Examining Phylogenetic Reconstruction Algorithms

Evan Albright
ichbinevan@gmail.com

Jack Hessel
jmhessel@gmail.com

Nao Hiranuma
hiranumn@gmail.com

Cody Wang
mcdy143@gmail.com

Abstract

Understanding the evolutionary relationships between organisms by comparing their genomic sequences is a focus of modern-day computational biology research. Estimating evolutionary history in this way has many applications, particularly in analyzing the progression of infectious, viral diseases. Phylogenetic reconstruction algorithms model evolutionary history using tree-like structures that describe the estimated ancestry of a given set of species. Many methods exist to infer phylogenies from genes, but no one technique is definitively better for all types of sequences and organisms. Here, we implement and analyze several popular tree reconstruction methods and compare their effectiveness on both synthetic and real genomic sequences. Our synthetic data set aims to simulate a variety of research conditions, and includes inputs that vary in number of species. For our case-study, we use the genes of 53 apes and compare our reconstructions against the well-studied evolutionary history of primates. Though our implementations often represent the simplest manifestations of these complex methods, our results are suggestive of fundamental advantages and disadvantages that underlie each of these techniques.

Introduction

Phylogenetic Trees

A phylogenetic tree, commonly referred as a tree of life, is a widely accepted representation of evolutionary relationship among species on the Earth [2]. The concept of a tree of life was first introduced by Charles Darwin in 1836 after his trip to Galapagos islands (Figure 1). Each node in a tree is called a taxonomic unit, or "taxa," for short. Internal nodes in a tree represent most common ancestors of their direct child nodes. The length of a branch in a phylogenetic tree is an indication of evolutionary distance. Depending on the particular reconstruction method used, branch length usually indicates either the estimated time it took for one species to evolve into another species, or the genetic distance between a pair of an ancestor and its descendant. We are particularly interested in bifurcating phylogenetic trees, meaning an ancestor can only have two direct descendants.

Phylogenetic trees are useful not only for describing the evolutionary history of multiple species but also for solving other real world problems. For instance, phylogenetic analysis of a virus can sometime help us track down the source of infectious, viral diseases such as SARS [27]. Phylogenetic trees are also used to find natural sources of new drugs or to develop effective treatments against diseases that are hard to cure [34]. Reconstruction also allows us to make predictions about poorly understood or extinct species. All these applications are dependent on our ability to reconstruct phylogenetic trees from information available to us.

Here, we implement, apply, and compare several popular multiple sequence alignment algorithms (MSAs) and phylogenetic reconstruction methods. Because these algorithms are so commonly applied, state-of-the-art implementations for each exist. We acknowledge our programs lack the nuance and optimizations found in these refined and expert versions. However, we argue that results derived from our bare-bones implementations reflect basic advantages and disadvantages that underlie each method.

All phylogenetic algorithms examined by our group can be labeled as "cladistics." Cladistic methods attempt to identify relationships based on shared, inherited characteristics amongst individuals. Cladisitcs group organisms using similarities derived from common ancestors and splitting events, with evolutionary history in mind. In contrast to cladistic methods, "phenetic" methods shift their emphasis to morphological similarities independent of ancestry. Generally, phenetic methods are not considered to be state-of-the-art and we do not consider them further here.

For our study, we execute our algorithms on both synthetic data and real-world genetic sequences. Synthetic data is useful because the underlying evolutionary history can be known completely. We implement a data generator

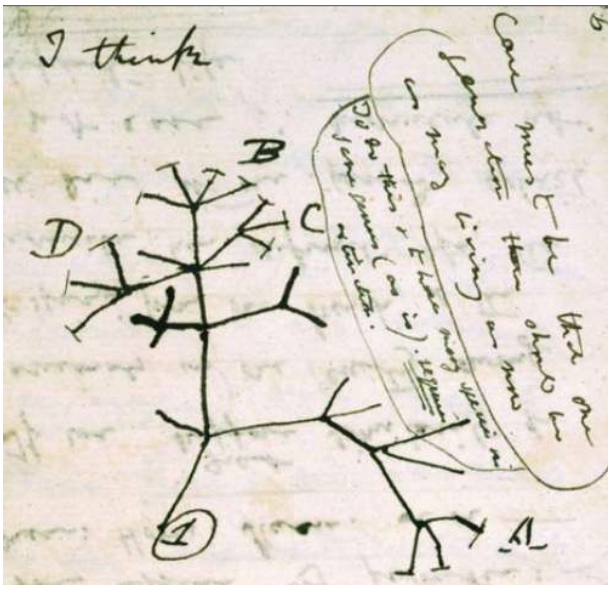

Figure 1: The first phylogenetic tree drawn by Charles Darwin on his notebook on mid-July, 1837.

that creates a random, but exactly determined, phylogenetic tree, and outputs genomic sequences that might be produced given that tree as the true evolutionary history.

Due to a lack of commonly agreed upon historical data, it is usually impossible to determine whether or not a constructed phylogenetic tree is accurate when using actual genomic sequences. One of the few areas within phylogenetics wherein experts generally agree is the evolutionary history is apes. We choose to use apes as our study case to examine algorithmic performance because there exists a commonly accepted baseline to compare our results against.

Phylogenetics is a field with significant nuances and complexity. For instance, there are multiple issues confounding tree construction: chiefly homoplasy, whereby some nucleic characters are likelier to evolve convergently, and horiztonal gene transfer, whereby genes may be transferred outside the parent-child relationship. Because each of our bare-bones implementations makes simplifying assumptions that ignore these complexities, applying these methods to real genes might help us assess their sensitivity to these uncontrolled for processes.

Project Goals

There is no single best algorithm in phylogenetic reconstruction. It's possible to produce pathological cases where certain algorithms perform better than others, and there exists substantial variability in potential inputs, depending on the specific organisms to be evaluated. Furthermore, exploration is too costly with large numbers of lengthy sequences for exhaustive search to be feasible. To explore an entire set of possible topologies, for instance, a problem with n input species would require $(2n-5)!/[(n-3)!2^{n-3}]$ constructions [24]. Thus, it is our goal to explore the contexts in which our algorithms are most effective, analyzing their efficiency and efficacy in producing a tree of relationships amongst a set of taxa.

After conducting a literature review and determining which types of algorithms were most frequently utilized, we decided to implement the following six phylogenetic reconstruction algorithms, three multiple sequence alignment algorithms, and two tree comparison methods.

- Multiple Sequence Alignments:
  - **Clustal-W (CW)**
    A progressive alignment method which computes a "guide tree" based on possible all pairwise sequence alignments. The guide tree is then collapsed to produce the final multiple sequence alignment.
  - **MUSCLE (MSC)**
    An iterative alignment method similar to Clustal-W that foregoes computing initial pairwise alignments in favor of later iterative improvement.
  - **Center Star (CS)**
    A simplistic alignment method that identifies the "center" sequence which all other sequences are aligned to. The pairs of the center and every other sequence are combined into the final multiple alignment.
- Tree Reconstruction Algorithms:
  - **Neighbor-joining (NJ)**
    A distance matrix method that attempts to reconstruct a tree through an agglomerative clustering approach.
  - **Maximum Likelihood (ML) with hill-climbing and progressive topology searches**
    A method which uses known mutation rates between nucleotides to recursively evaluate the "likelihood" of a given tree. We use the principle of ML to produce a final tree using two topology search heuristics, "hill-climbing" and "progressive" (referred to as MLH and MLP, respectively).
  - **Maximum Parsimony (MP) with hill-climbing and progressive topology searches**
    An evaluation based in the concept of parsimony, wherein trees with the least mutation required to group taxa are highly valued. We use the principle of MP to produce a final tree using two topology search heuristics, "hill-climbing" and "progressive" (referred to as MPH and MPP, respectively).

  - **Monte-Carlo Markov Chain (MCMC or MC)** An implementation of the Metropolis-Hastings algorithm which allows us to sample from a distribution of the most likely evolutionary trees.

- Tree Comparison Metrics:
  - **Pairwise Pathlength Distance (PPLD)** A distance metric that relies on the distances between all pairs of species in a given tree.
  - **Quartet Distance (QD)** A distance metric that measures the topological distance between input trees using quartet reductions.

A total phylogenetic reconstruction consists of a sequence alignment step followed by a tree reconstruction step. In total, we have 18 possible phylogenetic reconstruction methods, given our three sequence aligners and our six tree reconstructors.

Previous studies of the relative efficiencies and correctness of these algorithms are extensive but inconsistent. Most studies reach a consensus that distance-matrix based algorithms (i.e. NJ) generally outperform MP in both correctness and efficiency, regardless of nucleotide substitution rates. This is because of MP only uses sequence information from informative sites, and because it cannot adjust for multiple mutations [41]. Other studies claim that with uniform rates of evolution among branches, distance methods are inferior to parsimony both with short sequences with low rates (~ 0.01) and with long sequences with high rates (~ 0.1), and were slightly superior in the other cases [25].

In comparing NJ and ML methods, study results also vary. Saito, Naruya, and Imanishi maintain that when constant rates of nucleotide substitution rates among sites are assumed, the NJ method showed slightly better performance than ML, but inferior to ML when substitution rates varied drastically [37]. In contrast, Hasegawa, Masami, and Fujiwara find that NJ is also robust to heterogeneity of evolutionary rates among sites given that heterogeneity is considered in estimating the multiple-hit effect [23].

