The Dissertation Committee for Ganeshkumar Ganapathysaravanabavan certifies that this is the approved version of the following dissertation:

Algorithms and Heuristics for Combinatorial Optimization in Phylogeny

Committee:

Vijaya Ramachandran, Supervisor

Tandy Warnow, Supervisor

Warren A. Hunt Jr.

Robert Jansen

Bernard Moret

Algorithms and Heuristics for Combinatorial Optimization in Phylogeny

by

Ganeshkumar Ganapathysaravanabavan, B. E.; C. S

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2006

To my parents

Acknowledgments

Recently, an ex-colleague from my software-engineering days in Bangalore six years ago told me how I had dearly wanted to go back to India, be a professor and do research after my Ph.D. here at Texas. Honestly, I don't remember having said that, and what's worse, I don't even remember having had such a thought! But forgetting, I think, is more common than making up memories, and so I decided to trust my friend and not my memory. If six years are enough to lose even traces of strongly held convictions, I am certain they are enough to forget deeply owed gratitude. So, before I attempt to thank all those who have helped make my graduate student existence meaningful, let me state this: if I am leaving out anyone, as I most certainly am, it is not because I am not grateful, but only because I am forgetful, which I hope can be forgiven.

First of all, I thank my most wonderful parents: their love, affection, and indeed, their very presence have formed the ground beneath my feet whose stability and security I have been fortunate enough to be blessed with. I hope I have the wisdom to never take their kindness for granted, and the mindfulness to be ever conscious of it. Through out school and college and graduate school they have always allowed me the freedom and space to pursue what I wanted to, and have never exerted the slightest pressure on me to conform to their expectations. I am grateful for all the love my grandparents, my sisters Uma and Vidya, my uncles Narayanan and Padmanabhan and my aunts Latha, Lekha, Geetha and Uma have shown me. My aunts and uncles have always treated me and my sisters like their own children, and we will always be indebted to them. I really regret that my maternal grandfather Sankaran and my maternal uncle Padmanabhan who were present when I started graduate school have not lived to see its conclusion. I always think that I owe my interest in reading, and consequently the ideas that have influenced me, to my paternal grandfather

Ganapathy who owned a bookshop. I still have fond recollections of reading his books and sitting by him during his public lectures on Hindu scriptures and mythology.

Needless to say, without my advisors Vijaya Ramachandran and Tandy Warnow, this dissertation would never have amounted to anything. Whatever I have learnt about research, I have learnt from them: from Vijaya the value of doing sound, complete and scholarly research, and from Tandy the value of empirical research that has practical consequences. Tandy has suggested almost all of the problems tackled in this dissertation, and Vijaya has kept me motivated and on course during Tandy's sabbatical absence and ever since, and has taken an active interest in me even though computational phylogeny was not at that time one of her major research focuses. Without her, I might have just drifted away aimlessly, I think. To Tandy I owe my postdoctoral fellowship at NESCent: without her prodding me I may not have even come around to applying for it, and I surely could not have written an effective proposal without her. One of the most important things I have learnt (or I hope I have learnt) from my advisors is effective communication. They have helped me prepare all the papers, posters and talks about my work, and the improvement in my ability to communicate well is perceptible even to me. The famous phylo lab practice talks - it has been fun to watch effective, well-structured talks emerge out of all the seemingly chaotic, and at times overwhelming, editing process. In Tandy, I have had an advisor who shares my enthusiasm for music. Here is hoping we will write an article on music some day!

I thank committee member and co-author Bob Jansen for introducing me to the area of biogeography, which I expect to be one my major research interests in the future. The biogeographical problems tackled in this dissertation could not have been formulated without his guidance. I thank committee members Bernard Moret and Warren Hunt for being encouraging and appreciative of my work. I thank Warren Hunt for encouraging me to think about really long term research objectives.

I have been a much happier person for my friends in Austin, Deepak, Garima, Greg, Negin, Ned, Michael, Priya, Patrick and Prabha. I have enjoyed all the time I have spent with them watching movies (including the silliest Tamil movies), going to restaurants, playing racquetball, tennis, soccer, uno, and mafia and, most of all, all the long conversations we have had. They have

been immeasurably (and, if you ask me, inexplicably!) kind to me, always willing to listen to my frustrations and the not infrequent existential belly-aches. I will miss them all immensely, and I do not think I can ever realistically hope to replace them. I want to thank Ned especially, for he has been my closest friend and consequently has borne the brunt of my harangues on art, music and literature. I like to think that the the truly important lessons I have learnt in Austin, I have learnt from him. I thank my dear friend Arthi, who also has just finished her Ph. D. in chemical engineering from NCSU. She has not been here in Austin, but she has always been there to support me when I needed it. I wish her all the success and happiness. My friend Mutt and his wife Kokila in Portland are two of the warmest and the most hospitable persons I have known. There was a time when I utterly had to get away from Austin, and I knew I could count on them to host me at a very short notice. Even though he lives far away in India, I would like to thank one of my best friends for many years, Bharadwaj, to who I have always turned when I needed the wisest counsel.

My former lab mates (and Tandy's former students) Li-San Wang, Usman Roshan, Luay Nakhleh and Jerry Sun made my initial years in the Phylo lab enjoyable. Luay's lone republican voice helped me refine my liberal ideology too, I think. As the earliest participants in our practice-talk rituals, I owe to them too, the effectiveness of my first few talks. I also thank my current lab mates Serita, Shel, Kevin, David and Francois. I have enjoyed our Friday afternoon lab meetings that David and Francois helped organize.

I would like to thank the Department of Computer Sciences graduate co-ordinator Gloria, Laurie at the Center for Computational Biology and Bioinformatics, and Gem Naivar for making my time here at the department a really smooth sailing. It has been a real pleasure to interact with them and a real privilege to have known such wonderful people. They have always gone way beyond what was fairly required of them to make sure that we students are taken care of. For instance, it was really not Laurie's responsibility to make sure there was a vegetarian lunch ear-marked especially for me at the Wednesday Systematics seminar, but she did it any way. Similarly, it was really not Gloria's responsibility that I found some one to take over my regalia rental, but she helped me with it anyway. They have truly deserved all the awards and accolades that have come their way. Similarly, Katherine Utz, our graduate secretary, goes well beyond her call to make sure

the incoming students feel really welcome.

I thank the University of Texas at Austin for having such a variety of facilities and courses on offer, not just in Computer Sciences, though I wish the fine arts library were a little lenient about their late fees! I have enjoyed sitting through the courses in music theory and literature. Finally, I thank Austin for being such a weird city, and helping me meet the coolest people I have known.

GANESHKUMAR GANAPATHYSARAVANABAVAN

The University of Texas at Austin

August 2006

Algorithms and Heuristics for Combinatorial Optimization in Phylogeny

Publication No. _____

Ganeshkumar Ganapathysaravanabavan, Ph.D.

The University of Texas at Austin, 2006

Supervisors: Vijaya Ramachandran, Tandy Warnow

The goal of phylogeny is to infer the evolutionary history of the numerous and diverse species on earth. The evolutionary history is usually represented in the form of a phylogenetic tree. Reconstruction of the phylogenetic history of all life on earth is one of the central goals of biology and is also a problem that has many practical applications, ranging from drug design to conservation of biodiversity. My dissertation addresses some of the combinatorial optimization problems that arise in phylogenetic reconstruction and the related field of historical biogeography.

Two of the most important methods for reconstructing phylogenies are the optimization problems Maximum parsimony (MP) and Maximum likelihood (ML). Maximum parsimony is a purely combinatorial optimization problem, whereas ML requires an explicit probabilistic model of evolution. Both MP and ML are NP-hard problems, and the best current methods for obtaining good MP and ML solutions are based on some form of local search. Hence, developing better local-search heuristics for ML and MP is a problem of primary importance in phylogeny. Most current

local searches are based on the Tree Bisection and Reconnection (TBR) operation for moving from tree to tree. In my dissertation, I consider an alternative local-search move called Edge-Contract-and-Refine (ECR) and explore its theoretical properties in detail. We prove that the neighborhood structure induced by the ECR move on the search space has many attractive properties. These properties suggest it can complement the TBR move in local-searches. We provide fast algorithms for computing the 2-ECR neighbor of a tree with the best MP score during a local search, and for computing a random ECR neighbor. Further, we conduct an extensive comparative study of popular ML heuristics involving both large biological and simulated datasets. We also investigate the impact of the tree used to start the search on the quality of the results obtained. Our results suggest that of the currently available ML software, PHYML, RAxML and GARLI are the three best ML methods. Our results also suggest that the starting tree affects the performance of the search. However, the correlation between the quality of the starting tree and the quality of the final solution is not straightforward.

Phylogenetic reconstruction methods like MP and ML often return not a single tree but hundreds, even thousands, of trees with the same optimality score. Consensus phylogenetic trees are often used to summarize the common features observed in all or most of the resultant trees. Maximum Compatible Subtree (MCT) is one such consensus tree. In my dissertation, I provide a fixed-parameter tractable algorithm for the MCT problem when the input trees have a bounded degree. Further, I provide fast polynomial time approximation algorithms for the complement of the MCT problem.

The goal of historical biogeography is to infer the history of the geographic distribution of organisms in the light of their evolutionary history. One of the most important tools in historical biogeographical inference is the comparison of phylogenies of different groups of species that share their geographic distribution. Common patterns observed among these different phylogenies suggest a shared geographic history. So that they can be compared, the phylogenies are converted to area cladograms, where the taxa are the geographic locations of organisms. Until very recently, area cladograms have been compared only visually. In my dissertation, we formalize the problem of comparing area cladograms in the form of the Maximum Agreement Area Cladogram (MAAC)

problem, and provide a fast polynomial-time algorithm for computing the MAAC of two area cladograms. Further, we develop distance metrics for the comparison of area cladograms.

Contents

Chapter 1 Introduction **1**

1.1 The Different Phases of a Phylogenetic Analysis 3

1.2 The Phylogeny Problem 5

1.2.1 Character Encoding of Phylogenetic Trees and the RF distance 6

1.3 Maximum Parsimony 9

1.4 Maximum Likelihood and Bayesian Analysis 12

1.4.1 Stochastic Models of Single-Site Evolution. 12

1.4.2 Bayesian Analysis 16

1.5 Evaluating Phylogenetic Reconstruction Methods: Simulation Studies 17

1.5.1 Model Trees 18

1.6 Local-Search Heuristics For Solving MP and ML 19

1.6.1 The Tree-Transformation Operations 21

1.7 Phylogenetic Consensus Methods 27

1.8 Biogeography 36

1.9 Overview of Our Contributions 38

Chapter 2 The *p*-Edge-Contract-and-Refine Tree Transformation 42

2.1 The Search Space of Trees Under the *p*-ECR Move 43

2.1.1 Comparison Between *p*-ECR and TBR Neighborhoods 44

2.1.2 Diameter of the *p*-ECR Search Space 47

2.2 Efficient Hill-Climbing with 2-ECR 53

2.2.1 The Three Labels Technique 54

2.2.2 Computing an optimal 2-ECR neighbor 55

2.2.3 Random Sampling from the *p*-ECR and the *p*-sECR Neighborhoods 58

2.3 Structural Properties of the *p*-ECR Move 72

2.3.1 Irreducibility and Elementary Bipartite Graphs 73

Chapter 3 Factors Affecting Maximum Likelihood Heuristic Searches 82

3.1 Introduction 82

3.2 Experimental Methodology 85

3.2.1 Datasets 86

3.2.2 Calculating the ML Score of a Fixed Tree Topology 93

3.2.3 The Use of Random Starting Trees 94

3.3 Experimental Results on Real Datasets 97

3.4 Experimental Results on Simulated Datasets 107

3.5 Evaluation of Intermediate Trees in the ML Heuristics 114

3.6 Conclusions 121

Chapter 4 The Maximum Compatible Tree Consensus Method 123

4.1 Fixed Parameter Tractable Algorithm for MCT 124

4.1.1 Computing an MCS of two rooted trees 126

4.1.2 Algorithm for the MCS problem of *k* rooted trees with bounded degree . . 129

4.1.3 Extension to unrooted trees. 129

4.2 Approximation Algorithms 130
4.2.1 Basics 130
4.2.2 The Efficient 4-Approximation Algorithm 133
4.2.3 A slower algorithm with a better approximation ratio 137

Chapter 5 Pattern-Identification in Biogeography **143**
5.1 Introduction 143
5.1.1 Historical Biogeography 144
5.1.2 Our Contributions 150
5.2 Distance Measures Between Area Cladograms 151
5.2.1 The Character Encoding Cannot Distinguish Between Area Cladograms . . 151
5.2.2 The Edge-Contract-and-Refine Distance Metric for Area Cladograms . . . 152
5.2.3 The MAAC Distance Metric Between Area Cladograms 156
5.3 Algorithm for the Maximum Agreement Area Cladogram Problem 159
5.3.1 Basic Dynamic Programming Algorithm for MAAC 159
5.3.2 Testing Isomorphism Between Two Rooted Area Cladograms 163

Chapter 6 Conclusion **166**

Bibliography **171**

Vita **185**

Chapter 1

Introduction

Life on earth exhibits enormous diversity – there are 1.75 million species that have been discovered and described, and yet these 1.75 million species represent only a fraction of all life on earth. The diversity spans many levels: at the highest, *phenotypic*, level the species differ on manifest traits like the ability to photosynthesize and anatomical features, and at the lowest, *genotypic*, level the species differ in the composition of their genetic material. Yet the best evidence strongly suggests that all this diversity has a common origin. Throughout the history of life, new species, with their own distinct genetic material and manifest traits, *evolved* from pre-existing species. Thus, all known diversity, at all levels, can be traced back to a single common ancestor.

