TreeTOP: Using Triples to Quickly Reconstruct Time-Oriented Phylogenetic Trees

A thesis submitted for the degree
Bachelor of Advanced Computing (Research and Development)

24 pt Honours project, S2/S1 2022–2023

By:
Robert Neil McArthur

Supervisors:
Prof. Gavin Huttley
Dr. Yu Lin
Dr. Ahad N. Zehmakan

Australian National University

School of Computing
College of Engineering, Computing and Cybernetics (CECC)
The Australian National University

December 2023

Declaration:

I declare that this work:

- upholds the principles of academic integrity, as defined in the University Academic Misconduct Rules;
- is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the class summary and/or Wattle site;
- is produced for the purposes of this assessment task and has not been submitted for assessment in any other context, except where authorised in writing by the course convener;
- gives appropriate acknowledgement of the ideas, scholarship and intellectual property of others insofar as these have been used;
- in no part involves copying, cheating, collusion, fabrication, plagiarism or recycling.

May, Robert Neil McArthur

Acknowledgements

This thesis, or indeed my time at university, would not have been possible without the support of so many people along the way.

I would like to begin by expressing my absolute sincerest of thanks to the late Dr. Yu Lin, who tragically passed away towards the end of last year. Yu Lin was a truly exceptional person of great character and intellect. He was an incredibly fun and kind person who I learnt so much from under his supervision of my honours. I remember very fondly our conversations, and I only wish I had the opportunity to learn more from him.

I would also like to extend my deepest gratitude to my supervisors Prof. Gavin Huttley and Dr. Ahad N. Zehmakan. Gavin and Ahad's supervision have been truly exceptional, and I feel like I would not have been able to come close to this level of work without them. I'm extremely grateful to Gavin, who took me into his lab at the beginning of 2022 despite my lack of university biology study. He has taught me all of the biological knowledge necessary for this thesis, and has assisted tremendously with direction with ideas for computational methods to help tackle the problem. He even allowed me to present parts of my work at the end of 2022 at Phylomania in Tasmania! I've thoroughly enjoyed his brilliant dry sense of humour. I am also incredibly grateful to Ahad, who stepped in halfway through this project. He has had brilliant insights into graph based methods which allowed me to quickly enumerate different ideas for what was the main hurdle of this assignment (the supertree method) which I had previously spent far too much time on. More so than just academic supervision, I have thoroughly enjoyed just our simple more general chats. All of my supervisors have provided exceptionally fantastic supervision over the course of my honours. I feel truly indebted to them, and this work simply wouldn't have at all been possible without them.

I would also like to thank the other members of the Huttley lab, who have provided great insights and feedback on various topics during chats and our weekly meetings. In particular, I wanted to pay a very special thanks to Kath who provided exceptional feedback from proofreading of the thesis. She provided so many numerous suggestions when it came to making biological concepts clearer to a reader, as well as enhancing the clarity of the overall thesis more broadly. I feel like the quality of this thesis is a definite level higher thanks to her suggestions.

I would also like to pay a massive thanks to my friends, for keeping me sane throughout my honours. There are so many people who are worth thanking but in particular I would like to thank my board gaming friends Pranay (who also helped proofread my thesis!), Elly, and Tanya, as well as my friend and roommate Damon. They have all been tremendous pillars of support throughout my university journey. I have thoroughly enjoyed every single moment I have spent with them, and I look forward to seeing how the next chapters of our lives evolve moving forwards.

Finally, but most importantly, I would like to thank all my family and in particular dedicate this thesis to my parents Adrian and Luci McArthur. They have both sacrificed a lot throughout all stages of my life to enable me to get to where I am today. None of the above would have been possible without them. I am eternally grateful for all of the opportunities they have given me to allow me to maximise my potential and get to where I am now. Love you mum and dad!

This thesis was also undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

Abstract

Time-oriented phylogenetic tree reconstruction is the task of converting the DNA sequences of a set of taxa into a tree that displays the evolutionary history and relationships between organisms over time. There are different types of algorithms for producing these phylogenetic trees, the most meaningful of these being maximum-likelihood methods, as they aim to find the best possible tree which fits some model of evolution. A major challenge with this class of methods, however, is the size of the tree space that needs to be explored to find the best phylogenetic tree. Given a set of n taxa, there are $(2n-3)!!$ possible rooted phylogenetic trees (Felsenstein, 1978). To address this issue, state-of-the-art algorithms such as IQTree (Minh et al., 2020) employ some form of search over the tree space to make the problem tractable. Kaehler (2017) has more recently conjectured a method whereby under a strand-symmetric Markov model of evolution, for three taxa, it is possible to very quickly find the maximum-likelihood rooted phylogenetic tree in a way that skips the numerical optimisation step. We call such three taxa, rooted phylogenetic trees "triples". This gives rise to the idea of a new type of algorithm whereby we use these maximum-likelihood triples to puzzle together the full phylogenetic tree. This presents new challenges, given that for n taxa, there are $\binom{n}{3}$ total triples, and these triples are not necessarily consistent given the inherent noisiness of the data. The algorithms must be robust enough to deal with these problems. This thesis presents three main contributions to this area. The first is an algorithm called TripleTree, which can quickly convert a set of triples into phylogenetic trees of modest sizes (the time taken to compute all the triples being the primary bottleneck). We also present Spectral Cluster Supertree, a modified version of the well-studied Min-Cut Supertree algorithm (Semple and Steel, 2000), which is much more scalable in terms of merging a group of rooted phylogenetic trees into a single rooted phylogenetic tree. The thesis culminates by combining these with the disk-covering method divide-and-conquer framework (Roshan et al., 2004) to present TreeTOP: an algorithm which uses triples to quickly reconstruct **t**ime-**o**riented **p**hylogenetic **trees**. Under known triples for upper thousands of taxa, TreeTOP is capable of reconstructing phylogenetic trees in tens of minutes for known triples and hours for DNA sequences of length 30,000.

Table of Contents

Chapter 1

Introduction

Phylogenetics is the study of the evolutionary history and relationships between groups of organisms. Central to phylogenetics is the idea of a phylogenetic tree, a diagram showing how life has evolved and branched over time. All life on earth shares a common ancestor, but very similar species, such as humans and gorillas, would share a much more recent common ancestor than either of those and a cat, for instance. Phylogenetics serves as a powerful framework that untangles life's evolutionary history.

Phylogenetic trees have an extraordinary range of applications. They have been used in epidemiology - enhancing our understanding of the HIV-1 virus (Korber et al., 2000) and detecting new variants of SARS-CoV-2 (Tang et al., 2021). They have been used in forensics - assisting in criminal convictions (González-Candelas et al., 2013), conservation biology of whales (Baker and Palumbi, 1994), cancer diagnosis (Abu-Asab et al., 2006), and for developping new drugs (Searls, 2003). Moreover, phylogenetic trees provide valuable insights into our own tree of life. The ability to construct large phylogenetic trees efficiently in a statistically meaningful way is of considerable importance.

Phylogenetic trees can be constructed from the DNA sequences of a set of organisms. A DNA sequence can be thought of as a digital code that encodes the instructions for life. It can be represented using a sequence of characters 'A', 'C', 'G' and 'T', each corresponding to biological molecules that make up the DNA. DNA can undergo several forms of mutation as organisms reproduce. These include insertion, where new segments of DNA are added to the sequence; deletion, where regions of DNA are removed from the sequence; and substitution, where characters in the sequence may be randomly modified to any other. Mutation events such as insertion and deletion require the DNA sequences to be aligned for comparison. Evolutionary models, biologically informed Markov models, can be used to represent these substitution events. These allow for statistically meaningful phylogenetic trees that show how a set of organisms have evolved and diverged over time to be generated.

Two main categories of algorithms for reconstructing phylogenetic trees are statistically meaningful maximum-likelihood techniques and fast heuristic-based methods. The heuristic-based methods can be very quick in generating phylogenetic trees; however, they do not necessarily rely on an evolutionary model and are thus a less statistically meaningful generation process (Saitou and Nei, 1987; Sneath and Sokal, 1973). On the other hand, maximum-likelihood numerical optimisation methods aim to find the optimal phylogenetic tree for a given set of taxa under an evolutionary model. Optimising to an evolutionary model means the results of these methods are significantly more statistically meaningful; however, this also means that it can take a much longer time to compute. For a set of n taxa (the leaves of the phylogenetic tree), there are $(2n-3)!!$ possible rooted phylogenetic trees (Felsenstein, 1978). In this context, rooted means the phylogenetic tree shows the direction of time and how the taxa have diverged from their collective common ancestor (the root). Clearly, enumerating all trees and optimising to an evolutionary model quickly becomes a computationally infeasible task as the problem size grows. Some maximum-likelihood methods, such as IQ-TREE (Nguyen et al., 2015) (which we will later briefly discuss), improve upon this by employing a search over the tree space. While maximum-likelihood techniques are statistically meaningful, they take much longer to resolve than heuristic-based methods. As many typical applications of phylogenetic trees concern large numbers of taxa, using maximum-likelihood techniques to generate statistically meaningful trees within a reasonable time-bound is of clear importance.

A work by Kaehler (2017) proved that the root is identifiable for three taxa under a particular evolutionary model (specifically, a non-stationary, strand-symmetric nucleotide substitution model). That is, for those three taxa, we can form a rooted phylogenetic tree displaying how they have diverged over time. More recently, Kaehler has conjectured that there is a numerical trick involving eigendecomposition that allows efficient resolution of maximum-likelihood rooted phylogenetic trees for three taxa in a way that skips the numerical optimisation process (Kaehler, Personal Communication), generating so-called rooted triples. This gives rise to the possibility of an entirely new class of algorithm, whereby for a set of taxa, we can employ this process to all groups of three to form a new set of these rooted triples. These triples can then be used as building blocks that can be puzzled together to reconstruct the complete rooted phylogenetic tree describing the full set of taxa,

This work provides three important contributions to this problem. First is an algorithm called TripleTree which, given a collection of triples, can efficiently reconstruct rooted phylogenetic trees for small problem sizes. The second is Spectral Cluster Supertree, a modification called Min-Cut Supertree (Semple and Steel, 2000), that has proofs relating to desirable properties the resulting supertree topology obeys. Our final contribution is the result of combining these techniques with the disk-covering method, a divide-and-conquer framework that has proofs relating to its convergence rate (Huson et al., 1999a). The combination gives TreeTOP: an algorithm that uses triples as a building block to reconstruct time-oriented phylogenetic trees from DNA sequences efficiently.

The proof by Kaehler (2017) that a strand-symmetric DNA substitution model can recover the chronological root for three sequences, thus identifying rooted triples, is a fundamental component for exploiting time-orientation. The conjecture by Kaehler regarding the feasibility of a decomposition solution in identifying the root is appealing but not essential for the viability of our approach. For now, we rely instead upon a simpler process for determining triples that will be later described. We show that our new method of using triples to reconstruct rooted phylogenetic trees in TreeTOP is feasible and scalable for handling large quantities of taxa.

We discuss the extensive background for our solution to this problem in chapter 2. In chapter 3, we present a new algorithm for constructing rooted phylogenetic trees from these rooted triples and a modified method for merging multiple rooted phylogenetic trees together efficiently. The chapter culminates as we present a new algorithm relying on these results that is able to scale to efficiently resolve phylogenetic trees with many thousands of taxa. We analyse the performance of these new algorithms in chapter 4 before concluding and providing direction for future work in chapter 5.