In comparing estimations of tree branch lengths, previous work suggests that when a low nucleotide substitution rates (~ 0.01) is assumed, NJ, MP, and ML are equally successful, while for higher rates (~ 0.1), ML is slightly better [25].

Markov Chain Monte Carlo (MCMC) methods are useful when an estimate of the posterior distribution of phylogenetic trees under specific prior assumptions is desired [39]. Methods utilizing MCMC are easily extended to Bayesian analysis, and, consequently, tree reconstructions using this algorithm are often phrased in that context. Compared with NJ, MP, and ML methods, MCMC has the advantage of being able to create a distribution of trees and constructing a "confidence set." In addition, this distribution of trees can be used to study the variability in any aspect of the phylogeny that is of interest. For instance, if you were interested in computing the variability in total tree diameter, MCMC could provide draws from an entire tree distribution to compute statistics and confidence intervals from. However, it gives poor estimates of the posterior probability of any individual tree when the number of taxa is large [39].

In terms of computational time, Saito, Naruya, and Imanishi conclude that NJ has the best performance [37], while others propose that when using different distance measures and nucleotide transition/transversion rate (R), NJ and ML perform differently [44]. When large data sets are considered, MCMC can be quite computationally intensive [39].

Our study aims at addressing these inconsistencies in the current literature through comparative efficiency and correctness analysis of these algorithms.

Case Study

Phylogenetic trees are useful tools for inferring information about public health issues. In the past, the Center for Disease control has used phylogenetic analyses to understand the origins of a lethal betacoronavirus associated with kidney and respiratory complications [8]. Also, Burr describes how phylogeny reconstruction has been used to infer useful information regarding quickly evolving diseases, including HIV and influenza [5].

Because of the potential for phylogenetic reconstruction algorithms to uncover useful information regarding public health problems, another goal of our project is to apply our algorithmic studies to a real-world data set. Our case-study involves examining the genomic sequences of 53 apes. While the evolutionary properties of apes and viruses differ considerably, the relationships between primates are far better understood. Furthermore, there exist commonly accepted ape phylogenies against which we can compare the outputs of our algorithms. Executing our algorithms on real data allows us to determine our different algorithms' sensitivity to the previously addressed major evolutionary complications, homoplasy and horizontal gene transfer.

Parallelism

Many phylogenetic tree reconstruction algorithms are particularly well suited to parallelization. Previous work demonstrates the capacity of parallelism to meaningfully speedup phylogenetic reconstruction algorithms. For instance, Schmidt et al. provide TREE-PUZZLE, a software package containing parallelized components of the ML reconstruction algorithm [40]. For a large reconstruction problem, it took their sequential algorithm 5.5 months

to terminate, whereas their parallel implementation computed the same tree in two weeks using only 12 threads. For sequence alignment algorithms, there exist parallel implementations of CLUSTAL family algorithms that achieve 10x speedup when running on 16 CPUs [6]. Clearly, there are situations in which parallelism can be meaningfully applied in the reconstruction of phylogenies.

For this project, our goal is not to compete with the existing, state-of-the-art implementations; we do not approach alignment and tree construction purely from the perspective of performance increase. Rather, we plan to take advantage of instances where parallelism offers obvious potential for improvement. For many of our tree reconstruction methods, site independence between adjacent genomic sites is assumed, and separate operations are executed for each site. This provides us a means of normalizing our parallel analysis, and simplifies our question significantly: which of our implementations are able to parallelize over site independence most effectively?

For programming on multi-core processors, there are several popular libraries available. We choose to use OpenMP, a well-supported API that supports shared memory parallelism on many popular multi-core processor architectures.

Multiple Sequence Alignment Algorithms

In order to reconstruct phylogenetic trees from multiple sequences of different lengths, the sequences first need to be aligned to uniform length. Pairwise alignment, where gaps are inserted into two different strings, can be solved by a simple dynamic programming algorithm which guarantees an optimal alignment for user defined cost parameters. The best way to align multiple sequences, on the other hand, is an open question, and we aim to analyze several popular methods.

ClustalW

Introduced in 1994 [45], Clustal-W is considered to be a so-called "progressive alignment" algorithm. At the time of its inception, it represented dramatic progress in alignment sensitivity combined with other existing tools, and is still the most widely used MSA program [11]. There are three main steps of the algorithm:

1. **The Distance Matrix/Pairwise Alignment.** Given n sequences, all $\binom{n}{2}$ pairwise alignments are computed. This step is accomplished with a simple dynamic program, as is a standard practice [45]. Optimal alignment is guaranteed with this approach given a table of scores for matches and mismatches between sequence characters and penalties for insertions or deletions. The scores of each pair of alignments are then calculated using the a simple sum-of-pairs (SP) measure, and a distance matrix is constructed based on these scores.

2. **Create a Guide Tree.** Next, a "guide tree" is constructed. This structure is used to guide the rest of the process. This guiding tree is calculated from the distance matrix in step 1, using Neighbor-Joining.

3. **Progressive Alignment Using the Guide Tree.** This final step is accomplished using a series of pairwise group alignments. Progressively larger groups of sequences are aligned following the branching order in the guide tree we created in the second step using a dynamic program. Instead of defining a penalty matrix, to align groups of sequences, a site-wise frequency function over all sequences in a group is defined. After the entire tree has been "collapsed" up from tips to root, we are left with a set of n aligned sequences, each with the same total length [45].

Merits

The main advantage of the progressive strategy used by ClustalW is its speed and relative robustness [30]. ClustalW also requires much less memory than other programs. Therefore, it is suggested that it should be used on aligning small number of unusually long sequences [11].

Critiques

Because it was one of the first popular multiple sequence alignment algorithms, many improvements have been made to CW. For instance, Clustal-W has been improved through better decision making during multiple alignment (e.g. when to change weight matrix) and the accuracy and appropriateness of parameterization. However, no significant improvements have been made since 1994, and several modern methods (e.g. MAFFT, MUSCLE, T-COFFEE) claim to achieve better performance in accuracy, speed or both [11].

Although widely used in a variety of cases, CW suffers from its greediness, as errors made in initial alignments cannot be corrected later when the progressively more sequences are merged together[43]. In addition, there is no way of quantifying whether the resulting alignment is good, or if the alignment is correct due to the algorithm's greedy nature.

MUSCLE

As the exploration of phylogenetic reconstruction has advanced, so too has the desire for highly scalable alignments. MUSCLE is an approach strikingly similar to CW with some modifications made to quickly reach the alignment phase. Introduced in 2004 [10], MUSCLE has been

gaining traction among the various MSA's for its use on alignments of large data sets of short sequences. MUSCLE has three main phases:

1. **Draft Progressive.** The first iteration focuses on speed over accuracy, quickly transforming the raw sequences into sloppily aligned sequences.
   (a) **kmer Distance.** A distance matrix is produced by converting the basic sequences to their amino acid sequences. From there, each sequence is given an identity of the frequencies of k-tuples in the amino acid chain. Euclidean distance is used on the series of frequencies to determine distances amongst the sequences.
   (b) **Guide Tree 1** Using a heuristic to cluster the sequences, the guide tree is formed. In this case, Neighbor Joining is used to compute a guide tree.
   (c) **Progressive Alignment** The alignments are produced identically to CW with the exception that a differing scoring system [12] is employed. The goal of this modified scoring system is that it accounts for gaps in sequences more accurately.
2. **Improved Progressive.** Now that the sequences are aligned, the process will be repeated using a different metric to improve the alignments.
   (a) **Kimura Distance.** Every distance between sequences is computed using the Kimura metric. Essentially the distance becomes how many matching characters each sequence shares.
   (b) **Guide Tree 2.** The guide tree is reproduced on the new distances in the same manner as before.
   (c) **Progressive Alignment.** The alignment process is repeated here as well, with the opportunity to only realign on sections of the new guide tree that differ from the old one to cut down on processing time.
3. **Refinement.** Here is where MUSCLE diverges most from CW. We improve the alignment through an iterative process. Upon visiting some edge while traversing the current guide tree, we cut the tree into two subtrees, realigning their profiles individually, reconnecting them, and realigning as a whole. If the action has netted an improvement, the change is kept, otherwise the refinement is considered to be done. This process is repeated until convergence or user satisfaction.

Merits

MUSCLE is best for large data sets (on the order of hundreds of species) with short length sequences [11], where the process of eliminating the initial pairwise alignment outweighs the cumbersome refinement process.

Easily the most defining feature of MUSCLE compared with CW is the refinement period, which improves upon the weak exploration of CW. This feature should lead to more accurate alignments.

Critiques

While some improvements have been made over the basis of CW, the refinement phase is still highly expensive to run, with little guarantee of dramatic improvement. The refinement phase caps the performance at an $O(n^3L^2)$ time, for n species of maximum sequence length L.