Phylogeny is the study of this evolution of diverse characteristics from a single origin. An evolutionary history is mostly, but not always, represented as a branching tree. These branching structures are called *phylogenetic trees*, or simply phylogenies. The exception to the branching structure occurs when two different lineages come together to engender a new lineage, as it happens, for example, during the process of hybridization of two different plant species.

Phylogenetic trees are constructed on the basis of any set of *characters* associated with the species. Characters such as behaviors or anatomical features are generally called *morphological* characters. However, diversity at all levels is driven by changes at the genotypic level, that is,

through random changes in biomolecular sequences like DNA and RNA. Hence, many phylogenies are now obtained from DNA, RNA or protein sequences. For other phylogenies, the variation in the *order* of genes in chromosomes forms the basis for phylogenetic reconstruction. In any event, phylogenies derived from variations observed in the biomolecular sequences are called *molecular* phylogenies.

The reconstruction of phylogenetic trees is both a fundamental problem in biology, and a problem that has many practical applications. As we will elaborate in Chapter 5, phylogenies are used to test biogeographic hypotheses about the history of geographic distribution of organisms. Phylogenies of co-distributed groups of species are compared, and common patterns are considered evidence of a common history of geographic spread and distribution [Jac04a, CLW95]. Through their use in historical biogeography, phylogenies also are finding widespread applications in biodiversity conservation strategies [PGB05, Erw91, MF98]. Phylogenies are also used to track evolution of diseases, and thus help design drugs and vaccines. The most famous such application is the development of influenza vaccines [BCS+99]. Other applications of phylogenies include multiple-sequence alignment, protein structure prediction and drug design [BW01].

The rest of the chapter is organized as follows: Section 1.1 briefly describes the various phases of a phylogenetic analysis. Section 1.2 formally defines the phylogeny problem and introduces the notion of the character encoding of a phylogenetic tree, which is fundamental to the study of phylogenetic trees. Section 1.3 describes the maximum parsimony phylogenetic reconstruction method. Section 1.4 describes probabilistic models of sequence evolution, and introduces the maximum likelihood and Bayesian analysis methods of phylogenetic reconstruction. Section 1.5 describes how the results of a phylogenetic analysis are evaluated. Section 1.6 provides a general outline of local-search heuristics for "solving" MP and ML, and discusses tree-rearrangement operations used in local-search heuristics. Here we also introduce the p-ECR tree-rearrangement operations, one of the main focuses of this dissertation. Section 1.7 describes some phylogenetic consensus methods that are used to summarize the features common to a set of phylogenetic trees.

In particular, the section describes the maximum agreement subtree, and the maximum compatible subtree methods, two related consensus methods that are another major subject of this dissertation. Section 1.8 discusses pattern-identification problems that arise in biogeography. Here we introduce the maximum agreement area cladogram problem, for which we provide algorithms in Chapter 5. Finally, Section 1.9 provides an overview of the contributions in this dissertation.

1.1 The Different Phases of a Phylogenetic Analysis

A phylogenetic analysis goes through many phases. To begin with, the scope of the phylogenetic study is determined. Operationally, this means that the group of taxa whose phylogeny is to be estimated is determined (taxa is plural for taxon, which stands for operational taxonomical unit which means a unit of classification). Then, the organisms, or tissue specimens from them, are collected from the field. Once the specimens are collected, the next step is to assemble, for each taxon, a sequence of characters; the phylogeny of the taxa will be constructed based on the the observed variations in the character sequences. If a molecular phylogeny is sought, the collected tissue samples are subject to a series of chemical processes in the lab to extract the biomolecular sequence of interest.

Once character sequences have been assembled for each taxon, the rest of the phylogenetic analysis is an entirely abstract computational endeavor, and goes through the following phases:

- **Multiple Sequence Alignment**. In this phase, the set of sequences are placed in the form of a matrix, with each sequence occupying a row in the matrix. Placing the sequences in this form asserts *positional homology*. That is, all nucleotides (or amino acids) in the same column are assumed to have evolved from a common ancestor. Two sequences are said to be homologous if they have evolved from a common ancestor. For the set of sequences to form a meaningful basis for phylogenetic reconstruction, homology among the sequences is a fundamental requirement. For sequence-based phylogenetic reconstruction, where each

individual site in the sequences is treated as an independent unit of data for phylogenetic reconstruction, the homology requirement extends to positional homology as described above. Note that the alignment process makes the length of all the sequences the same, and may introduce *gaps* in the sequences being aligned. There are many methods for multiple sequences, like *ClustalW* [THG94] and *MAFFT* [KMKM02, KKTM05].

- **Phylogeny Reconstruction**. The aligned set of sequences form the input for a phylogenetic reconstruction method such as *maximum parsimony (MP)*, *maximum likelihood (ML)* or *Bayesian* analysis [Fel03, SOWH96]. The objective of a phylogeny reconstruction method, in general, is to produce a tree whose leaves represent the input set of taxa. The reconstructed tree is the purported true evolutionary tree.

  – Maximum parsimony is an optimization problem based on the minimum evolution criterion. The true tree according to MP is the tree that minimizes the total number of times the characters change on its edges in order to show the observed diversity at the leaves.

  – Maximum likelihood is also an optimization problem, but requires an explicit probabilistic model of evolution. The true tree according to ML is the tree T that maximizes the conditional probability of observing the character data under the assumed model of evolution. That is, the best tree is that tree T that maximizes $Pr(D|T)$.

  – Unlike MP and ML, Bayesian analysis is not an optimization criterion, and the objective of a Bayesian analysis is not to infer one single "true" tree of evolution. Instead, one computes a probability distribution over the set of all possible phylogenetic trees for the input dataset, where the probability of any single tree is the conditional probability that that tree is the true tree given the data. That is, one computes for each tree T, the conditional probability $Pr(T|D)$. Like ML, Bayesian analysis is also performed under an explicitly specified probabilistic model of evolution.

Maximum parsimony is defined and described in detail in Section 1.3, and ML and Bayesian analysis receive a similar treatment in 1.4.

Apart from MP, ML and Bayesian methods, there are also *distance-based methods* for reconstructing phylogenies, In distance-based methods, the given character data is used to construct a matrix of distances between the taxa, and the matrix then forms the basis of some agglomerative method such as *Neighbor Joining (NJ)* [SN87] or *Unweighted Pair Group Method with Arithmetic Mean (UPGMA)* [SS73].

- **Post-processing.** Phylogenetic reconstruction methods that rely on optimization often return many trees that have the same optimality score, and the result of Bayesian analysis is, as we have seen, not one tree, but a probability distribution on the set of all trees. This necessitates *consensus* methods to summarize information present in all or a majority of the "good trees". In Section 1.7, we discuss four of these consensus methods: *strict consensus*, *majority consensus*, *maximum agreement subtree (MAST)*, and *maximum compatible subtree (MCT)*.

1.2 The Phylogeny Problem

In this section, we formalize the sequence-based phylogenetic estimation problem, and introduce the fundamental notions of the *character encoding* of a phylogeny, and the *Robinson-Foulds (RF)* distance between phylogenies.

We first describe how aligned biomolecular sequences and phylogenetic trees are modeled mathematically.

Aligned biomolecular sequences can be modeled as strings over an alphabet Σ. For example, the alphabet for DNA sequences is $\Sigma = \{A, C, T, G, \_\}$, where the symbol _ is used to denote gaps in the alignment. A sequence of length k is an ordered k-tuple from Σ^k, i.e., it is a member of Σ^k. Given n sequences of length k each, we can place each sequence as the row of an $n \times k$ matrix

and we call the i^{th} column of the matrix the i^{th} *site*. A collection on n such sequences (one for each species) is called a collection of aligned sequences. For RNA sequences $\Sigma = \{A, C, U, G, \_\}$, and for protein sequences Σ is the set of 20 amino acids and the gap character.

Mathematically, a **phylogenetic tree** is a rooted or unrooted tree whose leaves are labeled by distinct extant species. The internal nodes represent ancestral species and are unlabeled. There are $(2n-5)!! = (2n-5)(2n-7)\cdots 3\cdot 1$ unrooted binary trees and $(2n-3)!!$ rooted binary trees on n distinctly labeled leaves. An unrooted binary phylogenetic tree on n leaves has n external edges, one attached to each leaf, and $n-3$ internal edges. A rooted binary phylogenetic tree, on the other hand, has n external edge, but $n-2$ internal edges.

We now formally define the sequence-based phylogeny reconstruction problem.

Definition 1 *The Phylogeny Problem*

- **Input:** *A set S of n aligned sequences*
- **Output:** *A phylogenetic tree T with n leaves bijectively leaf-labeled by sequences in S. The tree T is supposed to represent the true evolutionary history of the set S of sequences.*

1.2.1 Character Encoding of Phylogenetic Trees and the RF distance

The character encoding is a fundamental notion associated with a phylogenetic tree. Each edge in a phylogenetic tree induces a partition of the set of leaves of the tree into two sets. The character encoding of a tree is the set of all such partitions induced by its edges. We now define the character encoding more formally.

Definition 2 *(The set $C(T)$, the bipartition encoding of T): Removing an edge e from a leaf-labelled tree T induces a bipartition π_e on its set S of leaves. We denote by $C(T)$ the set $\{\pi_e : e \in E(T)\}$. The set $C(T)$ is known as the* character encoding *of the tree T.*

Bipartitions are sometimes also called *splits*. The bipartitions corresponding to external edges (those with a leaf as one end-point) are called *trivial* bipartitions since they do not provide any information about the structure of the tree, whereas bipartitions corresponding to internal edges are called *non-trivial* bipartitions. We will not differentiate between edges and their induced bipartitions unless necessary; when comparing two edges of two trees (leaf-labeled by the same set of species) we say that the edges are equal if their induced bipartitions are equal. Bipartitions are sometimes also called *splits*.

The character encoding can be used to define the following distance measure between two unrooted phylogenetic trees:

Definition 3 *(Character Encoding Distance)*

The character encoding distance between two unrooted phylogenetic trees T and T' is $|C(T) - C(T')| + |C(T') - C(T)|$.

The character encoding distance is also called the *bipartition distance* or the *splits distance*. Buneman proved the following theorem, which is the reason why $C(T)$ is called an encoding of the tree T:

Theorem 1 *(Buneman [Bun71])*

The character encoding distance is a metric on the space of unrooted phylogenetic trees. In particular, two phylogenetic trees are isomorphic, with the leaf labels being preserved, if and only if their character encodings are identical.