Chapter 2

Background

The primary result of this thesis is TreeTOP, a technique that enables us to efficiently reconstruct rooted phylogenetic trees for many thousands of taxa. TreeTOP is an amalgamation of many different underlying methods - both existing, new and modified. This chapter covers the background necessary to understand the application area, the nature of the data and characteristic algorithms. It begins by introducing sequence alignments, phylogenetic trees and triples. The scalability challenges of existing methods are then discussed. From there, we build the basis for our new techniques by introducing: triples; a molecular clock approach for their estimation; the disk-covering divide-and-conquer algorithm; distance measures between phylogenetic trees; and finally, discussing in slightly greater detail some other phylogenetic reconstruction techniques.

2.1 Sequence Alignments

For the purposes of constructing a phylogenetic tree for a set of taxa, the DNA sequences must be compared. Sequence alignments allow us to achieve this by, for a set of these sequences, aligning the characters by identifying the most likely counterparts in all other sequences (Corpet, 1988). The result is a matrix with sequence names as row labels and columns being the characters descended from the same character in the common ancestor of the species.

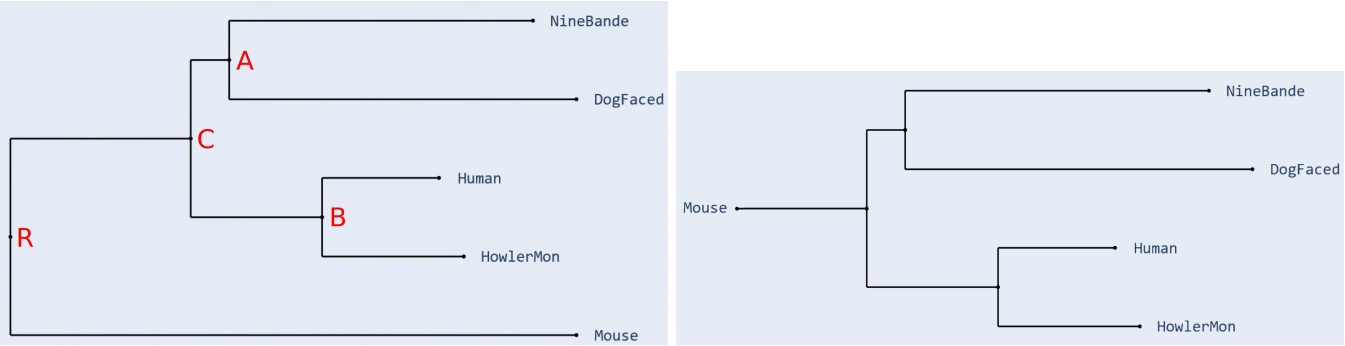
Figure 2.1 shows a short region of DNA sequence for five organisms and the positions at which the characters differ. All characters are displayed in the top row. A '.' indicates that the character is the same as the top row. Otherwise, the differing character is shown. In this example, the human and howler monkey sequences differ only at the second rendered G, displayed in the howler monkey's row. This would likely imply (at least for the displayed part of the alignment) that the human and howler monkey are closely related. The relationships between the aligned sequences can be quantified

using a distance measure. There exist many measures of distance that can be used. We primarily used an additive measure known as paralinear distance (Lake, 1994). It is important to note that this distance measure requires computing the determinant of a square matrix and thus can sometimes fail to compute for some problems. In such instances, we used the TN93 distance as a fallback (Tamura and Nei, 1993).

Figure 2.1: A sequence alignment for five taxa. Each row displays part of the DNA sequence for the respective organisms. A '.' is displayed if the character is the same as in the top row. (Knight et al., 2007).

2.2 Phylogenetic Trees

The previous section discussed how sequence alignment may be used to discover the shared ancestry of a group of taxa. A phylogenetic tree is a graph that displays these relationships. Each leaf node corresponds to a single taxon, and the internal nodes of the tree show where a speciation event has occurred.



(a) A rooted phylogenetic tree.

(b) An unrooted phylogenetic tree.

Figure 2.2: Simple phylogenetic trees for the sequence alignment in Figure 2.1. They display how the organisms are related to one another. The left image shows a rooted tree, with labelled internal nodes representing the common ancestor of the descending tips. The right image is the respective unrooted variant of the tree. (Knight et al., 2007).

Figure 2.2a displays a simple rooted phylogenetic tree for the set of five taxa used in the example sequence alignment in Figure 2.1. It displays how the organisms are related. For example, "Human" and "HowlerMon" appear together in the tree, sharing a common ancestor labelled **B**. As do "NineBande" and "DogFaced", labelled **A**. **A** and **B** also share a common ancestor labelled **C**, and then the root of the tree, labelled **R**, is the common ancestor of all nodes in the tree.

We note here that there are two types of phylogenetic trees - rooted and unrooted. The key difference between the two is that a rooted phylogenetic tree, such as the one in Figure 2.2a, shows the direction of time - the progression of divergence of taxa from the common ancestor (the root of the tree). Unrooted phylogenetic trees, of course, have no root and instead merely show how different organisms relate to one another. Figure 2.2b shows the respective unrooted tree that is formed by replacing the root of the tree labelled R with the "Mouse" node. All internal nodes of the tree would then become of equal degree, three, and the graph then would only show relationships between the taxa without encoding the direction of time. Many relevant algorithms which deal with phylogenetic trees work with the unrooted variant. Motivated by Kaehler's (2017) result, which identifies the conditions for identifying the root for a set of 3 taxa, this thesis is only concerned with generating rooted phylogenetic trees which capture more information than their unrooted counterparts.

In some cases, phylogenetic tees are multifurcating. That is, some internal nodes may have more than two children. This usually indicates that a method was unable to resolve precisely how the speciation event occurred at a given node for the immediate descendants. We will be restricting our method to only generating bifurcating phylogenetic trees. That is, every internal node will have exactly two children.

2.3 Scalability Challenges of Probabilistic Methods

Most phylogenetic reconstruction algorithms can be divided into two broad categories, heuristic methods and probabilistic methods (e.g. maximum-likelihood). Heuristic methods, such as neighbour joining (Saitou and Nei, 1987) and UPGMA (Sokal, 1958), can quickly scale to handle many taxa. These methods, while time efficient and scalable, generate less meaningful results than probabilistic methods that fit data to a model of evolution. Heuristic methods are also less efficient in their usage of information in the underlying data (Roch, 2010).

If a full maximum-likelihood approach was undertaken, one would need to explore the entire tree space to evaluate each tree. As the number of rooted trees is factorial with the number of taxa, this rapidly becomes infeasible to compute. State-of-the-art methods such as IQ-TREE (Nguyen et al., 2015; Minh et al., 2020) aim instead to maximise the likelihood of the resulting phylogenetic tree through an exploration of the tree space. Through personal communication with Minh Bui, depending on various factors, IQ-TREE may still take days to fully resolve tens of thousands of taxa.

2.4 Triples

2.4.1 Rooted Triples

This thesis proposes using the notion of a rooted triple as the core building block for phylogenetic tree reconstruction. Put simply, a rooted triple is a rooted phylogenetic tree for a set of three taxa. For three taxa $\{a, b, c\}$, suppose b and c share a common ancestor more recently than either with a. We write the triple $\{a, \{b, c\}\}$ to represent this phylogenetic tree. We may also state that a is the outgroup, while b and c form the ingroup. Figure 2.3 illustrates this with a further example, where the human and gorilla form the ingroup, and a cat is the outgroup. We will use these triples as the fundamental unit of information to reconstruct complete rooted phylogenetic trees.

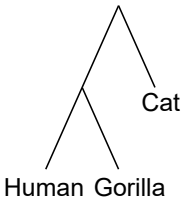


Figure 2.3: A simple rooted triple involving the human, gorilla and cat. A human and a gorilla, being great apes, share a more recent common ancestor than a cat. They thus appear together in the triple. We write this triple as $\{cat, \{human, gorilla\}\}$.

2.4.2 From Phylogenetic Trees to Rooted Triples

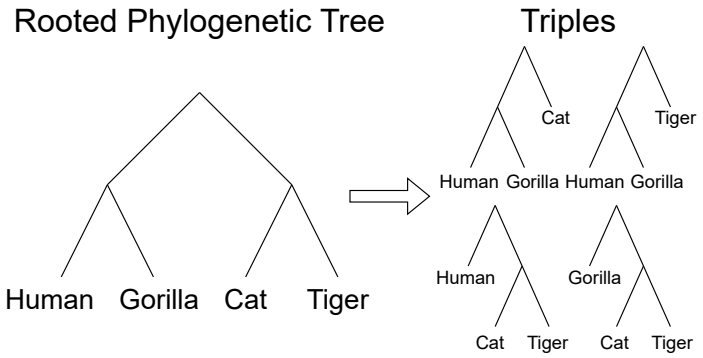


Figure 2.4: A simple rooted phylogenetic tree and the triples obtained from it. Triples can be constructed from any phylogenetic tree by seeing where a group of three taxa are partitioned.

Given a rooted phylogenetic tree, it is trivial to compute all corresponding triples. Figure 2.4 illustrates this with a small example phylogenetic tree and the triples obtained from it. Given three taxa, to find a triple, one must simply observe how they are split in the tree. Consider the three species, human, gorilla and cat. In the tree, the cat leaf is split from both the human and gorilla leaves at the root node. Hence, the triple is $\{cat, \{human, gorilla\}\}$. In larger, more complex phylogenetic trees, the split used to find the triple may occur at any internal node - not necessarily the root as in this example. This process can be done for all subsets of three taxa to compute all triples. For a set of n taxa, there are $\binom{n}{3}$ triples. This presents computational challenges as the size of the tree grows.

At the time of writing, no published algorithm exists for recovering rooted phylogenetic trees efficiently from a set of possibly inconsistent rooted triples. This thesis presents a means of solving this problem with three contributions. First, an algorithm that speedily converts triples into rooted phylogenetic trees for modest problem sizes, TripleTree. Second, an algorithm that can scale to merge together multiple large rooted phylogenetic trees into a single tree, Spectral Cluster Supertree. Finally, the main result of the thesis is TreeTOP, which combines these with a divide-and-conquer method known as the disk-covering method (Roshan et al., 2004) to efficiently convert these rooted triples into a rooted phylogenetic tree. This thesis aims to demonstrate the feasibility of this method.

2.5 Molecular Clock

Kaehler (2017) showed that for two or more sequences, the root of a phylogenetic tree can be identified when the sequences have evolved under a stochastic process that includes the constraint of DNA "strand-symmetry". Kaehler conjecture that a corollary of his proof allows us to compute this for three taxa in a way that skips the maximum-likelihood process (Kaehler, Personal Communication). As this remains to be proven, instead of using this method to estimate the rooted triples, we will instead rely on the molecular clock hypothesis. This will be done as proof of concept to demonstrate the feasibility of using triples to efficiently reconstruct large phylogenetic trees.

2.5.1 Definition

The strict molecular clock is the assumption that the rate at which evolution occurs (in terms of the rate at which substitutions occur) is the same constant value over a set of organisms (Zuckerkandl and Pauling, 1965). This assumption is known to be generally false, as the rate of evolution can vary significantly between different lineages (Yang and Nielsen, 1998). Still, this can be a powerful tool to assist in the process of rooting phylogenetic trees.

2.5.2 Molecular Clock Rooting

Molecular clock rooting provides a quick method for computing triples from DNA sequences. For a set of three organisms, this can be done using pairwise distances between the aligned sequences (Yang and Rannala, 2012). This can be achieved by setting the sequence with maximum summed distance to the others as the outgroup, with the remaining two sequences forming the ingroup. As the molecular clock assumption is not always valid, there are methods that could be used to determine whether the assumption approximately holds (Holder and Lewis, 2003). If the molecular clock assumption does not hold, then an alternative method could be used to form the triple such as IQ-Tree (Minh et al., 2020). For now, as Kaehler's (2017) result remains to be proven, we will rely on molecular clock rooting for triple generation. This is done as a simple technique so the feasibility of the TreeTOP algorithm can be assessed.

