the phylogenetic tree which led to those species. We are interested mainly in the structure of the tree, since that embodies the evolutionary relationships between the species. We are not as interested in the internal sequences or any branch lengths in the tree, although some algorithms learn this data and make use of it.

The problem of phylogenetic reconstruction is of interest to biologists, naturally, but also to mathematicians, statisticians, and computer scientists. Biologists are using algorithms to resolve ambiguities in the tree of life, and also to advance theories about the relation between mice and men, for example. To the other fields, the problem is related to problems of percolation theory and information flow over a noisy channel. They view evolution as transmitting information over time, which is a natural idea. Over time, information is lost to random mutations, but also information is duplicated and transmitted through replication. Of course, making copies is not the explicit goal.

There is an interplay between random mutations, which are ways of trying new configurations, and preserving the configurations which are proven to work.

In the inference of phylogenies, two central issues are the inference of quartet topologies and the use of ancestral sequences. In this dissertation, we implement and test a novel framework which proceeds by alternatively inferring quartet topologies and learning ancestral sequences. This framework provides a context in which to study the performance of methods for inferring quartet topologies as well as a motivation for optimizing their performance.

We study the performance of the framework with a variety of ancestral sequence methods. We show the framework is effective for both nucleotide data and genomic data.

In Chapter 2, we present the basic algorithm and compare the performance of our implementation with other programs. We extend the algorithm of Mihaescu et al. [MHR08] to incorporate more sophisticated ancestral learning techniques which require us to estimate edge lengths. We describe the performance of our framework using various combinations of ancestral sequence learning methods described therein and quartet methods described subsequently in this dissertation.

As an example of the difficulties of learning quartets, the sequence at an internal node with a distant child and a closely related child should be closer to the closely related child. Here the accuracy of quartet topology and distances comes into play, motivating further study of the performance of quartet methods for distance estimation.

In order to understand and improve our relative performance in comparison with previous methods we study a variety of methods to infer quartet topologies. We first study the relative performance of the standard distance-based methods with likelihood methods. We observe that likelihood performs better for some quartets. We provide insight into this phenomenon by showing that the estimates from likelihood have smaller confidence intervals than those provided by distance methods. Our methods for estimating the confidence intervals are presented in Chapter 3. Our implementation illustrates that these methods can be incorporated in the future to actually provide confidence intervals for edge lengths in programs (which is not done currently in any program to our knowledge.)

In Chapter 4, we give a theoretical proof that the framework is effective for genomic data. This follows a previous proof [MHR08] showing that essentially the same framework was theoretically effective for nucleotide data.

Chapter 2

Molecular Phylogeny

In this chapter, we describe our algorithmic framework for molecular phylogeny. In molecular phylogeny, we are concerned with sequences of characters, typically DNA. Each character evolves independently according to a Markov process.

2.1 Background

We assume that evolution proceeds independently on each character in the tree according to a Markov process. The parameters of the Markov process may change from edge to edge in the tree, but not from site to site for any given edge. For molecular data (i.e. A, C, G, T), the most general model is the general time reversible (GTR) model [Tav96], which only assumes that the transition probability matrix has non-zero determinant. More specific models include the Kimura 2 or 3 parameter models [Kim80, Kim81] (Figure 2.1), which group the transition and transversion probabilities. The simplest model is the Jukes-Cantor model [JC69], with one parameter for the transition probability among any characters.

The Jukes-Cantor process can be described as follows. With probability θ, no mutation occurs; the character is copied. With probability $1 - \theta$, a random mutation occurs. All characters are equally likely to occur, including the original character. The θ formulation is useful because we can multiply parameters for edges to get a composite parameter. By taking a logarithm,

$$d(e) = -\log \theta(e), \tag{2.1}$$

we get an additive metric.

With a formal model defined and with long enough sequences, the problem is well-formulated. There is one correct tree that we can say with high probability generated the sequences we observed. We can reconstruct that tree and its internal branch lengths accurately, if only we had enough trials. Of course, the amount of sequence data that is necessary depends on the method used. Daskalakis et al. [DMR06] shows that with certain reasonable restrictions on the branch

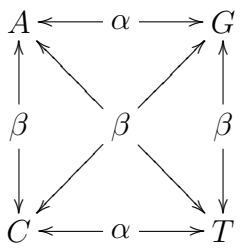


Figure 2.1: The Kimura two-parameter model of nucleotide substitution.

lengths, the amount of data needed to reconstruct a tree with n leaves is $O(\log n)$. This bound is tight.

2.1.1 Distance Estimation

Given two sequences, we can estimate the distance between them assuming the Jukes-Cantor model of evolution. First, we compute the normalized agreement between the two sequences, a and b, of length k

$$A(a,b) = \frac{1}{k}\sum_{i=1}^{k}[a_i = b_i]. \tag{2.2}$$

From that, we can estimate the parameter θ for the edge or path between a and b as,

$$\widehat{\theta}(a,b) = \frac{4}{3}\big(A(a,b) - \frac{1}{4}\big). \tag{2.3}$$

If the agreement is less than $1/4$, this value may be negative. In those cases, we consider it to be 0. Also, for convenience, we also use the equivalent additive distance estimate

$$\widehat{d}(a,b) = -\log\widehat{\theta}(a,b). \tag{2.4}$$

2.1.2 Four Point Method

The four point method is an unbiased estimator for the middle edge of a quartet of taxa, given the sequences at the leaves of the quartet and a distance function between sequences. Given a quartet $(ab|cd)$, the four point method estimates the length of the middle edge (x,y) as

$$\widehat{d}(x,y) = \frac{1}{4}\big(\widehat{d}(a,c) + \widehat{d}(a,d) + \widehat{d}(b,c) + \widehat{d}(b,d) - 2\widehat{d}(a,b) - 2\widehat{d}(c,d)\big). \tag{2.5}$$

Similarly, we can estimate the other edge lengths as

$$\widehat{d}(a,x) = \frac{1}{4}\big(2\widehat{d}(a,b) + \widehat{d}(a,c) + \widehat{d}(a,d) - \widehat{d}(b,c) - \widehat{d}(b,d)\big) \tag{2.6}$$

$$\widehat{d}(b,x) = \frac{1}{4}\big(2\widehat{d}(a,b) + \widehat{d}(b,c) + \widehat{d}(b,d) - \widehat{d}(a,c) - \widehat{d}(a,d)\big) \tag{2.7}$$

$$\widehat{d}(c,y) = \frac{1}{4}\big(2\widehat{d}(c,d) + \widehat{d}(a,c) + \widehat{d}(b,c) - \widehat{d}(a,d) - \widehat{d}(b,d)\big) \tag{2.8}$$

$$\widehat{d}(d,y) = \frac{1}{4}\big(2\widehat{d}(c,d) + \widehat{d}(a,d) + \widehat{d}(b,d) - \widehat{d}(a,c) - \widehat{d}(b,c)\big) \tag{2.9}$$

although these estimators are biased unlike the middle edge estimator.

We use the four point method primarily to determine the orientation of a quartet of nodes. The correct orientation is the one with the largest middle edge by this estimator. In fact, if we orient the quartet incorrectly, the middle edge estimate will have a negative expected value.

As a simplification, it is not necessary to do the full calculation to find the orientation of four nodes with the largest middle edge. It suffices to pick the smallest sum of side edges:

$$(ab|cd) \rightsquigarrow \widehat{d}(a,b) + \widehat{d}(c,d); \tag{2.10}$$

$$(ac|bd) \rightsquigarrow \widehat{d}(a,c) + \widehat{d}(b,d); \tag{2.11}$$

$$(ad|bc) \rightsquigarrow \widehat{d}(a,d) + \widehat{d}(b,d). \tag{2.12}$$

2.2 Related Work

For an overview of the field of phylogenetics, see the books by Semple and Steel [SS03] and Felsenstein [Fel04].

Distance-based methods define a metric on the tree, usually represented by a pairwise distance matrix. The original four-point method is due to Buneman [Bun71].

Maximum likelihood, although NP-hard and difficult to approximate, is used in heuristic programs to great effect, e.g. RAxML [Sta04, Sta06] and GARLI [Zwi06]. There are also Bayesian program using Markov chain Monte Carlo techniques, e.g. MrBayes [HR01, RH03, ADHR04].

There has been more theoretical work on distance-based algorithms and quartet methods in particular. Erdös et al. [ESSW99a, ESSW99b] proposed the short quartets method which clears the hurdle of information loss in long quartets by only piecing together quartets of bounded diameter. Their method reconstructs trees with polynomial sequence lengths as opposed to the exponential requirement of neighbor joining.

Daskalakis et al. [DMR06] proved optimal phylogenetic reconstruction results. Namely, under certain conditions on the edge lengths, you can reconstruct phylogenetic trees with $O(\log n)$ sequence data at the leaves. This result is optimal to a constant factor. Mihaescu et al. [MHR08] later gave a more efficient, polynomial time algorithm which achieved the same optimal bounds.

The key idea of our algorithm is to not just use the distances between leaves, but to also learn ancestral genomes. For molecular data, the ancestral approach has been shown to be more accurate than distance-based methods in theory and practice. In practice, likelihood methods (e.g. RAxML [Sta04, Sta06] and PhyML [GG03]) and parsimony methods (e.g. Fitch-Hartigan [Fit71, Har73]) which learn internal sequences significantly outperform distance methods. In theory, the recursive majority algorithm for learning ancestral sequences was introduced and analyzed by Mossel [Mos04]. The recursive procedure is also a key step in the logarithmic-sized phylogeny reconstruction algorithms of Daskalakis, Mossel, and Roch [DMR06] and Mihaescu, Hill, and Rao [MHR08].

2.3 Our Approach

Our algorithm maintains a forest of correct subtrees of T, infers characters on interior nodes of each tree, and uses that information to decide how to join these subtrees. The main tool for joining subtrees is the now classical four-point method, which can properly sort out the topology of a quartet given sufficiently accurate distances among the four positions. If we can estimate distances between internal nodes with sufficient accuracy, our algorithm will reconstruct the correct tree.

The algorithm begins by forming a forest consisting of each taxa as an isolated tree. At each step, it combines two subtrees into a larger tree. A tree T_f in the forest corresponds to a set of taxa and an induced subtree in the true evolutionary tree T. An edge in T_f may correspond to a path in T. The algorithm maintains the property that the set of trees in the forest corresponds to a disjoint induced subgraph in T.

The algorithm joins trees by locating the shortest path (in evolutionary distance) between two subtrees in the forest. Adding an edge corresponds to joining two subtrees, T_1 and T_2. That edge corresponds to a path in the evolutionary tree. By choosing the shortest such path and corresponding edge, the algorithm maintains the disjointness condition of the forest. By choosing to attach the edge to the appropriate place in T_1 and T_2, the algorithm maintains the correct topologies on the subtrees of the forest.

In the general case, the algorithm needs to find the correct pair of trees and edges, $e_1 = (a, b)$ and $e_2 = (c, d)$, in the forest to serve as the endpoints of the new edge. We assume that it tries all pairs of edges and estimates the distance of the connecting edge as follows. It removes edges $e_1 = (a, b)$ and $e_2 = (c, d)$, breaking their respective trees into two subtrees. It then learns internal sequences at a, b, c, and d from the separate subtrees. These internal sequences can then by used in the four point method to estimate the length of the edge connecting the two edges assuming that this is indeed the correct pair of edges. This follows since the sequences at a, b, c, and d are independent random variables; they are learned from entirely separate subtrees in the true evolutionary subtree. If we are at an incorrect pair of edges, (a, b) and (c, d), one or more of the subtrees will contain a portion of the path in the tree connecting the edges. This creates dependencies between the learning algorithm in the supposedly separate subtrees and makes the distance computation invalid. The algorithm contains a check for this case (essentially the four point method at adjacent edges).

Now, given that the middle edge distance output by the four point method is sufficiently well estimated by a distance estimator and learning method, the algorithm finds the shortest possible joining edge and proceeds. The efficient algorithm optimizes this approach by caching various sequence learning results.

We compared our algorithm to neighbor joining, PAUP*, and RAxML. The most interesting was RAxML, a likelihood-based program which is the best program we were able to find. Results are given later in Section 2.4. We found that our initial efforts, while good, were clearly inferior to RAxML's trees. One theory was that our method of learning sequences, simple majority at depth two, did not account for edge lengths (nor should it have to according to the theory), while RAxML was making full use of estimated edge lengths and every parameter it could get its hands on. The quartet method gives a reliable estimate of the middle edge of a quartet, so we decided to use that to estimate edge lengths. We used these edge lengths as a rudimentary weight in the majority calculation of the sequence learning process. This version is dubbed the weighted quartet algorithm and is, to date, the best performing algorithm we have. Its performance is comparable to RAxML's, with the exception of some particularly difficult small interior edges that RAxML seems to be better at learning than we are. Getting those last edges correct is the focus of the remainder of this research.

In addition to using edge weights to compute a simple weighted majority, we use those edge weights to make a Bayesian calculation of the probabilities of certain characters in sequences. The motivation behind this was to mirror the likelihood calculation that RAxML was performing during its learning process, as well as to accurately account for the probabilistic process that generated the data in the first place. This is tricky to get right, and simply propagating probabilities from the leaves all the way to the interior nodes of the tree was unsuccessful. Just as in the quartet method, it is necessary to round at some point and learn internal sequences, instead of always relying on just the leaf data. It is not clear why this is the case, or what is the best way to do Bayesian sequence learning.

We note that the method of picking two close trees and then doing a quartet walk is potentially more expensive than it needs to be, especially since we observed that the correcting quartet walk rarely occurs on trial runs. According to the theory, the length of the middle edge of the quartet joining two trees should be minimized at the join point. Therefore, if we simply tried joining between every pair of edges, then the correct place to join would be at the smallest middle edge, with no need to walk. This optimization simplifies the code and speeds it up. But, we eventually discovered that using this as a starting point was a good idea, as it usually eliminated the walk, but it is still necessary to check in case of small interior edges.

Another thing we tried was fixing up quartets after the fact. Our program would sometimes make mistakes on early edges, joining trees off by one vertex, because it did not have information on where the later vertices were in the tree. This appears to be a problem caused by the order of joining; the incorrect quartet can be detected and corrected once the tree is fully assembled. In retrospect, we realize that this is pretty much how RAxML is operating. It first generates a candidate tree with some simple, greedy method. Then, it spends most of its time trying out various local rearrangements on that tree which might improve its likelihood score. We initially

tried this using the quartet test as the check of whether or not a quartet needed to be flipped. We also later used a likelihood test, although it only considered the internal sequences on the quartet and not the entire tree as RAxML generally does.

2.3.1 Joining Metric

One option for the joining metric is parsimony. We join two trees at the point which increases the parsimony score the least. This approach, while not optimal, is simple to implement and compares favorably in practice to the `hsearch` (heuristic search) method of PAUP*. The problem of maximizing parsimony over an entire tree is NP-complete [FG82].

Another joining metric is the quartet test. We select two trees to join which have vertices which are close to each other by the corrected distance metric, a variation on Hamming distance which corrects for bias. It does not matter if we precisely get the smallest distance here, only that the distance between the two trees is smaller than some threshold. Given two close trees as a starting point, we then perform a quartet test to determine precisely where the join points should be. This quartet test performs reliably if the distances between the two trees is short. This method is more computationally expensive than the parsimony method, but gives better results.

2.3.2 Learning Edge Lengths

We use the four point method not only to decide where to join trees, but also to learn edge lengths. We can reliably learn the length of an edge in the middle of a quartet of sequences. We use this method, along with the sequence learning algorithm, to bootstrap other edge lengths in the tree.

Another edge learning technique we considered was maximum likelihood. Instead of using the four point method directly, we computed the maximum likelihood quartet and branch lengths from the sequences at the leaves of the quartet. This optimization is more expensive, but bounded in cost because we are just doing it for a quartet, not the entire tree.

2.3.3 Learning Internal Sequences

We have three methods for learning the sequence at an internal node from its two children.

The first is to take a simple majority of the four children at a depth of two from the node. If three or more share the same character, the parent gets that character. Otherwise, the parent character is unknown.

The second technique is similar to the first, but instead we take a weighted majority of the children. The weights we use on each branch are the copy probabilities, $\theta(e)$.

The third technique is Bayesian. We use the branch lengths to determine the conditional probabilities of each character at the parent. We carry these distributions through the tree as if we were computing the likelihood of the tree. We round the probabilities in order to determine an actual sequence at each internal node. This is crucial for the four point method.

2.3.4 Disjoint-Set Forests

As we join together trees into a forest, we need to be able to efficiently tell if two nodes are in the same tree. This calls for a disjoint-set forest. We use a modified version of the standard union-find algorithm [CLR90, Chapter 22].

2.3.5 Pruning

If the distance across the quartet is too large, our techniques to estimate the middle edge will get overwhelmed by the overall noise, especially if the middle edge itself is small. We have a technique to cope. When an edge is too long, we remove it from consideration entirely. Statistically, there is no bias, although we have reduced the sample size. It only makes sense to do this if the loss in signal is less than the noise being removed.

We implement this in the algorithm by assigning a weight of 0 to edges we want to eliminate. This implies that their neighbor will get weight 1, and we will just copy the more reliable sequence in its entirety rather than introduce excessive noise.

The log likelihood function sums the log likelihoods over leaf patterns seen at each nucleotide site. The effect of this is to decrease the influence of sites where the distribution at any of these sequences is flat. That is, the sites where the learning noise is high are automatically given lower weights compared to sites where the confidence in the learned character is higher.

We incorporate this idea into the distance method by adding a bit pruning heuristic into our approach. Ideally, such an approach would prune bits by trading off the signal added for each bit with the noise introduced by each bit on a per quartet basis. Experiments show that choosing thresholds appropriately can improve the performance of our approach.

In particular, one might set the threshold as a function of sequence length or even locally for a particular quartet based on the signal available in the associated learned sequences.

2.3.6 Post Facto Fixup

This heuristic makes a pass over the final tree using a quartet method. The theory is that when the tree is completely assembled and mostly correct, is is possible to correct earlier mistakes made with less information. The issue is, once again, that optimizing using the proxies available for a correct topology may not actually help given noise, and that our fixup methods do not necessarily globally optimize the measure. Moreover, for some of our methods, we may improve the score for a particular place, while decreasing the score corresponding to another location in the tree.