The most common application of the bipartition distance in phylogeny is as a measure of the topological difference between two trees that arise in a phylogenetic analysis. One of the trees might be the result of an MP, ML or Bayesian analysis, and the other might be a reference tree that the investigator considers to be a reasonable estimate of the true evolutionary tree. If, as described later in Section 1.5, *simulation studies* are employed in the analysis, the true tree is known, and

the investigator might want to estimate the difference between the estimated tree and the true tree. Usually, the differences are stated in terms of the *false positive rate* and the *false negative rate*, as defined below:

Definition 4 *(False Positive and False Negative Error Rates)*

If T is the known true tree or a reference tree and T' is the tree estimated by a phylogenetic reconstruction method for a set of n taxa, then the rate of false positives is $\frac{|C(T')-C(T)|}{n-3}$ and the rate of false negatives is $\frac{|C(T)-C(T')|}{n-3}$.

Intuitively, false positives are those splits of taxa that occur in the estimated tree, but not in the true tree or the reference tree. False negatives are those splits of taxa that occur in the true tree or the reference tree, but not in the estimated tree. Sometimes, the average error rate, the average of the false positive rate and the false negative rates is reported.

The bipartition distance is, in fact, identical to an edit distance between phylogenetic trees based on *contractions* and *refinements* of edges. A contraction of an edge in a tree collapses the edge and identifies its two end points, while a refinement expands a node of degree greater than three, called an *unresolved node*, into two nodes connected by an edge (see Figure 1.1).

The edit distance based on contractions and refinements was introduced by Robinson and Foulds [RF81], and is defined as follows:

Definition 5 *The Robinson-Foulds (RF) Metric. (From [RF81])*

The Robinson-Foulds distance between two unrooted leaf-labeled (not necessarily binary) trees T and T', denoted $RF(T,T')$, is defined to be the length of a shortest sequence of contractions and refinements that transforms T to T'.

Robinson and Foulds also proved that the RF distance between two trees is the same as the bipartition distance between them [RF81]:

Theorem 2 *For any two unrooted phylogenetic trees T and T', $RF(T,T') = |C(T) - C(T')| + |C(T') - C(T)|$.*

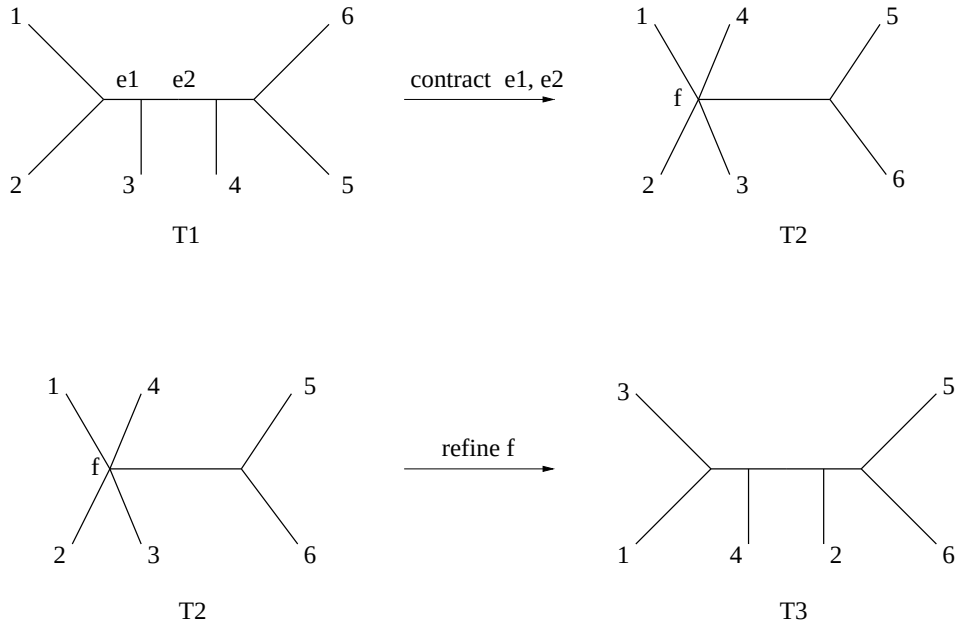


Figure 1.1: Edges e1 and e2 in T1 are contracted to produce T2; node f in T2 is the refined to produce T3.

In the phylogenetic literature, bipartition distance and RF distance are used interchangeably.

1.3 Maximum Parsimony

As mentioned earlier, maximum parsimony is based on the minimum evolution principle. An optimal MP tree is a tree that minimizes that total number of mutations needed to "explain" the observation of the given aligned sequences that label the leaves. We formalize this notion below. We begin by defining the *parsimony score* of a phylogenetic tree.

Definition 6 *Parsimony score of a tree*

Let S be a set of sequences of length k over the alphabet Σ*. Let* T *be a binary tree with leaf set S, and let* f *be an assignment of sequences of length k over the alphabet* Σ *to the internal nodes of* T*. For two such sequences x and y, let* $H(x,y)$ *be the Hamming distance between x and y. The score of* T *under the assignment* f*, denoted score*(T,f) *equals* $\sum_{(u,v)\in E(T)} H(f(u),f(v))$.

The parsimony score of T is denoted pscore(T) and is the minimum score(T, f) over all possible assignments f. The parsimony score of the tree is also called the length *of the tree.*

We now formally define the Maximum Parsimony (MP) problem.

Definition 7 *The Maximum Parsimony Problem*

Input: *Set S of sequences of length k over an alphabet* Σ.

Output: *A binary tree T whose leaves are bijectively labeled with sequences in S, such that the parsimony score of T, pscore(T), is minimum.*

Computing the Maximum Parsimony Score of a Fixed Tree.

Maximum parsimony is nothing but the Hamming-distance Steiner tree problem and has been known to be NP-hard for a long time [Fel03]. Although finding the most parsimonious tree is NP-hard, we can find the optimal labeling of the internal nodes of a given tree in polynomial time. The standard algorithm for this problem is by Fitch [Fit71].

Fitch's Algorithm. The algorithm operates as follows. First, the tree is rooted (arbitrarily), either at a leaf, or by subdividing an edge e and rooting the tree at the newly introduced node. The cost of the tree is then computed using dynamic programming. The algorithm is usually described as having two phases, where the first phase computes the length of the tree as well as a representation of candidate labels (strings over Σ^k that would produce optimal scores) for the root of each subtree of the tree; the second phase then actually produces a specific labeling for each node achieving the optimal score.

Note that each position within the strings can be handled separately, so it suffices to describe the fixed-tree maximum parsimony algorithm as though there were only one position to consider. Since we have rooted T (arbitrarily), for every internal node v in T, we can define the rooted subtree T_v, and also the children of v. We let $States_v$ denote the set of state assignments for the

node v (i.e., elements from Σ) which are part of an optimal assignment of states to all nodes in T_v so as to minimize the total parsimony score in T_v. We assume that T is binary, and that v's children are x and y (the algorithm can be applied more generally, however), and we similarly define $States_x$ and $States_y$. Then, the following equality holds (see [Fit71]):

$$
\begin{aligned}
&v \text{ is a leaf :} && States_v = \{\text{state of } v\} \\
&v \text{ has two children } x,\ y\text{ :} && States_v = \begin{cases} States_x \cap States_y \text{ if } States_x \cap States_y \neq \emptyset \\ States_x \cup States_y \text{ otherwise} \end{cases}
\end{aligned}
$$

This allows us to compute $States_v$ for every node v in T, from the bottom up. The optimal cost, i.e. the parsimony score, of T can also be calculated bottom-up at the same time: every time $States_x \cap States_y = \emptyset$ we increment the parsimony score of the tree by one. (Since we perform this computation for each site – i.e., position – independently, the sum of these values over all the sites is the parsimony score of the tree.)

In the second phase, we obtain the labeling on the internal nodes using a pre-order traversal. Once again, we can handle the positions (sites) independently. For the root r, we arbitrarily assign the state for r to be any element of $States_r$. Then visit the remaining nodes in turn, every time assigning a suitable state to the node v from its set $States_v$. When we visit a node v we will have already set the state of its parent, u. If the selected state for u is an element of $States_v$, then we use the same state; otherwise we pick a state arbitrarily from $States_v$.

This algorithm takes $O(nrk)$ time to compute the labeling of every node in T and the optimal length (i.e., maximum parsimony cost) of T, where $r = |\Sigma|$, $n = |S|$, and k is the sequence length.

1.4 Maximum Likelihood and Bayesian Analysis

As opposed to maximum parsimony, maximum likelihood and Bayesian analysis are probabilistic approaches to phylogenetic estimation. Both approaches assume a stochastic *model of evolution*. In Maximum Likelihood, given data D (the data is the same as in MP, a matrix of aligned sequences), we wish to estimate that tree T that maximizes the conditional probability $P(D|T)$, under a given model of evolution. As opposed to the MP and ML approaches, which are framed as optimization problems, in Bayesian analysis we wish to estimate, for each phylogenetic tree T, the conditional *posterior probability* $P(T|D)$, the probability of tree T being the "true evolutionary tree", given that D is the observed data. The rest of the section formalizes the above discussion.

1.4.1 Stochastic Models of Single-Site Evolution.

The stochastic models describe the evolution of a sequence down the purported evolutionary tree as a series of *site substitutions*. A site substitution in a DNA sequence, for example, is a mutation of one nucleotide into another. No insertions, deletions or duplications of sites occur. In standard site-substitution models, the sites are assumed to have evolved independently under identical processes. This is known as the *i.i.d.* assumption. The sequence of substitutions at a site is modeled as a Poisson process. [BNB96]:

- The numbers of substitutions occurring in two non-overlapping time intervals are independent random variables. That is, the current rate of substitutions is independent of the past.
- The probability of one substitution occurring in a time interval Δt is $\lambda\Delta t + o(\Delta t)$ for some constant $\lambda > 0$. That is, the probability that there is a substitution during a short time interval is proportional to the length of the interval, and does not depend on the location of the interval.
- The probability of two or more substitutions occurring in an interval of length Δt is $o(\Delta t)$.

That is, the chance of two events occurring in a short interval Δt tends to zero as Δt tends to zero.

In the above Poisson process, the number of events in a time interval of length t is a Poisson-distributed random variable with mean λ. Denoting this random variable by $X(t)$, we have: $Pr(X(t) = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$ [BNB96]. The parameter λ is the mean number of substitutions per unit time, or in other words, the *mean rate of substitutions*.

The Poisson process as described above is sufficient when there is only one type of event taking place. However, biomolecular sequence evolution consists of many types of events. In DNA sequence evolution for instance, there can be $12 = 4 \times 3$ different types of mutations. Hence, two further assumptions are made in modeling biomolecular sequence evolution:

- The evolution of sequences is assumed to occur at equilibrium. That is, the proportion of different nucleotides is assumed to remain constant across all sequences and over time.

- The individual mean rate of change to a given nucleotide is proportional to the equilibrium frequency of the nucleotide. That is, the mean rate of change from nucleotide i to nucleotide j is $\lambda_{ij}\pi_j$, where π_j is the equilibrium frequency of nucleotide j, and λ_{ij} is a constant of proportionality.

The *rate matrix* for the above model, denoted A, is an $r \times r$ matrix, where r is the number of characters, which is four for DNA or RNA evolution. The $(i,j)^{th}$ entry of A, denoted $A^{i,j}$, is the mean rate of change from nucleotide i to nucleotide j which is given by $\lambda_{ij}\pi_j$ as noted above.

Thus, the evolution of a single site of a biomolecular sequence on a tree T can be completely described by specifying the rate matrix and the time duration along each branch of T. The rate matrix and the time durations along each branch will be collectively called the *single site model parameters*, and will be denoted by M_s.

Various site substitution models can be derived by constraining the equilibrium base frequencies and the constants of proportionality to satisfy certain relationships. The simplest site-

substitution model is the *Jukes-Cantor (JC)* model [JC69]. In the JC model, all the site-substitutions occur at the same mean rate, and the nucleotides are assumed to be equally distributed at equilibrium. That is, all $\pi_j = 1/4$, and $\lambda_{ij} = \lambda_{kl}$ for $i \neq j$ and $k \neq l$. The equilibrium assumption implies that $\lambda_{ii} = -\sum_{j \neq i} \lambda_{ij}$.