2.6 Disk-Covering Method

The disk-covering method (Huson et al., 1999a,b; Roshan et al., 2004) describes a family of algorithms that employ a divide-and-conquer framework to approach the problem of phylogenetic tree reconstruction. These techniques are proved to, with high probability, reconstruct the true tree if the sequences grow polynomially with the number of leaves and arbitrary bounds placed on the edge lengths (Huson et al., 1999a). Of particular focus is DCM3 (Roshan et al., 2004) which employs its approach in four steps:

1. Using a guide tree to decompose the problem into overlapping subproblems.
2. Reconstructing the phylogenetic trees for the subproblems.
3. Merging the resulting phylogenetic trees together.
4. A refinement step.

For our purposes, we shall ignore the refinement step. The algorithm repeats these steps, iteratively improving the starting guide tree (which may be randomly initialised) until convergence to some phylogenetic tree. Of particular note is that the method was tailored to handle unrooted phylogenetic trees. As we must instead generate rooted topologies, an additional adjustment was required that will be later described.

2.6.1 Generating Subproblems

The main contribution of DCM3 was the provision of a technique to divide a set of taxa into overlapping subsets. This provides smaller problems one can more easily reconstruct phylogenetic trees for, the results of which can later be merged. The method assumes a guide tree, T, which can be initialised randomly and updated through successive iterations of the algorithm.

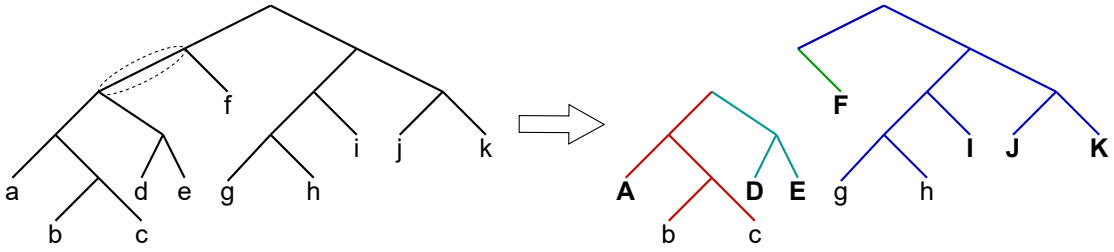


Figure 2.5: The removal of the circled edge splits the graph into the four coloured subtrees. The fully connected graph with nodes, the set of leaves closest to the root of each new subtree (capitalised and bolded), forms the short subtree for that edge.

Consider removing an internal edge, e, of a guide tree, T, from the graph in addition to the two internal nodes the edge connects. A special case is if the internal edge connects the root, in which case the other root edge and corresponding internal node are also removed. This disconnects the graph and creates four subtrees. Take the set of leaves being the closest leaves to the root of each subtree. We call the new fully-connected graph with nodes being this set the short subtree $K(e)$. This process is illustrated in Figure 2.5 for a single edge. By iterating over all internal edges e in T, the union of the nodes and edges of each $K(e)$ is called the short subtree graph, G. For the example, this is shown on the right of Figure 2.6.

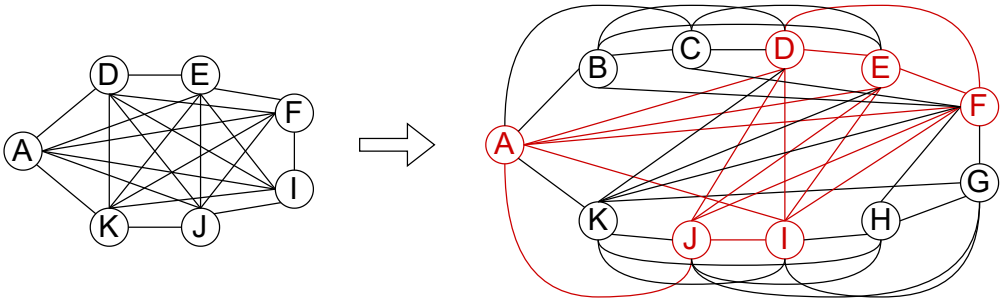


Figure 2.6: The left diagram displays the short subtree for the edge removed in Figure 2.5. By taking the union of all short subtrees for each edge in Figure 2.5, the short subtree graph on the right is obtained. The maximal clique separator is highlighted in red. This, and the components obtained when removing the separator from the graph, gives overlapping subproblems $\{A, B, C, D, E, F, I, J\}$ and $\{A, D, E, F, G, H, I, J, K\}$.

The short subtree graph is used to find the optimal partition of taxa into overlapping subsets. Optimality in this context minimises the size of the largest subproblem. The maximal clique separator, X, of the short subtree graph, G, is found. That is, the clique X of taxa that minimises the formula $\max_i |X \cup C_i|$, where C_i are the components of G after removing the nodes in X. The optimal partition can be computed in $\mathcal{O}(n^2)$ time, but a heuristic can be used to quickly estimate a suboptimal partition. Each $X \cup C_i$ then forms the subproblems to be solved. These subproblems can be recursively split further using the same process until they are smaller than a pre-specified maximal subproblem size. Figure 2.6 shows the subproblems generated by removing the maximal clique separator. The effect in decomposing problems is significantly more influential for larger graphs.

2.6.2 Solving Subproblems

Following the generation of the subproblems, phylogenetic trees are subsequently solved for each subproblem. The paper used TNT (Goloboff et al., 2008) to solve the subproblems; however, this can be substituted with any other reconstruction method. In this thesis, we will be using a new method, TripleTree, that relies on triples to reconstruct rooted phylogenetic trees for these subproblems.

2.6.3 Merging Solved Subproblems

The final step is to merge the solution to the solved subproblems. Supertree methods are those which transform a set of overlapping phylogenetic trees into a single phylogenetic tree containing all taxa. The DCM3 paper only dealt with unrooted phylogenetic trees and used the strict consensus merger (Day, 1985) method to form the supertree. The strict consensus merger method essentially contracts a minimum number of edges over the input trees such that the topology of the shared taxa form an identical backbone. Other taxa are then grafted onto this backbone. However, since our work must generate rooted phylogenetic trees, we will use a new technique called Spectral Cluster Supertree to perform this operation.

2.6.4 Parallelisation

A later paper (Coarfa et al., 2005) parallelised DCM3 using a controller-worker model. This significantly boosts the scale of problem sizes the algorithm can tackle. The controller sends to the workers either split, solver or merge commands. On receiving a split command, the worker splits a problem into overlapping subproblems and returns the result. On receiving a solve command, the worker reconstructs a phylogenetic tree for a given subproblem. Finally, on receiving a merge command, the worker returns the supertree of a set of solved subproblems. Another benefit to this parallelisation is that it allows the maximum subproblem size to be increased without causing a bottleneck. This can lead to a greater resolution of modification at each iteration, which may lead to faster convergence towards an estimated topology.

2.7 Measuring Tree Distances

This section defines two distance measures that can be used to compare phylogenetic trees. The metrics are used for the purposes of evaluation and additionally as a convergence measure for successive iterations of the disk-covering method.

2.7.1 Matching Distance

Matching distance (Lin et al., 2011) is a distance metric that can be used to evaluate the closeness of two phylogenetic trees. It can be regarded as a considerable improvement on other distance measures, such as Robinson-Foulds distance (Robinson and Foulds, 1981) which is known to lack robustness even under small changes - attaching a single leaf elsewhere in the tree can maximise the metric.

Every bifurcating phylogenetic tree can be represented as a set of bipartitions. The Robinson-Foulds distance measure can be rephrased as the minimum weight matching between the sets of bipartitions of the two input trees, with a weight of 0 if a node is shared between the trees and 1 otherwise. This binary measure clearly does not make use of all of the information inside the bipartitions. Matching distance instead uses the hamming distance between the bipartitions as the weight for the edge and then solves the minimum weight matching problem.

This type of minimum weight matching problem can be done in $\mathcal{O}\left(n^{\frac{5}{2}}\log(n)\right)$ time (Gabow and Tarjan, 1989). For our purposes, where a very quick convergence measure is required to avoid stalling the algorithm, this distance metric is not useful as a convergence measure. It is, however, a useful measure for comparing distances between the finally generated trees.

2.7.2 Jaccard Distance

The Jaccard Distance is a well-known metric for evaluating the dissimilarity between two sets A and B. The formula for the metric is given below:

$$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

The Jaccard Distance can also be applied to measure the dissimilarity between two phylogenetic trees. Given a phylogenetic tree, a set can be generated. Each internal node in the tree, excluding the root, corresponds to an element in the set containing all of that node's descendants. By computing this set for two phylogenetic trees, the Jaccard Distance between the two trees can be efficiently computed. This will later be used as a convergence measure in our algorithm.

2.8 Related Work

Felsenstein (1981) described a maximum-likelihood technique that constructed a phylogenetic tree by successively adding species. The tree begins as an unrooted two-species tree. When adding a new species, it is attempted to be added at every possible edge. The maximum likelihood from the DNA sequences of each resulting topology is computed using a dynamic programming algorithm. The tree topology is updated to the one which maximises the likelihood value. If the tree has more than four taxa, before attempting to add the next one, local rearrangements are made to the tree to see if the likelihood of the tree can be improved. The process continues one taxon at a time until a phylogenetic tree is constructed.

A modern, well-cited method for phylogenetic tree reconstruction using maximum-likelihood techniques is IQ-TREE (Nguyen et al., 2015). IQ-TREE employs a stochastic search algorithm over a set of initially generated trees to explore the tree space. At each iteration, one of the trees from the set is chosen at random. A sequence of nearest neighbour interchange (NNI) operations is applied to randomly perturb the tree. An NNI operation swaps two subtrees in a tree across an internal branch. To this randomly perturbed tree, a hill-climbing NNI search is then employed. Each NNI operation that increases an approximate likelihood of the resulting tree is stored. The best non-conflicting NNI operations are then simultaneously applied, and the likelihood is calculated by optimising the branch lengths under maximum likelihood. If the resulting likelihood is worse than the best single NNI operation, only that operation is instead applied.

From here, IQ-TREE replaces the worst tree in its set of trees with the new tree. The algorithm continues, randomly perturbing another tree selected from the set and optimising it. If, after 100 iterations, a new best tree in terms of maximum likelihood has not been found, the algorithm terminates, returning the best tree. IQ-Tree has undergone numerous improvements since its inception. The current version, IQ-TREE 2 (Minh et al., 2020), also includes automatic model selection (Kalyaanamoorthy et al., 2017), scalability improvements for computation across multiple cluster nodes; a memory saving mode, and other additional improvements.

There are other techniques that are used for phylogenetic reconstruction, such as RAxML (Stamatakis, 2014) and PhyML (Guindon et al., 2010), which contain their own optimisations for assisting in the search of the tree space. The final goal of the TreeTOP algorithm is to provide a new technique that can eventually rely on maximum-likelihood triples to efficiently reconstruct rooted phylogenetic trees. While the method which enables us to quickly calculate maximum-likelihood triples remains to be proven (Kaehler, 2017), we are temporarily using the strict molecular clock to compute these instead - an assumption that is known to be generally false. For now, we show the viability of using triples to compute large rooted phylogenetic trees with TreeTOP. We leave a complete comparison to existing methods as future work once the underlying maximum-likelihood triple technique is proven, and fair comparisons with respect to time and the accuracy of generated trees can be made.