2.4 Experimental Results

In this section, we experimentally evaluate our algorithm on molecular data.

2.4.1 Experimental Setup

We first generate random trees using `r8s` [San03]. By default, `r8s` generate ultrametric trees, so we must randomly perturb the output. Another factor in the difficulty of trees is the maximum normalized Hamming distance for any pair of vertices. We scale the branch lengths of the trees to control the difficulty of a dataset.

For each random tree, we generate sequence data. We start with a random sequence of chosen length at the (arbitrary) root. We evolve the sequence to get internal sequences and leaf sequences. The leaf sequences are the test data, while the original tree is the correct answer.

We feed the leaf sequences to all programs and compare the resulting trees with the model tree. The Robinson-Foulds distance between two trees measures the number of differing edges. We normalize this value over the number of possible edges to get the normalized error rate for each program on each input.

In our first set of experiments, we compared the following programs:

- `mp` – PAUP* maximum parsimony
- `nj` – PHYLIP neighbor joining
- `qw` – Our program using weighted majority sequence learning
- `r` – RAxML

Our programs take as input the leaf sequences in PHILIP format and output output a tree in NEWICK format. Other programs do not necessarily operate in this manner, so we wrap them in shell scripts so that they can be run in our test framework.

2.4.2 Results

We ran a series of experiments on trees with varying sizes and sequence lengths. The trees ranged from 100 to 400 taxa, and the sequence lengths ranged from 1000 to 5000. For each program, we plot the normalized Robinson-Foulds error versus the number of taxa. Each plot only contains data points of a fixed sequence length. As discussed before, normalized Hamming distance is supposed to be a proxy for the difficulty of the dataset. We ran experiments and have results for 6 datasets. Trees in datasets A (Figure 2.2), B (Figure 2.3), and C (Figure 2.4) all have normalized Hamming distances between 0.6 and 0.7. Trees in dataset D (Figure 2.5) have a normalized Hamming distance between 0.75 and 0.8. Trees in dataset E (Figure 2.6) have a normalized Hamming distance of roughly 0.5. Finally, trees in dataset F (Figure 2.7) have a normalized Hamming distance of about 0.25.

There are three factors which influence the difficulty of inferring a tree from its sequences. First, we expect all programs to make more errors with short sequence lengths, and converge to correctness as the sequence length increases. Second, we expect that increasing the number of taxa will increase the error rate, not only because there are more opportunities for error, but

also because more edges are deep in the tree and thus difficult to learn. Finally, the maximum normalized pairwise Hamming distance serves as a proxy for difficulty. We especially expect neighbor joining, which is a distance method, to be tripped up by high pairwise distances.

We find that in all datasets, neighbor joining generally performs worse than all other programs. As expected, this is especially true in dataset D, which has the highest normalized maximum Hamming distances. We also find the RAxML tends to be the best of the programs we tested in most cases. In most of the datasets, we find that maximum parsimony and our program fall between neighbor joining and RAxML. There is one exception. In dataset E, which has the lowest normalized maximum Hamming distances, we find that maximum parsimony does as well as RAxML.

The absolute differences between the programs are small. We are dealing with a constant factor difference. But these constant factors are significant, because when the learning error improves by a factor of 2 or 3, the probability of learning that edge incorrectly is 4 or 5 standard deviations out, and thus on the order of 1000 times more unlikely.

Our algorithm is sound. Not only is it a well-understood terminating algorithm with proven bounds, at each step of the algorithm we can know how confident we are of the edge we just added. It knows with some certainty that it is right at each step. When we do not have confidence, that is significant as well.

2.4.3 Illustrative Example

We evaluate our program versus neighbor joining and RAxML. This experiment is deeper and narrower than the previous experiments. We choose one interesting 100-taxa tree on which to plot results. For this tree, we generate sets of random sequences of varying length. For each sequence length, we generate 10 sets of random sequences. To evaluate the programs, we run them on the sample inputs and compare the resulting tree to the original tree. The metric we use for this comparison is normalized error, which is the percentage of internal edges which differ from the model tree. To generate a smooth and accurate graph, we average the results for the 10 inputs at each sequence length.

The results are plotted in Figure 2.8. On this dataset, we are more accurate than neighbor joining but less accurate than RAxML. The tree we selected is by no means special. We got similar results on many other trees. The size of the gap between the programs in this experiment seems to depend on the number of difficult edges in the tree. We do not have an exact characterization of what makes an edge difficult, be we do know that small edges deep in the tree are more difficult than large edges toward the boundary of the tree.

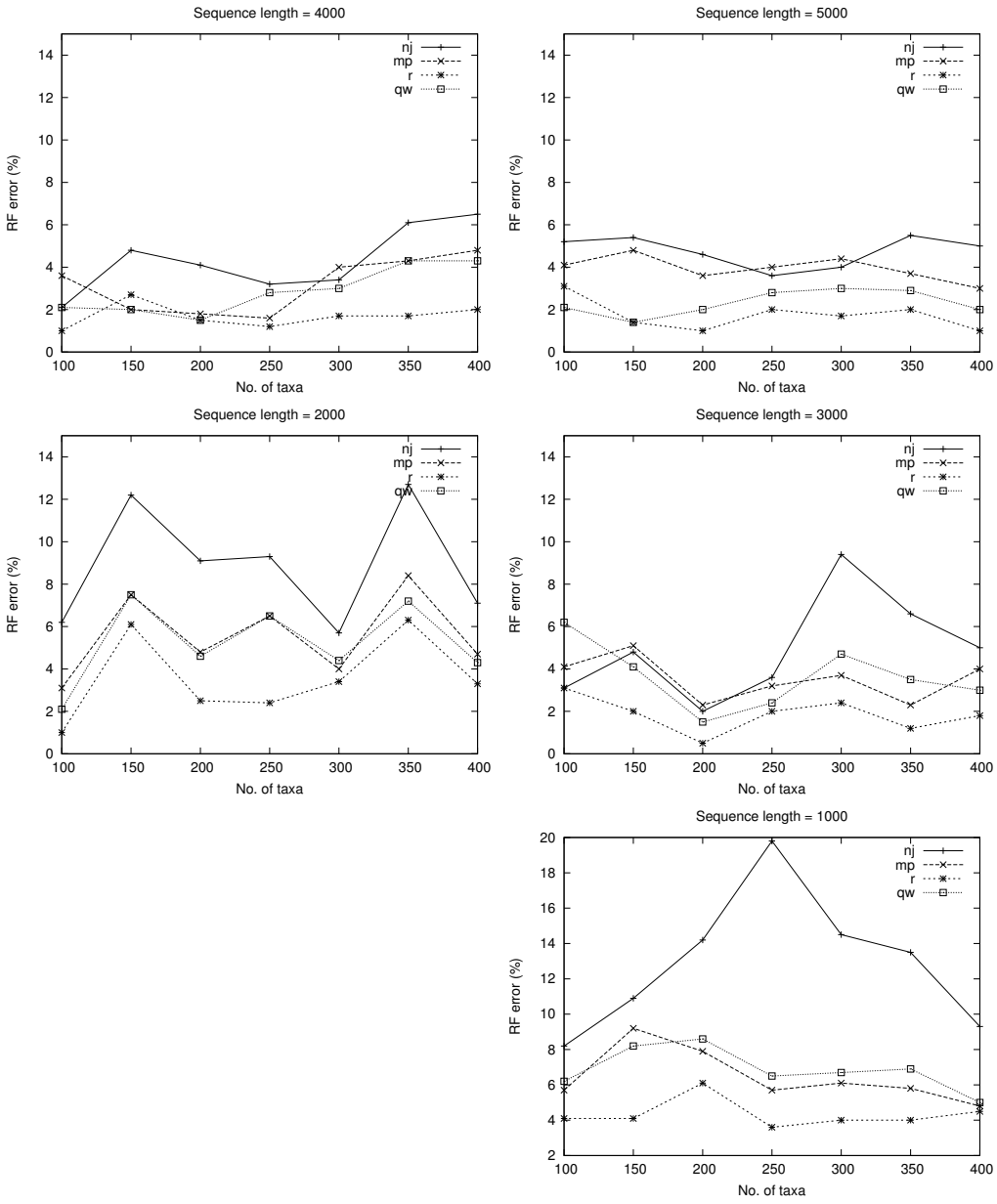


Figure 2.2: Dataset A. We compare neighbor joining (nj), maximum parsimony (mp), RAxML (r), and our weighted quartet program (qw). Trees in this dataset are not scaled, and have a maximum normalized Hamming distance of 0.6–0.7. The y-axis is the percentage of internal edges a program gets wrong. The independent variables are the number of taxa in a dataset (leaves) and the length of the sequences. At lower sequence lengths and higher number of taxa, RAxML makes the fewest errors while neighbor joining makes the most.

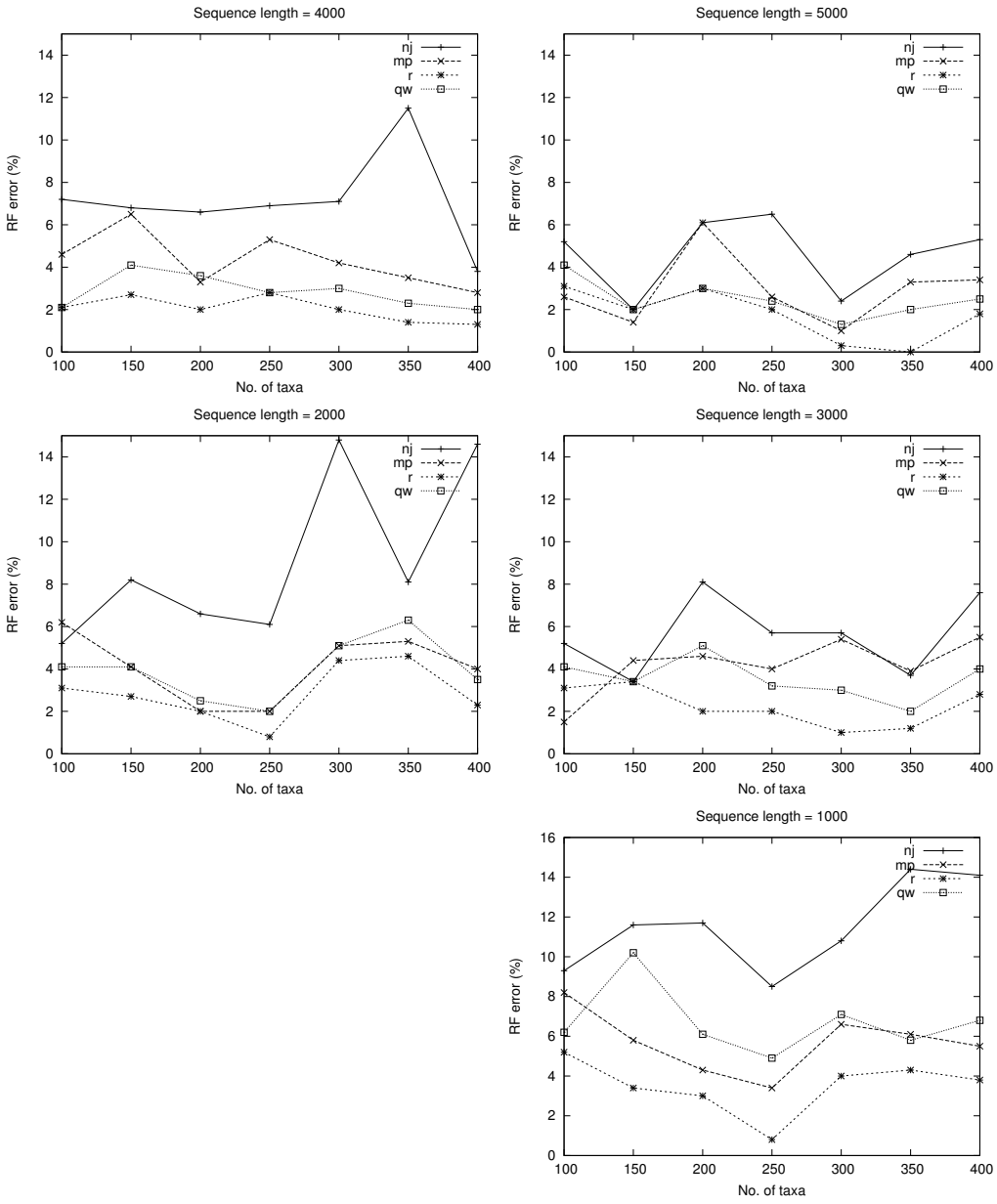


Figure 2.3: Dataset B. We compare neighbor joining (nj), maximum parsimony (mp), RAxML (r), and our weighted quartet program (qw). Trees in this dataset are not scaled, and have a maximum normalized Hamming distance of 0.6–0.7. The y-axis is the percentage of internal edges a program gets wrong. The independent variables are the number of taxa in a dataset (leaves) and the length of the sequences. As in dataset A, at lower sequence lengths and higher number of taxa, RAxML makes the fewest errors while neighbor joining makes the most.

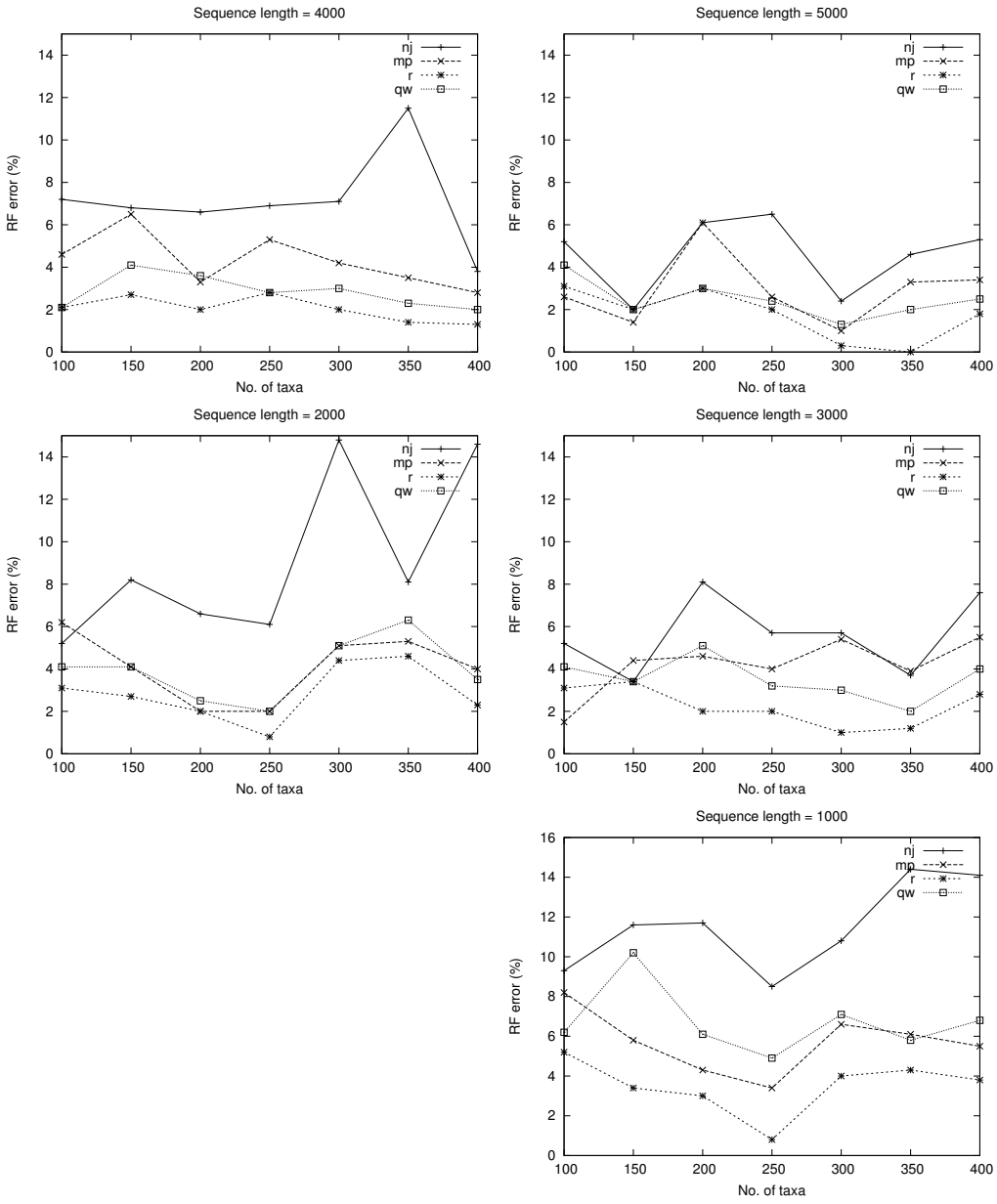


Figure 2.4: Dataset C. We compare neighbor joining (nj), maximum parsimony (mp), RAxML (r), and our weighted quartet program (qw). Trees in this dataset are not scaled, and have a maximum normalized Hamming distance of 0.6–0.7. The y-axis is the percentage of internal edges a program gets wrong. The independent variables are the number of taxa in a dataset (leaves) and the length of the sequences. As in datasets A and B, at lower sequence lengths and higher number of taxa, RAxML makes the fewest errors while neighbor joining makes the most.