A much more general model is GTR, or the *General Time Reversible Markov model.* In GTR, each of the twelve nucleotide substitutions can occur at its own rate, except that the matrix is constrained to be symmetric so as to preserve time-reversibility. Thus, $\lambda_{ij} = \lambda_{ji}$, for all i and j. The frequency distribution of the bases at equilibrium is arbitrary. In the General Markov (GM) model, the time-reversibility constraint is relaxed. See [Fel03] for a more detailed description of the various site-substitution models.

The Single Site Likelihood Score of a Tree. Given a dataset x in the form of a matrix of sequences, the data at the k^{th} site is the k^{th} column of the data matrix, and will be denoted x_k. The conditional probability $Pr[x_k|T, M_s]$ of observing x_k at the k^{th} site given that the true tree is T and the model of evolution is M_s is the *single site likelihood score* of the tree T under the model M_s.

Single site likelihood scores are computed as follows:

- From the model M_s, a set of *probability matrices*, one for each edge of T, is computed. Let Q_e be the probability matrix for the edge $e = (u, v)$ where u is the parent. Then, $Q_e^{i,j}$ is the conditional probability that the site k changes to state j across edge e, given that it was in state i at the node u. Given that the time duration across edge e is t, Q_e can be calculated as e^{At}, where A is the rate matrix. The matrix $Q_e = e^{At}$ is such that its eigenvalues are exponents of the eigenvalues of the matrix At [SS03].

- Once the matrices Q_e have been computed, the likelihood score of a tree can be computed in polynomial time, using a dynamic programming algorithm by Felsenstein [Fel81].

Evolution of Whole Sequences: Rates Across Sites. If all the sites evolve at the same rate, then the single site model M_s is sufficient to describe the evolution of the whole sequence. The

likelihood score of a tree T for the whole set of sequences is just the product of all the single site likelihood scores. However, the *i.i.d.* assumption does not mean that all the sites evolve at the same rate. The total rates of substitutions at different sites can be different, but be drawn from a distribution such as the gamma distribution [BNB96], and some sites may not vary at all, with some probability. The GTR+G+I model incorporates these possibilities. The name GTR+G+I stands for GTR with rates across sites drawn from a gamma distribution, with some sites being invariable. The GTR+G+I model is still an *i.i.d.* model. It is crucial to note that even under the rates across sites assumption, the ratios between the constants of proportionality of the rates of substitution λ_{ij} remain the same across all sites. It is only the sum of the individual rates of substitutions that differs across sites. Thus, under the GTR+G+I model, the rates of sequence evolution at different sites are just scaled versions of one another, with the scaling factors being drawn from the gamma distribution. Those sites for which the scaling factor is zero are the invariable sites. Thus, with the gamma-distributed rates across sites assumption, we need two more parameters to completely specify our model of evolution for whole sequences: (i) the probability of a site being invariant, denoted p, and (ii) the shape parameter α of the gamma distribution [BNB96].

With the gamma-distributed rates across sites assumption, the likelihood score of a tree T at a non-invariant site k now depends on the gamma distribution. This likelihood score is denoted $Pr[x_k|T,M_s,\alpha]$, and is calculated as follows. Let $g(r,\alpha)$ be the probability density function of the gamma distribution parameterized on the *shape parameter* α. Then, $Pr[x_k|T,M_s,\alpha] = \int Pr[x_k|T,M_s,r]g(r,\alpha)dr$, where $Pr[x_i|T,M_s,r]$ is the likelihood at site i conditional on the scaling factor being r.

It is hard to compute the above integral efficiently, Hence, in practice, various discrete approximations of gamma distributions are used, where the integral is replaced with summation as follows: $Pr[x_i|T,M_s] = \sum_j Pr[x_i|T,M_s,r_j]\psi_j$. In this equation the r_js are the *discrete rate categories* and the ψ_j is the probability of r_j being the scaling factor.

In the most commonly used discrete approximation of the gamma distribution from [Yan94], the number of rate categories is fixed to a constant k and the rates r_j are chosen so that each $\psi_j = 1/k$ and $\sum_j r_j/k = 1$. The rates (scaling factors) r_j depend on the gamma shape parameter α, and α is chosen so as to maximize the overall likelihood of data x on tree T, $Pr[x|T]$.

The Maximum Likelihood Score of a Tree Under GTR+G+I. The maximum likelihood (ML) score of a tree T with data x under the GTR+G+I model, $ML(T,x)$, is defined to be $sup_{M_s,p,\alpha} Pr[x|T,M_s,p,\alpha]$. The supremum, instead of the maximum, is used because a maximum may not exist, but the set of probabilities $Pr[x|T,M,p,\alpha]$ is bounded from above by 1. Maximum likelihood estimation is a statistically consistent method under the General Markov Model [Cha96b]. That is, as the sequence length approaches infinity, the probability that the maximum likelihood tree is the true tree goes to one, given that the model of evolution is correct. However, ML is not statistically consistent under more general models where sites do not evolve $i.i.d.$ [Cha96a, ST97], or when the model used to analyze the data differs from the model which generated it [Cha96a].

1.4.2 Bayesian Analysis

Bayesian analysis in phylogeny deals with estimating *posterior probabilities* associated with phylogenetic trees. That is, given data about extant species, we would like to estimate, for each phylogenetic tree T, the probability that the data evolved down tree T.

Posterior probabilities. Let $\mathcal{T}$ denote the set of all phylogenetic trees on n leaves. The posterior probability of a phylogenetic tree $T \in \mathcal{T}$ given the data D depends on the probabilistic model of evolution and the prior probability distribution on trees.

$$P[T|D] = \frac{P[D|T]P[T]}{\sum_{T' \in \mathcal{T}} P[D|T']P[T']}. \tag{1.1}$$

The quantity $P[D|T]$ is usually (i.e., in the most accepted Bayesian approaches) the *in-*

tegrated likelihood ([Ste02]) of the hypothesis tree T, obtained by integrating the probability $P[D|T,M]$ of observing the data given the tree topology T under the model M, over all possible values of M. By this token, $P[D|T]$ is not the ML score of the tree, since the ML score is the *maximum* value attained by $P[D|T,M]$ over all the parameters of the model of evolution M. Even then, $P[D|T]$ is not a probability distribution on the space of all tree topologies, and $P[D|T]$ could be defined to be any likelihood measure, including the ML score.

The quantity $P[T]$ is the *prior probability* of T. The denominator in the above equation is very hard to compute. However, it can be approximated well by Markov Chain Monte Carlo (MCMC) approaches [Mau96, LS99, HR01, HRNB01].

1.5 Evaluating Phylogenetic Reconstruction Methods: Simulation Studies

Simulation studies are widely used to evaluate phylogenetic reconstruction methods. Simulation studies play an important role in phylogeny because of two main reasons:

- A reconstruction algorithm has to be evaluated based on how close the inferred tree is to the true evolutionary tree. But in practice the true evolutionary tree is not known. Hence, we cannot evaluate our method against the true evolutionary tree. In contrast, in simulation studies the true tree is known, since it is the model tree that we explicitly construct.

- Most of the theoretical results on performance guarantees of algorithms offer only loose guarantees. Simulation studies can overcome this problem by providing actual figures on the performance of algorithms under various conditions.

A simulation study consists of the following steps:

Steps of a simulation study

1. We first decide on a stochastic site-substitution model of evolution, such as the ones described in Section 1.4, which will govern the evolution of sequences on a model tree T.

2. We then decide on a class of *model tree topologies* $\mathcal{T}$. The model trees might be either biologically-based, or based on a random process. This is explained briefly later.

3. For each model tree, we evolve sequences based on the probabilistic model of evolution; this produces a set of aligned sequences at the leaves of the trees. We then apply the phylogenetic reconstruction algorithm under evaluation to construct a tree on each set of sequences to obtain a set of inferred trees.

4. For each set of evolved sequences, we compare the topology of the inferred tree to that of the model tree. We use the false positive rate and the false negative rate for comparing trees (defined later in this section). We average the false positive rates of the inferred trees and do the same for false negative rates; thus, we can determine the average performance of the algorithm over various tree topologies from $\mathcal{T}$ under the probabilistic model of evolution.

1.5.1 Model Trees

Model trees function as true evolutionary trees in simulation studies. Ideally, we would want model trees which are most likely to occur in reality. Model trees are either biologically-based model trees that are constructed from biological data, or random model trees

Biologically Based Model Trees. The application of the phylogenetic reconstruction method ML on real biological data produces not just the purported optimal tree topology, but also the optimal branch lengths in the tree and the optimal values for the parameters of the model under which the reconstruction was performed. The resulting tree topology, together with its branch lengths and the model parameters, can be used as a model tree for simulating sequence evolution. Such model trees are called biologically based model trees.

The biggest advantage from using biologically based model trees is that we may get biologically realistic branch lengths. Thus, when evolving sequences on a biologically based model tree, we can be confident that the process is not that far from reality. However, using biologically based model trees has some drawbacks. First, the number of different tree topologies is limited. Second, if a tree (along with edge lengths) is reconstructed using the same method we are investigating, then the experiment will be very biased and not very informative. Third, the model tree will be biased towards the properties of the biological data from which it was constructed. Most of the readily available data are RNA sequences which are slow evolving; hence, biologically based model trees from RNA sequences will be biased towards the properties of RNA data. Fourth, if the data are not well-aligned, then the resulting tree will be inaccurate; multiple alignment of data is still a difficult problem and most biologists only trust hand alignments. Lastly, depending upon how we sample the biological data, we could get model trees which are very far from reality. None of these issues have been resolved in the biology community.

Random Model Trees. Random trees are generated from a probability distribution on trees. There are various distributions from which to generate random model trees. The beta splitting model [Ald95, Ald01] is a general model which encapsulates the various probability distributions used in the phylogenetic literature. In [Ald95, Ald01], the author has also compared tree topologies from the beta splitting model to trees constructed on real biological data; he has suggested parameters to the model that would produce biologically realistic model trees.

1.6 Local-Search Heuristics For Solving MP and ML

Maximum parsimony is the Hamming-distance Steiner tree problem and has been known to be NP-hard for a long time [Fel03], and ML has recently been shown to be NP-hard [CT05a, CT05b]. ML is, however, much harder to solve than MP in practice, given the many real-valued optimizations

involved in estimating the optimal ML tree. Distance-based methods like NJ are fast polynomial-time methods, but they are not as accurate as MP or ML [NRJ+01, NMR+02, MW03].

Local-search heuristics, also called hill-climbing heuristics, are the most popular and successful methods currently for obtaining good solutions for MP and ML fast. Local-search heuristics work as follows:

- Phase I: Obtain a set of initial trees by using some fast non-optimal algorithm.
- Phase II: For each initial tree, perform successive local moves that modify the tree topology to possibly improve the MP or ML score, until a local optimum is reached. In the case of ML heuristics, the local moves can also modify either the branch lengths or the model parameters and leave the topology unchanged. Here, a local optimum refers to a tree none of whose local-move neighbors has a better ML or MP score. Local moves that modify the tree topology are of special interest, since they are applicable in ML as well as MP heuristics. Moreover, even in ML search heuristics, the ultimate interest is often only in the tree topology. Tree rearrangement operations are described in detail in Section 1.6.1.

In Phase II, it is possible that there is more than one neighbor with a better score than the current candidate solution. In such cases, one such neighbor is arbitrarily selected to be the next candidate solution, and the search continues.

While a very basic hill-climbing search has been described above, many variants of the basic technique exist. One such variant, the *Parsimony Ratchet* ([Nix99]), has been a very successful MP heuristic, and is described below. For ML, recently heuristics based on genetic algorithms have been developed [Zwi06]. However, all hill-climbing methods in general evaluate trees one after another looking to improve the (MP or ML) score, and in the worst case will need to look at all trees in order to find the best.

Parsimony Ratchet. In the parsimony ratchet local-search strategy, escaping a local optimm is achieved by perturbing the input set of sequences. The ratchet in essence repeats the following

steps: (i) Starting from any tree, a local search is performed until a local optimum is reached. (ii) at this point a certain fraction of the characters are re-weighted (usually the weight of a random 25% of the sites is doubled). That is, the parsimony score due to those characters are multiplied by the weighting factor, and a new TBR local search is performed starting from the previous local optimum until a new local optimum is reached. All the characters are now again equally weighted, and steps (i) and (ii) are repeated until the desired level of optimization is achieved.