Chapter 3

Methodology

This chapter presents the methodology behind TreeTOP: an efficient algorithm that uses triples to quickly reconstruct time-oriented phylogenetic trees for many taxa. The chapter begins with a brief discussion of the considerations that arise when handling imperfect data and other complexity challenges. We present TripleTree, an algorithm that can rapidly reconstruct a rooted phylogenetic tree using pre-computed triples. We also present Spectral Cluster Supertree, an enhancement to an existing supertree method that can scale to efficiently handle substantially more taxa. The chapter culminates by combining these two techniques with the disk-covering method to formally present the major contribution of this thesis, the TreeTOP algorithm.

3.1 Robustness Requirements

3.1.1 Handling Noisy Data

A crucial requirement for the algorithms is that they must be resilient to noisy data. Biological data has noise introduced to it over time due to the inherent randomness of biological processes. Consequently, this may mean that the rooted triples used by the algorithm may exhibit inconsistencies between them. This implies that in many cases, it may be impossible to fully resolve the ground truth phylogenetic tree, let alone a phylogenetic tree. Therefore, when formulating our methods, it is necessary that they are robust enough to handle the noisy information and form a best estimate.

3.1.2 Scalability

Another key consideration when working with triples to reconstruct phylogenetic trees is the sheer number of triples involved. Given a set of n taxa, there are $\binom{n}{3}$ triples, which is a fast-growing cubic. For example, a set of 10,000 taxa corresponds to approximately 167 billion triples. This is infeasible to compute, store, or even iterate over. The algorithms must be robust enough to scale to handle the quantity of this data. This motivated the use of the disk-covering method as a component of our approach. By dividing the problem into smaller overlapping subproblems, it is possible to compute a solution avoiding the computation of every triple - we only use the information that we need.

3.2 TripleTree: Efficiently Constructing Trees from Triples

We present a new algorithm, called TripleTree, that is able to very efficiently reconstruct a rooted phylogenetic tree given a set of triples. The algorithm consists of an initialisation phase, followed by a merge and aggregation step, which repeats until the rooted phylogenetic tree is formed.

We say a triple supports an unordered pair of taxa $\{a, b\}$ if the triple contains a and b as its ingroup. In the initialisation phase, a mapping from unordered pairs of taxa to integers is formed, $S(x)$, which is equal to the number of triples that support the unordered pair x of taxa.

A phylogenetic tree, T, is initialised as a graph with no edges, the set of nodes being the set of taxa. In the merge phase, the unordered pair, x, which maximises $S(x)$ is chosen. That is, the unordered pair with maximal support. Then in T, a new node labelled x is inserted, along with edges that connect the elements of x with this new node. x is additionally removed from S.

The aggregation step updates S to find the support for merging another node with the newly inserted node, $x = \{a, b\}$. For each node y that does not yet have a parent node in T, we compute the new support $S(\{x, y\}) = Agg(S(\{a, y\}), S(\{b, y\}))$. $\{a, y\}$ and $\{b, y\}$ are then both removed from the mapping. Here, Agg is some aggregation function. The max, as well as arithmetic, geometric and harmonic mean aggregation functions are investigated, and their effect on the accuracy of the reconstructed phylogenetic tree is later investigated in the evaluation section.

TripleTree subsequently repeats the merge and aggregation phases until there are no nodes left to merge and returns the resulting rooted phylogenetic tree. Pseudocode of the algorithm is provided on the following page:

Algorithm 1: The TripleTree Algorithm

Function `TripleTree(`*triples*`):`
- $T \leftarrow$ An edgeless graph with nodes being the taxa in triples
- $S \leftarrow$ `Initialise(`*triples*`)`
- **while** *T is not a tree* **do**
  - $x \leftarrow$ `Merge(`T,S`)`
  - `Aggregate(`S, x`)`
- **end**

Function `Initialise(`*triples*`):`
- $S \leftarrow$ Empty map from unordered pairs to integers. Default values for keys are 0.
- **for** *$triple \in triples$* **do**
  - $S[triple.ingroup]$+=1
- **end**
- **return** S

Function `Merge(`T, S`):`
- $x \leftarrow$ `max(`S`)`
- $T \leftarrow$ Add node x. Add edges making elements of x adjacent to new node.
- `del` S[x]
- **return** x

Function `Aggregate(`T, S, x`):`
- **for** *$y \in T.parentlessNodes$* **do**
  - $S[\{x, y\}] \leftarrow$ `Agg(`$[(S(\{i, y\})$ **for** $i \in x]$`)`
  - `del` $S[\{i, y\}]$ **for** $i \in x$
- **end**

Implementing the support mapping S as a priority queue built on a Fibonacci heap after the `Initialise` function, TripleTree runs in $\mathcal{O}\left(n^3\right)$ amortised time, where n is the number of taxa. This is due to the existence of $\binom{n}{3}$ triples for n taxa. The algorithm requires worst case $\mathcal{O}\left(n^2\right)$ memory complexity to store the unordered pairs in S. With a Fibonacci heap priority queue implementation, the merge function takes $\mathcal{O}\left(\log\left(n\right)\right)$ amortised time. The aggregate function takes $\mathcal{O}\left(p \log\left(n\right)\right)$ amortised time, where p is the number of parentless nodes, with $\log\left(n\right)$ coming from the delete complexity of Fibonacci heaps. Noting that at each iteration of the while loop, the number of parentless nodes decreases by one, and $n - 1$ iterations are required to form the phylogenetic tree, excluding the initialise function the algorithm would run in $\mathcal{O}\left(n^2 \log\left(n\right)\right)$ amortised time. In practice, it is indeed the generation of the triples and initialisation of the S mapping that is the bottleneck of the algorithm. So much so that using a hash map instead of any form of priority queue for S has minimal effect on the running time.

3.3 Supertree Methods for Rooted Graphs

The final phase of the disk-covering method (Roshan et al., 2004) was to use a supertree algorithm to merge solved subproblems into a single phylogenetic tree. Unlike the assumptions of the disk-covering method, which is concerned with generating unrooted topologies, we aim to generate rooted phylogenetic trees. This section presents a modification to a well-studied rooted supertree method that is able to scale better to become capable of merging significantly larger rooted phylogenetic trees.

3.3.1 Min-Cut Supertree

Min-Cut Supertree (Semple and Steel, 2000) is a well-studied method for merging rooted topologies. It has proofs relating to its ability to produce rooted supertrees with desirable properties. The algorithm recursively finds partitions of the taxa based on the input trees to construct the rooted supertree.

Proper Cluster Graph

Two taxa are said to belong to a proper cluster if they, from the root, lie on the same side of the tree. For a set of input trees T, Min-Cut Supertree computes what is known as the proper cluster graph. The nodes of the graph are the set of taxa appearing on any tree in T. An edge connects two taxa in the proper cluster graph if they belong to a proper cluster in any tree in T. Each edge is weighted by the number of trees the proper cluster appears in. Figure 3.1 illustrates this with an example.

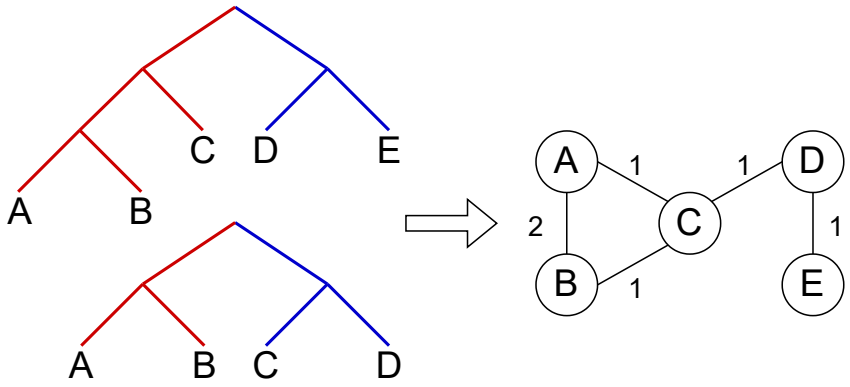


Figure 3.1: A proper cluster graph (right) for an example set of two input trees (left). The trees are coloured to more easily identify proper clusters. Each edge in the proper cluster graph is weighted by the number of input trees where the corresponding nodes are proper clusters.

If there are any edges in the proper cluster graph with a weight equal to the number of input trees, every tree supports that proper cluster. To enhance the efficiency of the approach, those edges are contracted. That is, the two nodes are merged into a single

node with edges attached to the old nodes now connected to the new. It is important to note that duplicate edges with different weights from either old node do not pose a problem for contraction. Due to the definition of a proper cluster and the fact that since the taxa in the new node appear in all trees, duplicate edges should have equal weights.

Finding the Best Partition

If the proper cluster graph is disconnected, then the components of the proper cluster graph are computed. Otherwise, every edge which lies on any min-cut of the proper cluster graph is removed, and the components of this graph are computed. These components partition the taxa into disjoint sets. For each set of taxa, the input trees are first induced on the set. That is, any taxa not present in the set is temporarily removed from the tree. Min-Cut Supertree is then recursively called on these induced subtrees.

If there are two or fewer taxa when Min-Cut Supertree is called, the method immediately returns a phylogenetic tree, making them adjacent to a new root. For each tree the recursive call returns, Min-Cut Supertree combines them by connecting the root of the returned trees to the root of a new tree, thereby recursively creating the full rooted phylogenetic tree. Figure 3.2 demonstrates this, continuing the example in Figure 3.1. Multifurcating nodes (those with more than two children) in the resulting tree can be randomly resolved to form a bifurcating tree.

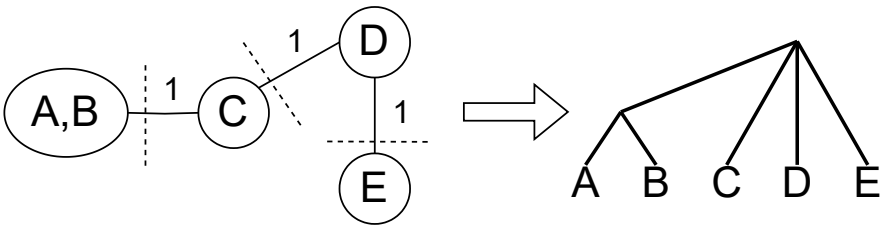


Figure 3.2: After contraction, any edge on any min cut of the proper cluster graph (left) is removed. Min-Cut Supertree is recursively called, making the results adjacent to a new root in the returned tree (right).

Improvements to Min-Cut Supertree

Min-Cut Supertree has several limitations relating to time complexity for large trees. Most notable is the step of finding every edge which lies on any min-cut of the proper cluster graph. Page (2002) improved upon the Min-Cut Supertree method by using an algorithm that finds all edges on any min-cut more efficiently, in addition to contracting more nodes in the proper cluster graph by introducing the notion of uncontradicted edges. Despite these enhancements, this method still took too long for the size of problems we wished to scale to. As a consequence, Spectral Cluster Supertree has been created which is able to calculate supertrees for much larger problem sizes.

3.3.2 Spectral Cluster Supertree

Spectral Cluster Supertree is a new supertree method for forming rooted supertrees heavily inspired by Min-Cut Supertree. It uses spectral clustering instead of min-cut to separate the proper cluster graph.

Spectral Clustering

Spectral clustering (Shi and Malik, 2000; Ng et al., 2001) is a clustering technique that can be applied to form a normalised cut of a graph. Spectral clustering can be intuitively explained by considering how heat may spread through a graph (Lee, 2014).