Center Star

Center Star does not construct a guide tree or use clustering methods in creating a multiple sequence alignment [22]. Instead it identifies the most "central" sequence which when aligned to every other sequences has the lowest total distance between itself and all other sequences. This method is by far the simplest and near the fastest of the mentioned algorithms, and can be shown to produce an alignment no worse than two times the optimal alignment's total distance. Center star operates as follows:

- **Center Sequence Identification.** By computing an $n \times n$ matrix of the hamming distance between each sequence and finding the sequence which minimizes this value, the center of the alignments can be found.
- **Center Sequence Matching.** Every sequence is pairwise aligned to the center sequence.
- **Combination.** By combining the center sequences between each pair alignments that have been aligned uniquely we can find the conglomerate sequence, which gains the spacing of both its component center sequences. From there we align the matched sequences to the conglomerate and repeat until all the sequences have been combined into a multiple alignment.

Merits

Center Star is easily implemented and runs in $O(n^2L^2)$, where n is the number of input sequences, and L is the length of the longest input sequence.

Critiques

The algorithm can be described as a "quick and dirty" method of generating a multiple alignment with only the guarantee that it is at most twice the optimal alignment's accuracy.

Sequence Alignment Evaluations

Using our experimental set up, we generated several alignments on the same set of sequences using the listed alignment methods. We must note that our sequences were aligned using the author of the MUSCLE algorithm's available code and not our own implementation of MUSCLE. Our evaluation of the success of an alignment uses the total distance between every possible pair of sequences in the set, multiplied by the length of the alignment to weight longer alignments as worse than shorter ones. Formally, let L be the sequence length, α be the total number of non-matching indices between two sequences, and β be the total number of non-dual-gap positions.

$$distance = \frac{\alpha_{i,j}}{\beta_{i,j}} * L$$

The total distance is simply the sum of distance between all possible pairs of sequences in the set. We completed 14 runs for each alignment method for several numbers of sequences. Using the distance equation the performance of Clustal-W (CW), Center Star (CS), and MUSCLE (MSC) were tested against each other in a student's T test to obtain the following t-statistics:

Table 1: Alignment Method Performance Check

Method Comparison	Real	Synthetic
CW-CS	-2.265	-0.666
CW-MSC	-8.252	-2.016
CS-MSC	-5.899	-1.309

Each entry relates to the significance of the former method being worse than the latter using our evaluation. Hence in the analysis of the real sequences of DNA every result was significant at $p = .05$ meaning we get a hierarchy of $CW > CS > MSC$ in terms of accuracy on real data, with respect to our test statistic. However on the much shorter synthetic sequences of $\frac{1}{8}$ length the real data we only see significance at the same level in the CW-MSC comparison.

Tree Reconstruction Algorithms

Neighbor Joining Method

Reconstruction of phylogenetic trees generally involves inference of phylogenies consisting of large amounts of gene sequences. First proposed by Naruya and Masatoshi [38] the neighbor-joining (NJ) method is frequently used to construct phylogenetic trees of life because of its known accuracy and relatively fast computational speed. The use of neighbor-joining is so wide spread, in fact, that it has become a common baseline to compare newly proposed reconstruction algorithms to [29].

Neighbor joining itself is a very simple algorithm. First, a distance metric is used to compute pair-wise distances between all possible pairs of input sequences, a common practice in so-called "distance matrix" methods. Based on this distance matrix, a similarity metric is used to find the two most similar sequences, which can be thought of as nodes in the eventual phylogenetic tree. Once discovered, these two sequences, now considered as leaf nodes, are connected by a common parental node. The distance between this new parental node and all other nodes represented in the distance matrix is estimated. This process continues, until there are only two nodes remaining. At this point, the algorithm has greedily and hierarchically constructed a phylogeny.

We determine the asymptotic running time of our implementation as follows. For a dataset of n species, $O(n^2)$ pairwise distance computations must be executed, which take $O(m)$ time, where m is the length of sequence. After the distance matrix is computed, $O(n)$ node merges must be executed. For each node merge, $O(n^2)$ distance computations must be executed to determine the average distance between all tips from each other. This leads to a running time of $O(mn^2 + n^3)$.

Merits

Tamura, Nei, and Kumar demonstrated the accuracy of NJ trees for inferring very large phylogenies using the reports of their computer simulations [43]. Given pairwise distances estimated using biologically realistic models of nucleotide substitution, reports show that the accuracy of NJ trees decline only by about 5% when the number of sequences used increases from 32 to 4,096 (128 times) even in the presence of extensive variation in the evolutionary rate among lineages or significant biases in the nucleotide composition and transition/transversion ratio. These results suggest the promising prospects for inferring large phylogenies using the NJ method.

Furthermore, the use of the NJ method has some appealing theoretical guarantees. Atteson shows that, for appropriately long and accurate multiple sequence alignments, NJ will produce the correct phylogeny for certain data; a formal bound is provided for the requirements of the inputs [1]. Though this is an important result regarding the theoretical guarantee of this algorithm, the formal conditions required for data inputs are often not met in practice. Despite this, NJ still performs exceptionally well. Mihaescu et al. [29] provide an extension of Attesons theorem that explains this apparent discrepancy, and quantify the usefulness of NJ in practice.

Finally, NJ is considered one of the most computationally efficient algorithms. As previously addressed, it runs in $O(n^3)$ [9].

Critiques

A distance-based method, NJ has the disadvantage of discarding the actual character data in the sequences [3]. Since it is designed to produce only one tree, it can obscure ambiguities in data. Although ambiguities can be uncovered by using resampling methods, if used alone, NJ programs may give misleading bootstrap frequencies because they do not suppress zero-length branches and/or are sensitive to the order of terminals in data. In addition, resampling can be employed with parsimony methods, which are far more efficient than NJ methods [15].

Markov Chain Monte Carlo for Bayesian Analysis

Traditional methods for phylogenetic inference select a single "best" tree, while a Bayesian approach expresses the uncertainty in phylogeny and in the parameters of the sequence mutation model with a posterior probability distribution [26]. The computational aspects of the problem can be efficiently solved by the Markov Chain Monte Carlo (MCMC) algorithms. While the Bayesian approach is the most commonly used Markov Chain based reconstruction method, because of its relative complexity, we assume a uniform prior, and use the Metropolis-Hastings algorithm to simply draw a phylogeny from a distribution proportional to tree likelihood. Though many applications of MCMC also utilize a uniform prior, our approach differs because we don't attempt to infer mutation rates and other complex parameters. This makes a our Bayesian analysis more similar to a likelihood-based inference because the distribution we draw from is unaffected by the prior [36].

MCMC technique was first introduced by Yang and Rannala in 1997 [49]. It was later developed by Mau, Newton, and Larget [28]. We adopted a simplified version from Larget and Simon [26]. In their original paper, they presented two MCMC algorithms: $GLOBAL$, in which all branch lengths in topologies are adjusted with every cycle, and $LOCAL$, which allows for the same changes but with different probabilities. After an initial burn-in period, the Markov Chain runs for 2,000 cycles with $GLOBAL$ followed by $LOCAL$ to complete tree parameter proposals and outputs the final tree. We implemented both algorithms under the molecular clock assumption.

Metropolis-Hastings Algorithm

The Metropolis-Hastings (M-H) algorithm is used to sample from distributions that are difficult to sample from directly. In many cases, a function proportional to the desired underlying probability distribution is known; in the case of phylogenetic trees, we use the likelihood scoring function to sample trees from regions of tree space with high likelihood. However, it is often the case that the statespace is so large (possibly infinite) that a required normalizing constant cannot be computed efficiently.

More specifically, the algorithm relies on an underlying stochastic process known as a Markov Chain to produce samples from a distribution proportional to some function. For phylogenetic trees, the state space is that of all possible phylogenies. The algorithm starts at a given tree and randomly makes transitions between states through a two step process. First, a local transition is "proposed" by the algorithm. Next, based on the nature of that proposed transition, it is either accepted or rejected. If the new state is rejected, the current state is repeated in the sequence. M-H imposes restrictions upon these acceptance/rejectance transition probability densities. It is relatively simple to show that such a process has a "limiting distribution" proportional to the given objective function, if the transitions are handled properly.

However, it is clear this process does not reach its limiting distribution immediately. The first transitions in particular are very dependent on where the chain was started. However, if one were to let the process continue for an infinite amount of time, the process would reach stationarity, and the current tree would represent a tree sampled from the desired distribution [26]. Therefore, the long-run frequencies of sampled tree topologies are arbitrarily close to their desired frequencies. The time it takes for the chain to come close enough to stationarity is commonly referred to as the "burn in" time [26].

M-H is used in constructing the Markov chain on tree parameters, which include topologies and associated branch lengths. We use transitions originally proposed by [26] as follows.

$GLOBAL$ with a Molecular Clock

This algorithm changes all branch lengths with every cycle. Under the molecular clock assumption, the pair of distances to adjacent leaves are maintained to be equal for each internal node.

$LOCAL$ with a Molecular Clock

This algorithm modifies the tree only in a small neighborhood of a randomly chosen internal branch, and each different modification is made with different probabilities. It is also implemented under the molecular clock assumption.