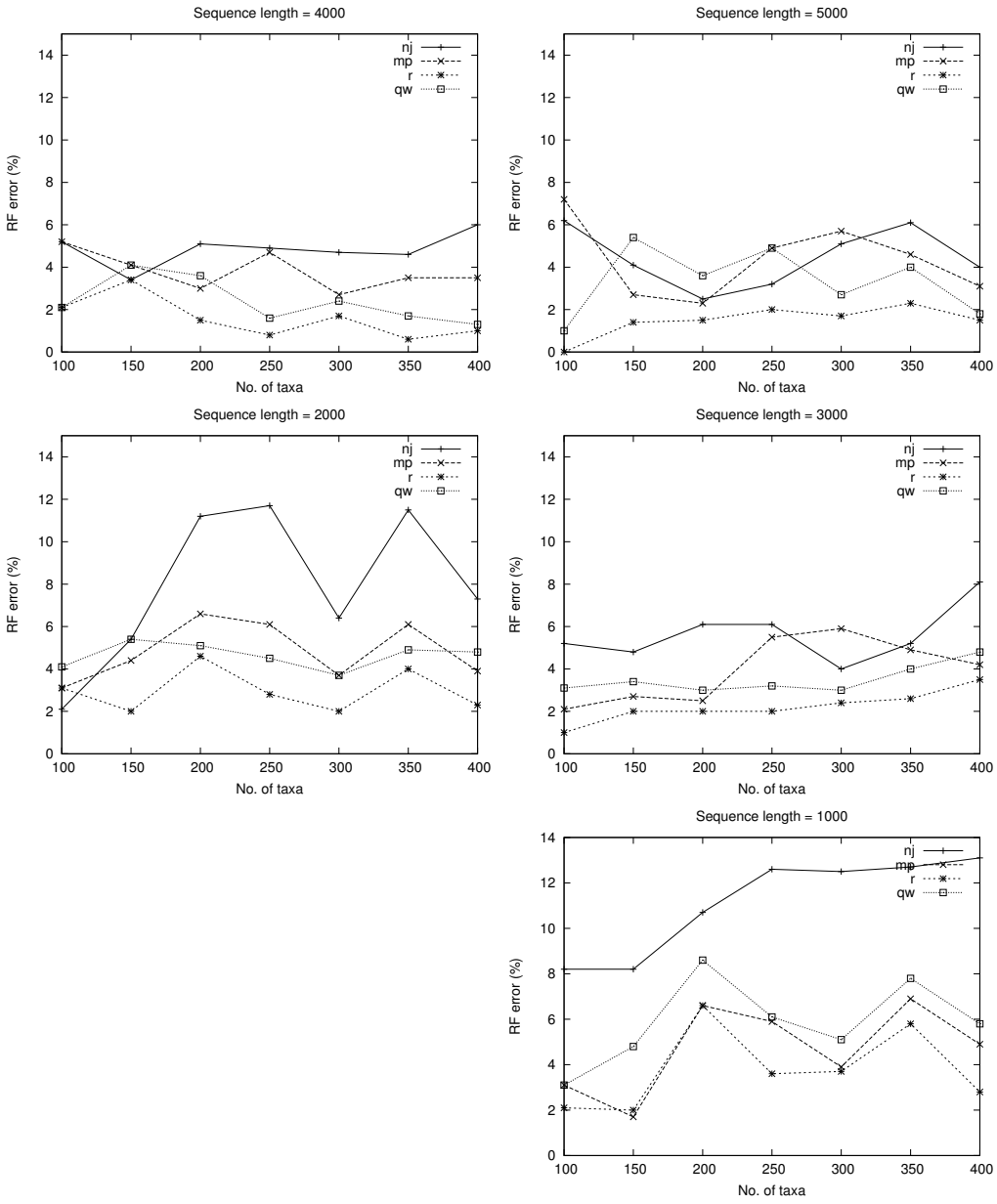


Figure 2.5: Dataset D. We compare neighbor joining (nj), maximum parsimony (mp), RAxML (r), and our weighted quartet program (qw). Trees in this dataset are scaled by a factor of 2, and thus have a maximum normalized Hamming distance of 0.75–0.8. The y-axis is the percentage of internal edges a program gets wrong. The independent variables are the number of taxa in a dataset (leaves) and the length of the sequences. The large pairwise distances cause neighbor joining to especially poorly, while RAxML only has a clear advantage on trees with sequence length 1,000.

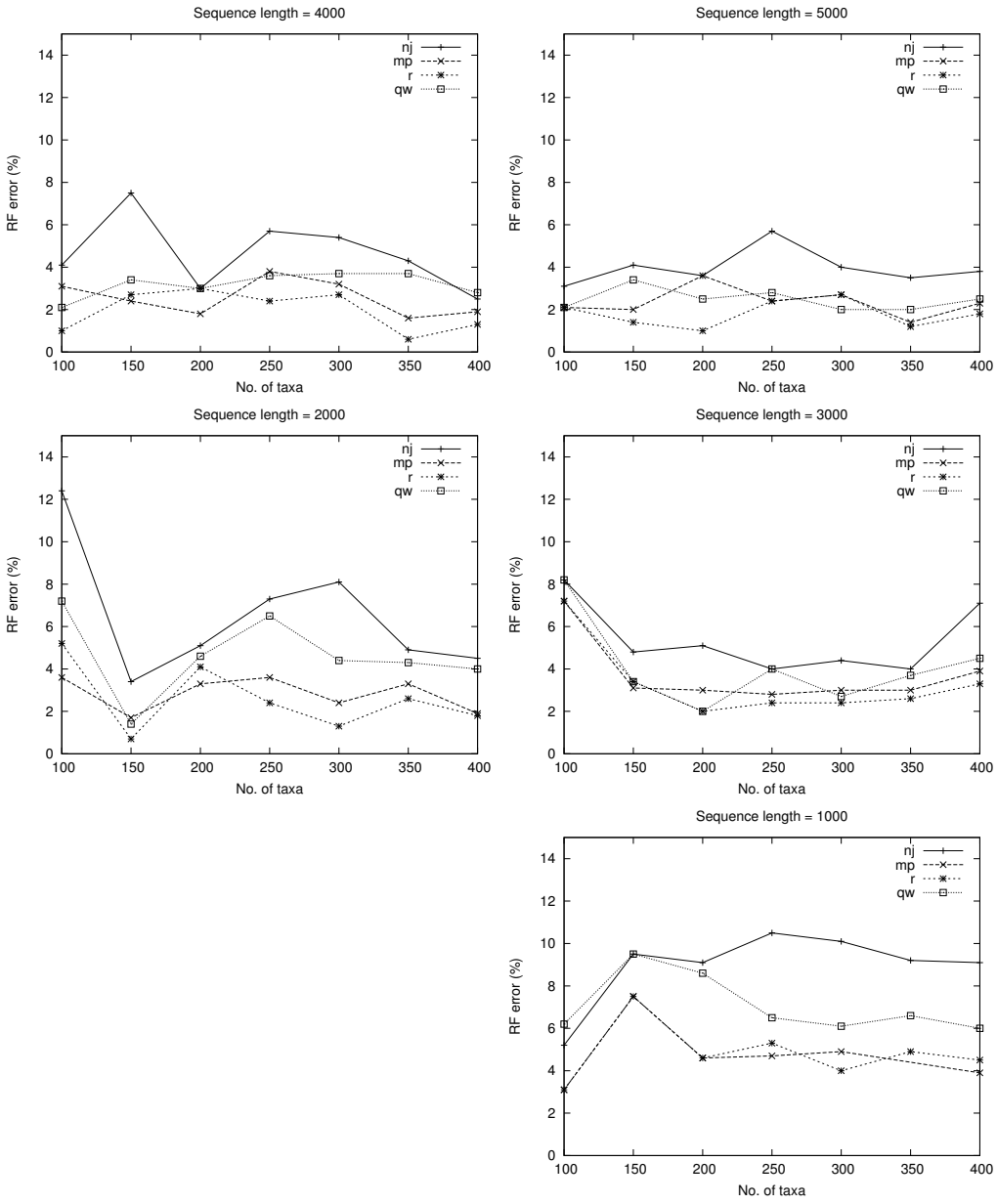


Figure 2.6: Dataset E. We compare neighbor joining (nj), maximum parsimony (mp), RAxML (r), and our weighted quartet program (qw). Trees in this dataset are scaled by a factor of 0.5, and thus have a maximum normalized Hamming distance of about 0.5. The y-axis is the percentage of internal edges a program gets wrong. The independent variables are the number of taxa in a dataset (leaves) and the length of the sequences. All programs do well due to the small pairwise distances, except for neighbor joining on some trees with sequence length 1,000.

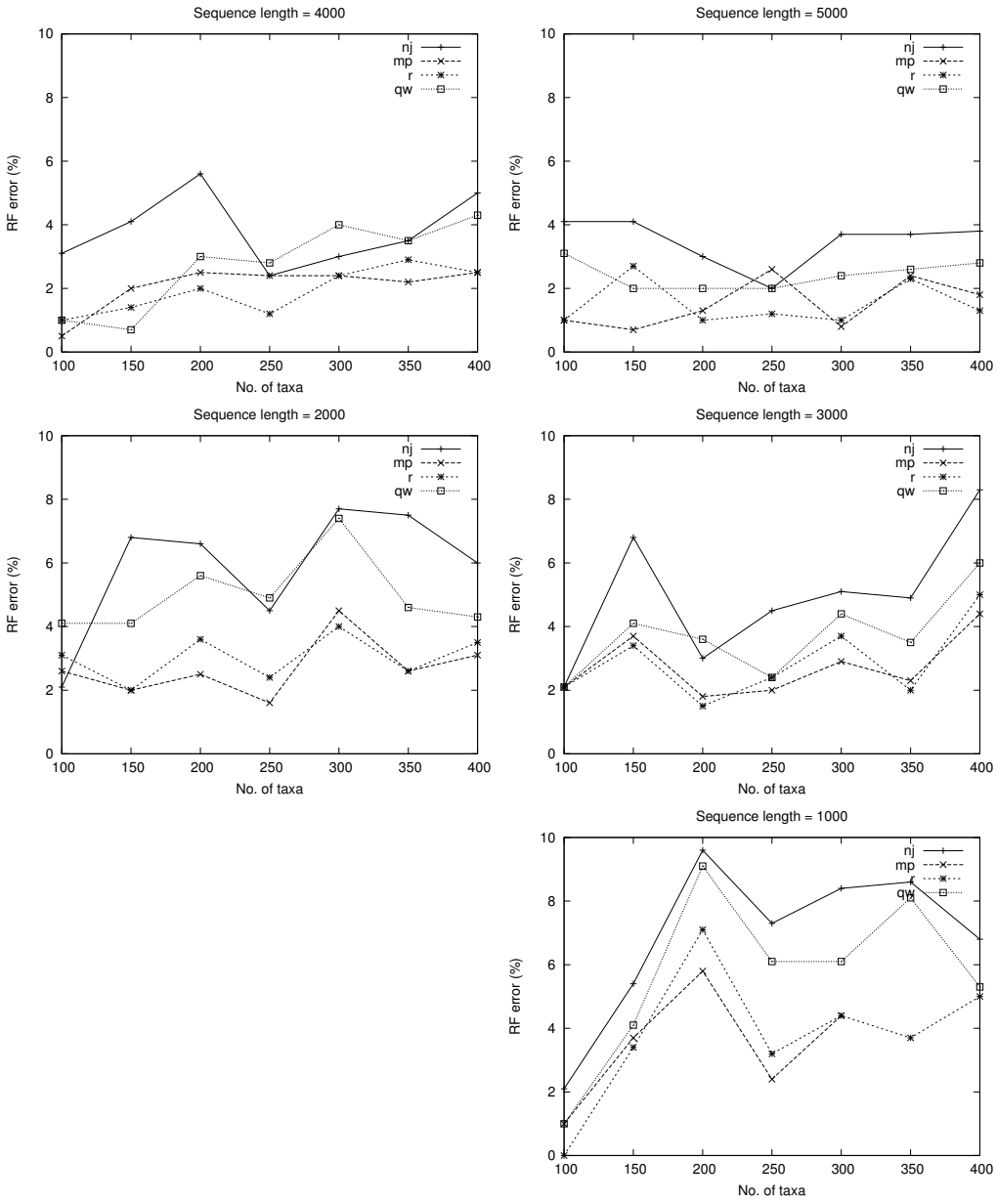


Figure 2.7: Dataset F. We compare neighbor joining (nj), maximum parsimony (mp), RAxML (r), and our weighted quartet program (qw). Trees in this dataset are scaled by a factor of 0.25, and thus have a maximum normalized Hamming distance of about 0.25. The y-axis is the percentage of internal edges a program gets wrong. The independent variables are the number of taxa in a dataset (leaves) and the length of the sequences. In this low mutation regime, all programs do well, but RAxML and maximum parsimony appear to be the clear winners.

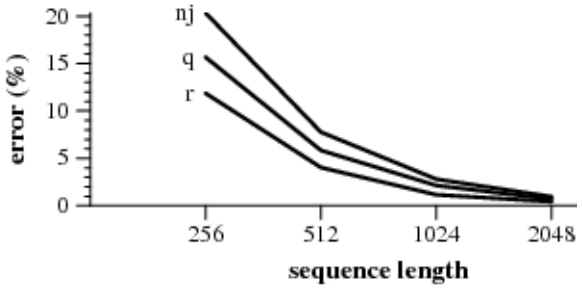


Figure 2.8: We compare neighbor joining (nj), RAxML (r), and our quartet program (q) on one particular 100 taxa tree. Each point in the graph is the result of generating 10 sets of sequences for that tree and running the programs on those sequences. The y-axis is the percentage of internal edges a program gets wrong, averaged over the 10 runs. The x-axis is the sequence length. Two trends are clear. First, as the sequence length increases, the error rate of all programs tends to 0. Second, there is a clear order among the programs. RAxML makes fewer errors than our quartet program on average, and our program makes fewer errors than neighbor joining on average.

Chapter 3

Learning Quartets

We compare two methods for learning quartets, a distance method and a likelihood method. The input is a set of four sequences or distributions, and the output is the proper quartet arrangement along with some measure of a confidence. Note, there are only three possible arrangements here (Figure 3.1).

3.1 Distance Method

Given a set of sequences a, b, c, d and estimated pairwise distances, denoted $\widehat{d}(a, b)$, the distance method estimates the middle edge to be

$$\frac{\widehat{d}(a,c) + \widehat{d}(a,d) + \widehat{d}(b,c) + \widehat{d}(b,d) - 2\widehat{d}(a,b) - 2\widehat{d}(c,d)}{4}. \tag{3.1}$$

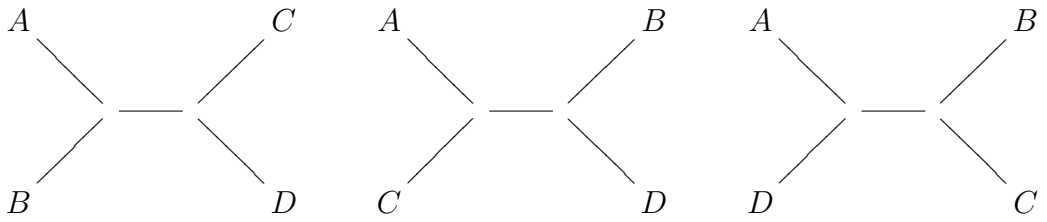


Figure 3.1: Three possible quartet arrangements.

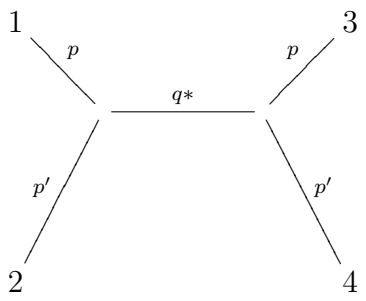


Figure 3.2: A difficult quartet: $p = 20\%$, $p' = 60\%$, and $q = 30\%$.

3.2 Likelihood Method

The likelihood method optimizes a log likelihood function

$$LL(T|D) = \sum_i \log L(T|D_i) \tag{3.2}$$

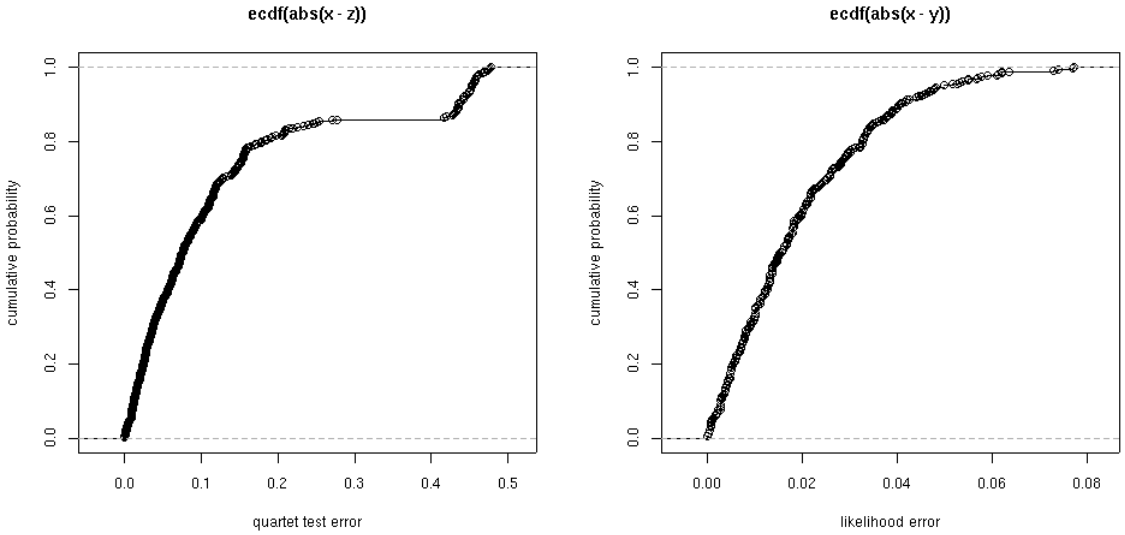
where T is the quartet with branch lengths, and D_i is the sequence data observed at the ith character. This function is optimized by any means necessary, which for us means either Newton's method or a simple grid search since the search space is relatively small and can be searched exhaustively.

3.3 Why Likelihood Works

We perform an experiment where for a given model quartet, we generate sets of 1000 random characters and then evaluate the two algorithms on the data. We repeat this many times in order to get a complete picture of the accuracy of the algorithms at learning the middle edge. Note that the value of the middle edge we expect the algorithms to learn is the empirical value based on the two internal sequences of the quartet, not the model value. Although, on average, the empirical value equals the model value, we wish to evaluate how close the algorithms come to learning the empirical value on any given trial, not the variance of the empirical value itself (which is a function of the sequence length).

We give some results for one particularly bad quartet in the Felsenstein zone, with $p = 20\%$, $p' = 60\%$, and $q^* = 30\%$ (Figure 3.2). These values are straight expected mutation rates, not adjusted copy probabilities. For this case, we give some empirical graphs of the absolute error of each method versus the true empirical values. To collect this data, we fix the outer edges and let the likelihood optimizer figure out the optimal value of q. This gives an unfair advantage to the likelihood method.