Consider a graph heat distribution problem with the n nodes numbered from 1 to n. Let the vector $u = (u_1, u_2, \ldots, u_n)$ be the initial heat distribution of each node in the graph. Let W be a random walk matrix over the graph, with $W_{i,i} = \frac{1}{2}$, and $W_{i,j} = \frac{1}{2d}$ if an edge connects node i to node j, with d being the degree of node i. W contains only zeroes elsewhere. The heat distribution of the graph after t time steps is given by $W^t u$.

The W matrix is eigendecomposable with eigenvalues μ_i and eigenvectors v_i. Sort the eigenvectors by decreasing eigenvalue. By writing u in terms of W's eigenvectors, the heat distribution after t time steps can be formulated as below.

$$u = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \tag{3.1}$$

$$W^t u = \mu_1^t \alpha_1 v_1 + \mu_2^t \alpha_2 v_2 + \ldots + \mu_n^t v_n \tag{3.2}$$

All eigenvalues of W are less than or equal to 1. Equation 3.2 thus represents how heat decays in the graph over a period of time steps. The first eigenvalue/eigenvector pair is entirely uninteresting, with $\mu_1 = 1$ and every element of v_1 equal. This represents the graph eventually reaching a steady state of heat. What is most interesting is the second eigenvalue/eigenvector pair. Being the second-largest eigenvalue, this is the slowest part of the equation to decay over time. The values in this eigenvector thus correspond to regions in the graph in which heat may remain trapped in for the longest number of time steps. By partitioning this eigenvector into two clusters by, for example, using k-means clustering, the graph is separated over a bottleneck. This process can be applied in a similar fashion starting from more general types of graphs (Shi and Malik, 2000).

Spectral Cluster Supertree

Over the proper cluster graph, where edges are weighted based on how many input trees have the nodes on the same side of the root, the main bottleneck of the graph should naturally occur over the true root of the input trees. We present a new variant of Min-Cut Supertree, called Spectral Cluster Supertree, which differs at a single step. Instead of deleting every edge on any min-cut of the proper cluster graph to partition it, we use

spectral clustering to efficiently separate the graph into two separate components relying on eigendecomposition. The remainder of the process is identical, using these components to induce the new trees for the recursive call. Experimentally, Spectral Cluster Supertree was able to run significantly faster than Min-Cut Supertree, allowing us to handle significantly larger problem sizes. We perform a comparison of these methods in the evaluation chapter.

3.4 TreeTOP

3.4.1 Overview

The TreeTOP algorithm combines TripleTree, Spectral Cluster Supertree, and the disk-covering method to reconstruct time-oriented phylogenetic trees. The parallel version of the disk-covering method (Coarfa et al., 2005) is used to help distribute the multiple tasks. Given some guide tree, the disk-covering method splits the taxa into overlapping subproblems. These subproblems are less than some pre-specified maximum subproblem size. The respective triples for each subproblem are generated using molecular clock rooting. Paralinear distance (Lake, 1994) between the sequences is used for this process, with TN93 distance (Tamura and Nei, 1993) used as a fallback when the paralinear distance cannot be calculated. The TripleTree algorithm uses the generated triples to reconstruct rooted phylogenetic trees for the subproblems. Finally, Spectral Cluster Supertree merges these trees together to form the guide tree for the next iteration of the algorithm, and the process continues.

Algorithm 2: A single iteration of a sequential version of the TreeTOP algorithm

```
Function TreeTOP(guideTree, maxSize):
    if guideTree.size ≤ maxSize then
        triples ← MolecularClockTriples(guideTree.taxa)
        return TripleTree(triples)
    end
     // Split the guide tree into overlapping subsets of taxa.  Returns
     a list containing the guide tree induced on each of these sets.
    splits ← DCMSplit(guideTree)
    resolvedSubtrees ← map(TreeTOP(·, maxSize), splits)
    return SpectralClusterSupertree(resolvedSubtrees)
```

3.4.2 Convergence

A convergence measure was included in TreeTOP as a termination condition for successive iterations of the algorithm. The Jaccard distance was employed to quantify the difference between the guide tries at each iteration. If this distance measure is sufficiently low for a number of iterations, the threshold and number specified by the user, TreeTOP terminates returning the sufficiently converged phylogenetic tree. A maximum number of iterations can also be specified for cases where the algorithm does not converge.

Chapter 4

Evaluation

This chapter evaluates the TripleTree, Spectral Cluster Supertree and TreeTOP algorithms. All experiments were performed on the National Computing Infrastructure (NCI) supercomputer Gadi on a single 2.9GHz Cascade Lake CPU. The chapter initially describes data generation methods used, so that the algorithms could be assessed. The results of TripleTree, Spectral Cluster Supertree, and TreeTOP are then sequentially discussed. The chapter concludes with a brief note on the correctness of the implementations. The code is provided on GitHub[1].

4.1 Data

Section 3.1.1 highlighted how the algorithms must be sufficiently robust to handle the presence of noise in data. This section describes two distinct methods that were used to generate such datasets. The first randomly generates binary trees from which triples may be sampled. The second method describes how DNA sequences were simulated over a published phylogenetic tree. By comparing the results of the algorithms to the known ground truth trees (something that is not generally known in nature), the capability of the algorithms to handle noise can be properly assessed.

4.1.1 Triples from Randomly Generated Trees

To assess the performance of the algorithms under known information, randomly generated phylogenetic trees were generated. We employ an aggregation algorithm, which starts by creating a set of n nodes. At each iteration, two nodes are randomly removed from this set, merged together, and reinserted into the set. The process repeats until the set is of length two, resulting in a nested set of bipartitions that can be translated

[1] https://github.com/rmcar17/TreeTOP

into a rooted phylogenetic tree. An example of this process is outlined below with four taxa.

$$\{a, b, c, d\}$$
$$\{a, c, \{b, d\}\}$$
$$\{a, \{c, \{b, d\}\}\}$$

Triples can easily be sampled from the randomly generated tree as described in section 2.4.2. This method provides us with mass randomly generated topologies that can be used to assess the performance of the algorithms under known information. It is important to note that there is no consideration of the edge length in this process.

Corrupting Triples

For these randomly generated topologies, noise can be simulated by "corrupting" a percentage of the triples. Given a triple $\{a, \{b, c\}\}$, it can be corrupted by randomly swapping one of the ingroups for that outgroup. That is, randomly modifying the triple to form either $\{b, \{a, c\}\}$ or $\{c, \{a, b\}\}$.

It is important to note that inserting noise into the data in this manner is not realistic when compared to the noise that is present for biological sequences. For the sequences, the noise is likely to occur over patterns in the data. That is to say, noise is very likely not independent over the sequences, but it is with the triple corruption method. Corrupting triples in this manner, however, gives us a simple tool that can be used to evaluate the algorithm's robustness to noise more generally for otherwise known triples.

4.1.2 Simulated DNA Sequences

A published phylogenetic tree containing over 10,000 Archaea and Bacteria was used to assist simulation (Zhu et al., 2019). The tree was midpoint rooted to transform it into a rooted phylogenetic tree. Additionally, any multifurcating node within the tree was made bifurcating by randomly resolving, setting the new branch length to be equal to the minimum branch length in the tree. To make the data conform to the molecular clock, the branch lengths of the leaves of the tree were extended such that the distance from the root to every leaf was equal.

From the resulting tree, sequence alignments were simulated for varying numbers of taxa using cogent3 (Knight et al., 2007). Given n randomly selected taxa on the tree, a subtree containing only those taxa was obtained. DNA sequences of length 30,000 were simulated over the resulting subtree using a strand-symmetric general nucleotide Markov substitution process. The parameters for the process were estimated from a sequence alignment of 3 bacterial species (Kaehler et al., 2015).

The process was used to generate a dataset of simulated DNA sequence alignments. The number of taxa in this dataset varied from 500 to 10,000 with an increment of 500 so that the scalability of TreeTOP could be properly assessed. By comparing the results of TreeTOP, to the ground truth tree, this dataset allowed us to evaluate the algorithm's effectiveness in handling inherently noisy DNA sequences.

4.2 TripleTree

Phylogenetic trees were randomly generated under the first model for up to 210 taxa. For these trees, the corruption process was applied to triples generated from these trees at noise levels of 0%, 5%, 10%, 15% and 25%. TripleTree was run over the different noise levels under four candidate aggregation functions: max, arithmetic mean, geometric mean, and harmonic mean. The matching distance between the estimated tree and the ground truth tree was recorded in addition to the time taken to generate the tree.

4.2.1 Effect of Aggregation Method on Distance from Ground Truth

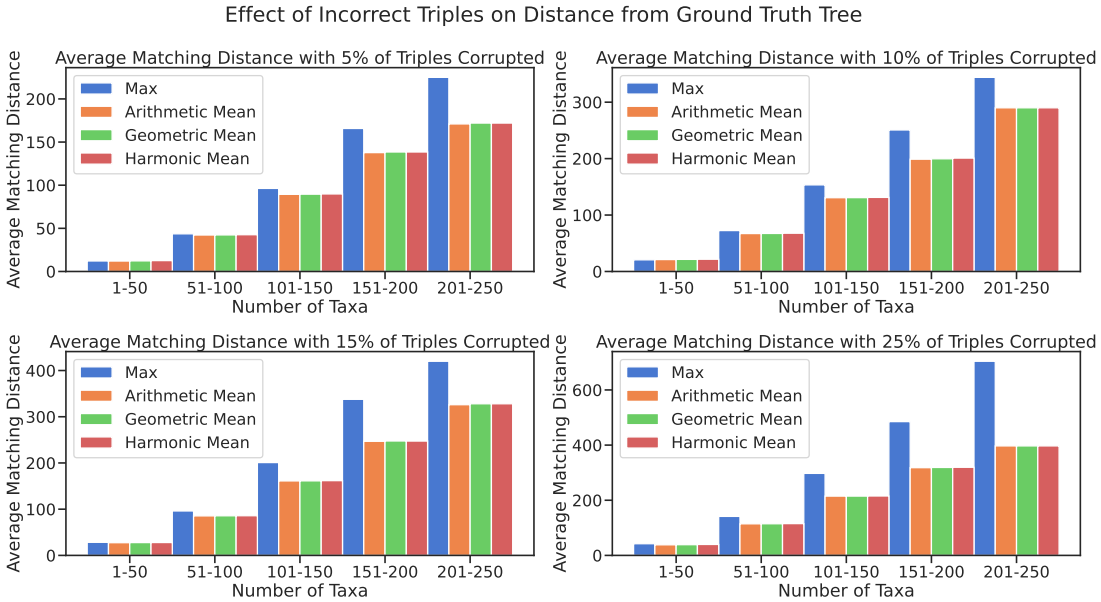


Figure 4.1: The average matching distance between the generated trees and the ground truth trees for varying levels of corruption. Results are bucketed based on the number of taxa in the trees. A smaller distance is better.

Figure 4.1 displays the average matching distance between the ground truth trees and the result of TripleTree over the triples for different levels of noise. All aggregation methods were able to fully reconstruct the ground truth tree when there was no noise in the triples, so those results are not displayed.

Figure 4.1 reveals that the max function consistently performed the worst of the four aggregation function candidates. The three alternative mean functions performed very similarly. Upon closer inspection of the raw results, each of the mean functions at times performed better than the others for some problem instances. Table 4.1 displays the average matching distance for all problems. The table shows that the arithmetic mean aggregation function, on average, performed the best for all problems. All later experiments involving TripleTree use the arithmetic mean aggregation function.

Table 4.1: The average matching distance for all problems for TripleTree for different aggregation functions. A smaller distance is better, and the best distance is bolded.