Merits and Critiques

Compared to Yang and Rannala's approach [49], this algorithm can work with much larger trees. It generates a posterior distribution of phylogenetic trees, giving it substantial advantages over traditional ML algorithms. In addition, the calculation of an acceptance probability of a proposed tree sums over the unknown data at the internal nodes, a rapid and accurate process with pruning algorithm [18]. Since MCMC depends on the underlying likelihood model, data sequences generated by the best fitted model would likely differ from genuine data regarding composition of amino acids, locations of stop codons, and other biologically relevant features [26]. A more fundamental problem underlying an MCMC algorithm is its ability to correctly identify the posterior probabilities of the collection of highly probable tree topologies. It is difficult for a particular simulation to visit new regions of parameter space once it gets stuck in an old region [26]. Likewise, it is relatively difficult to transition between islands of high posterior probabilities. Thus, this model may often yield inconsistent results when applied to the same sets of data. Finally, specification of a prior distribution of model parameters can strongly influence the estimation of its posterior parameter distribution [36]. Thus depending on the nature of the sequence data being examined, a prior distribution of parameters must be chosen carefully for MCMC algorithms to produce a rather accurate posterior distribution.

Topology Search Algorithms

Thus far, we have introduced two methods for producing phylogenetic trees from multiple aligned sequences. We present two more reconstruction algorithms that fall into a greater class of tree "scoring" algorithms. Scoring algorithms define an objective scoring function, and the user can utilize a variety of algorithms to search through tree space. In this study, we implemented two types of tree searching algorithms.

The first approach, originally introduced by [17], is a heuristic approach that begins with a tree that contains just two randomly chosen species. The final tree is iteratively built up from this simpler tree by adding one species at a time. To add the k^{th} species to a tree with $k-1$ species, the algorithm considers inserting this new species on each internal edge. The scoring function is then executed on the resulting $2k-5$ topologies, and the best is chosen. These steps are repeated until all of the species in the observed data set are added to the tree.

The second approach begins with a randomly constructed tree containing all n species, inserted arbitrarily, and uses hill-climbing to arrive at a local optimum. The algorithm proceeds as follows. We produce the "neighborhood" of the current tree by swapping subtrees of every internal edge. For each internal edge, there are two unique neighbors created. An outline of this approach is presented here...

1. Construct initial tree and determine its score.
2. Construct a set of "neighboring trees" using Nearest Neighbor Interchange [46].
3. If any neighboring tree is better than the current tree, select the best one and use as starting point for new round of rearrangements.
4. Repeat steps 2 and 3 until a tree that is better than all of its neighbors is found. This tree is a local optimum.

The former method runs significantly faster due to its smaller topology search space. However, the outcome of the algorithm will depend on the order of addition of the species. On the other hand, the latter method produces a tree that is independent of species ordering, but the run time is significantly worse because it requires more scorings of larger trees.

Maximum Parsimony

Maximum Parsimony is a scoring method used for inferring phylogenies. "The maximization of parsimony," or preferring the simplest of otherwise equally adequate theories is the guiding principle in this method. With the assumption that evolution is inherently a parsimonious process, Maximum Parsimony values phylogenetic trees where the least evolution is required to group taxa together [19].

The objective function we attempt to minimize is tree "length." Tree length refers to the minimum number of mutations required to explain a given topology. To determine the length of a given tree, Fitch's Algorithm [20] is used.

1. Root the tree at an arbitrary internal branch.
2. Visit an internal node x for which no state set has been defined, but where the state sets of x's immediate descendants (y,z) have been defined.
3. If the state sets of y,z have common states, then assign these to x.
4. If there are no common states, then assign the union of y,z to x, and increase tree length by one.
5. Repeat until all internal nodes have been visited and return the length of the current tree.

Merits

The principle of constructing a maximally parsimonious tree takes advantage of Occam's razor, which, in this context, states that the topology which assumes the least amount of total evolutionary events is the most likely to occur. Because mutations are relatively unlikely, the tree of "minimal evolution" is likely a good approximation of the actual evolutionary history of a a system.

The principles guiding the Maximum Parsimony approach frees the algorithm from making other assumptions, unlike other models used in phylogenetic reconstructions.

Furthermore, Fitch's algorithm lends itself exceptionally well to parallelization. It runs in $O(nmk)$, where n is the number of leaves, k is the number of states, and m is the sequence length. Parallelization of the algorithm can reduce the runtime to $O(nk)$ since processing of each nucleotide position is independent. Because a "read-only" tree traversal must be executed for each site in a genome, according to Gustafson's Law of parallelization, the speedup we can achieve on this portion of the program is exactly equal to the number of processors we have available, because no sequential operations must take place. Depending on the size of the topology and the length of the genomes, either a multi-core CPU or GPGPU implementation might be effective.

Critiques

Unfortunately, evolution is not a completely parsimonious process, though it is assumed to be in Fitch's original method. Unlikely events, which can cause massive change to occur in genomic sequences, do occur in organic evolution. For instance, gene duplication is a type of mutation that causes multiple copies of DNA segments to be reinserted into the original genome. Assuming such a mutation is surely not the most likely explanation for evolution in some cases, but it occurs nonetheless.

Fitch's original algorithm for determining the minimum number of mutations for a given topology does not produce edge weights. In other reconstruction algorithms, edge weights offer a sense of evolutionary distance, and this information is simply not present in topologies reconstructed using parsimony methods. (check to see if there are any parsimony-based edge-weight generating algorithms)

Maximum Parsimony is not a statistically consistent method in finding the true best tree given sufficient data. Consistency refers to the monotonic convergence on the correct answer with the addition of data, and Maximum Parsimony lacks this consistency under the category of situations called "long branch attraction" [16]. Under these situations, there are high levels of substitutions for two characters and low levels of substitutions for others. The more data we collect, the more we tend towards finding the wrong tree.

In addition, because Maximum Parsimony uses heuristic methods in searching tree space, the most parsimonious tree is not guaranteed to be obtained. However, this problem is not unique to this algorithm; any algorithm that uses an optimality criterion is subject to the same critique.

Maximum Likelihood

Scoring phylogenetic trees using a Maximum Likelihood approach was first proposed by Felsenstein in 1981 [17]. Continued advancement in computational power has overcome the method's inherent high computational costs, and the Maximum Likelihood approach became one of the most popularly used methods of phylogenetic reconstruction. In more recent history, the Maximum Likelihood approach has been used to uncover the confounding evolutionary history of some species, such as the giant panda and dolphins [48, 32].

The core concept of the Maximum Likelihood approach is to find a phylogenetic tree that has the highest "likelihood," given an observed set of DNA sequences. Note that the likelihood of a tree is the probability of a given tree yielding the observed outcome; it is not the probability of a tree being the correct one [17].

Assumptions

The Maximum Likelihood approach makes three assumptions to make computation faster and easier.

1. Nucleotide substitution happens site independently. This means that the mutation rate of a nucleotide at one location is not affected by any other nucleotide at a different location.

2. Two lineages evolve independently after speciation. In other words, any pair of two species will not affect the evolutionary processes of each other.

3. Any lineage has the same probability of nucleotide base substitutions. For example, if species A had 20 percent chance of cytosine mutating to thymine, the same rate applies to species B.

Likelihood calculation

By the first assumption of site-independent evolution, the likelihood of an entire tree can be calculated by computing the likelihood values site by site, and multiplying them all together. The likelihood of a tree at one DNA site is then calculated by multiplying the probability of each segment of a tree and the prior probability of the root. The probability of a tree segment is computed using the segment length, mutation rates, and the nucleotide bases of the two nodes in

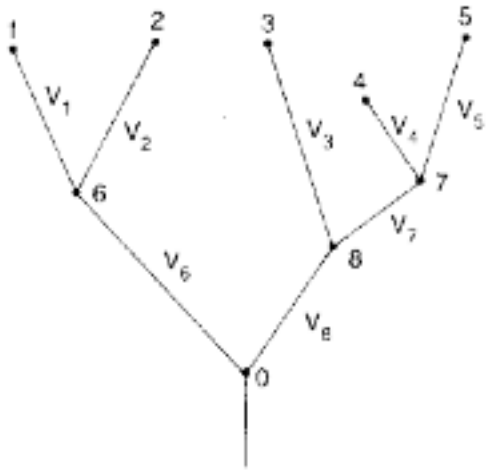

Figure 2: An example tree for likelihood calculation. This figure is taken from the paper written by Feslenstein 1981

the segment. We do not know the nucleotide bases of the internal nodes, since they represent common ancestors that are now extinct. To account for this, the likelihood is computed as the sum of all possible assignments of nucleotide bases to the internal nodes [17]. The following equation computes the likelihood of the tree in Figure 2.

$$L = \sum_{s_0}\sum_{s_6}\sum_{s_7}\sum_{s_8} \pi_{s_0} P_{s_0 s_6}(v6) P_{s_0 s_6}(v6) P_{s_6 s_1}(v1) \\ P_{s_6 s_2}(v2) P_{s_0 s_8}(v8) P_{s_8 s_3}(v3) P_{s_8 s_7}(v7) P_{s_7 s_4}(v4) \\ P_{s_7 s_5}(v5)$$

where $P_{s_x s_y}(v)$ denotes the probability of a nucleotide s_x changing to s_y, given branch length v.

Branch optimization

To find a tree with the highest likelihood, we first start by maximizing the likelihood of a topology. Note that, with different branch length assignments, one topology can represent multiple different tree structures. By iteratively optimizing the length of all branches within in a topology, we can compute the Maximum Likelihood value that can be achieved by the topology (see (Felsenstein 1998) for the optimization of a single branch). Using the topology searching methods, we can then compare the Maximum Likelihood values of topologies to find a topology whose Maximum Likelihood is maximum. Then, the tree structure that gives the Maximum Likelihood value for the best topology is outputted as a resulting phylogenetic tree.