An important aspect of any local search heuristic is how the heuristic modifies the current tree topology to generate new tree topologies, or in other words, the *the tree-rearrangement operation* used by the heuristic. The tree-rearrangement operation exerts a lot of influence on the nature of the heuristic: for example, it determines how the heuristic moves through the search space of trees and how many trees are explored in a given amount of time. In the following section, we discuss a currently common tree-rearrangement move, *tree bisection and reconnection (TBR)* and a relatively new move the *edge-contract-and-refine (ECR)*. The latter move is one of the major subjects of this dissertation. Tree-rearrangement moves are sometimes also called local moves, and we will use either term interchangeably throughout this dissertation.

1.6.1 The Tree-Transformation Operations

The Tree Bisection and Reconnection (TBR) Move. In a TBR move an edge is removed from a tree T, creating subtrees S_1 and S_2, and then a new edge is added between the midpoints of any two edges in S_1 and S_2, creating a new tree. Throughout the operation any internal node of degree two is suppressed.

In current heuristics for ML and MP, two special cases of the TBR move are also commonly used. We now describe those special cases.

The Subtree Prune and Regraft (SPR) Move. In an SPR move on a tree T, as in a TBR move, an edge is removed creating subtrees S_1 and S_2. Suppose the removed edge was e, and suppose it bifurcated edges f_1 in S_1 and f_2 in S_2. In a TBR move, the new edge that is introduced can

bifurcate any edge in S_1 and any edge in S_2. But in the SPR move, the new edge is constrained to bifurcate either f_1 or f_2. Thus, the SPR move is a special case of the TBR move. Intuitively, in an SPR move, a subtree S is pruned off a point in T, and then re-attached at a different point in the truncated tree T.

The SPR and TBR moves are illustrated in Figure 1.2.

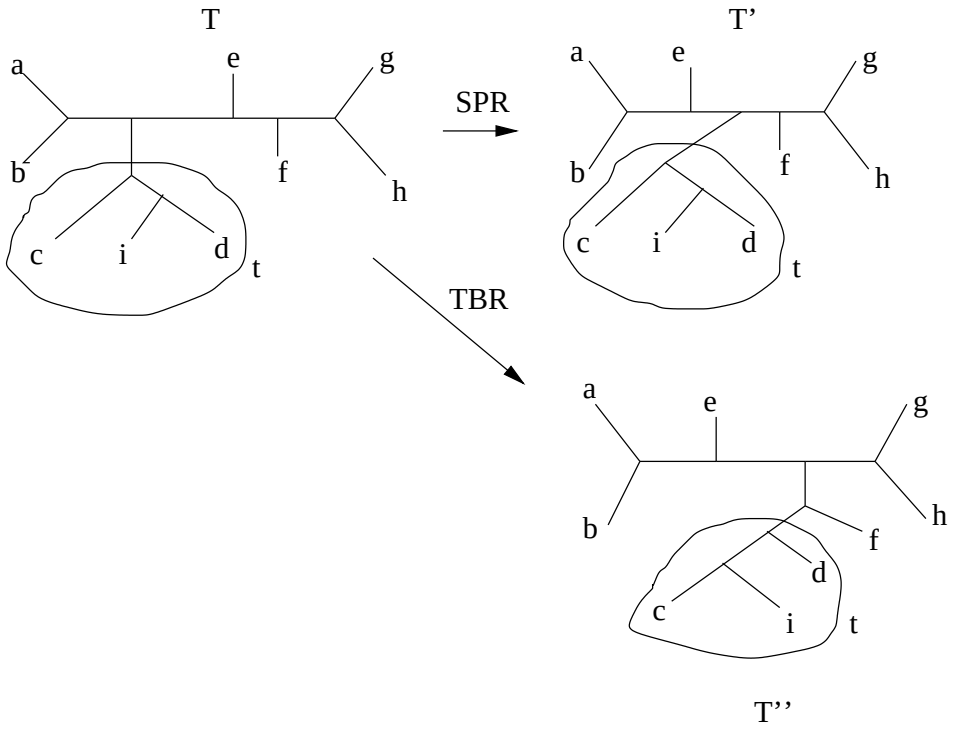


Figure 1.2: Tree T' is one SPR move away from T, while T'' is one TBR move away.

The Nearest Neighbor Interchange (NNI) Move. The NNI move swaps one rooted subtree on one side of an internal edge e with another on the other side. This is illustrated in Figure 1.3. Thus, the NNI move is an SPR move where there is only one internal edge between the point of re-attachment of the pruned subtree and the point of detachment. Note that an NNI move is also equivalent to contracting the edge e, and then resolving the resultant tree into a binary tree.

The Edge-Contract-And-Refine (ECR) Move. In this dissertation, I consider a newer tree-rearrangement move, the ECR move, in detail. The ECR tree-rearrangement operation on a phylogenetic tree is a sequence of edge contraction operations followed by refinements of unresolved

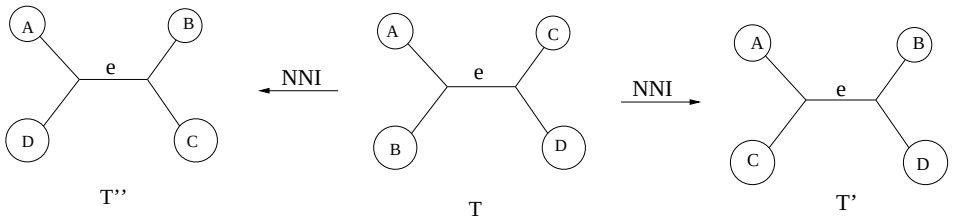


Figure 1.3: Tree T can be transformed into either T' or T'' with one NNI move

nodes. The p-ECR operation is a parameterized version of the ECR operation, where p edges are contracted. The trees $T1$ and $T5$ in Figure 1.4 are separated by one 2-ECR operation. A special case of the p-ECR move where the contracted edges are constrained to form a subtree will be called the p-sECR move.

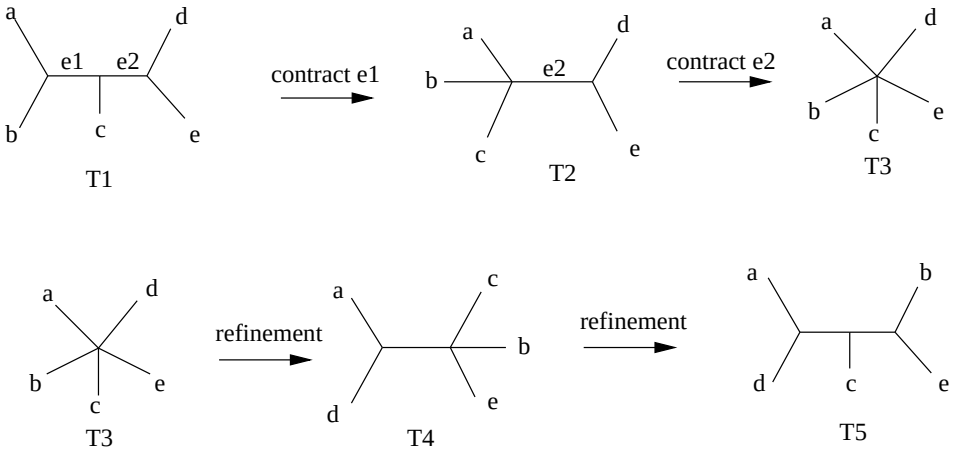


Figure 1.4: Two edges are contracted in T1 to produce T3, which is then refined to produce T5.

Our motivation for studying the ECR move is as follows: in the maximum parsimony ratchet as well as other popular hill-climbing heuristics available in software such as PAUP [Swo96] and TNT [Gol99], the TBR operation has been the local move of choice. The ML heuristics implemented in the programs PHYML ([GG03]), RAxML ([SOL05]) and GARLI [Zwi06] use the *Subtree-Prune-and-Regraft* (SPR) or *Nearest Neighbor Interchange* (NNI) move, the special cases of the TBR move. Thus almost all current ML and MP heuristics employ only some form of the TBR move. However, as noted previously, the tree-rearrangement move is a big part of the heuristic, and there may be potential benefits in employing a diverse suite of moves during a search. For

instance, a tree that is a local optimum under one move may not be one under a different move. However, among the current moves, NNI and SPR are subsumed by TBR.

The ECR local move is distinct from TBR, and is not subsumed by it, as we show in later in this dissertation, in Chapter 2. The ECR transformation has a natural appeal and has been proposed more than once independently, but has not yet been widely employed in local-search heuristics.

- In [SAH94] an experimental comparison of local searches based on p-sECR moves for different values of p is presented, and evaluated with regard to the quality of local optima generated.

- Subsequently the p-ECR move has appeared implicitly rather then explicitly in the local-search heuristic *sectorial-search* [Gol99]. In sectorial-search, a tree is transformed through contractions of edges and subsequent refinements, but the edges to be contracted are chosen using some specific heuristic, and so the number of edges contracted can vary during the search. The refinements are usually determined heuristically, as optimal tree refinement is NP-Hard [BSWY98] when the optimality criterion is maximum parsimony.

The ECR operation as defined here was first defined in our earlier work [GRW03]. In this dissertation, I consider ECR as a local move complementary to TBR, and prove several results that support this view. For instance, I show that only a small number of trees that are either one p-ECR move *or* one TBR move from a given tree are both one p-ECR move *and* one TBR move away from the given tree, for $p > 1$. This suggests that the two moves explore the tree space in a very distinct manner.

Our study of the p-ECR local move is motivated by the following questions:

- How many trees can be reached from a given tree by applying a single local move, or in other words, how large is the *neighborhood* of a tree under the local move? The size of the neighborhood determines how likely the local move is to get stuck in a local optimum. The

smaller the neighborhood, the more likely the local move is to get stuck in a local optimum soon. However, larger neighborhoods slow down the search.

- What is the largest distance between any two trees in the search space under the local move, or, what is the *diameter* of the search space under the local move? In parameterized local moves such as the p-ECR move, obtaining the diameter as a function of p might suggest appropriate values of p to be used during searches.

- If more than one move is being used in the heuristic, how do the properties of the different local moves interact? In particular, how do the properties of p-ECR and TBR interact?

- How quickly can a p-ECR neighbor of a given tree T that has the least MP score among all the p-ECR neighbors of T be computed? How quickly can a random p-ECR neighbor of a given tree be generated? These questions determine how fast the local move is able to explore the tree space.

We conclude this section with previous results on the neighborhood sizes and diameters of the search spaces induced by the TBR, NNI and SPR moves. A complete overview of our results for the p-ECR move will be provided in Section 1.9. For the sake of completeness, we provide formal definitions of the notions of the neighborhood of a tree under a local move, and the diameter of the search space under a local move.

Definition 8 *Neighborhood of a Tree Under a Tree-Rearrangement Operation*

The neighborhood of a binary leaf-labeled tree T under a tree-rearrangement operation is the set of all trees that can be obtained from T by one move.

Definition 9 *Distance Between Two Trees Under a Tree-Rearrangement Operation*

The distance between two trees on the same set of leaves under a given tree-rearrangement operation is the minimum number of operations needed to convert one tree to another. The notation $\delta_\Theta(T_1, T_2)$ denotes the distance between trees T_1 and T_2 under the operation Θ.

Definition 10 *Diameter of the Search Space Under a Tree-Rearrangement Operation*

The diameter of the search space under a tree-rearrangement operation is the maximum distance between two trees under that operation.

For a tree-rearrangement operation Θ, let $G_\Theta = (U, E)$ be an undirected graph such that U is the set of all unrooted phylogenetic trees on a given fixed number of leaves, and for two trees u and v, $(u, v) \in E$ if and only of $\delta_\Theta(u, v) = 1$. Thus, the diameter of the search space under Θ equals the diameter of G_Θ, and will be denoted $\Delta(G_\Theta)$.

The Size of the NNI, SPR and TBR neighborhoods. The size of the NNI neighborhood of an unrooted binary tree is fixed, and equals $2n-6$ [Rob71]. This can be seen from the fact that there are $n-3$ internal edges in any unrooted binary tree, and each of these edges accounts for exactly two NNI neighbors. Similarly, the sizes of the SPR and TBR neighborhoods can be calculated. The size of the SPR neighborhood of an unrooted binary tree is $(2n-6)(2n-7)$, and the size of the TBR neighborhood is at most $(2n-3)(n-3)^2$ [AS01].