Aggregation Method	Max	**Arithmetic Mean**	Geometric Mean	Harmonic Mean
Matching Distance	83.8	**69.4**	69.6	69.9

4.2.2 Performance

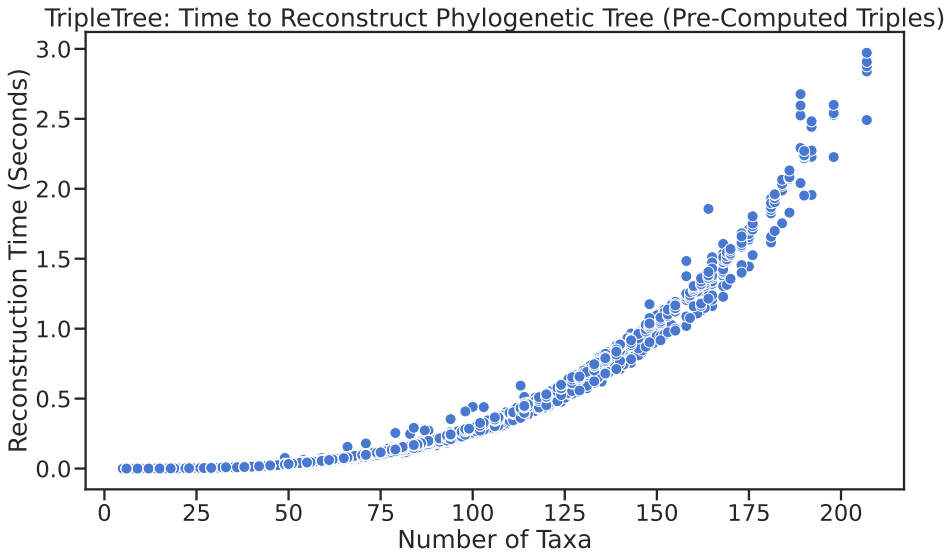


Figure 4.2: The time taken for TripleTree to reconstruct phylogenetic trees using the arithmetic mean aggregation function for all problems.

Figure 4.2 displays the time TripleTree took using the arithmetic mean aggregation function to reconstruct a phylogenetic tree for all problems in the previous section, given the pre-computed triples. The results show that once the triples are generated, TripleTree can very quickly reconstruct phylogenetic trees for these problem sizes. In fact, it is capable of solving for 100 taxa in half a second and 200 taxa in two to three seconds.

The primary limiting factor for the execution of TripleTree is the time it can take to generate the input triples. A set of n taxa yields $\binom{n}{3}$ triples - a large cubic. The 10,00 taxa in the DNA sequence dataset corresponds to approximately 166 billion triples. Even for much smaller numbers of taxa, the time for molecular clock rooting to generate the triples scales with the length of the DNA sequences. To circumvent this limitation, the overall TreeTOP method uses the disk-covering method to divide the taxa into sufficiently small overlapping subsets. This reduces the overall compute time by significantly reducing the number of triples that are required to be calculated.

4.3 Spectral Cluster Supertree

As was discussed in section 3.3, Spectral Cluster Supertree was used to scale the Min-Cut Supertree process for much larger trees. The main source of complexity behind Min-Cut Supertree was the need to find every edge on any min-cut of the proper cluster graph. For particular types of graphs, where many possible min-cuts may exist, early experiments showed that Min-Cut Supertree took an impractically long time to construct the supertree. This was despite implementing other described optimisations.

To assess the improvement in performance for Spectral Cluster Supertree, an experiment was performed for a single iteration of the TreeTOP algorithm for randomly generated trees that had known triples. During the merge phase of the algorithm, both the Min-Cut Supertree and Spectral Cluster Supertree methods were run. A 10-minute timeout was applied to Min-Cut Supertree to allow the experiments to complete within a reasonable timeframe, and Spectral Cluster Supertree was run without any timeout. The percentage of problems Min-Cut Supertree timed out for was recorded, in addition to the time taken to successfully resolve the supertree for each algorithm.

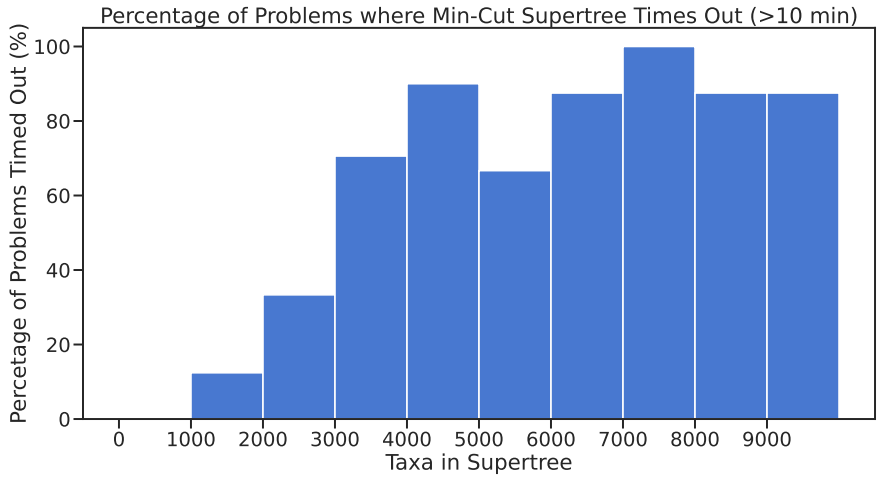


Figure 4.3: The percentage of problems for different intervals of taxa where Min-Cut Supertree failed to construct a supertree within 10 minutes.

Figure 4.3 shows the percentage of problems where Min-Cut Supertree failed to find a supertree within ten minutes. For relatively small problems of size 1,000-2,000, the algorithm can begin to take a significant amount of time to resolve. Beyond 4,000 taxa, on average approximately 80% of problems take longer than ten minutes to resolve.

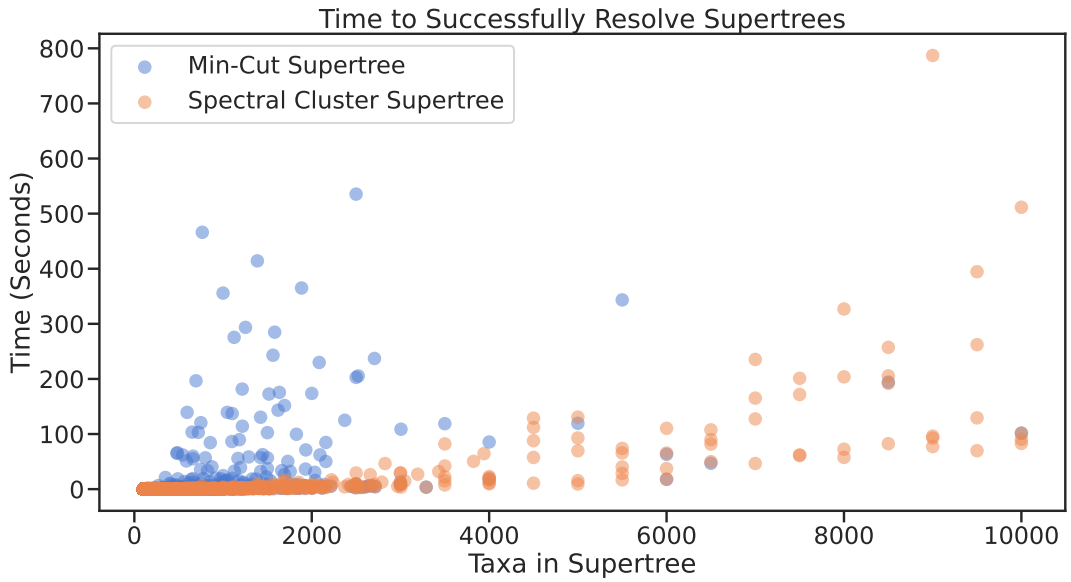


Figure 4.4: The time taken for Min-Cut Supertree and Spectral Cluster Supertree to resolve a supertree. The graph does not include problems where Min-Cut Supertree timed out.

Figure 4.4 displays the time taken for two algorithms to successfully resolve supertrees of varying sizes. The results for Min-Cut Supertree shown in the graph are skewed downwards as it does not include problems that took more than ten minutes to resolve for that algorithm. There is a relatively high variance in the results, likely due to the varying number of recursions both methods may require. As shown in the figure, Min-Cut Supertree begins to take comparatively long periods of time to find supertrees for relatively small problem sizes. Spectral Cluster Supertree is much more consistent here, always taking less than 100 seconds to find supertrees for problems with less than 4,000 taxa. Even though Spectral Cluster Supertree can, at times, take a relatively long time to compute a supertree (such as one data point at 9,000 taxa), it is apparent that it is significantly more scalable than Min-Cut Supertree.

4.4 TreeTOP

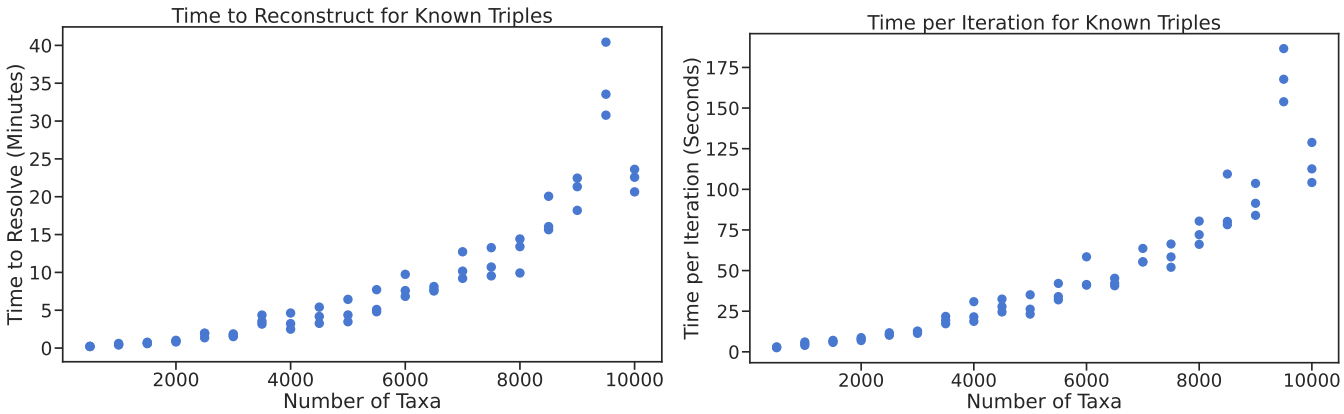
Experiments were performed on the TreeTOP algorithm for problems with known triples and for problems where the triples had to be computed from simulated DNA sequences. For each of these experiments, the program was run over 12 processes with 1 task distributor and 11 workers.

4.4.1 Results for Known Triples

Rooted phylogenetic trees were randomly generated with 500 to 10,000 taxa with an increment of 500. For TripleTree, triples were computed from the ground truth phylogenetic tree. In this experiment, the disk-covering method recursively divided the taxa into overlapping subsets with a maximal subproblem size of 100. The experiments were run until TreeTOP fully converged. Convergence here was defined by the Jaccard distance being equal to zero between the trees of three successive iterations. The initial guide tree was randomly generated for the taxa in the problem.

Time Results

The total and per iteration time for TreeTOP to fully converge to the ground truth tree is displayed in Figure 4.5. The results show that when the triples are free of noise and known, the algorithm is capable of converging to the ground truth tree for thousands of taxa in the order of minutes. TreeTOP is capable of solving a rooted phylogenetic tree with 5,000 taxa in approximately 5 minutes and 10,000 taxa in approximately 20 minutes, often taking fewer than 20 iterations.