Merits and Critiques

While the Maximum Likelihood method might produce more accurate trees when compared to other reconstruction methods, it is computationally costly, mainly due to the likelihood calculations in the branch optimization process. The big-O runtime estimation of the algorithm is $O(mn^6)$ with the progressive topology and $O(kmn^5)$ with the hill-climb approach, where k is the number of hill-climb iterations, m is the length of DNA sequences, and n is the number of species. The Maximum Likelihood algorithm also assumes that the nucleotides mutate site-independently, and thus, it does not accurately account for changes such as insertion or deletion. Commonly accepted solutions to this indel problem are (i) remove all sites in which any gap appears, (ii) assign an imaginary nucleotide for gaps, and (iii) treat gaps as missing data. In this project, we used the option (ii), as Evans et al showed that the option (iii) can have deleterious effects on the reconstruction [14].

Tree Comparison Algorithms

In order to properly compare alignment and reconstruction methods, we require methods of comparing their ultimate outputs. We decided to use two different tree-distance metrics; one focused on absolute distance between species that utilized branch lengths, and one concerned with overall tree topology. Together, we believe these metrics summarize the differences and similarities between trees effectively. Note that these two metrics are part of a larger class of comparison methods known as "dissimilarity metrics;" this means that higher values indicate greater dissimilarity.

Quartet Distance: Topological Metric

To compare the similarity of topologies numerically, we employ a "quartet" based method, first proposed for this purpose by Estabrook, McMorris and Meacham [13]. A quartet is a phylogenetic tree with only four species, divided by two internal nodes as in Figure 3. To "reduce" a phylogeny to a quartet given four species, first, remove all non-desired species from the tree. Next, remove edges and internal nodes shifting your remaining species appropriately until you are left with only a quartet. This quartet should maintain some structural properties of the original tree; the remaining species now represent reductions of subtrees in their original topological formations. Computing a phylogenetic reduction is $O(n)$ because it requires a single traversal of the tree. Many previous studies have employed quartet distance as a means of analyzing the similarity of phylogenetic topologies in both theoretical an practical settings. For instance, Puigbo et al. [35] utilize this metric in their analysis of horizontal gene transfer in prokaryote evolution.

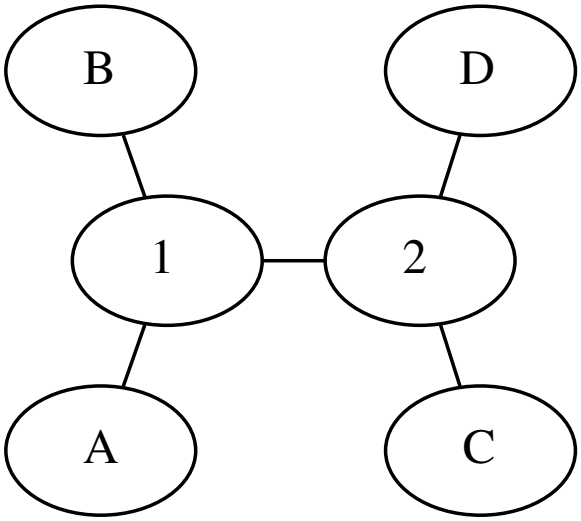

Figure 3: An example of a quartet, denoted as $AB|CD$.

To compute the quartet-distance between any two topologies, say T_1 and T_2, that contain the same n species, we compute all size-4 subsets $\{a, b, c, d\}$ of the n species and count the number of times the reduction of T_1 to $\{a, b, c, d\}$ doesn't match the reduction of T_2 to $\{a, b, c, d\}$. The brute force algorithm, as described, is $O(n^5)$ because there exist $O(n^4)$ size-4 subsets of the n species, and for each subset, two $O(n)$ reductions must be computed.

Fortunately, there exist algorithms that reduce the time complexity of this computation significantly; Brodal et al. [4] present an $O(nlogn)$ solution to the quartet-distance problem, which represents the current state-of-the-art. For our purposes, we utilize a simpler $O(n^3)$ solution, originally proposed by Christiansen et al. [7].

Central to this improved algorithm is the fact that, given any three species $\{a, b, c\}$ within a bifurcating phylogeny, there is exactly one internal node such that the paths between $\langle a, b\rangle$, $\langle b, c\rangle$, and $\langle a, c\rangle$ intersect at that node. Deleting such an internal node and its incident edges would result in three subtrees, each containing one of a, b, and c. Denote the subtrees as T^a, T^b, and T^c. For each species $x \in T^a - \{a\}$, we can determine that the quartet of our original tree restricted to $\{a, b, c, x\}$ will be $ax|bc$. Symmetric principles apply to T^b and T^c. While further algorithmic detail is not necessary, this is the primary observation that allows us to reduce our time complexity to $O(n^3)$, as we now restrict our consideration to species subsets of size three.

Pairwise Path Distance: Branch Length Metric

Because quartet distance doesn't account for branch lengths and many of our tree reconstruction algorithms produce weighted topologies, a secondary metric that accounts for this additional information is required. First proposed by Williams and Clifford [47], we utilize a version of pairwise pathlength distance similar to that proposed by Steel and Penny [42]. The focus of this comparison method is computing all the pathlength between all pairs of species in a given phylogeny.

More specifically, pairwise pathlength distance can be computed as follows. Given two trees with associated branch lengths T_1 and T_2 each containing species $\{S_1, S_2 \ldots S_n\}$, consider a fixed ordering of all possible species pairs $\langle (S_1, S_2), (S_1, S_3) \ldots (S_{n-1}, S_n)\rangle$. Consider $\vec{d_1}, \vec{d_2}$, the ordered pairwise pathlength distances between the species specified in the ordering for T_1 and T_2. After normalizing these vectors such that each of their largest components is equal to one, the pairwise path distance between T_1 and T_2, then, is given by

$$d_{path}(T_1, T_2) = ||\vec{d_1} - \vec{d_2}||_2 \quad (1)$$

where $|| \cdot ||_2$ is the L_2 (Euclidean) norm.

To compute pathlengths between all pairs of nodes in a weighted graph, we use the Floyd-Warshall algorithm [21], which performs the desired computation in $O(|V|^3)$ in a general graph. In our case, $O(|V|^3) = O(n^3)$, where n is the number of species in a tree. This term dominates the computation of pairwise path distance, and represents the overall runtime of our approach, given two phylogenies.

Experiments

Methods

To compare our algorithms, we design two basic experiments, one for synthetic data, and one for real data.

Synthetic Data Experiments

Our random data generator is capable of producing testing examples $\langle T, D\rangle$ where T is a randomly generated phylogenetic tree containing n species, and D is a set of n sequences generated based on that synthetic tree. We can use D as input to a total of 18 combinations of the 3 multiple sequence alignment algorithms and the 6 reconstruction algorithms. The output of these algorithms can be then be compared to the true tree T and the tree using either of our distance metrics.

Our random data generator is governed by several input parameters, including the number of desired species, the global mutation rate, and the starting sequence length. Because of our limited computational resources, we were only able to execute a subset of the large number of possible experiment. In total, we completed tree reconstructions

from all possible pairs of alignment and reconstruction algorithms in the Cartesian product {Clustal-W, Center Star} $\times$ {Neighbor Joining, Maximum Parsimony Progressive, Maximum Parsimony Hill-Climbing, Maximum Likelihood Progressive, Maximum Likelihood Hill-Climbing, MCMC with likelihood}. MCMC was ran for 200 iterations.

We executed each of these 12 reconstruction methods on 14 randomly generated datasets with varying number of species. The randomly generated datasets we used had a total number of species between four and eight. Furthermore, we have a constant mutation parameter and seed the mutation process with sequences of length 200. This gives us five distinct datasets.

For each of the five datasets and 12 reconstruction methods, we evaluate the performance of our algorithms over 14 trials. To evaluate their outputs, we compute the average quartet distances and the average pairwise path distances, normalized to $[0, 1]$. Notably, one of our reconstruction methods, MP, does not produce meaningful branch length predictions, so for any analysis associated with MPP or MPH, we only use quartet distance.

Furthermore, we measure the average runtime of each algorithm in each scenario to quantify the computational efficiency of each approach.

Questions we address with experiment one include...

1. Do different algorithms perform significantly better or worse when there are different numbers of species in the dataset?
2. Does algorithm runtime depend on problem difficulty, rather than problem size?

Real Data Experiments

We have a dataset $\langle T, D \rangle$ where T is the commonly accepted phylogeny for 53 species (50 primates and 3 non-primates), and D are the DNA sequences of the mitochondrial cytochrome c oxidase subunit 1 (COX1) of those real species [33]. We decided to use the phylogeny of great apes because it is well-studied and commonly agreed upon [33]. This makes the commonly accepted ape phylogeny a great candidate for a "ground truth" to compare against.

COX1 is a popular choice for phylogenetic reconstruction because it is highly conserved due to its involvement with aerobic respiration [31]. To produce varying numbers of species in our input data, we can select random subsets of the 53 extant species for analysis.

1. 14 test examples $\{\langle T, D \rangle\}$ with 5 species.
2. 14 test examples $\{\langle T, D \rangle\}$ with 8 species.

Due to the computational intensity of the experiments, we were only able to run Clustal-W alignments paired with our six reconstruction method for each of these 28 datasets.