In Figure 3.3, we see that the likelihood method is a better estimator of the middle edge length than the quartet method. In order to understand the behavior of the likelihood method, we examine



(a) Distance Method

(b) Likelihood Method

Figure 3.3: Quartet test error versus likelihood test error. These are plots of the empirical cumulative distribution of error for these two methods on one particular quartet.

the statistics of the likelihood function. In Figure 3.4, we plot the expected value of $LL(q)$ with error bars at one standard deviation. Examining the figure, it appears that $LL(q)$ can be potentially maximized at a point far from the optimal value q^* without deviating from the error bars.

In fact, a better way to narrow down the permissible range is to analyze the difference between $LL(q)$ and $LL(q^*)$. Assuming q^* is the model edge length, define

$$\Delta LL(q) = LL(q) - LL(q^*). \tag{3.3}$$

If $\Delta LL(q) > 0$ then q *might* maximize LL. Thus, we compute the expected value and variance of $\Delta LL(q)$ instead of $LL(q)$ and get a much tighter bound on likely values of q. In particular, as shown in Figure 3.5, the likelihood optimizer will likely output a value of q between 0.24 and 0.36 (the true value is 0.30). This analysis matches the empirical error for the likelihood method from Figure 3.3, where we observe that the error is less than 0.06 with 95% probability.

3.4 Quartet Likelihood Experiment

In this section, we test the hypothesis that RAxML's advantage is due to its use of likelihood. We compare RAxML, dnaml, and our own quartet likelihood optimizer. Details on running RAxML and dnaml are in Appendix C.

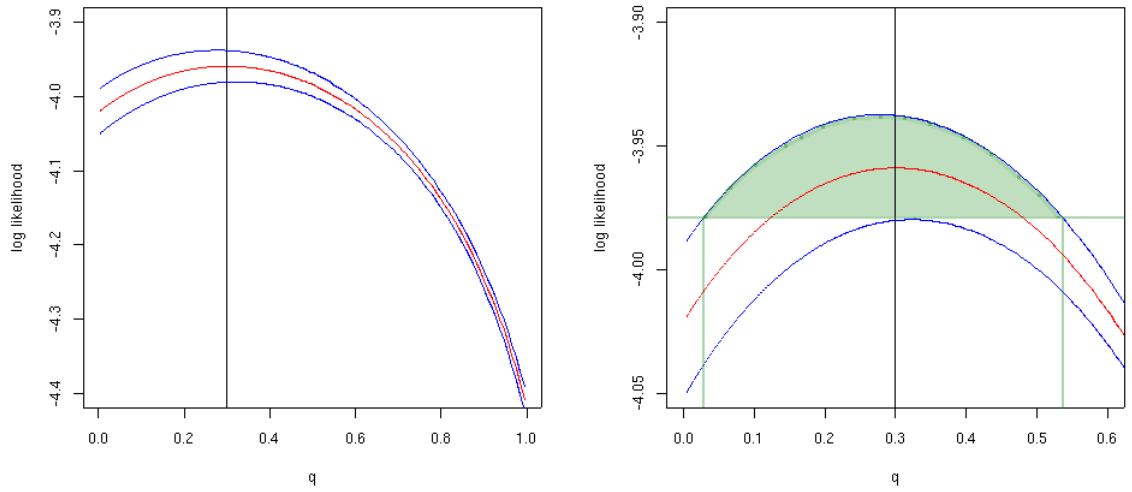


Figure 3.4: $E[LL(q)] \pm \sigma$. The middle curve is the expected value, while the outer curves represent one standard deviation. The right graph is a zoomed version of the left graph. The shaded region shows that that q can lie between 0.03 and 0.55 without deviating from the error bars.

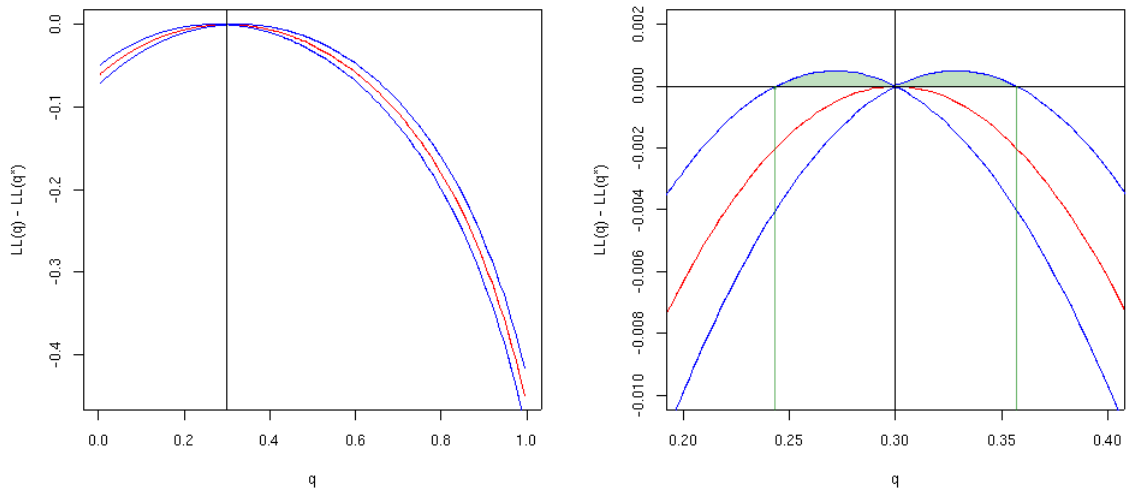


Figure 3.5: $E[\Delta LL(q)] \pm \sigma$. The middle curve is the expected value, while the outer curves represent one standard deviation. The right graph is a zoomed version of the left graph. The shaded region shows q can lie between 0.24 and 0.36 without deviating from the error bars.

Table 3.1: Quartet Showdown

	percent correct			best likelihood		
n	ll	RAxML	dnaml	ll	RAxML	dnaml
100	66.8	69.4	64.9	81.8	9.2	9.0
250	71.5	76.5	70.3	83.7	9.8	6.5
500	72.1	79.6	74.8	83.4	10.1	6.5
1000	76.1	82.7	78.4	84.5	9.7	5.8

The test dataset for this experiment consists of 1,000 quartets each with sequences lengths of 100, 250, 500, and 1,000. Each edge of a quartet has a length randomly generated by the expression $-\log r$, where r is picked uniformly at random from the range $[10^{-6}, 1)$. Values less than 10^{-6} are excluded to avoid numerical problems with the various algorithms. Given branch lengths for a quartet, we then generate sequences for the quartet.

Our likelihood optimizer uses the Newton-Rhapson method in an attempt to directly optimize the likelihood function over the space $\theta \in [0, 1]^5$. To avoid a bad initial position, we split the search space into $2^5 = 1024$ half-planes. We search each half-plane separately and pick the maximum over all searches.

Each method is given the leaf sequences of the quartet and asked to determine the correct quartet as well as its edge lengths. The likelihood optimizer determines branch lengths and likelihood values for the three possible quartet orientations. For each program, we are primarily interested in whether or not it picked the correct quartet orientation. We also independently compute the likelihoods for the quartets returned by each program.

The results of the experiment are in Table 3.1. We find that RAxML is better at inferring the correct topologies than a straight likelihood optimizer. We find this to be the case even though our likelihood optimizer knows the evolutionary model is Jukes-Cantor, while RAxML optimizes under the general GTR-Γ model. Our likelihood optimizer produces quartets and edge lengths which have a higher likelihoods than the quartets returned by RAxML. RAxML's advantage does not appear to be in its likelihood optimizer.

We also ran dnaml as an alternative to our likelihood optimizer to verify that our optimizer is effective. Indeed, our likelihood optimizer, albeit slower, produces better likelihood values than dnaml.

Chapter 4

Gene Order Phylogeny

In this chapter, we give a polynomial time algorithm for inferring phylogenies from gene order data. Our basic contribution is a method for estimating distance and an associated learning method which can be used to infer adjacencies at internal nodes. This leads to a polynomial time algorithm for reconstructing a phylogeny on a set of n taxa from gene order data consisting of $N = O(\log n)$ genes. We give an ancestral reconstruction technique and an associated distance metric that can be used in the framework of Mihaescu et al. [MHR08].

4.1 Whole Genome Phylogeny

We have discussed the problem of phylogenetic reconstruction in terms of molecular data, i.e. A, C, G, and T. One can take a longer view by looking at the order of genes in a genome. In theory, genes are conserved over a much longer period of time than one could conceivably align sequences by characters. These genes get moved around by large evolutionary events, such as inversions, transversions, and transpositions. Algorithms that operate on the whole genome have the potential to solve the broader problem of how very different species relate to each other, rather than just the closely related species.

For molecular data, the most effective approaches for inferring phylogenies are based on reconstructing internal sequences. For gene order data, the problem of reconstructing internal genomes remains a challenge. For example, even finding the median of three signed permutations under the inversion distance or breakpoint metrics is NP-hard [Cap99, PS98].

Previous approaches, as embodied for example in GRAPPA [MTWW02], use various heuristics and branch and bound techniques to find good solutions for small trees. Due to the exponential nature of their methods, they need to piece together these small tree solutions for larger trees. Moreover, the algorithms are not shown to converge to the correct tree even if they were to solve the NP-hard problems optimally.

4.2 Our Approach

The key idea of our algorithm is to not just use the distances between leaf genomes, but to also learn ancestral adjacencies. For molecular data, the ancestral learning approach is more accurate than distance-based methods in theory and practice. The recursive majority algorithm for learning ancestral sequences was introduced and analyzed by Mossel [Mos04]. This recursive procedure is also a key step in the phylogeny reconstruction algorithms of Daskalakis, Mossel, and Roch [DMR06] and Mihaescu, Hill, and Rao [MHR08], which only require logarithmic length sequences (as a function of the number of taxa).

Our algorithm is based on the recursive majority learning algorithm and the tree joining methods of Mihaescu et al. We maintain a forest of correct subtrees with inferred adjacencies on the interior nodes. We use the inferred adjacencies to decide where to join two trees in the forest using the now classical four-point method. The four-point method can sort out the topology of a quartet given sufficiently accurate distances among the four nodes. Our contribution is to show that sufficiently accurate distances can be recovered at internal nodes using gene-order data. After joining two trees, we infer the internal adjacencies using the recursive majority algorithm and proceed with the rest of the algorithm until the entire tree is reconstructed.

It is not immediately apparent how one can reconcile character-based models of evolution, which assume that each character evolves independently, with the Nadeau-Taylor model for gene-order data. We simplify the gene-order data by showing that it is sufficient to represent a genome by its set of adjacencies rather than its full signed permutation. We also show that it is okay to assume that these adjacencies evolve independently as if they were characters. This simplification leads to a distance estimator and ancestral learning algorithm which we can plug directly into the algorithm of Mihaescu et al.

4.3 Definitions and Notation

In this section, we define the representation of a genome along with the Nadeau-Taylor [NT84] model of evolution.

A *genome* consists of a set of N genes. We assume each genome has the same set of N genes and each gene appears exactly once. A genome is described by a signed circular permutation of the set $[N]$. We call this permutation the *gene order* data. Let G be a genome with signed ordering $g_1, g_2, \ldots, g_N$. Since G is circular, any rotations are equivalent (e.g. $g_2, \ldots, g_N, g_1$). Any signed reversals are also equivalent (e.g. $-g_N, -g_{N-1}, \ldots, -g_1$).

Remark. The number of unique gene-orders on N genes is $2^{n-1}(n-1)!$

Genomes evolve through a serious of evolutionary events. We consider three classes of genome rearrangement events:

- An *inversion* between indices i and j, for $i \leq j$, reverses the interval between g_i and g_j,

resulting in the genome

$$g_1, \ldots, g_{i-1}, -g_j, \ldots, -g_i, g_{j+1}, \ldots, g_N.$$

- A *transposition* on G acts on indices i, j, k, with $i \leq j$ and $k \notin [i, j]$, by transplanting the interval between g_i and g_j and inserting it immediately after g_k. This results in the genome

$$g_1, \ldots, g_{i-1}, g_{j+1}, \ldots, g_k, g_i, \ldots, g_j, g_{k+1}, \ldots g_N.$$

- An *inverted transposition* is an inversion composed with a transposition. Like a transposition, it acts on indices i, j, k, with $i \leq j$ and $k \notin [i, j]$, resulting in the genome

$$g_1, \ldots, g_{i-1}, g_{j+1}, \ldots, g_k, -g_j, \ldots, -g_i, g_{k+1}, g_N.$$

The *Nadeau-Taylor* model of genome evolution uses only genome rearrangement events. The model assumes that each of the three types of events obeys a Poisson distribution on each edge, with the three means for the three types of events in some fixed ratio. Within each class, all events occur with equal probability.

In this chapter, we further simplify the model by assuming that only inversions occur.

Definition 1. Genes i and j are said to be adjacent in the gene-order if either i is followed immediately by j or $-j$ is followed immediately by $-i$. Such a pair (i, j) is called an *adjacency*.

Remark. We note that (i, j) and $(-j, -i)$ are the same adjacency. We usually view a gene-order as the set of its adjacencies.

In the phylogeny reconstruction, we assume that there is a model tree T with n leaves rooted at ρ, and for each edge $e \in T$, a real number $D(e)$ specifying the mutation rate. The genome on every node of T evolves from the genome at ρ. For edge $e = (u, v) \in T$, with u being the parent of v, the genome at v is the result of applying $D(e) \cdot N$ random inversions to the genome at u. Note that $D(\cdot)$ induces a tree metric on nodes in T. Given a set of observed genomes on the leaves, denoted δT, we want to reconstruct the model tree T.

Our method is based on learning ancestral genomes. More specifically, our algorithm maintains a set of correct subtrees of T, infers sets of adjacencies on interior nodes of each tree using genomes on its leaves, and uses that information to decide how to join these subtrees.

We now define notions of the distance between two random genomes. For node $u \in T$, let u be a random variable denoting the genome at node u. We abuse the notations further by denoting also by u the set of all adjacencies in u. For subtree T' of T rooted at u, such that $\delta T' \subseteq \delta T$, we let $\tilde{u}(T')$ denote the set of learned adjacencies at node u by the recursive procedure on tree T'. Note that $\tilde{u}(T')$ is a random variable. For two random genomes u and v, let

$$\widehat{\theta}(u, v) = |u \cap v|/N$$

be the fraction of common adjacencies, and

$$\theta(u, v) = \mathbf{E}[|u \cap v|/N]$$

be its expectation. We also define the estimated distance to be

$$\widehat{d}(u, v) = \frac{\log \widehat{\theta}(u, v)}{N \log(1 - 2/N)}$$

and the true distance to be

$$d(u, v) = \frac{\log \theta(u, v)}{N \log(1 - 2/N)}.$$

Intuitively, since $\theta(u, v)$ will be approximately $(1-2/N)^{D(u,v)N}$ (see Section 4.6), $d(u, v)$ approximates $D(u, v)$.

To extend the distance notation to sets of learned adjacencies, we let

$$d(\tilde{u}(T'), u) = D(\tilde{u}(T'), u) = \frac{\log(\mathbf{E}\left[|\tilde{u}(T') \cap u|\right]/N)}{N \log(1 - 2/N)},$$

and

$$\widehat{d}(\tilde{u}(T'), u) = \frac{\log(|\tilde{u}(T') \cap u|/N)}{N \log(1 - 2/N)}.$$

For a fixed set A of adjacencies, we also define notions of distances with respect to A as follows.

$$\theta_A(u, v) = \mathbf{E}\left[|u \cap v \cap A|/N\right]$$
$$d_A(u, v) = -\log \theta_A(u, v)$$

We define $\widehat{\theta}_A(\cdot, \cdot)$ and $\widehat{d}_A(\cdot, \cdot)$ accordingly.

4.4 Empirical Distance Theorems

We now state theorems regarding empirical distances. These theorems are proved later in Sections 4.6 and 4.7.

Theorem 1 *For any $\epsilon > 0$ and $M > 0$ there exists an $N = O(\log n)$ such that for two random genomes G and G' the following are true:*

1. $\Pr[|\widehat{d}(G, G') - D(G, G')| > \epsilon] < 1/n^3$ *when* $D(G, G') \leq M + \epsilon$*;*
2. $\Pr[\widehat{d}(G, G') < M] < 1/n^3$ *when* $D(G, G') > M + \epsilon$*.*

Theorem 2 *For any $\epsilon', \epsilon'' > 0$, there exists $N = O(\log n)$ such that for tree $T' \subseteq T$, rooted at u, where $\delta T' \subseteq \delta T$, the following are true.*

1. *For a set of adjacencies A, such that $|A \cap u| \geq \beta N$ for some $\beta \leq 1$, let*

$$\mu'_{\tilde{u}} = \mathbf{E}[|\tilde{u}(T') \cap u \cap A|].$$

Then, we have

$$\Pr[||\tilde{u}(T') \cap u \cap A| - \mu'_{\tilde{u}}| > \epsilon' \mu'_{\tilde{u}}] < 1/n^3.$$

2. $\Pr[|\tilde{u}(T') - u| > \epsilon'' N] < 1/n^3$.

4.5 Four-Point Method

We now describe the four-point method, as used in this chapter. Denote by $(a, b|c, d)$ a quartet Q such that a and b form a cherry and c and d form a cherry. Given a distance metric $\widehat{d}$ on the four leaves, the four-point method (FPM) returns the quartet topology $Q = (x, y|z, t)$ that minimizes the sum $\widehat{d}(x, y) + \widehat{d}(z, t)$ over all permutations (x, y, z, t) of (a, b, c, d). Let

$$\mathrm{ME}(\widehat{d}; x, y|z, t) = \frac{1}{4}(\widehat{d}(x, z) + \widehat{d}(x, t) + \widehat{d}(y, z) + \widehat{d}(y, t) - 2\widehat{d}(x, y) - 2\widehat{d}(z, t)).$$

If d_Q is any tree metric corresponding to the quartet topology Q, $\mathrm{FPM}(d_Q) = Q$. Let e be the length of the middle edge of Q. We also have that $\mathrm{ME}(d_Q; a, b|c, d)$ returns e. If we are given another approximate metric $\widehat{d}$ such that for all pairs $x, y \in \{a, b, c, d\}$, $|\widehat{d}(x, y) - d_Q(x, y)| < \delta$, then $|\mathrm{ME}(\widehat{d}; x, y|z, t) - e| < 2\delta$, and if $\delta < e/2$, we get that $\mathrm{FPM}(\widehat{d}) = Q$.