(a) Total time (minutes).

(b) Time per iteration (seconds).

Figure 4.5: The total and per-iteration time for TreeTOP to converge to the ground truth tree with known triples.

The results for 9,500 taxa appear to be an outlier in the results, and the time per iteration graph indicates this is not a result of needing to perform more iterations. Usually, the most costly operation in the algorithm is the final Spectral Cluster Supertree merge to generate the next guide tree, which is not distributed between the processors. The more unbalanced a tree is, the more recursive calls of Spectral Cluster Supertree are required. This can lead to many calls to the spectral clustering part of the algorithm over large proper cluster graphs. Though, it is unusual that this appears in the graph only in the 9,500 taxa case. Greater experimentation is required to investigate this further.

Regardless, being able to fully resolve a rooted phylogenetic tree for many thousands of taxa in less than an hour is a significant result. It is important to qualify that these time results ignore the true compute cost of calculating the triples, as they are sampled from the ground truth tree instead of using conventional probabilistic methods over the DNA sequences to compute them. The results clearly demonstrate the tractability of using triples to reconstruct time-oriented phylogenetic trees. Critically, we identify an opportunity to eliminate one of the key tautological dilemmas facing the field. To construct a phylogenetic tree for a collection of taxa, we require a sequence alignment. However, the best method to compute sequence alignments requires a tree. This work suggests an opportunity to incorporate the sequence alignment step into the triple estimation step as a means of eliminating this circular dependency (Kaehler, 2017).

Distance Results

Figure 4.6 shows the Jaccard distance from the ground truth tree at each iteration for a subset of the experiments. As defined in section 2.7.2, a distance of 1 means 100% of the internal nodes (excluding the root) have different descendants, whereas a distance of 0 indicates the trees are identical.

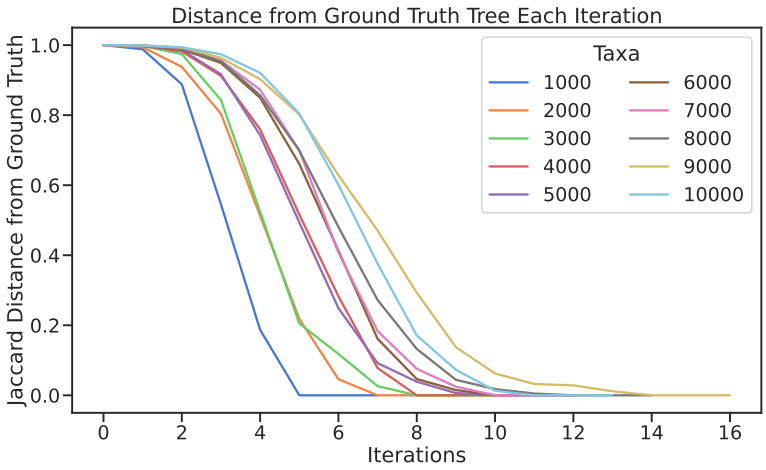


Figure 4.6: Jaccard distance from the ground truth tree at each iteration for a subset of experiments with varying taxa.

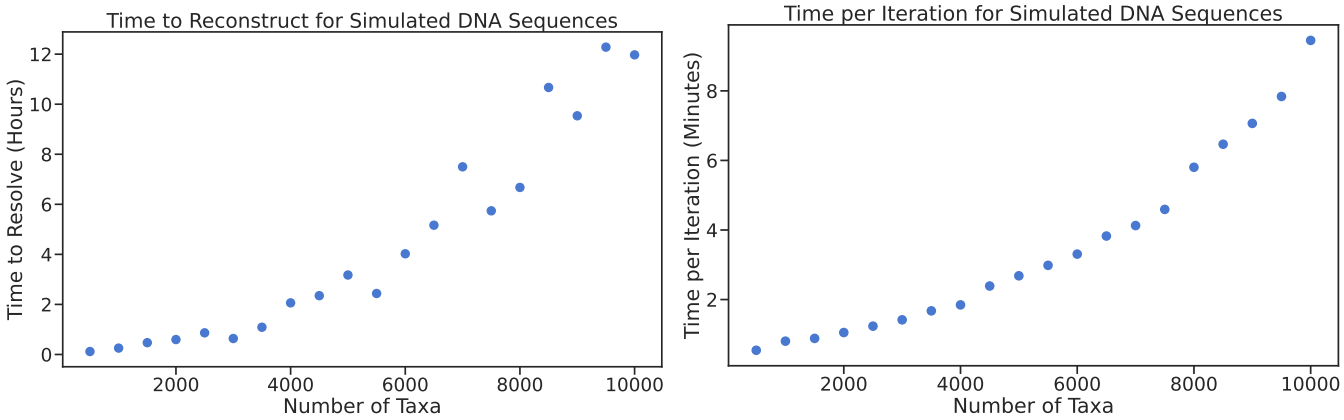
The guide tree was randomly initialised, in most cases causing the initial distance to be 1. From there, the distance from the ground truth tree tends to follow a sigmoidal shape, taking a number of iterations to place taxa in the correct general regions of the tree before the distance quickly improves as many components of the tree are solved simultaneously. This is followed by some final refinement.

One interesting observation Figure 4.6 highlights is that the number of iterations required to solve the problem appears to grow logarithmically in the number of taxa. This is a useful characteristic to have when scaling to handle greater quantities of taxa, particularly if the time per iteration can be improved through the exploitation of further parallelism.

4.4.2 Results for Simulated DNA Sequences

A similar experiment was performed to evaluate TreeTOP's performance on simulated DNA sequences. The number of taxa in each sequence alignment varied from 500 to 10,000, with an increment of 500. In these experiments, the disk-covering method component divided the taxa into overlapping subsets of maximal size 140. Molecular clock rooting was used to generate triples for the disk-covering method. In this experiment, if after six iterations the Jaccard distance to the ground truth tree did not improve, the method was terminated. When running TreeTOP without the ground truth tree, some maximum number of iterations should instead be set. Alternatively, or in combination, some sufficiently small distance threshold between successive iterations can be used to terminate the program.

Time Results



(a) Total time (hours).

(b) Time per iteration (minutes).

Figure 4.7: Time results for TreeTOP's convergence given DNA sequences of length 30,000.

Figure 4.7 displays the total and per iteration time for TreeTOP for the reported experiment. Using the simulated DNA sequences, TreeTOP can take a significantly longer time to terminate in comparison to the known triples case - taking in the order of hours to reconstruct rooted phylogenetic trees for many thousands of taxa. The algorithm takes approximately 3 hours to reconstruct a rooted tree for 5,000 taxa and is capable of handling 10,000 taxa within 12 hours. Being able to resolve the rooted phylogenetic trees for this quantity of data is fast relative to the reported performance of current tools but still takes significantly longer than the known triples case.

There are predominantly two reasons for the relative increase in time compared to the known triples case. First is the approximately fivefold increase in time per iteration. Previously where it took 100 seconds for an iteration of TreeTOP to unfold for the known triples, for the DNA sequences it now takes approximately 10 minutes. This is in small part due to the larger maximum subproblem size that was specified (which was set so more significant changes could be made to the tree at each iteration and to reduce the impact of noise). In larger part, however, this was also due to the extra overhead in requiring to repeatedly compute triples for sequences of length 30,000 which is very significant. Some form of caching of results will prove quite useful, which is discussed in section 5.1 as future work.

The second predominant reason for the time increase is due to the extra number of iterations it takes to resolve rooted phylogenetic trees of this size. This will be discussed to a greater extent in the next section.

Distance Results

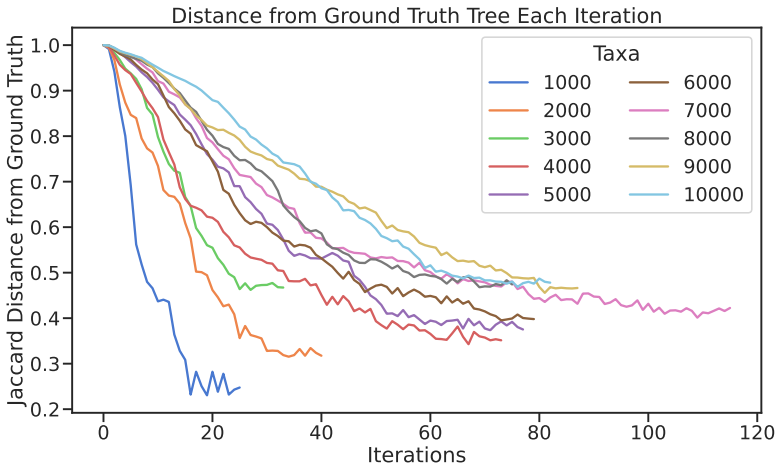


Figure 4.8: Jaccard distance from the ground truth tree at each iteration for a subset of experiments with varying taxa.

Figure 4.8 illustrates the Jaccard distance from the ground truth tree at each iteration of the TreeTOP algorithm for a subset of the experiments. As it portrays, the algorithm requires a significantly larger number of iterations to terminate compared to the previous known triples case.

The distances generally follow a similar sigmoidal shape compared to the results for known triples (noting that the first few iterations appear compressed in the figure). However, now the algorithm appears at times to spend some time perturbing the tree through some local optima.

Focussing on the 1,000 taxa run, it very early makes extremely rapid progress towards the ground truth tree. Then, for the region of Jaccard distance between 0.4 and 0.5, it spends a short number of iterations there at a local optimum. After sufficiently perturbing the tree, it begins to again make rapid progress towards the ground truth tree before cycling near a distance of 0.2. During the horizontal temporary local optima, looking internally at the Jaccard distance between successive iterations revealed that the tree is still significantly updating here. It just may take time to restructure the tree so that the Jaccard distance begins to lower again. For the final cycling behaviour, due to the noise inherent in the triples, it can be impossible to fully reconstruct the true phylogenetic tree. In such cases, the algorithm reaches a point where no further improvements can be made and may cycle through several best guesses.

In separate experiments, some runs of the TreeTOP algorithm were able to get significantly closer to the ground truth tree. It seems that for some instances, for example in the 3,000 taxa run, the termination condition was too strict for it to escape a local optima within a sufficient number of iterations.

It is also worthwhile to discuss that the Jaccard distance metric is not necessarily the most robust for this task. For example, when TreeTOP was run over a set of 10 taxa, even though a single taxon appeared misplaced in the result, it was possible for the distance measure to reach close to 0.4. It would be interesting to perform a similar experiment measuring the matching distance at each iteration. However, for large problem sizes, this may take too long to compute. Hence, working on a better and more performant alternative to Jaccard distance is important for future work. Alternatively, the task of computing the matching distance could be distributed to another process.

4.5 Correctness of Implementations

TripleTree and TreeTOP were each run for many different randomly generated trees of varying shapes and sizes for known triples with no corruption. Each algorithm was always able to fully resolve the ground truth tree, supporting the correctness of those algorithms. TreeTOP's correctness is reliant on Spectral Cluster Supertree, supporting the correctness of that algorithm as well.

Chapter 5

Concluding Remarks

5.1 Future Work

There are many areas for future work relating to TreeTOP and using triples to generate rooted phylogenetic trees. Within TreeTOP, a heuristic method could be used to seed the initial guide tree. This could give us a much better starting point and would enable us to converge on a solution in fewer iterations.

An additional area for further research is improvements to TreeTOP's convergence measure. As was stated in section 4.4.2, the current use of the Jaccard distance is not the most robust measure for evaluating the convergence of phylogenetic trees. One possible idea in this area is to incorporate the matching distance (Lin et al., 2011) for this purpose. Even though it can be slow for very large trees, it may be possible to hide this computation cost by delegating it to a separate process as iterations continue. An alternative idea here may be to use a similar stopping criterion as used by IQ-TREE (Nguyen et al., 2015), whereby if the maximum likelihood of the tree has not improved for a number of iterations, the algorithm is terminated.