Questions we address with experiment one include...

1. Do the random data results match the real data results?
2. Which algorithm performs fastest on the real data?
3. Which algorithm produces the most accurate tree on the real data?

Reconstruction Hypotheses

We hypothesize that the method with Center Star and Neighbor Joining has the shortest average running time on randomly generated data due to its algorithmic simplicity. We believe that Clustal W/Maximum Likelihood and Clustal/Markov Chain Monte Carlo will perform better than other methods in terms of the accuracy of tree reconstruction on the random data because these algorithms make fewer "binding" local decisions that might cause a build-up of errors.

Parallel Experiment

We execute the sequential and parallel version of our code on 4 randomly generated datasets consisting of four, five, six and seven species, and measure the resulting speedup on a machine with 8 processors. Of our implementations, Neighbor Joining does not assume site independence and it is consequently excluded from our analyses. Because each algorithm's asymptotic runtime is simply multiplied by a constant factor representing genomic length in accordance with the site independence assumption (i.e. an $O(n^2)$ computation for a single site becomes an $O(mn^2)$ computation for m sites) we suspect that speedup between our algorithms will be roughly the same, and consistent over varying numbers of species.

Results

Synthetic Data Experiments

- **Pairwise Distance.** Figure 4 illustrates the accuracy of our tree outputs in terms of pairwise distance. Notably, Neighbor Joining and Maximum Likelihood Progressive consistently did better than other reconstruction algorithms. On the other hand, the trees produced by the Markov Chain Monte Carlo method did significantly worse than trees produced by any other reconstruction method. Choice of multiple sequence alignment algorithms did not have a visible effect on the accuracy of a resulting tree.

- **Quartet Distance.** The results for the quartet distance analysis is represented in Figure 5. Reconstruction

with Neighbor Joining, Maximum Parsimony Progressive, and Maximum Likelihood Progressive methods outperformed other methods. Again, choice of multiple sequence alignment algorithms did not have a visible effect on the accuracy of a resulting tree.

Real Data Experiments

Due to an unexpected bug found late in our Maximum Likelihood algorithm and the constraint of time, we were not able to obtain complete results for the real data experiments. However, experiments are running in progress and results will be available soon.

Parallel Experiment

Results from our parallel tests are illustrated in Figure 6. Clearly, our results were not consistent with our hypothesis; speedup deviated significantly from the theoretical 7-8x for all algorithms. This is an indication that parallel theory does not always align with parallel implementation and significant care must be taken to achieve optimal performance.

The speedup of MLH and MLP were fairly consistent over different numbers of species, but only a 3x speedup was achieved. MPH and MPP were no faster (and sometimes slower) than their sequential counterparts; this is likely a result of their relatively low runtimes. In these cases and at this scale, the overhead associated with creating threads counteracted the parallel speedup attained.

Most interesting were our speedup results for MCMC. Here, speedup decreased significantly as the number of species increased. Because MCMC is parallelized in the same way as MLH and MLP (all three methods use a parallel version of the likelihood computation) this result was particularly unexpected.

Running Time

Runtime analysis of both synthetic (Figure 8) and real data (Figure 9) suggests the following:

1. Neighbor Joining gave the fastest performance.

2. Maximum Likelihood performed worse than Maximum Parsimony.

3. Monte Carlo Markov Chain performed slower than Maximum Likelihood for smaller synthetic data sets, and faster than Maximum Likelihood hill-climbing for bigger synthetic data sets.

These results are similar when using Center Star and MUSCLE alignment algorithms.

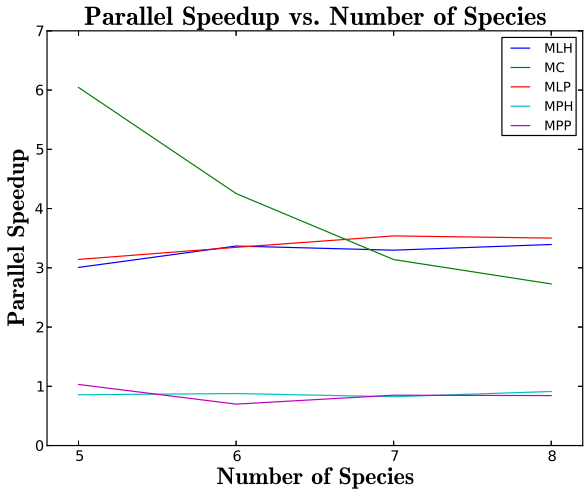

Figure 6: Parallel speedup of our implementations on several test synthetic datasets (bp = 200) of varying size on a machine with 8 processors. We allowed OpenMP to use 64 threads.

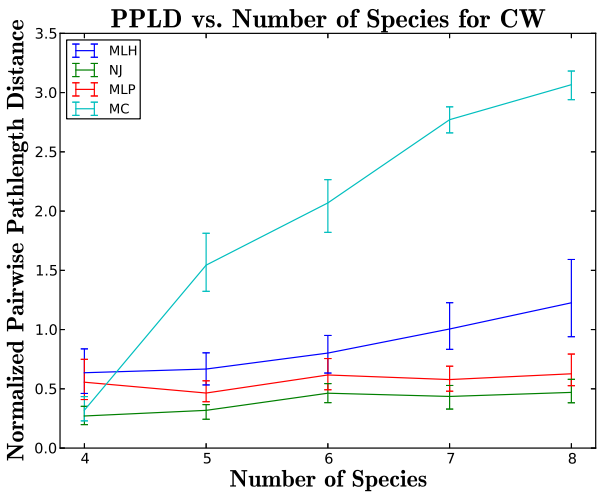

Figure 7: Pairwise pathlength distance of several methods over varying numbers of species. Results indicate that MCMC and MLH might become relatively less accurate as the input size increases.

Discussion

Tree Accuracy

In terms of pairwise distance metric, ML Progressive outperformed ML Hill-climb. The only difference between the two reconstruction algorithms was their topology searching method: progressive vs. hill-climb approach. The difference in performance between these two algorithms can be explained as follows. The downside of the progressive

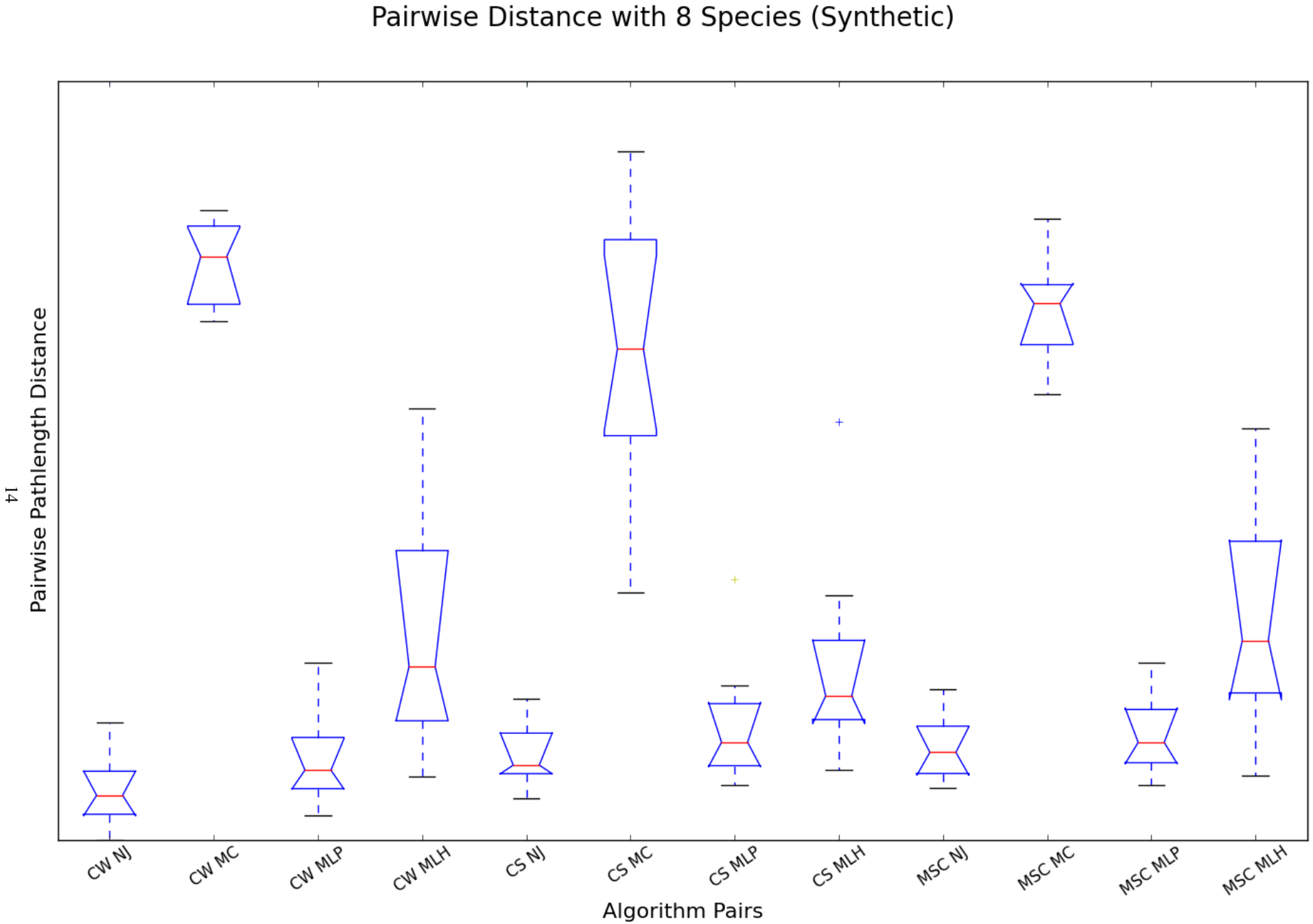

Figure 4: The resulting total PPLD distance between each method's generated trees and the source tree which is assumed to be correct. Higher values are considered to be further from the original tree.