NNI, SPR and TBR Distances. The distances induced by NNI, SPR and TBR moves have been discussed in [AS01, LTZ96, HJWZ96]. Note that every NNI move is an SPR move, and that every SPR move is a TBR move. Hence we have the following result from [Mad91].

Observation 1 *For any two unrooted leaf-labelled binary trees T and T' on the same set of leaves,*

$$\delta_{TBR}(T, T') \leq \delta_{SPR}(T, T') \leq \delta_{NNI}(T, T').$$

It is also known that all these distances are finite (Robinson showed this for the NNI distance in [Rob71]). Computing the NNI distance, SPR distance and TBR distance between two trees is NP-hard (see [DHJ$^+$97], [BS04] and [AS01] respectively).

NNI, SPR and TBR Diameters.

The NNI diameter of the search space of unrooted phylogenetic trees on n leaves is $\Theta(n \log n)$. More precisely,

Theorem 3 *(From [LTZ96])*

$\frac{n}{4}\log_2 n - o(n\log n) \leq \Delta(G_{NNI}) \leq n\log_2 n + O(n)$.

The SPR and TBR diameters for the same search space of phylogenetic trees on n leaves is smaller, in $O(n)$, and so smaller than $\Delta(G_{NNI})$. More precisely, we have:

Theorem 4 *(From [AS01])*

1. $n/2 - o(n) \leq \Delta(G_{SPR}) \leq n-3$*; and,*
2. $n/4 - o(n) \leq \Delta(G_{TBR}) \leq n-3$.

1.7 Phylogenetic Consensus Methods

The result of an MP or ML phylogenetic analysis is often not a single tree but many trees, for example, the set of all optimal phylogenetic trees. The goal of phylogenetic *consensus* methods is to extract information that is common to all or a majority of trees that result from an analysis. One such consensus method, the *maximum compatible subtree (MCT)* method, and a complement of this method are the subjects of Chapter 4 of this dissertation.

The *strict consensus* and the *majority consensus* methods work by contracting edges in the input trees, and are also the most widely used consensus methods. The *maximum agreement subtree (MAST)* and MCT work be eliminating leaves from the input trees, and are much less widely used than strict consensus and majority consensus.

In this section, we first define the four above-mentioned consensus methods, and the complement of the MCT. We then motivate our interest in the MCT method and its complement. We then describe the previous results on MAST, MCT and its complement. A polynomial-time dynamic programming algorithm for computing a maximum agreement subtree of two trees due to Steel and Warnow from [SW93] forms the basis of our algorithm for the MCT problem. The same

algorithm also forms the basis of our pattern-identification algorithm for biogeography, described in Section 1.8 and the subject of Chapter 5. Hence, we conclude this section with a description of the above MAST algorithm, which we call the Steel-Warnow algorithm.

The strict consensus tree of a set of trees is the tree encoded by the set of bipartitions common to all the trees in the given set:

Definition 11 *Strict Consensus Tree*

For a given set of trees $\{T_1, T_2, \ldots, T_k\}$*, the strict consensus tree* T^* *is such that* $C(T^*) = \{\pi | \pi \in C(T_i)$ *for all* $1 \leq i \leq k\}$.

The majority consensus tree of a set of trees is the tree encoded by the set of bipartitions present in more than half of all the trees in the given set:

Definition 12 *Majority Consensus Tree.*

For a given set of trees $\{T_1, T_2, \ldots, T_k\}$*, the majority consensus tree* T^* *is such that* $C(T^*) = \{\pi |$ *the number of trees* T_i *such that* $\pi \in C(T_i)$ *is greater than* $k/2\}$.

The strict consensus and the majority consensus of a set of three trees are shown in Figure 1.5.

The strict consensus of k trees on n leaves is computable in $O(nk)$ time (see [Day85]). A linear time *randomized* algorithm for computing the majority consensus was presented in [ACJ03].

Before we can define the maximum agreement subtree problem, we need to define the notion of *restricting* a phylogenetic tree to a subset of its leaves.

Definition 13 *(Restriction of a tree to a set of leaves): The restriction of a tree* T *to a set* X *of leaves, denoted* $T|X$*, is the tree obtained by deleting the leaves not in* X *from* T *and then suppressing all internal nodes of degree 2.*

Definition 14 *Maximum Agreement Subtree (MAST).*

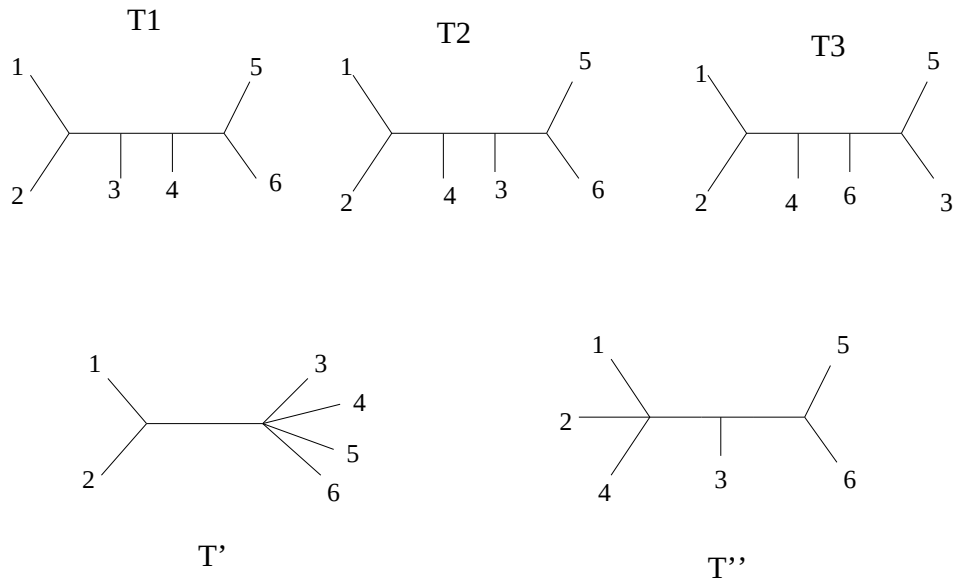


Figure 1.5: Tree T' is the strict consensus of T1, T2 and T3, while T" is the majority consensus.

For the MAST problem, the input is $\mathcal{T}$, a collection of trees, each leaf-labelled by the same set S. The task is to find a maximum cardinality set $A \subseteq S$ so that the for any two trees T and T' in $\mathcal{T}$, $T|A$ and $T'|A$ are isomorphic (with the leaf-labels preserved by the isomorphism). Here, $T|A$ denotes the tree obtained by restricting T to the leaves labelled by elements of A, and suppressing degree two nodes. Such a set A is called a maximum agreement subset for the collection of trees $\mathcal{T}$, and the restricted tree on the set of taxa A is called the maximum agreement subtree (MAST) for the collection $\mathcal{T}$.

Before we define the MCT problem, we define what it means for a set of trees to be compatible:

Definition 15 *(Tree compatibility): A set of trees $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ on the set of leaves S is said to be* compatible *if there is a tree T^* that is a common refinement of trees in $\mathcal{T}$. In other words, all trees in $\mathcal{T}$ can be obtained by contracting some edges in T^*.*

Definition 16 *Maximum Compatible Subtree (MCT).*

For the MCT problem the input is $\mathcal{T}$, a collection of trees, each leaf-labelled by the same set S. Here, the task is to find a maximum cardinality $A \subseteq S$ so that the set $\{T|A : T \in \mathcal{T}\}$ has a common refinement. Such a set A is called a maximum compatible set (MCS) for the collection of trees $\mathcal{T}$, and the restricted tree on the set of taxa A is called the maximum compatible subtree (MCT) for the collection $\mathcal{T}$.

We now define the complement of the MCT (or MCS) problem:

Definition 17 *Complement of the Maximum Compatible Subset (CMCS).*

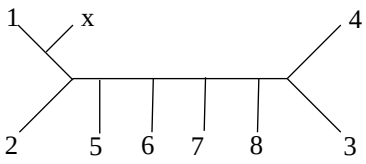
For the CMCS problem the input is $\mathcal{T}$, a collection of trees, each leaf-labelled by the same set S. Here, we seek a minimum cardinality subset X of S such that $\{T|(S-X) : T \in \mathcal{T}\}$ is a compatible set of trees.

Note that if all the trees in the input collection are binary, any maximum agreement subset is a maximum compatible subset (and vice versa).

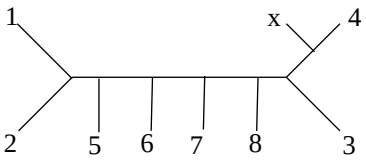
Biological Motivation for the MCT Problem. The problem with standard consensus methods such as the strict consensus is that they can return highly unresolved trees. The strict consensus tree is highly unresolved when one species, say *x*, is sufficiently distantly related to the other species so that the reconstruction method is not able to meaningfully locate species *x* in a phylogenetic tree, and is not able to distinguish between phylogenies that differ only with respect to the placement of *x*. This is illustrated in Figure 1.6. Trees T and T' in the figure are identical except for the placement of the taxon *x*, yet the strict consensus is tree is completely unresolved and conveys no information. The majority consensus tree can be similarly poorly affected. Taxa such as *x* are called "rogue" taxa.

Alternative consensus approaches such as MAST [FG85] were proposed so as to identify and eliminate rogue taxa from the phylogenetic analysis and get a well-resolved consensus tree. This is again illustrated in Figure 1.6, where the MAST approach is seen to correctly identify and

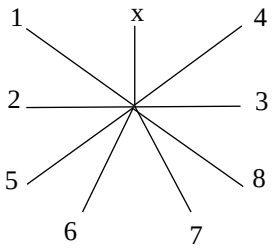
eliminate the rogue taxon *x*, and return the maximum agreement subtree which is more informative than the strict consensus in this case.



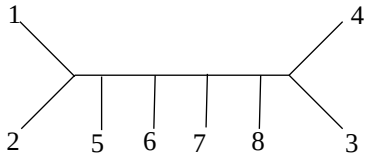
T



T’



Strict Consensus (T, T’)



MAST(T, T’)

Figure 1.6: Trees T and T’ differ only in the placement of taxon *x*, but this difference is enough to ensure that they do not share any non-trivial bipartitions. As a consequence, the strict consensus tree is completely unresolved. But trees T and T' become identical once *x* is eliminated, and this identical subtree is their maximum agreement subtree.

Software for the MAST problem exists in the PAUP [Swo96] software package, but is not very useful since the size of the MAST is often not large enough to be interesting (David Swofford, personal communication). The problem seems to be that requiring complete agreement between the trees, even when restricted to a smaller set of taxa, is too strong a requirement.

A potential explanation for this phenomenon lies in the difficulty in correctly resolving “short edges” in the true tree. In trees resulting from a parsimony analysis, the short edges are those edges across which the normalized Hamming distance is very small. In general, short edges can be identified as those edges with low bootstrap support value (refer to [Fel03] for discussions on the bootstrap measure). In such situations, optimal solutions for problems such as Maximum Parsimony or Maximum Likelihood may differ significantly in terms of how they resolve around these short edges. This results in having the MAST of the set of trees being very small. On the

other hand, the identification and contraction of short edges in each of the ML or MP trees could result in a collection of trees that, while not binary, have a common refinement on some large subset of leaves. This suggests the following approach:

- First, contract all short enough edges, and
- Second, identify and delete a minimum set of leaves so that after these leaves are deleted and degree two nodes suppressed, the resultant trees share a common refinement (i.e. are compatible). If a consensus tree is desired, the minimum common refinement (i.e., the maximum compatibility subtree, or MCT) of the resultant trees is returned.

The first step is straightforward (though establishing how short is short enough needs to be studied), but the second step is essentially the CMCS problem.

Note that there is a relationship between the number of leaves in the MAST (maximum agreement subtree) of a set of trees and its MCT (maximum compatibility subtree). By definition, the number of leaves in a MCT is at least as big as that of the MAST, and this can often be significantly larger. See Figure 1.7 for an example of two trees and their MAST and MCT, for which there is a difference in sizes. Thus, the MCT problem is a better model for consensus trees, as it retains more information and is less affected by noise in the input.