Another promising idea is the implementation of caching to store the computed triples. One of TreeTOP's main benefits is that by dividing the problem, many triples never need to be calculated. In fact, it is very likely that the method is currently computing the same triples many times over as parts of the tree are resolved, so caching here would significantly improve performance. An additional optimisation technique that could be employed is the detection of regions in the tree that do not update between iterations. It may be possible to lock such regions, temporarily treating them as a single node until another condition may unlock them. This may significantly increase performance by reducing the problem size.

One significant area for future work is the actual implementation of Kaehler's (2017) conjectured method for efficiently finding the maximum likelihood solution to triples. The current molecular clock rooting method for resolving triples is efficient, but lacks biological significance given that the assumption of the molecular clock is known to be in general false. Proving and implementing the conjecture method may lead to more accurate and biologically meaningful results.

Another area for possible future work is the combination of this technique with others. Other current methods are much more well-studied than the triple-based approach. It may be possible to replace the TripleTree component of the TreeTOP algorithm with methods such as IQ-TREE (Minh et al., 2020) to solve sufficiently small problems and attain meaningful results with the various models of evolution they allow.

5.2 Conclusion

This thesis presented three new algorithms targeting the problem of quickly reconstructing time-oriented phylogenetic trees from DNA sequences. Using triples as a starting point, we devised TripleTree - an algorithm capable of efficiently converting a collection of triples into a phylogenetic tree. We also presented an enhanced version of a supertree method, which we call Spectral Cluster Supertree, able to scale much better than existing methods to find rooted supertrees. The work finally culminated by combining these methods with the disk-covering method to create TreeTOP - a divide-and-conquer algorithm that, for 10,000 taxa, can reconstruct rooted phylogenetic trees for known triples in approximately 20 minutes. TreeTOP also exhibits the ability to find optima for DNA sequences of length 30,000 within 12 hours. We have demonstrated the viability and scalability of using triples as an approach to reconstruct large rooted phylogenetic trees with promising results, and there is ample potential for further development of this technique.

Bibliography

Abu-Asab, M.; Chaouchi, M.; and Amri, H., 2006. Phyloproteomics: what phylogenetic analysis reveals about serum proteomics. *Journal of proteome research*, 5, 9 (2006), 2236–2240. [Cited on page 1.]

Baker, C. and Palumbi, S., 1994. Which whales are hunted? a molecular genetic approach to monitoring whaling. *Science*, 265, 5178 (1994), 1538–1539. [Cited on page 1.]

Coarfa, C.; Dotsenko, Y.; Mellor-Crummey, J.; Nakhleh, L.; and Roshan, U., 2005. Prec-i-dcm3: a parallel framework for fast and accurate large scale phylogeny reconstruction. In *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, vol. 2, 346–350. IEEE. [Cited on pages 12 and 21.]

Corpet, F., 1988. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, 16, 22 (1988), 10881–10890. [Cited on page 5.]

Day, W. H., 1985. Optimal algorithms for comparing trees with labeled leaves. *Journal of classification*, 2 (1985), 7–28. [Cited on page 12.]

Felsenstein, J., 1978. The number of evolutionary trees. *Systematic zoology*, 27, 1 (1978), 27–33. [Cited on pages v and 2.]

Felsenstein, J., 1981. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17 (1981), 368–376. [Cited on page 14.]

Gabow, H. N. and Tarjan, R. E., 1989. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18, 5 (1989), 1013–1036. [Cited on page 13.]

Goloboff, P. A.; Farris, J. S.; and Nixon, K. C., 2008. Tnt, a free program for phylogenetic analysis. *Cladistics*, 24, 5 (2008), 774–786. [Cited on page 12.]

González-Candelas, F.; Bracho, M. A.; Wróbel, B.; and Moya, A., 2013. Molecular evolution in court: analysis of a large hepatitis c virus outbreak from an evolving source. *BMC biology*, 11, 1 (2013), 1–13. [Cited on page 1.]

Guindon, S.; Dufayard, J.-F.; Lefort, V.; Anisimova, M.; Hordijk, W.; and Gascuel, O., 2010. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0. *Systematic biology*, 59, 3 (2010), 307–321. [Cited on page 14.]

Holder, M. and Lewis, P. O., 2003. Phylogeny estimation: traditional and bayesian approaches. *Nature reviews genetics*, 4, 4 (2003), 275–284. [Cited on page 10.]

Huson, D. H.; Nettles, S. M.; and Warnow, T. J., 1999a. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *Journal of computational biology*, 6, 3-4 (1999), 369–386. [Cited on pages 2 and 10.]

Huson, D. H.; Vawter, L.; and Warnow, T. J., 1999b. Solving large scale phylogenetic problems using dcm2. In *ISMB*, vol. 99, 1. [Cited on page 10.]

Kaehler, B. D., 2017. Full reconstruction of non-stationary strand-symmetric models on rooted phylogenies. *Journal of Theoretical Biology*, 420 (2017), 144–151. [Cited on pages v, 2, 3, 7, 9, 10, 14, 30, and 36.]

Kaehler, B. D.; Yap, V. B.; Zhang, R.; and Huttley, G. A., 2015. Genetic distance for a general non-stationary markov substitution process. *Systematic biology*, 64, 2 (2015), 281–293. [Cited on page 24.]

Kalyaanamoorthy, S.; Minh, B. Q.; Wong, T. K.; Von Haeseler, A.; and Jermiin, L. S., 2017. Modelfinder: fast model selection for accurate phylogenetic estimates. *Nature methods*, 14, 6 (2017), 587–589. [Cited on page 14.]

Knight, R.; Maxwell, P.; Birmingham, A.; Carnes, J.; Caporaso, J. G.; Easton, B. C.; Eaton, M.; Hamady, M.; Lindsay, H.; Liu, Z.; Lozupone, C.; McDonald, D.; Robeson, M.; Sammut, R.; Smit, S.; Wakefield, M. J.; Widmann, J.; Wikman, S.; Wilson, S.; Ying, H.; and Huttley, G. A., 2007. Pycogent: a toolkit for making sense from sequence. *Genome Biology*, 8, 8 (Aug 2007), R171. doi:10.1186/gb-2007-8-8-r171. https://doi.org/10.1186/gb-2007-8-8-r171. [Cited on pages 6 and 24.]

Korber, B.; Muldoon, M.; Theiler, J.; Gao, F.; Gupta, R.; Lapedes, A.; Hahn, B.; Wolinsky, S.; and Bhattacharya, T., 2000. Timing the ancestor of the hiv-1 pandemic strains. *science*, 288, 5472 (2000), 1789–1796. [Cited on page 1.]

Lake, J. A., 1994. Reconstructing evolutionary trees from dna and protein sequences: paralinear distances. *Proceedings of the National Academy of Sciences*, 91, 4 (1994), 1455–1459. [Cited on pages 6 and 21.]

Lee, J., 2014. The unreasonable effectiveness of spectral graph theory: A confluence of algorithms, geometry, and physics. https://simons.berkeley.edu/events/open-lecture-unreasonable-effectiveness-spectral-graph-theory-confluence-algorithms-geometry. [Cited on page 20.]

Lin, Y.; Rajan, V.; and Moret, B. M., 2011. A metric for phylogenetic trees based on matching. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9, 4 (2011), 1014–1022. [Cited on pages 13 and 35.]

Minh, B. Q.; Schmidt, H. A.; Chernomor, O.; Schrempf, D.; Woodhams, M. D.; Von Haeseler, A.; and Lanfear, R., 2020. Iq-tree 2: new models and efficient methods for phylogenetic inference in the genomic era. *Molecular biology and evolution*, 37, 5 (2020), 1530–1534. [Cited on pages v, 7, 10, 14, and 36.]

Ng, A.; Jordan, M.; and Weiss, Y., 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14 (2001). [Cited on page 20.]

Nguyen, L.-T.; Schmidt, H. A.; Von Haeseler, A.; and Minh, B. Q., 2015. Iq-tree: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular biology and evolution*, 32, 1 (2015), 268–274. [Cited on pages 2, 7, 14, and 35.]

Page, R. D., 2002. Modified mincut supertrees. In *Algorithms in Bioinformatics: Second International Workshop, WABI 2002 Rome, Italy, September 17–21, 2002 Proceedings 2*, 537–551. Springer. [Cited on page 19.]

Robinson, D. F. and Foulds, L. R., 1981. Comparison of phylogenetic trees. *Mathematical biosciences*, 53, 1-2 (1981), 131–147. [Cited on page 13.]

Roch, S., 2010. Toward extracting all phylogenetic information from matrices of evolutionary distances. *Science*, 327, 5971 (2010), 1376–1379. [Cited on page 7.]

Roshan, U. W.; Warnow, T.; Moret, B. M.; and Williams, T. L., 2004. Rec-i-dcm3: a fast algorithmic technique for reconstructing phylogenetic trees. In *Proceedings. 2004 IEEE Computational Systems Bioinformatics Conference, 2004. CSB 2004.*, 98–109. IEEE. [Cited on pages v, 9, 10, and 18.]

Saitou, N. and Nei, M., 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4, 4 (1987), 406–425. [Cited on pages 2 and 7.]

Searls, D. B., 2003. Pharmacophylogenomics: genes, evolution and drug targets. *Nature Reviews Drug Discovery*, 2, 8 (2003), 613–623. [Cited on page 1.]

Semple, C. and Steel, M., 2000. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105, 1-3 (2000), 147–158. [Cited on pages v, 2, and 18.]

Shi, J. and Malik, J., 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 8 (2000), 888–905. [Cited on page 20.]

Sneath, P. H. and Sokal, R. R., 1973. Numerical taxonomy. (1973). [Cited on page 2.]

Sokal, R. R., 1958. A statiscal method for evaluating systematic relationships. *Univ Kans sci bull*, 38 (1958), 1409–1438. [Cited on page 7.]

Stamatakis, A., 2014. Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30, 9 (2014), 1312–1313. [Cited on page 14.]

Tamura, K. and Nei, M., 1993. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular biology and evolution*, 10, 3 (1993), 512–526. [Cited on pages 6 and 21.]

Tang, J. W.; Tambyah, P. A.; and Hui, D. S., 2021. Emergence of a new sars-cov-2 variant in the uk. *Journal of Infection*, 82, 4 (2021), e27–e28. [Cited on page 1.]

Yang, Z. and Nielsen, R., 1998. Synonymous and nonsynonymous rate variation in nuclear genes of mammals. *Journal of molecular evolution*, 46 (1998), 409–418. [Cited on page 9.]

Yang, Z. and Rannala, B., 2012. Molecular phylogenetics: principles and practice. *Nature reviews genetics*, 13, 5 (2012), 303–314. [Cited on page 10.]

Zhu, Q.; Mai, U.; Pfeiffer, W.; Janssen, S.; Asnicar, F.; Sanders, J. G.; Belda-Ferre, P.; Al-Ghalith, G. A.; Kopylova, E.; McDonald, D.; et al., 2019. Phylogenomics of 10,575 genomes reveals evolutionary proximity between domains bacteria and archaea. *Nature communications*, 10, 1 (2019), 5477. [Cited on page 24.]

Zuckerkandl, E. and Pauling, L., 1965. Evolutionary divergence and convergence in proteins. In *Evolving genes and proteins*, 97–166. Elsevier. [Cited on page 9.]