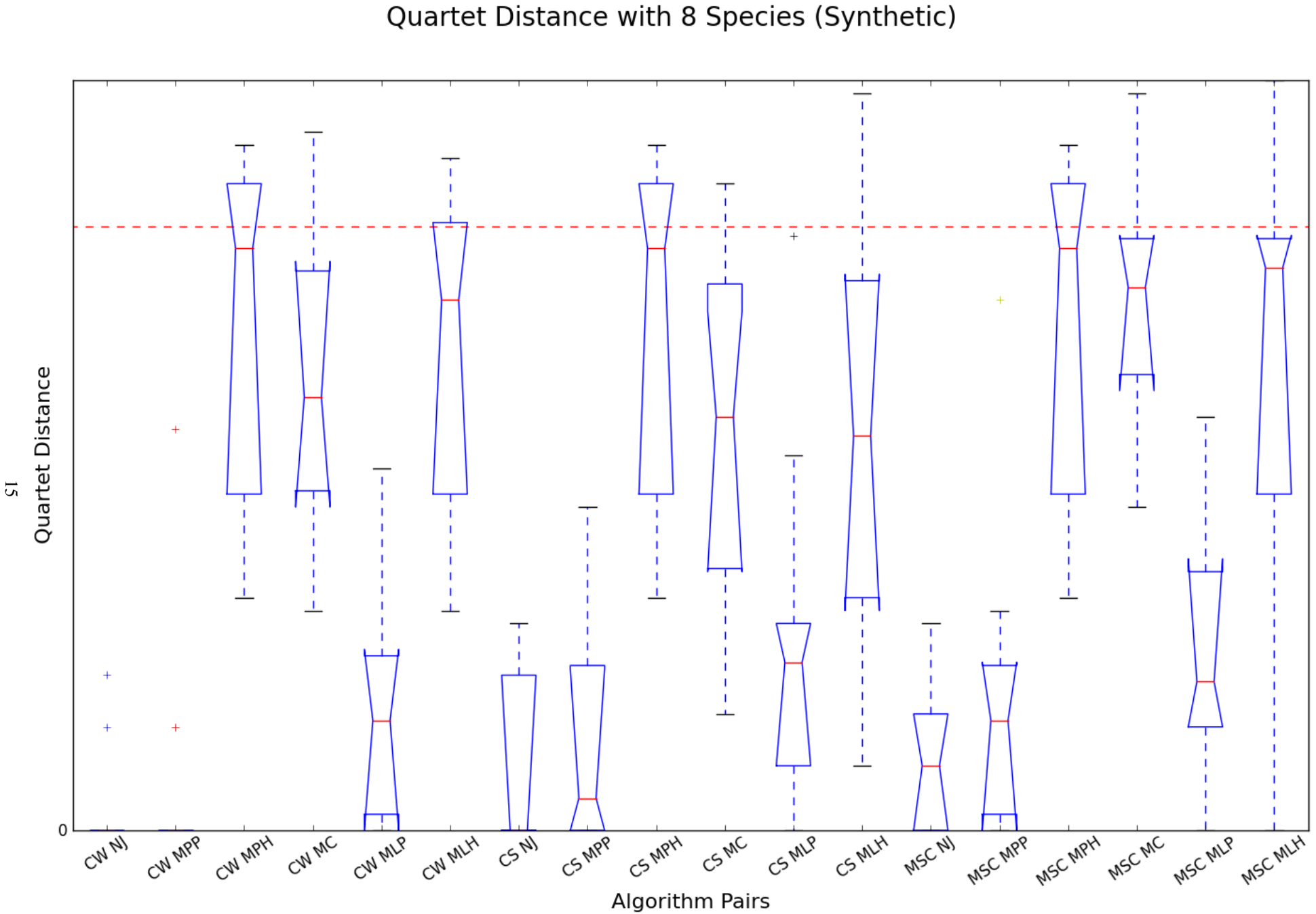

Figure 5: The resulting total quartet distance between each method's generated trees and the source tree which is assumed to be correct. Higher values are considered to be further from the original tree. The red dotted line represents the distance between a randomly guessed tree and the original tree.

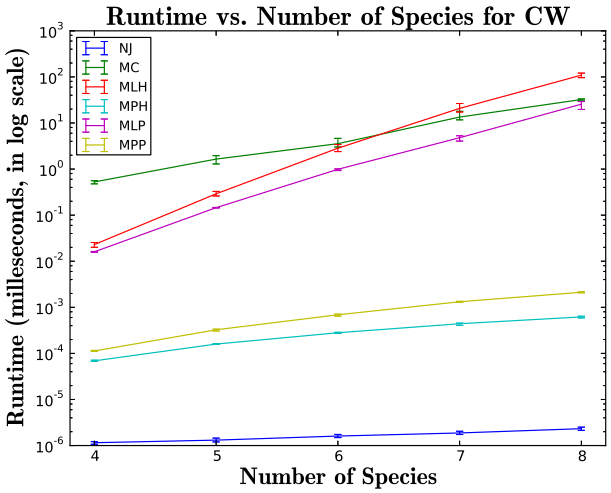

Figure 8: Runtime comparison of reconstruction algorithms on synthetic data sets using Clustal-W as our alignment algorithm.

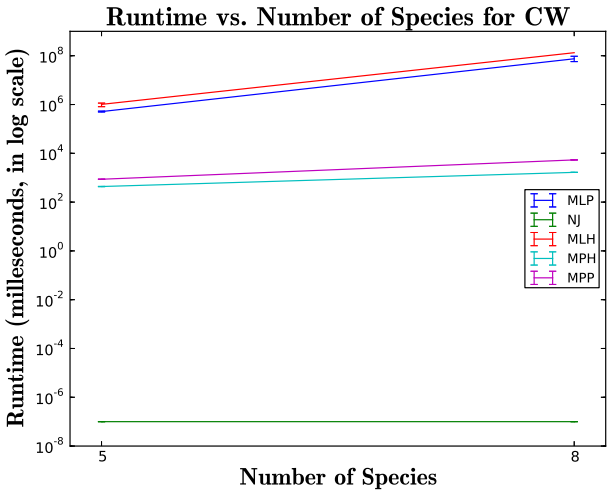

Figure 9: Runtime comparison of reconstruction algorithms on real data sets using Clustal-W as our alignment algorithm.

approach is that it makes local decisions when searching through the space of possible topologies, and thus, a resulting tree topology can sometimes be unreliable. However, this does not have a big impact when trees are evaluated on pairwise distance, because it only measures the distances between pairs of leaf nodes; pairwise distance does not account for the position of a node within a tree. On the other hand, the hillclimb approach can sometime gets caught in a local optimum. In our case, it is likely that the downside of the hillclimb approach had a larger impact on resulting trees.

For quartet distance metric, Maximum Parsimony also achieved results with equally high accuracies as NJ and ML methods. This is in agreement with [25], which suggests that under low nucleotide substitution rates, NJ, MP, and ML should be equally successful. Hill-climbing approaches performed significantly worse than progressive approaches in terms of quartet distance metric. Again, this is likely due to the fact that hill-climbing approaches can sometimes only find the local optimal topology rather than the true global optimum.

NJ produced accurate results in our study for both pairwise distance and quartet distance. This was in accordance with [37], which suggested that NJ performs slightly better than ML methods under constant nucleotide substitution rates.

MCMC performed significantly worse than other reconstruction algorithms in terms of both quartet and pairwise distance metrics. This is likely because we did not run the algorithm long enough to find a reasonable global optimum. In order to find trees close to the global optimum in our sample space, the suggested number of iterations was 2000 [26]. Due to the time constraints in our project, we only ran 200 iterations.

In Figure 7, we compare the correctness of tree output of various algorithms when problems increase in size. Notably MCMC becomes increasingly less accurate when the number of species increases. This is likely a reflection of the fact that tree space is less able to be explored in a fixed 200 iterations when more species are added. Furthermore, using likelihood and hill climbing appears to become less correct and more variable for larger problems as well. Because the objective function increases significantly in complexity as the number of species increases, it's likely the case that getting caught in local optima becomes increasingly common. On the other hand, NJ and MLP perform relatively consistently, indicating their potential accuracy on larger datasets.

Running Time

The expected efficiency of NJ is consistent with our experimental results. ML, on the other hand, is considered computationally costly with progressive topology ($O(mn^6)$) and with the hill-climb approach ($O(kmn^5)$) due to the branch optimization process. MCMC, which uses ML's likelihood calculations, is also computationally expensive. These theoretical observations also agree with our experimental results. MP had a performance speed that fell in between NJ and ML, which also fits our expectation.