We note that the analysis of Theorem 1 implies that FPM on a quartet of diameter $O(\log n)$ succeeds with high probability when $N = poly(n)$. In Appendix B.1, we show how to modify the algorithm of Mihaescu et al. so that it does not use the recursive learning procedure and only calls FPM on quartets of diameter $O(\log n)$. Thus, this implies that the reconstruction is possible for $N = poly(n)$, without the ancestral learning algorithm. However, with the recursive learning algorithm, we can reduce N down to $O(\log n)$.

4.6 Estimating the true evolutionary distance

Let G and G' be genomes such that G evolves to G' according to the Nadeau-Taylor model with k evolution events. Let $A(G, G') = |G \cap G'|$ be the number of their common adjacencies. Let $\mu = \mathbf{E}[A(G, G')]$ be its expectation. There have been studies on μ, e.g., see [Wan01, Eri02]. One can compute the exact value of μ given k. Here, we are interested in the question of how close is $A(G, G')$ to μ. Given k, we will prove the following bounds on μ:

$$N\left(1 - \frac{2}{N}\right)^k \leq \mu \leq 1 + (N - 1)\left(1 - \frac{2}{N - 1}\right)^k$$

From these bounds, it can be seen that our distance

$$d(G, G') = \frac{-\log(\mu/N)}{N} \cdot \log\big(1 - (1 - 2/N)\big)$$

is approximately k/N. Hence, if $A(G, G')$ is close to μ, our distance $\widehat{d}(G, G')$ will be close to $k/N = D(G, G')$ as well.

In this section we show that $A(G, G')$ is concentrated around its mean. Hence, our approximation of k is accurate, i.e., $\widehat{d}(G, G')$ is close to $D(G, G')$ with high probability for $N = O(\log n)$. More precisely, this section is devoted to proving the following theorem.

Theorem 1. *For any $\epsilon > 0$ and $M > 0$ there exists an $N = O(\log n)$ such that for two random genomes G and G' the following are true:*

1. $\Pr[|\widehat{d}(G, G') - D(G, G')| > \epsilon] < 1/n^3$ *when* $D(G, G') \leq M + \epsilon$;
2. $\Pr[\widehat{d}(G, G') < M] < 1/n^3$ *when* $D(G, G') > M + \epsilon$.

First we compute the probabilities that a random inversion will create or break a certain adjacency. Using the notation from Wang and Warnow [WW01], we compute the probabilities of randomly destroying an arbitrary adjacency (separation) and randomly recreating a previously destroyed adjacency (unification). Given that an inversion cuts a circular permutation in two distinct places, and that there are N places to cut, the total number of possible inversions is $\binom{N}{2} = N(N-1)/2$. An adjacency can be broken by an inversion that cuts at the adjacency and at any of the other $N - 1$ places. Thus, the probability of separating an arbitrary adjacency is

$$s = \frac{N - 1}{N(N-1)/2} = \frac{2}{N}.$$

Now we consider reconstructing a broken adjacency (i, j). Assume without loss of generality that i is positive and precedes j. There are two possibilities. If j is positive, it is impossible to rejoin i and j in one inversion. If j is negative, there is precisely one inversion that brings j next to i, namely $(i + 1, j)$. Thus the unification probability for an arbitrary broken adjacency is

$$u = \frac{2}{N(N-1)}.$$

We start with the lower bound. We define the random variable Z^L to be the number of adjacencies remaining after k steps, assuming that they only break and are never reconstructed. This is clearly a lower bound on the true number of adjacencies remaining. We can compute the expected value of Z^L by examining one adjacency. At each time step, that adjacency will be broken with probability $2/N$. It survives k steps with probability $(1 - 2/N)^k$. By linearity of expectation,

$$\mu_L = \mathbf{E}\left[Z^L\right] = N\left(1 - \frac{2}{N}\right)^k.$$

To get a concentration bound is not so simple, though, because the adjacency breaking events are not independent. At any time, at most two adjacencies can be broken. Even though the events are not mutually independent, we show in Section B.4 that they have negative dependencies. Therefore, as observed by Dubhashi and Ranjan [DR98], we can still use Chernoff's bound to obtain the following result.

Lemma 1. *Let* $\mu_L = \mathbf{E}[Z^L] = N(1-2/N)^k$. *Then* $\Pr[Z^L < (1-\delta)\,\mu_L] < \exp(-\mu_L\delta^2/2)$.

Now we consider the upper bound. We define the random variable Z^H to be the number of adjacencies remaining after k steps *ignoring* the signs. The analysis of the upper bound is more involved since we must consider recreated adjacencies.

We first analyze the expected value $\mathbf{E}[Z^H]$, paying attention to how it changes over time. We consider genomes in this analysis *unsigned* circular genomes. We define events regarding the presence of adjacencies in the genomes. Because of linearity of expectation, it suffices to analyze one particular adjacency. Consider a fixed pair of consecutive genes $a = (g_i, g_{i+1})$. Let E_k be the event that there is no breakpoint at a in G_k, i.e., g_i and g_{i+1} are adjacent in G_k. The following claim calculates the probabilities $\Pr[E_k|E_0]$ and $\Pr[E_k|\overline{E}_0]$ by solving recurrences as in Wang and Warnow [WW01]. This calculation appears more detailed than necessary to analyze the expected value of Z^H, but we shall use the results again later to prove a concentration bound.

Claim. The following are true for any $k \geq 0$:

$$\Pr[E_k|E_0] = \left(1-\frac{1}{N}\right)\left(1-\frac{2}{N-1}\right)^k + \frac{1}{N} \tag{4.1}$$

$$\Pr[E_k|\overline{E}_0] = \frac{1}{N}\left(1-\left(1-\frac{2}{N-1}\right)^k\right) \tag{4.2}$$

Proof. We review bounds on creating and destroying a given adjacency. The probability that a random inversion creates a given adjacency is $\frac{2}{N(N-1)}$. Also, given that some adjacency a is present in G_k, the probability that it is destroyed is $2/N$.

As in Wang and Warnow [WW01], we analyze these probabilities using recurrences. Let $P_k = \Pr[E_k]$. We have the following recurrence:

$$P_{k+1} = P_k\left(1-\frac{2}{N}\right) + (1-P_k)\,\frac{2}{N(N-1)}$$
$$P_0 = 1$$

This recurrence reduces to

$$\begin{aligned} P_{k+1} &= \frac{2}{N(N-1)} + \left(1-\frac{2}{N}-\frac{2}{N(N-1)}\right)P_k \\ &= \frac{2}{N(N-1)} + \left(1-\frac{2}{N-1}\right)P_k \end{aligned}$$

which solves to

$$P_k = \left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N-1}\right)^k + \frac{1}{N}.$$

The analysis of $\Pr[E_k|\overline{E}_0]$ is nearly identical. We have the same recurrence, but with initial condition $P_0 = 0$. This solves to

$$P_k = \frac{1}{N}\left(1 - \left(1 - \frac{2}{N-1}\right)^k\right).$$

□

Let random variable Y_t denote the number of adjacencies between G_0 and G_t ignoring the signs. Let $Z_t^H = \mathbf{E}[Z^H|Y_t]$. Note that $Z_0, Z_1, \ldots$ is a martingale sequence. Using these two expressions derived above, we have that

$$Z_t^H = Y_t(\Pr[E_{k-t}|E_0]) + (N - Y_t)(\Pr[E_{k-t}|\overline{E}_0]),$$

which expands to

$$\begin{aligned} Z_t^H &= Y_t\left(\left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N-1}\right)^{k-t} + \frac{1}{N}\right) + (N - Y_t)\left(\frac{1}{N}\left(1 - \left(1 - \frac{2}{N-1}\right)^{k-t}\right)\right) \\ &= \left(\frac{Y_t}{N} + \frac{N - Y_t}{N}\right) + \left(Y_t\left(1 - \frac{1}{N}\right) - \frac{N - Y_t}{N}\right)\left(1 - \frac{2}{N-1}\right)^{k-t} \\ &= 1 + (Y_t - 1)\left(1 - \frac{2}{N-1}\right)^{k-t}. \end{aligned}$$

By plugging in $Y_0 = N$ at $t = 0$ in the previous expression, we get the expected value of Z^H.

Corollary.

$$\mu_H = \mathbf{E}[Z^H] = 1 + (N-1)\left(1 - \frac{2}{N-1}\right)^k.$$

We now state the concentration lemma, whose proof, following the balls and bins analysis of Kamath et al. [KMPS95], uses Azuma's inequality.

Lemma 2.

$$\Pr\left[|Z^H - \mu_H| \geq \lambda\right] \leq 2 \cdot \exp\left(-\frac{\lambda^2(N-1)}{8(N^2 - \mu_H^2)}\right).$$

Proof. We have to resort to Azuma's Inequality for martingale sequences with bounded differences. We mainly follow Kamath et al. [KMPS95]. Call the random variable we want to analyze X. Let time t denote the stage at which the first t random inversions have been applied. Let $\mathcal{F}_t$

denote all the set of events at time t. Let $X_t = E[X|\mathcal{F}_t]$ denote the conditional expectation of X at time t. Note that the sequence $X_0, X_1, \ldots X_k$ forms a martingale. If we can get the bound on $|X_t - X_{t-1}|$, we can use Azuma's inequality, stated as follows, to get the concentration bound of X.

Azuma's Inequality. *Let $X_0, X_1, \ldots$ be a martingale sequence such that for each t,*

$$|X_t - X_{t-1}| \leq c_t$$

where c_t may depend on t. Then, for all $k \geq 0$ and any $\lambda > 0$,

$$\Pr[|X_k - X_0| \geq \lambda] \leq 2 \cdot \exp\left(-\frac{\lambda^2}{2\sum_{t=1}^{k} c_t}\right).$$

Suppose that we are at time $t-1$; thus, the values of Y_{t-1} and Z_{t-1}^H are determined. To use Azuma's Inequality, we need to analyze the difference random variable $\Delta_t = Z_t^H - Z_{t-1}^H$:

$$\begin{aligned}\Delta_t &= \left(Y_t - \frac{N}{N+1}\right)\left(1 - \frac{2}{N}\right)^{k-t} - \left(Y_{t-1} - \frac{N}{N+1}\right)\left(1 - \frac{2}{N}\right)^{k-t+1} \\ &= \left(1 - \frac{2}{N}\right)^{k-t}\left(Y_t - Y_{t-1} + \frac{2}{N}Y_{t-1} - \frac{2}{N+1}\right).\end{aligned}$$

To finish, we note two things. First, $Y_{t-1} \leq N$. Second, in any one time step, we can break up to 2 adjacencies. (It is also possible, but unlikely, to recreate up to 2 adjacencies.) Thus, we can bound $|Y_t - Y_{t-1}| \leq 2$. This gives us

$$\Delta_t \leq 4 \cdot \left(1 - \frac{2}{N}\right)^{k-t}.$$

Setting $c_t = 4(1-2/N)^{k-t}$, we obtain that

$$\sum_{t=1}^{k} c_t^2 = \sum_{t=1}^{k} 16 \cdot \left(1 - \frac{2}{N}\right)^{2(k-t)} = 16 \cdot \frac{1-(1-2/N)^{2k}}{1-(1-2/N)^2} = 4 \cdot \frac{N^2 - \mu_H^2}{N-1}.$$

Plugging this into Azuma's inequality gives the desired result. □

We are ready to prove Theorem 1.

Proof of Theorem 1: Let $k = N \cdot D(G, G')$. We first consider the case where $D(G, G') \leq M + \epsilon$.

We start with the lower bound. We have $\mu_L = N(1-2/N)^k$. Using Lemma 1, we have

$$\begin{aligned}
&\Pr\left[\widehat{d}(G,G') > D(G,G') + \epsilon\right] \\
&= \Pr\left[Z = N(1-2/N)^{\widehat{d}(G,G')\cdot N} < N(1-2/N)^{(D(G,G')+\epsilon)\cdot N}\right] \\
&\leq \Pr\left[Z_L < N(1-2/N)^{(D(G,G')+\epsilon)\cdot N}\right] \\
&= \Pr\left[Z_L < N(1-2/N)^{D(G,G')\cdot N}(1-2/N)^{\epsilon\cdot N}\right] \\
&= \Pr\left[Z_L < \mu_L(1-(1-(1-2/N)^{\epsilon\cdot N}))\right] \\
&\leq \Pr\left[Z_L < \mu_L(1-(1-e^{-2\epsilon}))\right] \\
&< \exp\left(\frac{-\mu_L(1-e^{-2\epsilon})^2}{2}\right).
\end{aligned}$$

Note note that if $D(G,G') \leq M$, $\mu_L = \Omega(N)$. Therefore, the above probability is at most $1/2n^3$ for some $N = O(\log n)$.

We turn to the upper bound. We have $\mu_H = N(1-2/N)^k + 2N/(N-1)(1-(1-2/N)^k) \leq N(1-2/N)^k + 2$. Thus,

$$\begin{aligned}
&\Pr\left[\widehat{d}(G,G') < D(G,G') - \epsilon\right] \\
&= \Pr\left[Z = N(1-2/N)^{\widehat{d}(G,G')\cdot N} > N(1-2/N)^{(D(G,G')-\epsilon)\cdot N}\right] \\
&\leq \Pr\left[Z_H > N(1-2/N)^{(D(G,G')-\epsilon)\cdot N}\right] \\
&= \Pr\left[Z_H > \mu_H + N(1-2/N)^{(D(G,G')-\epsilon)\cdot N} - \mu_H\right] \\
&\leq \Pr\left[Z_H > \mu_H + N(1-2/N)^{(D(G,G')-\epsilon)\cdot N} - N(1-2/N)^{D(G,G')N} - 2\right] \\
&\leq \Pr\left[|Z_H - \mu_H| > N(1-2/N)^{D(G,G')N}((1-2/N)^{-\epsilon N} - 1) - 2\right] \\
&< 2\exp\left(-\frac{(N(1-2/N)^{D(G,G')N}((1-2/N)^{-\epsilon N} - 1) - 2)^2(N-1)}{8(N^2-\mu_H^2)}\right) \\
&= e^{-\Omega(N)} < 1/2n^3,
\end{aligned}$$

if $D(G,G') \leq M$, for some $N = O(\log n)$.

Therefore, for $N = O(\log n)$, $\Pr[|\widehat{d}(G,G') - D(G,G')| > \epsilon] < 1/n^3$, if $D(G,G') < M + \epsilon$.

The upper bound above also implies the theorem in the case where $D(G,G') > M + \epsilon$. To see this, consider another genome G'', such that $D(G,G'') = M + \epsilon$. Note that $\widehat{d}(G,G'')$ statistically dominates $\widehat{d}(G,G')$ and $\Pr[\widehat{d}(G,G'') < M] < 1/n^3$. ∎

Let $G_0 = G$, $G_k = G'$, and G_i denote G_0 after the i-th random inversions. Let random variable Z be $A(G_0, G_k)$. To show that Z stays close to the bounds above, we introduce two other random variables Z^L and Z^H defined as follows.

Let Z^L denote the number of adjacencies in G_0 which remain in all $G_1, G_2, \ldots,$ and G_k, i.e., Z^L is the number of adjacencies which have never been broken. The analysis of the expectation and variance of Z^L appears in the work of Wang [Wan02], where this model is called the *box model*.

Let Z^H denote the number of adjacencies between G_0 and G_k ignoring the signs. For example, $(1, 2)$ is considered as an adjacency in genome $(0, 1, -2, 3)$.

The expectations of Z^L and Z^H can be expressed analytically. For each adjacency $a = (x, x') \in G_0$, let an indicator random variable Z_a^L be 1 if x and x' remain adjacent in all G_i, for $0 \leq i \leq k$. Since a random inversion breaks a given adjacency with probability $2/N$, we have

$$\mathbf{E}\left[Z_a^L\right] = \Pr\left[Z_a^L = 1\right] = (1 - 2/N)^k.$$

Therefore,

$$\mathbf{E}\left[Z^L\right] = N\mathbf{E}\left[Z_a^L\right] = N(1 - 2/N)^k.$$

To analyze the upper bound, we note that a random inversion breaks a given adjacency with probability $(n-3)/\binom{N}{2} = 2/N - 2/\binom{N}{2}$, and it destroys a given breakpoint, i.e., creating an adjacency, with probability $2/\binom{N}{2}$. Later in this section, we show that this leads to

$$\mathbf{E}\left[Z^H\right] = N\left(1 - 2/N\right)^k + \frac{2N}{N-1}\left(1 - (1 - 2/N)^k\right).$$

Not only are Z^L and Z^H lower and upper bounds of μ, but also Z dominates Z^L and Z^H in turn dominates Z. For any x, $\Pr[Z \leq x] \leq \Pr[Z^L \leq x]$, and $\Pr[Z \geq x] \leq \Pr[Z^H \geq x]$. Therefore, to get concentration bounds for Z, one needs only to show that Z^L and Z^H concentrate around their means.

To prove Theorem 1, we need two concentration lemmas for Z^L and Z^H. For Z^L, we prove that the underlying random variables for Z^L are negatively dependent [DR98] and then use Chernoff's bound. For Z^H, following the balls-and-bins analysis of Kamath et al. [KMPS95], we use Azuma's inequality to show the concentration result.

4.7 Learning Ancestral Adjacencies

In this section we will show how to recover, up to an *a priori* bounded error, the set of adjacencies at the interior nodes of a phylogenetic tree from the adjacencies at the leaves of the tree, by means of a recursive algorithm.

Definition 2. Let $T = (V, E)$ be a tree rooted at ρ with boundary (leaf-set) δT.

Let T' denote a rooted binary tree with n nodes. Let B denote the tree induced by the first two top-most levels of T'. Let r denote B's root, let a, b, c and d denote its 4 leaves, and let r_1 and r_2 denote two internal nodes, where r_1 (r_2) is the parent of a and b (c and d). Let T'_u denote a subtree

of T' rooted at u. Recall that $\tilde{a}(T'_a), \tilde{b}(T'_b), \tilde{c}(T'_c)$, and $\tilde{d}(T'_d)$ denote the predicted adjacencies at nodes a, b, c, and d, respectively. Since T' is clear in this section, for brevity, we omit T' and write $\tilde{a}, \tilde{b}, \tilde{c}, \tilde{d}$, and $\tilde{r}$ for $\tilde{a}(T'_a), \tilde{b}(T'_b), \tilde{c}(T'_c), \tilde{d}(T'_d)$, and $\tilde{r}(T')$.