Previous Work on MAST. The maximum agreement subset problem was introduced in [FG85], and a polynomial time algorithm for computing a MAST of two trees was first described in [SW93]. In [SW93], a running time of $0(n^{4.5}\log n)$ is reported, however, we show that the algorithm, in fact, runs in $O(n^{2.5}\log n)$ time, where n is the number of leaves in the trees. Computing a MAST is NP-hard for three or more trees (whether rooted or unrooted; in general, the unrooted MAST problem can be solved by solving a polynomial number of rooted MAST problems) [AK97]. For computing a MAST of k rooted trees, an $O(kn^3 + n^d)$ algorithm (with d being the maximum degree of a node in some tree) was described in [FCPT95].

Previous Work on MCT. The maximum compatible subset problem was initially posed by Hamel

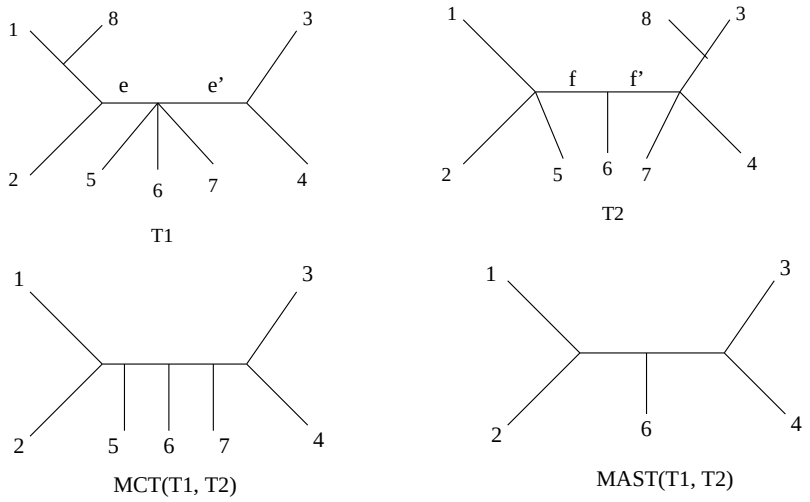


Figure 1.7: The maximum compatible tree (MCT) and the maximum agreement subtree (MAST) of two unrooted trees T1 and T2 are shown. If the edges e and e' in T1, and f and f' in T2 are short edges which are contracted, then T1 and T2 categorically differ in only where they place taxon 8. The MCT of T1 and T2 is larger than MAST(T1, T2), and reflects this observation.

and Steel in [HS96], where it was shown to be NP-hard for six or more trees. Hein et al. show in [HJWZ96] that the problem is NP-hard for two trees even if one of the trees has bounded degree. The problem is solvable in polynomial time for any number of trees when all trees are of bounded degree [HJWZ96]. In our work, we show that the MCT problem is polynomial-time solvable when all the input trees have a bounded degree. Further, we present a 4-approximation algorithm, and a slower 3-approximation algorithm for the CMCS problem. A detailed overview of these contributions is presented in Section 1.9.

The Steel-Warnow MAST Algorithm. We now give a brief summary of the algorithm from [SW93] for computing the MAST of two rooted trees.

We describe the algorithm for two rooted binary trees, for the sake of clarity. The algorithm can be easily extended to compute a MAST of two arbitrary unrooted trees. In our description, the expression $MAST(T, T')$ denotes a maximum agreement subset of the leaves of T and T'. Let T and T' be two given binary rooted phylogenies on n leaves. Let $V(T)$ and $V(T')$ be the set of all nodes of T and T' respectively. Denote by T_v the subtree of T rooted at a node v in $V(T)$. Similarly denote by T'_w the subtree of T' rooted at a node w in $V(T')$. Let $c(v)$ and $c(w)$ be the

set of children of nodes v and w respectively. The dynamic programming algorithm for MAST operates by computing $MAST(T_v, T'_w)$ for all pairs of nodes (v,w) in $V(T) \times V(T')$ "bottom-up". We now show how to reduce computing $MAST(T_v, T'_w)$ to computing a small number of smaller MAST computations $MAST(S, S')$ where S and S' are subtrees of T_v and T'_w respectively, with at least one of them being a proper subtree.

To begin with, the $MAST(T_v, T'_w)$ is easy to compute when either v or w is a leaf. So in the following discussion assume neither v nor w is a leaf.

Let L^* be a MAST of T_v and T'_w, and let T^* be the corresponding MAST tree. Then there exist homeomorphisms mapping T^* to a rooted subtree of T_v and to a rooted subtree of T'_w. Let p be the (not necessarily proper) descendant of v such that the root of T^* is mapped to p. Similarly let q be the descendant of w in T' such that that the root of T^* is mapped to w. Then, $MAST(T_v, T'_w)$ in fact equals $MAST(T_p, T'_q)$.

The vertex p may be actually v or it might be a descendant of v. Similarly q may be w or some descendant of w. Based on the location of p and q, we have the following cases.

1. *Vertex p is a proper descendent of v.* In this case, T_p is a proper subtree of T_v, and $MAST(T_v, T'_w)$ equals $MAST(T_p, T'_w)$. Since p is a proper descendant of v, there exists $v_{max} \in c(v)$ such that $MAST(T_p, T'_w)$ equals $MAST(T_{v_{max}}, T'_w)$. The node v_{max} maximizes $MAST(T_{v_i}, T'_w)$ over all nodes $v_i \in c(v)$.

2. *Vertex q is a proper descendent of w.* In this case, $MAST(T_v, T'_w)$ equals $MAST(T_v, T'_q)$. Since q is a proper descendant of w, there exists $w_{max} \in c(w)$ such that $MAST(T_v, T'_q)$ equals $MAST(T_v, T'_{w_{max}})$. The node w_{max} maximizes $MAST(T_v, T'_{w_j})$ over all nodes $w_j \in c(v)$.

3. *Vertex p equals v and vertex q equals w.* Let trees T_1^* through T_k^* be the maximal rooted subtrees of the MAST tree T^*. Then, each subtree T_i^* is homeomorphic to a subtree of T_{v_j} for some $v_j \in c(v)$. Now, there does not exist a node $v_j \in c(v)$ such that two different maximal rooted subtrees $T_{i_1}^*$ and $T_{i_2}^*$ of T^* are both isomorphic to subtrees of T_{v_j}. To see this,

let x be a leaf in $T^*_{i_1}$ and let y be a leaf in $T^*_{i_2}$. The least common ancestor of x and y in T^* is the root of T^*, but the least common ancestor in T_1 is a descendant of v_j. This violates the assumption that the root of T^* is homeomorphically mapped to node v in T_v. A similar argument holds with respect to the homeomorphism between T^* and T'_w.

Let each maximal rooted subtree T^*_i of T^* be homeomorphic to a subtree of T_{v_i} in T_v and a subtree of T'_{w_i} in T'_w (assume that the children of v and w have been re-numbered so that the above statement is true). A maximum agreement subtree T^* can be computed as follows: compute a maximum agreement subset for each pair of subtrees T_{v_i} and T'_{w_i} and take the disjoint union. The pairs of nodes (v_i, w_i) can be identified as follows: let $G_{v,w} = (c(v), c(w), W)$ be a complete weighted bipartite graph on the sets of vertices $c(v)$ and $c(w)$. The weight of an edge (x,y) is given by $W((x,y)) = |MAST(T_x, T_y)|$. Then, the pairs of vertices (v_i, w_i) correspond to a maximum weighted bipartite matching (MWBM) in $G_{v,w}$.

The above discussion suggests a straightforward dynamic programming algorithm: we do not know which of the above three cases is true, but we do know that one of them is true. Hence we solve the subproblems corresponding to all the three cases and then choose the largest solution.

The running time of the above algorithm is in $O(n^2)$ for trees of bounded degree: there are $O(n^2)$ subproblems and the running time for each subproblem is dominated by the MWBM computation. Using the algorithm from [GT89], the MWBM problem can be solved in time $O(p \cdot q\sqrt{(p+q)}\log(p+q))$ if the roots of the two subtrees whose MAST is being computed have degrees p and q. Thus, if the maximum degree of a node in either tree is bounded from above by a constant d, the total running time is in $O(n^2 d^{2.5}\log d)$, which is in $O(n^2)$. If the degree is not bounded, a straightforward bound on the running time is $O(n^{4.5}\log n)$, which is in fact the bound stated in [SW93].

A careful analysis of the algorithm, however, reveals that the running time is in fact $O(n^{2.5}\log n)$, even when the maximum degree is not bounded. If the two sets of vertices in the bipartition of a

bipartite graph have p and q vertices, then the running time of the MWBM algorithm in [GT89] is, as noted above, $O(p \cdot q \cdot \sqrt{(p+q)}\log(p+q))$. The MAST algorithm performs the MWBM computation on graphs $G_{v,w}$, for each pair (v,w) with $v \in V(T)$ and $w \in V(T')$. Let n_1 and n_2 be the number of leaves in T and T' respectively, with $n = max(n_1, n_2)$. Also, let d_u be the degree of node u in either tree. If we let $T(n_1, n_2)$ denote the running time of the MAST algorithm, we have:

$$\begin{aligned} T(n_1,\ n_2) &\leq c \sum_{u \in T_1} \sum_{v \in T_2} (d_u d_v \sqrt{(d_u + d_v)} \log(d_u + d_v)) \\ &\leq c\sqrt{(n_1+n_2)}\log(n_1+n_2) \sum_{u \in T_1} d_u \sum_{v \in T_2} d_v \\ &\leq c\ n_1 n_2 \sqrt{(n_1+n_2)}\log(n_1+n_2) \\ &\leq cn^{2.5}\log n. \end{aligned}$$

1.8 Biogeography

Biogeography is the study of spatial and temporal distribution of organisms. Biogeographers seek not only to understand ecological processes that influence the distribution of living organism over short periods of time (e.g., climatic stability, effect of area) but also to uncover events occurring in the distant past (e.g., continental drift, glaciation, evolution) which have resulted in the geographic distribution observed today.

Biogeography and Phylogeny. One of the ways of understanding the geographic distribution of species is by using the associated phylogeny. This approach is illustrated thus: consider a group of islands, each island containing multiple ecological zones (for example, each island can comprise a coastal ecological zone and a mountain ecological zone). Suppose our goal is to understand the observed geographic distribution of species on the islands. One hypothesis about the distribution could be that species dispersed from an ecological zone in an island to similar zones in other is-

lands and then differentiated. This process is called *inter-island colonization*. Another hypothesis could be that dispersal between the islands happened first followed by dispersal to the different ecological zones and differentiations. This process is called *adaptive radiation*. The crucial idea is that we might be able to infer which of the above two hypotheses is responsible for the observed distribution: inter-island colonization is suggested by taxa in different islands but the same ecological zone forming a monophyletic group, and adaptive radiation is suggested if species on the same island in different ecological zones form a monophyletic group. Thus, the branching structure of the phylogeny of a set of taxa can be used to differentiate between competing hypotheses concerning the observed geographic distribution of the set of taxa. Moreover, *a consistent pattern* observed in the phylogenies of different genera of species on the same geographic area will imply a stronger support for the particular hypotheses suggested by the pattern.

Area Cladograms. Before looking for common patterns in the phylogenies of different sets of species in the same geographic area, the phylogenies are converted to *area cladograms*. Area cladograms are rooted or unrooted trees (as are phylogenies) whose leaves are labeled with *geographic areas* instead of taxa. To obtain the area cladogram for a set of species local to a set of areas, we start with the phylogeny for the set of species and for each leaf, we replace the taxon label with the label of the area in which the taxon is found. Note that in an area cladogram many leaves may share the same label and a single leaf may have many labels.

In view of the above discussion, the problems of comparing cladograms and identifying patterns within them become important. In this dissertation, we present an algorithm for identifying the common patterns in two area cladograms, a related distance metric between area cladograms, and an algorithm to identify if two area cladograms are identical. A detailed overview of these contributions is presented in Section 1.9.

1.9 Overview of Our Contributions

My contributions fall into the following categories:

- Analysis of the space of phylogenetic trees under the p-ECR local-move operation (Chapter 2).
- An experimental evaluation of current ML heuristics in order to understand the factors that affect the performance of ML heuristics (Chapter 3).
- Algorithms for the MCT consensus problem (Chapter 4), and
- Algorithms and metrics for comparing area cladograms in biogeography (Chapter 5).