Parallelization

Our parallelization results were promising but highly inconsistent. The main downside of our consideration was its relative simplicity; parallelizing in accordance with the site independence assumption is a good place to start, but it is only a first step. Many other aspects of these algorithms, such as

recursive tree traversals, can be easily and completely parallelized, and it would be exciting to pursue new avenues of optimization, particularly on specialized hardware (MapReduce clusters, GPGPUs) in future work.

The results for MCMC, where speedup decreased with increasing input size, were among the most intriguing in our entire study. Perhaps, in this case, the sequential version of this algorithm benefits from some under-the-hood compiler optimization that the other likelihood-based methods (MLH, MLP) do not.

In total, our investigation of parallelism in phylogenetic reconstruction is a small first step towards understanding the complexity involved in this rich field of optimization.

Conclusion

Based on our experiments with both synthetic and real data, and our analysis of both run-time efficiencies and the accuracies of our algorithms, we conclude that for data sets with similar properties to those of our data (i.e. short sequences, low and constant nucleotide substitution rates), Neighbor Joining should be used in order to achieve the best efficiency and accurate results. Maximum Likelihood with progressive tree search creates equally accurate trees, but is far more computationally expensive.

However, due to the complexity of the real-world data sets and their varying characteristics, algorithms should be carefully chosen in order to obtain accurate results. Based on our results, we cannot determine the total superiority of a specific reconstruction method. In addition, there is no guarantee that the details of our implementation match those in the literature we surveyed. Nonetheless, our study provides a comparative approach that future research alike can undertake.

In the future, we would like to examine more types of synthetic data (perhaps varying mutation characteristics) and optimize our implementations in accordance with modern advancements to get a better sense of the current state of the field.

Acknowledgments

First and foremost, we would like to acknowledge Sherri Goings for her continued advising and support throughout this project. We also thank Robert Dobrow for his assistance with Markov Chain Monte Carlo and Mike Tie for maintaining our computational resources.

References

[1] K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25(2-3):251–278, 1999.

[2] D. Baum. Reading a phlogenetic tree: The meaning of monophletic groups. *Nature Edcation*, 1, 2008.

[3] F. S. Brinkman and D. D. Leipe. Phylogenetic analysis. *Bioinformatics: a practical guide to the analysis of genes and proteins*, pages 323–358, 2001.

[4] G. S. Brodal, R. Fagerberg, and C. N. Pedersen. Computing the quartet distance between evolutionary trees in time o (n log n). *Algorithmica*, 38(2):377–395, 2004.

[5] T. Burr. Predicting virus evolution. In M. A. Mahdavi, editor, *Bioinformatics - Trends and Methodologies*. InTech, 2011.

[6] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. J. Gibson, D. G. Higgins, and J. D. Thompson. Multiple sequence alignment with the clustal series of programs. *Nucleic acids research*, 31(13):3497–3500, 2003.

[7] C. Christiansen, T. Mailund, C. N. Pedersen, and M. Randers. *Computing the quartet distance between trees of arbitrary degree*. Springer, 2005.

[8] M. Cotten, T. T. Lam, S. J. Watson, A. L. Palser, V. Petrova, P. Grant, O. G. Pybus, A. Rambaut, Y. Guan, D. Pillay, et al. Full-genome deep sequencing and phylogenetic analysis of novel human betacoronavirus. *Emerging infectious diseases*, 19(5):736, 2013.

[9] R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of computational biology*, 9(5):687–705, 2002.

[10] R. C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[11] R. C. Edgar and S. Batzoglou. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373, 2006.

[12] R. C. Edgar and K. Sjölander. A comparison of scoring functions for protein sequence profile alignment. *Bioinformatics*, 20(8):1301–1308, 2004.

[13] G. F. Estabrook, F. McMorris, and C. A. Meacham. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Systematic Biology*, 34(2):193–200, 1985.

[14] S. Evans and T. Warnow. Phylogenetic analyses of alignments with gaps.

[15] J. S. Farris, V. A. Albert, M. Källersjö, D. Lipscomb, and A. G. Kluge. Parsimony jackknifing outperforms neighbor-joining. *Cladistics*, 12(2):99–124, 1996.

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

[17] J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17(6):368–376, 1981.

[18] J. Felsenstein. Statistical inference of phylogenies. *Journal of the Royal Statistical Society, Series A*, 146:246–272, 1983.

[19] W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[20] W. M. Fitch, E. Margoliash, et al. Construction of phylogenetic trees. *Science*, 155(760):279–284, 1967.

[21] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[22] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of mathematical biology*, 55(1):141–154, 1993.

[23] M. Hasegawa and M. Fujiwara. Relative efficiencies of the maximum likelihood, maximum parsimony, and neighbor-joining methods for estimating protein phylogeny. *Molecular phylogenetics and evolution*, 2(1):1–5, 1993.

[24] V. H. HEYWOOD and J. McNEILL. Phenetic and phylogenetic classification. *Nature*, 203(4951):1220–1224, 1964.

[25] M. K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11(3):459–468, 1994.

[26] B. Larget and D. L. Simon. Markov chain monte carlo algorithms for the bayesian analysis of phylogenetic trees. *Molecular Biology and Evolution*, 16:750–759, 1999.

[27] W. Li, Z. Shi, M. Yu, W. Ren, C. Smith, J. H. Epstein, H. Wang, G. Crameri, Z. Hu, H. Zhang, et al. Bats are natural reservoirs of sars-like coronaviruses. *Science*, 310(5748):676–679, 2005.

[28] B. Mau, M. A. Newton, and B. Larget. Bayesian phylogenetic inference via markov chain monte carlo methods. *Biometrics*, 55(1):1–12, 1999.

[29] R. Mihaescu, D. Levy, and L. Pachter. Why neighbor-joining works. *Algorithmica*, 54(1):1–24, 2009.

[30] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, 2000.

[31] X. C. Oxidase and X. Lipoproteins. Electron transport and oxidative phosphorylation. *Advances in Enzymology and Related Areas of Molecular Biology*, 21:73, 2009.

[32] R. Peng, B. Zeng, X. Meng, B. Yue, Z. Zhang, and F. Zou. The complete mitochondrial genome and phylogenetic analysis of the giant panda (¡ i¿ ailuropoda melanoleuca¡/i¿). *Gene*, 397(1):76–83, 2007.

[33] P. Perelman, W. E. Johnson, C. Roos, H. N. Seuánez, J. E. Horvath, M. A. Moreira, B. Kessing, J. Pontius, M. Roelke, Y. Rumpler, et al. A molecular phylogeny of living primates. *PLoS genetics*, 7(3):e1001342, 2011.

[34] S. Pillai, B. Good, S. Pond, W. J.K., M. Strain, D. Richman, and S. D.M. Semen-specific genetic characteristics of humans immunodeficiency virus type 1 env. *Journal of Virology*, 79:1734–1742, 2005.

[35] P. Puigbò, Y. I. Wolf, and E. V. Koonin. The tree and net components of prokaryote evolution. *Genome biology and evolution*, 2:745, 2010.

[36] S. S. Qian, C. A. Stow, and M. E. Borsuk. On monte carlo methods for bayesian inference. *Ecological Modelling*, 159(2):269–277, 2003.

[37] N. Saitou and T. Imanishi. Relative efficiencies of the fitch-margoliash, maximum-parsimony, maximum-likelihood, minimum-evolution, and neighbor-joining methods of phylogenetic tree construction in obtaining the correct tree. *Mol. Biol. Evol*, 6(5):514–525, 1989.

[38] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[39] L. Salter. Algorithms for phylogenetic tree reconstruction. In *Proceeding of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, volume 2, pages 459–465. Citeseer, 2000.

[40] H. A. Schmidt, K. Strimmer, M. Vingron, and A. von Haeseler. Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3):502–504, 2002.

[41] J. Sourdis and M. Nei. Relative efficiencies of the maximum parsimony and distance-matrix methods in obtaining the correct phylogenetic tree. *Molecular biology and evolution*, 5(3):298–311, 1988.

[42] M. A. Steel and D. Penny. Distributions of tree comparison metricssome new results. *Systematic Biology*, 42(2):126–141, 1993.

[43] K. Tamura, M. Nei, and S. Kumar. Prospects for inferring very large phylogenies by using the neighbor-joining method. *Proceedings of the National Academy of Sciences of the United States of America*, 101(30):11030–11035, 2004.

[44] Y. Tateno, N. Takezaki, and M. Nei. Relative efficiencies of the maximum-likelihood, neighbor-joining, and maximum-parsimony methods when substitution rate varies with site. *Molecular Biology and Evolution*, 11(2):261–277, 1994.

[45] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

[46] M. S. Waterman and T. F. Smith. On the similarity of dendrograms. *Journal of Theoretical Biology*, 73(4):789–800, 1978.

[47] W. Williams and H. Clifford. On the comparison of two classifications of the same set of elements. *Taxon*, pages 519–522, 1971.

[48] X.-G. Yang. Bayesian inference of cetacean phylogeny based on mitochondrial genomes. *Biologia*, 64(4):811–818, 2009.

[49] Z. Yang and B. Rannala. Bayesian phylogenetic inference using dna sequences: a markov chain monte carlo method. *Molecular biology and evolution*, 14(7):717–724, 1997.