Our prediction uses the recursive majority algorithm on these quartets. The predicted adjacency set $\tilde{r}$ is

$$\{(x, y) : (x, y) \text{ is an adjacency in at least 3 predicted sequences at the leaves}\}.$$

With this definition, we never get inconsistency, i.e., if $(x, y) \in \tilde{r}$, for any $y' \neq y$, $(x, y') \notin \tilde{r}$. The goal is to correctly recover a large fraction of r.

Definition 3. Adjacencies in $\tilde{r} \cap r$ are *recovered* adjacencies, while adjacencies in $\tilde{r} - r$ are *accidental* adjacencies. We make a particular note for adjacencies in $\tilde{r}$ that, after being destroyed, get recreated during the evolution process. An adjacency i is *recovered in the box model* if it gets recovered even if we discard all recreated adjacencies.

To analyze $\tilde{r}$, we consider three types of adjacencies in $\tilde{r}$, (1) adjacencies recovered in the box model, (2) other recovered adjacencies, called recreated adjacencies, and (3) accidental adjacencies. Let X_b, X_r, and X_a denote the number of adjacencies of each type, respectively. Later we shall see that $\tilde{r}$ mostly consists of adjacencies in the first type.

4.7.1 Adjacencies recovered in the box model

We first analyze X_b. This gives a lower bound on the expected number of recovered adjacencies. We assume the box model. Let θ denote the lower bound on the probability that a given adjacency is preserved by the evolution process on one edge of the tree. If we assume that along the edge of the tree, at most k evolutionary events occurs, $\theta \geq (1 - 2/n)^k$. The next lemma analyzes the probability that an adjacency is recovered. It shows that for some θ it is possible to recover an adjacency with constant probability.

Lemma 3. *For some θ, there is a constant α such that, if an adjacency at node $v \in a, b, c, d$ is recovered recursively with probability at least α, an adjacency i at r is recovered with probability at least α.*

Proof. Consider some adjacency $i \in r$. We compute the probability that adjacency i is recovered. There are 2 cases.

Case 1. Adjacency i remains in all genomes at all nodes a, b, c, and d. This case occurs with probability at least θ^6; in this case, adjacency i can be recovered if it is recovered in at least 3 nodes, which occurs with probability at least $4\,\alpha^3(1 - \alpha) + \alpha^4$.

Case 2. Adjacency i remains in only three nodes, which happens with probability at least $4\,\theta^5(1 - \theta)$. In this case, adjacency i is recovered with probability at least α^4.

Combining the two cases, we have that adjacency i is recovered with probability at least

$$\alpha' = \theta^6(\alpha^4 + 4\,\alpha^3(1-\alpha)) + 4\,\theta^5(1-\theta)\alpha^3.$$

We want $\alpha' \geq \alpha$. From numerical experimentation, this is true for $\alpha = 0.94$ when $\theta > 0.988$. Therefore, for some value of the parameter θ, the expected fractions of adjacencies that we recover is at least α. □

Lemma 3 implies the following corollary.

Corollary. *For some θ, there is a constant α such that $\mathbf{E}[X_b] \geq \alpha N$.*

We not only want X_b to be large, since we use the set of learned adjacencies to estimate the closeness of two nodes in the tree, we want X_b to be near its expectation with high probability. We now show that X_b does not deviate much from $\mathbf{E}[X_b]$. For adjacency $i \in r$, let indicator random variable $W_i = 1$ if and only if adjacency i is recovered in the box model. Note that the W_i's are not independent. However, in Section B.4, we show that they are negatively dependent. As observed in Dubhashi and Ranjan [DR98], the Chernoff bound still applies. Therefore, we have the following lemma.

Lemma 4. *Let $\mu_b = \mathbf{E}[X_b]$. The following are true:*

$$\Pr[X_b < (1-\delta)\mu_b] < \exp\left(-\mu_b\delta^2/2\right);$$
$$\Pr[X_b > (1+\delta)\mu_b] < \exp\left(-\mu_b\delta^2/3\right).$$

The analyses of the numbers of accidental adjacencies and recreated adjacencies together with the proof of Theorem 2 are given in Appendix B.2.

4.8 The algorithm

Our algorithm is essentially the same as the one in Mihaescu et al. [MHR08], but, instead of using distances between pairs of binary random variables, we measure the distance between two genomes by the expected number of evolution steps between them. Here, we describe the algorithm for completeness.

We are given a model tree T, such that the mutation rate on each edge is bounded, i.e., for all $e \in T$,

$$f < D(e) < g.$$

We give an algorithm that reconstructs T if g is small enough. More specifically, our algorithm works if $g < d_0/2$, where $d_0 = 0.006$.

Let $\epsilon > 0$ be a constant such that $\epsilon < f/100$, and $2g + 6\epsilon = g' < d_0$. Let α be the constant implied by Lemma 3 for $\theta = e^{-2g'}$. Let $\beta = -0.5 \log \alpha$. Finally let $M = 10g + 6\beta + 100\epsilon$.

Let $N = O(\log n)$ as implied by Theorem 1 and Theorem 2 for $\epsilon = \epsilon' = \epsilon/4$ and $\epsilon'' = (\epsilon/4) \cdot e^{-M}$.

In what follows, we say that an event happens *with high probability* if the event occurs with probability at least $1 - O(1/n^3)$.

The following central lemma shows that with these parameters the four-point method succeeds with high probability for quartets with diameter, under D, at most M.

Lemma 5. *Let a, b, c, d be nodes in the tree T and let $Q = (a, b|c, d)$ be the subtree they induce on T. Let T_a, T_b, T_c, T_d be subtrees of T rooted at a, b, c, d respectively, such that $\delta(T_x) \subseteq \delta(T)$ for all $x \in \{a, b, c, d\}$. If the diameter of Q under D is at most M and the length of the middle edge of Q is e, then*

$$\mathrm{FPM}(\widehat{d}; \tilde{a}(T_a), \tilde{b}(T_b), \tilde{c}(T_c), \tilde{d}(T_d)) = Q$$

and

$$|\mathrm{ME}(\widehat{d}; \tilde{a}(T_a), \tilde{b}(T_b)|\tilde{c}(T_c), \tilde{d}(T_d)) - e| < 2\epsilon,$$

with high probability.

Proof. The proof uses Theorem 1 and 2 to bound the sizes of $\tilde{u} \cap \tilde{v}$ for pairs of sets of learned adjacencies. To make sure that FPM is correct, we show that $\widehat{d}(\tilde{a}, \tilde{b}) + \widehat{d}(\tilde{c}, \tilde{d})$ is less than both $\widehat{d}(\tilde{a}, \tilde{c}) + \widehat{d}(\tilde{b}, \tilde{d})$ and $\widehat{d}(\tilde{a}, \tilde{d}) + \widehat{d}(\tilde{b}, \tilde{c})$.

We show that

$$\widehat{d}(\tilde{a}, \tilde{b}) + \widehat{d}(\tilde{c}, \tilde{d}) < \widehat{d}(\tilde{a}, \tilde{c}) + \widehat{d}(\tilde{b}, \tilde{d})$$

with high probability. The case for $(a, d|b, c)$ is similar. Let $\tilde{u} = \tilde{u}(T_u)$, for $u \in \{a, b, c, d\}$.

Let $A_{uv} = u \cap v$, and $\tilde{A}_{uv} = \tilde{u} \cap \tilde{v}$ for $u, v \in \{a, b, c, d\}$. From Theorem 1, the following hold with high probability:

1. $\widehat{d}(a, b) \leq D(a, b) + \epsilon$, i.e., $|A_{ab}| \geq N(1 - 2/N)^{D(a,b)N + \epsilon N}$,
2. $\widehat{d}(c, d) \leq D(c, d) + \epsilon$, i.e., $|A_{cd}| \geq N(1 - 2/N)^{D(c,d)N + \epsilon N}$,
3. $\widehat{d}(a, c) \geq D(a, c) - \epsilon$, i.e., $|A_{ac}| \leq N(1 - 2/N)^{D(a,c)N - \epsilon N}$,
4. $\widehat{d}(b, d) \geq D(b, d) - \epsilon$, i.e., $|A_{bd}| \leq N(1 - 2/N)^{D(b,d)N - \epsilon N}$.

We derive the upper bound for $\widehat{d}(\tilde{a}, \tilde{b})$. Since $D(a, b) < M$, $|A_{ab}| = \Omega(N)$. Let $A_{\tilde{a}b} = \tilde{a} \cap A_{ab}$. We have also that

$$\mathbf{E}[|A_{\tilde{a}b}|] \geq N(1 - 2/N)^{D(a,b)N + \epsilon N} \cdot (1 - 2/N)^{D(\tilde{a},a)} = N(1 - 2/N)^{D(a,b)N + \epsilon N + D(\tilde{a},a)N}.$$

From Theorem 2, we have that

$$|A_{\tilde{a}b}| \geq (1 - \epsilon')\mathbf{E}[|A_{\tilde{a}b}|],$$

i.e.,

$$|A_{\tilde{a}b}| \geq N(1 - 2/N)^{D(a,b)N + \epsilon N + D(\tilde{a},a)N}(1 - \epsilon'),$$

for some ϵ', with high probability.

Let $A_{\tilde{a}\tilde{b}} = \tilde{b} \cap A_{\tilde{a}b}$; thus,

$$\mathbf{E}[A_{\tilde{a}\tilde{b}}] = |A_{\tilde{a}b}|(1-2/N)^{D(\tilde{b},b)N}.$$

Now applying Theorem 2 again, we have that

$$|A_{\tilde{a}\tilde{b}}| \geq (1-\epsilon')\mathbf{E}[|A_{\tilde{a}\tilde{b}}|],$$

i.e.,

$$|A_{\tilde{a}\tilde{b}}| \geq (1-\epsilon')^2 N(1-2/N)^{D(a,b)N+\epsilon N+D(\tilde{a},a)N+D(\tilde{b},b)N},$$

with high probability. This implies that

$$\begin{aligned}
|\tilde{A}_{ab}| &\geq (1-\epsilon')^2|A_{ab}|(1-2/N)^{D(a,\tilde{a})N+D(b,\tilde{b})N} \\
&\geq (1-\epsilon')^2 N(1-2/N)^{D(a,b)N+D(a,\tilde{a})N+D(b,\tilde{b})N+\epsilon N} \\
&\geq e^{-2\epsilon'} N(1-2/N)^{D(a,b)N+D(a,\tilde{a})N+D(b,\tilde{b})N+\epsilon N} \\
&\geq (1-2/N)^{2\epsilon' N} N(1-2/N)^{D(a,b)N+D(a,\tilde{a})N+D(b,\tilde{b})N+\epsilon N} \\
&= N(1-2/N)^{D(a,b)N+D(a,\tilde{a})N+D(b,\tilde{b})N+\epsilon N+2\epsilon' N},
\end{aligned}$$

where the third inequality is true because

$$e^{-2\epsilon'} \leq 1-2\epsilon'+2\epsilon'^2 \leq 1-\epsilon',$$

if $\epsilon' \leq 1/2$. Hence,

$$\begin{aligned}
\widehat{d}(\tilde{a},\tilde{b}) &\leq D(a,b)+D(a,\tilde{a})+D(b,\tilde{b})+\epsilon+2\epsilon' \\
&= D(a,b)+D(a,\tilde{a})+D(b,\tilde{b})+3\epsilon/4,
\end{aligned}$$

with high probability. A similar argument shows that

$$\widehat{d}(\tilde{c},\tilde{d}) \leq D(c,d)+D(c,\tilde{c})+D(d,\tilde{d})+3\epsilon/4,$$

with high probability.

We turn to the lower bound for $\widehat{d}(\tilde{a},\tilde{c})$. Let $A'_{\tilde{a}\tilde{c}} = A_{ac} \cap \tilde{a} \cap \tilde{c}$. By two applications of Theorem 2 as in the previous bound for $\widehat{d}(\tilde{a},\tilde{b})$, we have that

$$\begin{aligned}
|A'_{\tilde{a}\tilde{c}}| &\leq (1+\epsilon')^2 N(1-2/N)^{D(a,c)N+D(\tilde{a},a)N+D(\tilde{b},b)N-\epsilon N} \\
&\leq (1-2/N)^{-\epsilon' N} N(1-2/N)^{D(a,c)N+D(\tilde{a},a)N+D(\tilde{b},b)N-\epsilon N} \\
&= N(1-2/N)^{D(a,c)N+D(\tilde{a},a)N+D(\tilde{b},b)N-\epsilon N-\epsilon' N}.
\end{aligned}$$

Now, there might be adjacencies in $A_{\tilde{a}\tilde{c}} - A'_{\tilde{a}\tilde{c}}$, i.e., those which are accidental adjacencies in $\tilde{a}$ and $\tilde{c}$. Part (2) of Theorem 2 implies that $|A_{\tilde{a}\tilde{c}} - A'_{\tilde{a}\tilde{c}}| \leq 2\epsilon'' N \leq (\epsilon/2)N(1-2/N)^{MN}$, with high probability. Thus,

$$\begin{aligned}
|A_{\tilde{a}\tilde{c}}| &\leq N(1-2/N)^{D(a,c)N+D(\tilde{a},a)N+D(\tilde{b},b)N-\epsilon N-\epsilon' N} + (\epsilon/2)N(1-2/N)^{MN} \\
&\leq (1+\epsilon/2)N(1-2/N)^{D(a,c)N+D(\tilde{a},a)N+D(\tilde{b},b)N-\epsilon N-\epsilon' N} \\
&\leq N(1-2/N)^{D(a,c)N+D(\tilde{a},a)N+D(\tilde{b},b)N-\epsilon N-\epsilon' N-(\epsilon/2)N},
\end{aligned}$$

with high probability. i.e.,

$$\begin{aligned}
\widehat{d}(\tilde{a},\tilde{c}) &\geq D(a,c) + D(a,\tilde{a}) + D(c,\tilde{c}) - \epsilon - \epsilon' - \epsilon/2 \\
&= \widehat{d}(\tilde{a},\tilde{c}) \\
&\geq D(a,c) + D(a,\tilde{a}) + D(c,\tilde{c}) - \epsilon,
\end{aligned}$$

with high probability.

A similar argument shows that $\widehat{d}(\tilde{b},\tilde{d}) \geq D(b,d) + D(b,\tilde{b}) + D(d,\tilde{d}) - \epsilon$, with high probability.

Let e denote the middle edge of Q. Thus,

$$\begin{aligned}
\widehat{d}(a,c) + \widehat{d}(b,d) - \widehat{d}(a,b) - \widehat{d}(c,d) &\geq D(a,c) + D(b,d) - D(a,b) - D(c,d) - 4\epsilon \\
&= 2D(e) - 4\epsilon > 0,
\end{aligned}$$

if $D(e) > 2\epsilon$, which is true. Thus, $\widehat{d}(a,c) + \widehat{d}(b,d) > \widehat{d}(a,b) + \widehat{d}(c,d)$, as required.

The previous calculation also shows that $|\widehat{d}(x,y) - D(x,y)| < \epsilon$ for $x, y \in \{a,b,c,d\}$ with high probability. Therefore, $|\mathrm{ME}(\widehat{d}; \tilde{a}(T_a), \tilde{b}(T_b) | \tilde{c}(T_c), \tilde{d}(T_d)) - e| < 2\epsilon$ with high probability. □

The main algorithm maintains a forest F of subtrees of the model tree T, satisfying these two invariants: (1) All trees have edge lengths less than g', and (2) All subtrees are edge-disjoint.

The correctness and the running time of the algorithm follows from the same proof as in Mihaescu et al. [MHR08] and Lemma 5.

4.9 Implementation

To implement our algorithm for gene-order data, we reused our program for molecular data. All that is necessary is to convert the input of signed circular permutations into adjacencies. The adjacencies of a genome with N genes can be converted into character sequences of length $2N$ over an alphabet of size $2N$. From there, the molecular algorithm applies directly.

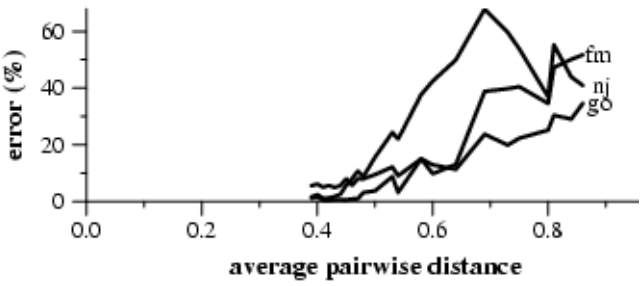


Figure 4.1: We compare neighbor joining (nj), FastME (fm), and our gene-order program (go) on 37-gene datasets. The y-axis is the percentage of internal edges a program gets wrong. The x-axis is the normalized average pairwise distance for a dataset. It is a property of the dataset, not an independent variable. We find that our program does slightly worse on low mutation datasets, but slightly better on datasets with higher average pairwise distances. Neighbor joining is the worst of the three programs in most cases.

4.10 Experimental Results

We followed the outlines of the simulation study presented in [MTWW02]. We generate 160 leave trees using `r8s` [San03] and then perturb the edge lengths by choosing a random number s in the range $[-c, c]$ and multiplying by e^s. We try values of s in 1, 2, 3, 4, and 5. The higher the number, the further we get from the ultrametric tree that is handed to us by `r8s`.

We then scale the resulting trees by multiplying each edge by a constant factor in 0.025, 0.05, 0.1, 0.2, 0.4, and 0.8. The purpose of this step is to give us trees with a wide range of inversion distances between pairs of nodes. Average pairwise distance turns out to be a good proxy for the difficulty of reconstructing a tree.

We run 3 trees for each parameter setting and generate 10 gene order datasets for each tree. We then use a Poisson model using the edge weight as the mean to pick a number of inversions and choose uniformly among them. We further ensure that every edge has at least one inversion event. Thus, the empirical tree is fully resolved.

We compare our program to distance methods using distances computed with the EDE correction [MTWW02]. We ran both the FastME algorithm [DG02], and the neighbor joining algorithm [SN87] from the PHYLIP package [Fel05].

We processed the data and sorted by our choice of c, and then sorted according to average pairwise inversion distance among the genomes (which tops out at the length of the genomes). We plot the results for c equal to 2 in Figures 4.1 and 4.2. We chose s-factor 2 since it gave the broadest range of possible branch lengths. The results for the other s-factors are similar, but not presented here. Essentially, as indicated by the graph, we do a bit worse for very low average distance (i.e., most branches have one inversion) and better for large average distance (i.e., when the evolutionary rate is quite high.)

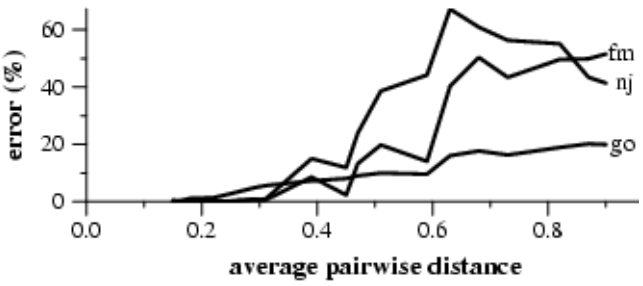


Figure 4.2: We compare neighbor joining (nj), FastME (fm), and our gene-order program (go) on 120-gene datasets. The y-axis is the percentage of internal edges a program gets wrong. The x-axis is the normalized average pairwise distance for a dataset. It is a property of the dataset, not an independent variable. We find that our program does slightly worse on low mutation datasets, but much better on datasets with higher average pairwise distances. Neighbor joining is once again the worst of the three programs, with an unexpected exception on the datasets with an average pairwise distance of 0.9 or greater.

4.11 Related Work

Distance-based methods have been widely used in phylogeny reconstruction. For gene order data, the first attempt by Blanchette, Kunisawa, and Sankoff [BKS99] applies the Neighbor Joining method to the breakpoint distance matrix. Since then, there have been several efforts to come up with better estimators for evolutionary distance than breakpoint distance or inversion distance: EDE [MWWW01], Approx-IEBP [WW01], and Exact-IEBP [Wan01]. Eriksen [Eri02] derives an analytic approximation of the expected inversion distance in terms of breakpoints. We point the reader to surveys by Moret, Tang, and Warnow [MTW05] and by Wang and Warnow [WW05].

Researchers are also interested in using other parameters beside the numbers of breakpoints or adjacencies. The concept of conserved intervals in genome rearrangements was introduced and analyzed by Bérard, Bergeron, and Chauve [BBC04]. Blin, Chauve, and Fertin [BCF05] then used conserved intervals as a distance metric in phylogeny reconstruction. Bergeron, Blanchette, Chateau, and Chauve [BBCC04] later extended these techniques to a general algorithm that reconstructs ancestral gene orders given a tree topology using conserved intervals.

References

[ADHR04] Gautam Alketar, Sandyha Dwarkadas, John P. Huelsenbeck, and Fredrik Ronquist. Parallel Metropolis-coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 20(3):407–415, February 2004.

[BBC04] Sèverine Bérard, Anne Bergeron, and Cedric Chauve. Conservation of combinatorial structures in evolution scenarios. In *Comparative Genomics*, pages 1–14, 2004.

[BBCC04] Anne Bergeron, Mathieu Blanchette, Annie Chateau, and Cedric Chauve. Reconstructing ancestral gene orders using conserved intervals. In *Proceedings of the 4th Workshop on Algorithms in Bioinformatics (WABI 2004)*, volume 3240 of *Lecture Notes in Computer Science*, pages 14–25, 2004.

[BCF05] Guillaume Blin, Cedric Chauve, and Guillaume Fertin. Genes order and phylogenetic reconstruction: Application to γ-proteobacteria. In *Comparative Genomics*, pages 11–20, 2005.

[BKS99] Mathieu Blanchette, Takashi Kunisawa, and David Sankoff. Gene order breakpoint evidence in animal mitochondrial phylogeny. *Journal of Molecular Evolution*, 49:193–203, 1999.

[Bun71] Peter Buneman. The recovery of trees from measures of dissimilarity. In D.G. Kendall and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1971.

[Cap99] Alberto Caprara. Formulations and hardness of multiple sorting by reversals. In *Proceedings of the Third Annual International Conference on Research in Computational Molecular Biology (RECOMB '99)*, pages 84–93, 1999.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1st edition, 1990.

[DG02] Richard Desper and Olivier Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 19(5):687–705, 2002.

[DMR06] Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Optimal phylogenetic reconstruction. In *Proceedings of the Thirty-Eigth Annual ACM Symposium on the Theory of Computing (STOC 2006)*, pages 159–168, 2006.

[DR98] Devdatt Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13(2):99–124, 1998.

[Eri02] Niklas Eriksen. Approximating the expected number of inversions given the number of breakpoints. In *Proceedings of the 2nd Workshop on Algorithms in Bioinformatics (WABI 2002)*, volume 2452 of *Lecture Notes in Computer Science*, pages 316–330, 2002.

[ESSW99a] Péter L. Erdös, Michael A. Steel, Lázló A. Székely, and Tandy J. Warnow. A few logs suffice to build (almost) all trees (I). *Random Structures and Algorithms*, 14:153–184, 1999.

[ESSW99b] Péter L. Erdös, Michael A. Steel, Lázló A. Székely, and Tandy J. Warnow. A few logs suffice to build (almost) all trees: part II. *Theoretical Computer Science*, 221(1–2):77–118, June 1999.

[Fel04] Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Massachusetts, 2004.

[Fel05] Joseph Felsenstein. PHYLIP (Phylogeny Inference Package) version 3.69, 2005. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

[FG82] Leslie R. Foulds and Ronald L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[Fit71] Walter M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

[GG03] Stéphane Guindon and Olivier Gascuel. A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52(5):696–704, 2003.

[Har73] John A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29:53–65, 1973.

[HR01] John P. Huelsenbeck and Fredrik Ronquist. MrBayes: Bayesian inference of phylogeny. *Bioinformatics*, 17(8):754–755, August 2001.

[JC69] Thomas H. Jukes and Charles R. Cantor. Evolution of protein molecules. In H. N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, 1969.

[JDP83] Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, 11(1):286–295, 1983.

[Kim80] Motoo Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120, December 1980.

[Kim81] Motoo Kimura. Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 78(1):454–458, January 1981.

[KMPS95] Anil Kamath, Rajeev Motwani, Krishna Palem, and Paul Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Random Structures and Algorithms*, 7(1):59–80, 1995.

[MHR08] Radu Mihaescu, Cameron Hill, and Satish Rao. Fast phylogeny reconstruction through learning of ancestral sequences. *CoRR*, `arxiv:abs/0812.1587`, 2008.

[Mos04] Elchanan Mossel. Phase transitions in phylogeny. *Transactions of the American Mathematical Society*, 356(6):2379–2404, 2004.

[MTW05] Bernard M. E. Moret, Jijun Tang, and Tandy Warnow. Reconstructing phylogenies from gene-content and gene-order data. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*, pages 321–352. Oxford University Press, 2005.

[MTWW02] Bernard M. E. Moret, Jijun Tang, Li-San Wang, and Tandy Warnow. Steps toward accurate reconstruction of phylogenies from gene-order data. *Journal of Computer and System Sciences*, 65(3):508–525, 2002.

[MWWW01] Bernard M. E. Moret, Li-San Wang, Tandy Warnow, and Stacia K. Wyman. New approaches for reconstructing phylogenies from gene order data. *Bioinformatics*, 17(1):165–173, 2001.

[NT84] Joseph H. Nadeau and Benjamin A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. In *Proceedings of the National Academy of Sciences USA*, volume 81, pages 814–818, 1984.

[PS98] Itsik Pe'er and Ron Shamir. The median problems for breakpoints are NP-complete. Technical Report TR98-071, The Electronic Colloquium on Computational Complexity, 1998.

[RH03] Fredrik Ronquist and John P. Huelsenbeck. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574, August 2003.

[San03] Michael J. Sanderson. r8s: Inferring absolute rates of molecular evolution and divergence times in the absence of a molecular clock. *Bioinformatics*, 29(2):301–302, January 2003.

[SN87] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: A new method, for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[SS03] Charles Semple and Mike Steel. *Phylogenetics*. Oxford University Press, Oxford, 2003.

[Sta04] Alexandros Stamatakis. *Distributed and Parallel Algorithms and Systems for Inference of Huge Phylogenetic Trees based on the Maximum Likelihood Method*. PhD thesis, Technische Universität München, Germany, October 2004.

[Sta06] Alexandros Stamatakis. RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, November 2006.

[Tav96] Simon Tavaré. Some probabilistic and statistical problems in the analysis of DNA sequences. In Robert M. Miura, editor, *Some Mathematical Questions in Biology: DNA Sequence Analysis*, volume 17 of *Lectures on Mathematics in the Life Sciences*, pages 57–86. American Mathematical Society, 1996.

[Wan01] Li-San Wang. Exact-IEBP: A new technique for estimating evolutionary distances between whole genomes. In *Proceedings of the First Workshop on Algorithms in Bioinformatics (WABI 2001)*, volume 2149 of *Lecture Notes in Computer Science*, pages 175–188, 2001.

[Wan02] Li-San Wang. Genome rearrangement phylogeny using Weighbor. In *Proceedings of the 2nd Workshop on Algorithms in Bioinformatics (WABI 2002)*, volume 2452 of *Lecture Notes in Computer Science*, pages 112–125, 2002.

[WW01] Li-San Wang and Tandy Warnow. Estimating true evolutionary distances between genomes. In *Proceedings of the Thirty-Third Annual ACM Symposium on the Theory of Computing*, pages 637–646, 2001.

[WW05] Li-San Wang and Tandy Warnow. Distance-based genome rearrangement phylogeny. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*. Oxford University Press, 2005.

[Zwi06] Derrick J. Zwickl. *Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion*. PhD thesis, The University of Texas at Austin, 2006.

Appendix A

Pseudocode

```
ST-QUARTET(V)
 1  E = ∅
 2  for each vertex v ∈ V
 3       E = E ∪ {(v, v)}
 4  for i = 1 to |V| − 1
 5       for each edge (u1, v1) ∈ E
 6            for each edge (u2, v2) ∈ E
 7                 if FIND-SET(u1) ≠ FIND-SET(u2)
 8                      cost = EST-ME(u1, v1, u2, v2)
 9                      if cost < best
10                           best = cost
11                           (x1, y1, x2, y2) = (u1, v1, u2, v2)
12       (u1, v1) = WALK-TREE(x1, y1, x2, y2)
13       (u2, v2) = WALK-TREE(x2, y2, x1, y1)
14       JOIN-TREES(u1, v1, u2, v2)

JOIN-TREES(u1, v1, u2, v2)
1  x = SPLIT(u1, v1)
2  y = SPLIT(u2, v2)
3  JOIN(x, y)
4  E = E ∪ {(x, y)}
5  LEARN(u1, v1, u2, v2, x, y)

SPLIT(u, v)
1  x = NEW-NODE()
2  E = E \ {(u, v)}
3  E = E ∪ {(u, x), (x, v)}
4  return x
```

```
LEARN(a, b, c, d, x, y)
 1  EST-ME(a, b, c, d, x, y)
 2  // Learn sequences on x and y.
 3  LEARN-SEQ(x → y)
 4  LEARN-SEQ(y → x)
 5  // Learn edge lengths surrounding the quartet.
 6  LEARN-EDGE(a, x)
 7  LEARN-EDGE(b, x)
 8  LEARN-EDGE(c, y)
 9  LEARN-EDGE(d, y)
10  // Relearn sequences on x and y.
11  LEARN-SEQ(x → y)
12  LEARN-SEQ(y → x)
13  // Propagate new sequence data down both sides.
14  PROPAGATE(x, y)
15  PROPAGATE(y, x)

PROPAGATE(u, parent)
1  for v ∈ Adj[u]
2      if v ≠ parent
3          LEARN-SEQ(u → v)
4          PROPAGATE(v, u)

WALK-TREE(u, v, e)
1  x = DIR(u, e)
2  y = DIR(v, e)
3  if x = v and y ≠ u
4      return WALK-TREE(v, y, e)
5  if y = u and x ≠ v
6      return WALK-TREE(x, u, e)
7  return (x, y)
```

Appendix B

Technical Lemmas

B.1 Reconstruction for polynomial-length data without ancestral gene order learning

The analysis of Theorem 1 (see Section 4.6) implies that FPM on a quartet of diameter $O(\log n)$ succeeds with high probability when $N = poly(n)$. Note that the algorithm of Mihaescu et al. can be modified so that it does not use the recursive learning procedure and we only call FPM on quartets of diameter $O(\log n)$. The only subroutine that uses the learning algorithm is $\mathrm{Dir}(a, c, T', T'')$, which returns the node paired with $\tilde{c}$ in the quartet split $\mathrm{FPM}(\widehat{d}; \tilde{x}, \tilde{y}, \tilde{z}, \tilde{c})$ where $\tilde{x}, \tilde{y}, \tilde{z}$ are the learned sets of adjacencies on the neighbors of a using their edge-disjoint maximal subtrees T'_x, T'_y, T'_z of T'. Now, instead of using $\tilde{w}$, for $w \in \{x, y, z\}$, we choose some leaf w' of T'_w, with minimum depth. The distance from a to w' is at most $O(\log n)$, because the minimum degree of every interior node in T' is at least 3. For $\tilde{c}$, we choose some leaf c' of T'' such that the path from c to c' in T'' contains at most $O(\log n)$ edges. Again, c' exists because the degree of every interior node of T'' is 3. Therefore, the diameter of the quartet for FPM is bounded by $O(\log n)$, and FPM is correct with high probability. Function Dir returns the neighbor w of a such that w' is paired up with c' in the quartet split returned by FPM. Thus, this implies that the reconstruction is possible for $N = poly(n)$.

B.2 Proofs of the recursive learning procedure

B.2.1 Accidental adjacencies

In this section we analyze random variable X_a, the number of accidental adjacencies. With an analysis similar to the one in Lemma 3, we can show that $\mathbf{E}[X_a] = O(1)$. However, we want a high probability bound.

To simplify the analysis, we consider genomes as unsigned. Multiplying the bounds here by 4

gives bounds for signed genomes.

Let $\mathcal{A}$ denote the set of all unsigned adjacencies. Let $\mathcal{A}'$ denote the set $\mathcal{A} - r$. Let $N' = |\mathcal{A}'| = \binom{N}{2} - N$ denote the number of possible adjacencies not in r. We are interested in the number of common accidental adjacencies created by independent evolutionary processes on two or more edges in the tree.

Let kN denote the upper bound on the number of evolution events on any edge. We now show that in order to analyze X_a, it is enough to consider a random set of size $2kN$ from $\mathcal{A}'$ as the accidental adjacencies on any edge. Since we want to get an upper bound on the number of accidental adjacencies, we consider all adjacencies in $\mathcal{A}'$ that are created during the process; call this set B. Note that it is possible that an evolution event preserves all current adjacencies, since it might act upon a single gene x, thus flipping its sign, and the result is not observable when we treat genomes as unsigned. In that case, we say that the two adjacencies containing x are created as well.

Each application of a random reversal creates at most 2 new adjacencies; therefore, at most $2kN$ adjacencies are created. We consider their distribution. First note that in any evolution step, a pair of created adjacencies completely determines each other. But between two evolution steps, the choices are independent. Thus, we can divide the set of created adjacencies B into two sets B_1 and B_2, such that each B_i is a random subset of size at most kN, but B_1 and B_2 completely determine each other. The worst dependency to our analysis occurs when B_1 and B_2 are disjoint. Therefore in the worst case, the set B is a random subset of $\mathcal{A}'$ of size $2kN$.

We are ready to prove the lemma stating that with high probability, $X_a = O(\log n)$.

Lemma 6. *If $N = \Omega(\log n)$, there exists an evolution rate k such that if the number of evolution steps on each edge is at most kN, the set of accidental adjacencies in $\tilde{r}$ is of size $O(\log n)$ with probability at least $1 - 1/n^3$.*

Proof. Recursively, we assume that for each leaf, the learning algorithm returns at most $c \log n$ accidental adjacencies, for some c. As discussed above, we can assume that the set of accidental adjacencies is a random subset of $\mathcal{A}'$ of size $c \log n$. Since $N' = |\mathcal{A}'| = \binom{N}{2} - N$, we have $N' = \Omega(\log^2 n)$.

To get an upper bound on the number adjacencies, we count the number of adjacencies that (1) appear as accidental adjacencies on at least two of the leaves or (2) are created by the evolution process on at least 2 of the 6 edges in this level of recursion. One can verify that any accidental adjacencies returned fall into at least one of these two groups.

Case 3 (1). We analyze the number of common accidental adjacencies returned by two of the leaves. The probability that, for a fixed pair of leaves, the number of common accidental adjacencies is at least l is at most

$$\binom{c \log n}{1} \frac{\binom{N'-l}{c \log n - l}}{\binom{N'}{c \log n}} \leq \frac{(c \log n)^{2l}}{l!(N'-l)^l} = \frac{1}{l!} \left(\frac{c^2 \log^2 n}{N' - l} \right)^l < \frac{1}{12n^3},$$

when $l > c' \log n$, for some c', and $N = \Omega(\log n)$. Therefore, the number of accidental adjacencies in this case is at least $\binom{4}{2} l = 6l$ with probability at most $1/2n^3$.

Case 4 (2). We consider accidental adjacencies which have been created on at least two edges of the tree in this level of recursion. Fix two edges e_1 and e_2, and consider common adjacencies created on these two edges. Denote the sets of adjacencies created on e_1 by B_1 and on e_2 by B_2. Now, B_1 and B_2 are random subsets of size $2kN$ from $\mathcal{A}'$ of size $N' = \binom{N}{2} - N$. For any given B_1 of size at most $2kN$, the probability that $B_1 \cap B_2$ is of size at least p is

$$\binom{2kN}{p} \frac{\binom{N'-p}{2kN-p}}{\binom{N'}{2kN}} \leq \frac{(2kN)^{2p}}{p!(N'-p)^p} = \frac{1}{p!} \left(\frac{4k^2N^2}{N'-p} \right)^p .$$

The term $\frac{4k^2N^2}{N'-p}$ is less than 1 if $k < 1/2\sqrt{2}$ and N is larger than some constant independent of p. Note that $p! > 2^p$ if $p > 4$; therefore setting $p = O(\log n)$ suffices to make the above probability less than $1/30n^3$. Since there are $\binom{6}{2} = 15$ possible pairs of edges, using a union bound, we conclude that there are more than $15p = O(\log n)$ with probability at most $1/2n^3$.

Combining the two cases, we have that the number of accidental adjacencies is more than $l + p = O(\log n)$ with probability at most $1/n^3$. □

B.2.2 Recreated adjacencies

In this section, we consider other learned adjacencies, i.e., those that are already in r, but if we discard all recreated adjacencies we would not recover it. Note that they are not considered by Lemma 4. We denote by $S \subseteq \tilde{r}$ the set of these adjacencies, and by X_r the number of them, i.e., the size of S. We call the adjacencies that remain at some leaf a, b, c, or d of B and are not destroyed by the evolution process on the edges of B *clean adjacencies*.

Again, to simplify the analysis, we consider all genomes as unsigned. We follow the same idea as in Section B.2.1. We first argue about the size and distribution the set of newly created adjacencies on any edge.

Lemma 7. *For N large enough, there exists a constant k such that for any edge where the number of evolution steps is at most kN, the number of recreated adjacencies is more than $c_1 \log n$ with probability at most $O(n^{-3})$, and the set of recreated adjacencies is dominated by a random subset of size $c_1 \log n$, for some constant c_1.*

Proof. There can be at most $2kN$ of them, if kN is the maximum number of evolution steps on any edge. For any adjacency to be destroyed and recreated, it must (1) belong to one of the $2kN$ adjacencies to be destroyed, and (2) get recreated, i.e., be among the $2kN$ created adjacencies.

We now look at the evolution process on an edge more closely. In the first evolution step, there are $N' = \binom{N}{2} - N$ adjacencies that can be created, none of them being candidates of recreated adjacencies. In general, in the j-th step, if $l < 2j$ adjacencies have been destroyed, there are at most $N' + l$ possible adjacencies that can be created, l of which are candidates for recreated

adjacencies. Now, as in the discussion about accidental adjacencies, for our purpose of proving the upper bound on the intersections of these sets, we can assume that the set of recreated adjacencies are generated as follows. There are N' adjacencies with $2kN$ candidates. We pick a random subset S' of size $2kN$ and the set of candidates in S' for the set of recreated adjacencies.

The probability that the number of recreated adjacencies is at least p is at most

$$\binom{2kN}{p}\frac{\binom{N'-p}{2kN-p}}{\binom{N'}{2kN}} \leq \frac{(2kN)^{2p}}{p!(N'-p)^p} = \frac{1}{p!}\left(\frac{4k^2N^2}{N'-p}\right)^p \leq \frac{1}{n^3},$$

if $k < 1/2\sqrt{2}$, N is larger than some constant independent of p, and $p = c_1 \log n$, for some c_1. The lemma thus follows. □

We now consider the learning process, and prove the bound on the size of S.

Lemma 8. *For N large enough, there exists a constant k such that if kN is the upper bound on the number of evolution steps on any edge, $X_r = O(\log n)$ with probability at least $1 - n^{-3}$.*

Proof. For each learned adjacency in S, either it appears as a clean adjacency in at least 3 leaves, or it gets recreated in one of the edges in this level. For adjacencies in the later case, Lemma 7 implies that there are at most $6 \cdot c_1 \log n$ with probability at most $O(1/n^3)$, since there are at most 6 edges in this level of recursion.

We turn to the former case. Assume that the recursive learning algorithm on each subtree returns at most $c' \log n$ recreated adjacencies. We prove the lemma by a detailed case analysis. There are 2 cases.

Case 5 (1). We consider adjacencies that get recreated in other recursive levels, but appear as clean adjacencies in less than two of the leaves. They must be recovered as accidental adjacencies in some of the leaves.

Lemma 6 states that there are at most $c_a \log n$ accidental adjacencies, for some c_a, with high probability. But note that this set can take on any subset of size $c_a \log n$ from a set of almost all adjacencies, while the adjacencies that we want are those destroyed by the evolution process. The probability that this set contains at least l adjacencies from r is at most

$$\binom{4kN}{l}\frac{\binom{N'-l}{c_a \log n - l}}{\binom{N'}{c_a \log n}} \leq \frac{(4kN \cdot c_a \log n)^l}{l!(N'-l)^l} \leq \frac{1}{n^3},$$

if $l = \Omega(\log n)$ when N is larger than some constant independent of l, because $N' = \Theta(N^2)$. Therefore, the number of recreated adjacencies of this type is more than $\binom{4}{2} \cdot 2 \cdot l = O(\log n)$ with probability at least $1 - O(1/n^3)$.

Case 6 (2). We consider the set of recreated adjacencies whose copies appear as clean adjacencies in at least three of the leaves. There are two subcases.

Subcase 2a: We consider the case of adjacencies which are returned as recreated in exactly one of the leaves.

Consider one leaf, say a. Let S_a denote the set of clean adjacencies in a which are returned as recreated adjacencies by the learning algorithm on T'_a. Let $W = |S_a|$. From the inductive hypothesis, we have that $W \leq c' \log n$ with high probability.

For $v \in \{r_2, a, b, c, d\}$, let θ_v denote the probability that an adjacency in the parent of v remains a clean adjacency at node v. Also, for leaf v, we denote by α_v the probability that a given adjacency is recovered in the box model. Let C_a denote the number of adjacencies in S_a such that it is recovered in the box model in exactly two other leaves. We have

$$\mathbf{E}[C_a] = W\Big(\sum_{\substack{x,y,z \in \{b,c,d\} \\ x \neq y \neq z}} \theta_{r_2}\theta_x\theta_y\alpha_x\alpha_y\big((1-\theta_z) + \theta_z(1-\alpha_z)\big)\Big).$$

By numerical experimentation in Mathematica, $\mathbf{E}[C_a] < 0.1845W < W/4$ when $\theta_v \geq 0.988$, for $v \in \{r_2, b, c, d\}$, and $\alpha_v \geq 0.91$, for $v \in \{b, c, d\}$. Note that C_a is a sum of negatively dependent indicator random variables, as argued in Lemma 9; thus, we can apply a Chernoff bound to prove that there exists a constant $\epsilon > 0$, where $0.1845 < 1/4 - \epsilon$, such that $C_a \leq (1/4 - \epsilon)W$ with probability at least $1 - 1/n^3$, if N is large enough.

Therefore, considering all four leaves, the number of recreated adjacencies in this case is at most $4 \cdot (1/4 - \epsilon)c' \log n < (1 - 4\epsilon)c' \log n$.

Subcase 2b: We consider the case of adjacencies which are returned as recreated adjacencies in at least two of the leaves. The probability that the number of common recreated adjacencies is at least l is

$$\binom{c' \log n}{l} \frac{\binom{N-l}{c' \log n - l}}{\binom{N}{c' \log n}} \leq \left(\frac{ec' \log n}{l}\right)^l \frac{(c' \log n)^l}{(N-l)^l} < \frac{1}{n^3},$$

if N is larger than some constant independent of l and $l = \Omega(\log n)$, since $N = \Theta(\log n)$.

Combining all cases, we have that with probability at least $1 - O(1/n^3)$ the number of recreated adjacencies returned from this level is at most $(1 - 4\epsilon)c' \log n + O(\log n)$, for some $\epsilon > 0$. If the constant hidden in the big-O term is c_2, we can set $c' = c_2/4\epsilon$ so that $(1 - 4\epsilon)c' \log n + c_2 \log n \leq c' \log n$, and the lemma follows. □

B.3 Concentration result for learned adjacencies

We prove the following theorem regarding the concentration of the size of the learned adjacency sets to its mean.

Theorem 2. *For any $\epsilon', \epsilon'' > 0$, there exists $N = O(\log n)$ such that for tree $T' \subseteq T$, rooted at u, where $\delta T' \subseteq \delta T$,*

(1) For a set of adjacencies A, such that $|A \cap u| \geq \beta N$ for some $\beta \leq 1$, let

$$\mu'_{\tilde{u}} = \mathbf{E}[|\tilde{u}(T') \cap u \cap A|].$$

Then, we have

$$\Pr[||\tilde{u}(T') \cap u \cap A| - \mu'_{\tilde{u}}| > \epsilon' \mu'_{\tilde{u}}] < 1/n^3.$$

(2) $\Pr\left[|\tilde{u}(T') - u| > \epsilon'' N\right] < 1/n^3$.

Proof. We start with (2). From Lemma 6, the number of accidental adjacencies is $O(\log n) < c' \log n$, for some c', with high probability if N is large enough. We can thus set $N > c'/\epsilon'' \log n = O(\log n)$ to get the bound.

We then prove (1). Assume without loss of generality that $A \subseteq u$. Recall that $|\tilde{u}(T') \cap u| = X_b + X_r$, where X_b is the number of adjacencies recovered in the box model and X_r is the number of recreated adjacencies recovered. Let random variable $X_A = |\tilde{u}(T') \cap u \cap A|$. The first step is to bound the contribution of X_b and X_r to X_A.

Let X'_b denote the number of adjacencies in $A \cap u$ recovered in the box model. Since the analysis of X_b uses indicator random variables, we can modify it to get the following. Let $\mu'_b = \mathbf{E}[X'_b]$. We have that $\mu'_b > 0.94|A|$, if the mutation rate is low enough, and $\Pr[|X'_b - \mu'_b]| > \epsilon N] < e^{-\epsilon^2 N/2\alpha}$, where $\alpha = \mathbf{E}[X'_b]/N$.

Since $X_r < c'' \log n$ for some c'' with high probability, if N is large enough, we have that $\mathbf{E}[X_r] \leq c'' \log n + N/n^3 \leq (c''+1) \log n$, because $N = O(\log n)$. However, we also have that $\mathbf{E}[X_A] \geq \mathbf{E}[X'_b] \geq 0.94|A| \geq 0.94\beta N$. Therefore, if we set $N > (4(c''+1)/(0.94\epsilon'\beta)) \log n = O(\log n)$, we get that $\mathbf{E}[X_r] \leq (\epsilon'/2)\mathbf{E}[X_A]$, and $\mathbf{E}[X'_b] \geq (1 - \epsilon'/2)\mathbf{E}[X_A]$.

Applying the concentration bound for X'_b discussed above proves the theorem. □

B.4 Negative dependencies in the box model

Definition 4. Random variables $Y_1, \ldots, Y_N$ are *negatively associated* if for every pair of disjoint sets $I, J \subset [N]$,

$$\mathbf{E}\left[f(Y_i, i \in I) g(Y_j, j \in J)\right] \leq \mathbf{E}\left[f(Y_i, i \in I)\right] \mathbf{E}\left[g(Y_j, j \in J)\right]$$

for all functions f and g which are both non-decreasing or both non-increasing.

The following lists a few facts regarding negatively associated random variables.

Proposition 1 (Joag-Dev and Proschan [JDP83])**.**

1. *Let $Y_1, Y_2, \ldots, Y_n$ and $Z_1, Z_2, \ldots, Z_m$ be two set of negatively associated random variables. If $(Y_1, \ldots, Y_n)$ and $(Z_1, \ldots, Z_m)$ are mutually independent, then the augmented vector $(Y_1, \ldots, Y_n, Z_1, \ldots, Z_m)$ are negatively associated.*

2. *Let $Y_1, Y_2, \ldots, Y_n$ be negatively associated random variables. Let $I_1, \ldots, I_k \subseteq [n]$[1] be disjoint index sets, for some k. For $j \in [k]$, let h_j be functions that are all non-decreasing or all non-increasing, and let $Z_j = h_j(Y_i, i \in I_j)$. Then vector $(Z_1, \ldots, Z_k)$ are negatively associated.*

[1] We denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

We consider the balls-and-bins experiment with N bins where two balls are thrown into different bins at the same time. This is an extension of results in Section 2.2 of [DR98], which considers the case where one ball is thrown in each round. For $i \in [N]$, let B_i be an indicator random variable which is 1 if one of the balls falls into bin i, and 0 otherwise.

Lemma 9. *Random variables $B_1, \ldots, B_N$ are negatively associated.*

Proof. Consider two subsets $I, J \subseteq [N]$, and non-decreasing functions $f : \mathbb{R}^{|I|} \to \mathbb{R}$, and $g : \mathbb{R}^{|J|} \to \mathbb{R}$. Let random variable $F = f(Y_i, i \in I)$ and $G = g(Y_i, i \in J)$. We want to show that $\mathbf{E}[FG] \leq \mathbf{E}[F]\mathbf{E}[G]$.

Since we can replace $f(x_1, x_2, \ldots)$ with $f(x_1, x_2, \ldots) - f(0, 0, \ldots)$ and $g(y_1, y_2, \ldots)$ with $g(y_1, y_2, \ldots) - g(0, 0, \ldots)$, we can assume that $f(0, 0, \ldots) = 0 = g(0, 0, \ldots)$.

Let event D be the event that one ball falls into some bin $i \in I$ and another ball falls into some bin $j \in J$. Let event E_I denote the event that at least one ball falls in to bins in I, and let event E_J denote the event that at least one ball falls into bins in J.

First, we consider $\mathbf{E}[FG]$. We have

$$\begin{aligned}\mathbf{E}[FG] &= \mathbf{E}[FG|D]\Pr[D] + \mathbf{E}[FG|\bar{D}]\Pr[\bar{D}] \\ &= \mathbf{E}[FG|D]\Pr[D],\end{aligned}$$

since $\mathbf{E}[FG|\bar{D}] = 0$. Because random variables F and G are independent given D,

$$\mathbf{E}[FG] = \mathbf{E}[F|D]\mathbf{E}[G|D]\Pr[D].$$

We write

$$\begin{aligned}\mathbf{E}[F] &= \mathbf{E}[F|E_I]\Pr[E_I] + \mathbf{E}[F|\bar{E}_I]\Pr[\bar{E}_I] \\ &= \mathbf{E}[F|E_I]\Pr[E_I] \\ &\geq \mathbf{E}[F|D]\Pr[E_I],\end{aligned}$$

where the last inequality follows from the fact that f is non-decreasing. Similarly, we have $\mathbf{E}[G] \geq \mathbf{E}[G|D]\mathbf{E}[E_J]$.

The lemma follows if

$$\begin{aligned}\mathbf{E}[FG] &= \mathbf{E}[F|D]\mathbf{E}[G|D]\Pr[D] \\ &\leq \mathbf{E}[F|D]\mathbf{E}[G|D]\Pr[E_I]\Pr[E_J] \\ &\leq \mathbf{E}[F]\mathbf{E}[G]\end{aligned}$$

i.e., if $\Pr[D] \leq \Pr[E_I]\Pr[E_J]$.

We have that

$$\begin{aligned}
\Pr[D] &= \frac{2|I||J|}{N(N-1)}, \\
\Pr[E_I] &= \frac{|I|\,(N-|I|) + |I|\,(|I|-1)\,/2}{\binom{N}{2}} \\
&= \frac{2N|I| - 2|I|^2 + |I|^2 - |I|}{N(N-1)} \\
&= |I|\frac{2N-|I|-1}{N(N-1)}, \\
\Pr[E_J] &= |J|\frac{2N-|J|-1}{N(N-1)}.
\end{aligned}$$

Thus,

$$\Pr[E_I]\Pr[E_J] = \frac{|I||J|(2N-|I|-1)(2N-|J|-1)}{N^2(N-1)^2}.$$

Therefore, it remains to check that

$$\frac{(2N-|I|-1)(2N-|J|-1)}{2N(N-1)} = \frac{4N^2 - 2N(|I|+|J|) - 4N + |I||J| + |I| + |J| + 1}{2N^2 - 2N} \geq 1.$$

Note that worst case occurs when $|I| + |J| = N$, so we assume that; thus, we need to check if $|I||J| + 1 = |I|(N-|I|) + 1 \geq N$ for $|I| = 1, 2, \ldots, N-1$, which is true. □

For each bin $1 \leq i \leq N$, let indicator random variable W_i be 1 if bin i is empty and 0 otherwise. With Lemma 9 and Proposition 1 in hand, we can show that the W_i's are negatively associated. For each evolution step s on the tree, we have a vector of indicator random variables $(B_1^s, \ldots, B_N^s)$, such that $B_i^s = 1$ if adjacency i is destroy in that step, and 0 otherwise. From Lemma 9, these random variables are negatively associated. We can concatenate all random vectors for all evolution steps on the tree; the resulting vector $\mathcal{B}$ is also negatively associated, from part 1 of Proposition 1. From $\mathcal{B}$, we can determine W_i, using only random variables associated with bin i. Therefore, random variables $W_1, \ldots, W_N$ are functions of disjoint subsets of negatively associated random variables, which are negatively associated by part 2 of Proposition 1.

Appendix C

Program Options

C.1 Neighbor Joining

For neighbor-joining, we used PHYLIP's `neighbor` program with default options.

C.2 RAxML

RAxML, version 7.0.4, was run with the following command:

```
raxmlHPC -m GTRGAMMA
```

C.3 DNAML

DNAML, version 3.69, was run with "Transition/transversion" ratio set to 1.0000 and "Speedier but rougher analysis" off (see Figure C.1).

```
Nucleic acid sequence Maximum Likelihood method, version 3.69

Settings for this run:
  U                 Search for best tree?  Yes
  T        Transition/transversion ratio:  1.0000
  F       Use empirical base frequencies?  Yes
  C                One category of sites?  Yes
  R           Rate variation among sites?  constant rate
  W                       Sites weighted?  No
  S        Speedier but rougher analysis?  No, not rough
  G                Global rearrangements?  No
  J   Randomize input order of sequences?  No. Use input order
  O                        Outgroup root?  No, use as outgroup...
  M           Analyze multiple data sets?  No
  I          Input sequences interleaved?  Yes
  0   Terminal type (IBM PC, ANSI, none)?  ANSI
  1    Print out the data at start of run  No
  2  Print indications of progress of run  Yes
  3                        Print out tree  Yes
  4       Write out trees onto tree file?  Yes
  5   Reconstruct hypothetical sequences?  No
```

Figure C.1: Settings for dnaml.