Results on p-ECR Local Operations. My results on the p-ECR local move focus on the following two aspects:

- Designing efficient hill-climbing searches using p-ECR, and
- Understanding the properties of the p-ECR search space.

With respect to designing efficient p-ECR local searches, we have obtained the following results:

- An $O(n^2)$ algorithm for computing the optimal (with respect to the MP score) 2-ECR neighbors of a given unrooted binary tree. The algorithm represents a running-time improvement of a factor of n over a brute-force algorithm. This result has been published in [GRW03].
- We present a $\Theta(p)$ expected-time algorithm to generate p-ECR neighbor of a tree uniformly at random. We also present two algorithms to generate a p-sECR algorithm uniformly at random: (a) a $\Theta(np^2)$ algorithm, and (b) a faster algorithm that takes only $\Theta(p)$ time to produce a random neighbor, but one that may fail to return a neighbor in some cases. The latter algorithm requires an $O(p^2)$ preprocessing at the beginning of the search.

With respect to understanding the properties of p-ECR search spaces, I have obtained the following results, published in [GRW04]:

- Asymptotically tight bounds for the diameter of tree-space under p-ECR and p-sECR moves as a function of p, showing that the diameter of the search space in both cases is in $\Theta(\frac{n\log n}{p\log p})$ (where n is the number of leaves in the trees). This result could be potentially useful in selecting a suitable range of values of p for performing local searches based on p-ECR operations.

- A comparison of the neighborhoods of a tree induced by TBR and p-ECR moves, showing that their intersection is of size $O(min\{n2^p, n^2p\})$. The neighborhoods themselves are much larger: there could be $\Theta(n^3)$ trees in the TBR neighborhood of a tree, as mentioned before, while the p-ECR neighborhood contains $\Omega(n^p2^p)$ trees. These results may help explain why the combination of the two moves improves upon the use of just one, as reported in [Gol99].

Apart from the above p-ECR results that relate to local-search heuristics, I have studied p-ECR purely with respect to its structural properties. My results, detailed below, have shown that the p-ECR local move has simple and elegant properties:

- We define the properties of *irreducibility* and *commutativity* of p-ECR operations, and observe a surprising connection between irreducible p-ECR operations and *elementary* bipartite graphs. We exploit this connection to develop an $O(n+p^2)$ algorithm to reduce a p-ECR operation into an equivalent sequence of irreducible (atomic) ECR operations. These results also have been published in [GRW04].

Experimental Evaluation of ML Heuristics. In Chapter 3, I evaluate the performance of four ML software PAUP [Swo96], PHYML [GG03], RAxML [SOL05], GARLI [Zwi06], a Bayesian analysis software MrBayes [HR01] used as an ML heuristic, and the MP heuristic ratchet used as an ML heuristic on real biological and simulated datasets. The experiments also chart the impact of starting trees on the quality of the final trees obtained by the various heuristics.

Results on Maximum Compatible Tree Consensus Method. The following are my results on the Maximum Compatible Tree (MCT) consensus method.

- A 4-approximation algorithm for the *complement* of the MAST problem, i.e., eliminating a minimum number of taxa so that the resulting trees on the rest of the taxa all share a common refinement. The algorithm runs in $O(k^2n^2)$ for k trees on n leaves.

- A 3-approximation algorithm for the same problem that runs in $O(k^2n^3)$ time for k trees on n leaves.

The above two results have been published in [GW02]. Apart from the above two results, I also obtained a $O(2^{2kd}n^{k+1})$ time algorithm for computing a maximum compatible tree of k unrooted trees on n leaves, with the maximum degree of any tree being $d+1$ [GW01]. Though this result was obtained independently, the result had been known previously [HJWZ96].

Biogeography. The following are my results in biogeography:

- We show that the equivalence between the edge-contract-and-refine metric ("RF-distance") and the bipartition metric ("character-encoding" metric) that holds for phylogenies *does not hold* for area cladograms. More specifically, we show that the bipartition metric, when extended to area cladograms, is not a metric. For the edge-contract-and-refine edit distance between two area cladograms we present a simple, but worst-case exponential-time algorithm. This edit distance can compare only area cladograms that are on the same number of leaves, and when each area labels the same number of leaves in both area cladograms

- We define another metric, for comparing two rooted area cladograms, which is based on the size of a largest common pruned subtree between the two area cladograms. We call a largest common pruned subtree a *maximum agreement area cladogram (MAAC)*, and the distance metric the MAAC metric. The MAAC distance metric can compare two arbitrary trees that are not necessarily on the same number of leaves, which is particularly useful when comparing area cladograms

- We present a polynomial-time algorithm for computing a MAAC of two rooted area cladograms. The algorithm is a dynamic programming algorithm that runs in $O(n^{2.5}\log n)$ time, where n is the maximum number of leaves in either cladogram. We also describe a linear-time algorithm to decide if two area cladograms are identical. In more recent work not included in this dissertation, a faster algorithm for computing a MAAC of two area cladograms is described, which runs in $O(n^2)$ time, and it is proved that computing the MAAC of k area cladograms is NP-hard. This work has not been included in this dissertation since it is joint work with our co-author Hai-son Le [GGJ+06].

Chapter 2

The p-Edge-Contract-and-Refine Tree Transformation

In this chapter, the p-ECR tree-rearrangement operation and our results on the analysis of the space of phylogenetic trees under the p-ECR operation are described. Recall from Section 1.6.1 that an edge-contract-and-refine (ECR) move contracts some edges in a tree and creates a different tree by refining the unresolved nodes created as a result of the contraction. The p-ECR move is the parameterized version of ECR where p edges are contracted at one time. Along with the p-ECR operation, we also study the p-sECR operation, in which the contracted edges form a connected subtree.

The rest of the chapter is organized as follows:

- In Section 2.1, we explore the nature of the search space under the p-ECR move:
  - We establish bounds on the size of a p-ECR neighborhood, and show that the size of the intersection of the p-ECR and TBR neighborhoods of tree, for $p > 1$, is asymptotically smaller than the size of either neighborhood (Subsection 2.1.1).
  - We establish tight asymptotic bounds on the diameters of the space under the p-ECR

move and the p-sECR move (Subsection 2.1.2).

- In Section 2.2, we provide fast algorithms for local search with the p-ECR move:
  - We provide an algorithm for computing a 2-ECR neighbor of a tree with the best MP score.
  - We provide algorithms for generating a uniformly random p-ECR neighbor of a tree and a uniformly random p-sECR neighbor of a tree.
- In Section 2.3, we analyze the structural properties of the p-ECR move in detail. Through a surprising connection to bipartite graphs with perfect matchings, we show how to transform any given p-ECR move into a sequence of *irreducible* or atomic smaller ECR operations.

2.1 The Search Space of Trees Under the p-ECR Move

We motivated the study of the search space under the p-ECR move in the Introduction, Section 1.6.1, but it is worth recalling the motivations here. Comparing the neighborhood structures of TBR and p-ECR is important for the following reason: most current heuristic searches for maximum parsimony and maximum likelihood employ the TBR move to move through the tree space. If the p-ECR and TBR neighborhoods are significantly different, then it might mean that a tree that is a local optimum under one move might not be so under the other move, and hence switching between p-ECR and TBR moves might be a good way of getting out of local optima.

Calculating the diameter of the space under the p-ECR move is important for the following reason: we would like local moves to induce search spaces with diameters comparable to that of TBR, whose diameter is $\Theta(n)$. Further, obtaining the diameter of the p-ECR search space as a function of p might give us ways to pick a range of values of p to use in a search, based on the diameter.

Before we present our results, we recall some notation first introduced in Chapter 1, Section 1.6.1.

Notation. In what follows, we will denote the neighborhood of a tree T under, say, the p-ECR operation, as $\Gamma_{p-ECR}(T)$. A local move such as p-ECR induces an undirected graph G_{p-ECR} on the space of trees; the vertices of G_{p-ECR} are the trees, and (u,v) is an edge in the graph if trees u and v are p-ECR neighbors. We will denote the diameter of the p-ECR search space by $\Delta(G_{p-ECR})$. Similar notation will apply for other local moves. We will let $r!!$ denote the product of all odd numbers up to r.

Recall that $C(T)$ denotes the character encoding or the set of all bipartitions of a tree T, and $RF(T,T')$ denotes the Robinson-Foulds distance between two trees T and T'.

2.1.1 Comparison Between p-ECR and TBR Neighborhoods

The following observation follows from Definition 5 and Theorem 2 in Chapter 1.

Observation 2 *Let T and T' be two unrooted binary leaf-labeled trees on n leaves, and let p be any integer between 1 and $n-3$. Then $RF(T,T') \leq 2p$ if and only if T and T' are at most one p-ECR move apart.*

We now provide a lower and upperbound on the size of the p-ECR neighborhood of a tree.

Lemma 1 *Let T be an unrooted binary leaf-labeled tree on n leaves. Then, $\sum_{k=1}^{p} \binom{n-3}{k} 2^k \leq |\Gamma_{p-ECR}(T)| \leq \binom{n}{p}(2p+1)!!$.*

Proof: For any tree T' in $\Gamma_{p-ECR}(T)$, $RF(T,T') \in \{2,4,6,\ldots,2p\}$. We will show that the number of trees T' in $\Gamma_{p-ECR}(T)$ such that $RF(T,T') = 2k$ is at least $\binom{n-3}{k}2^k$, and that will give us the result that we desire.

Let k be such that $1 \leq k \leq n-3$. For every way of choosing k edges in T, there are at least 2^k different k-ECR moves that can be performed on T: for each chosen edge, contract the edge

and refine the resulting unresolved node in one of two ways that results in the alteration of the bipartition corresponding to the edge. Since all the k bipartitions are altered, the resulting set of 2^k k-ECR neighbors is disjoint from any other set of similar 2^k k-ECR neighbors. Thus, there are at least $\binom{n-3}{k}2^k$ trees T' such that $RF(T,T') = 2k$. This completes our proof of the lower bound. For the upper bound, observe that for each of the $\binom{n}{p}$ ways of selecting p edges to contract, there are at most $(2p+1)!!$ neighbors ($(2p+1)!!$ is the number of unrooted leaf-labeled binary trees with p internal edges). This completes our proof. □

The following is the main result of this section, and shows that the size of the intersection of the p-ECR and the TBR neighborhoods is very small.

Theorem 5 *Let T be an unrooted binary leaf-labeled tree on n leaves. Then, for any p,*

$$|\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)| \leq min\{(2n-3)(p+1)2^{p+3}, 4(2n-3)^2(p+1)\}$$

.

Proof: Let $S = \Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)$, and let T' be in S. Then, $|C(T) - C(T')| \leq p$, since $T' \in \Gamma_{p-ECR}(T)$. Moreover, since $T' \in \Gamma_{TBR}(T)$, the edges in T corresponding to bipartitions in $C(T) - C(T')$ all must lie on a path, and the bipartitions corresponding to all edges on the path except three (the first edge, the last edge and the edge that is broken for the TBR move) must be in $C(T) - C(T')$. Hence, each such T' can be specified by three edges that lie on a path of length at most $(p+3)$. Now, the number of paths of length at most $(p+3)$ is at most $(2n-3)2^{p+3}$. This is because the number of paths of length exactly $p+3$ is at most $(2n-3)2^{p+2}$: fix one of the terminal edges of the path, and there are at most 2^{p+3} paths with a given terminal edge since the tree is binary. But in this manner each path will be counted at least twice, and hence there are at most $(2n-3)2^{p+2}$ paths of length exactly $(p+3)$. Summing over all p we get that the number of paths of length at most $(p+3)$ is at most $(2n-3)2^{p+3}$.

Also, each path of length at most $(p+3)$ corresponds to at most $(p+1)$ trees that are in S, since there are $(p+1)$ ways of choosing the edge that is broken for the TBR move. Hence we have that $|S| \leq (2n-3)2^{p+3}(p+1)$.

Moreover, the total number of paths in T is $(2n-3)^2$. For every tree in S, there is a path in T, and each path contributes at most $4(p+1)$ trees to S. Hence $|S| \leq 4(2n-3)^2(p+1)$.

Thus, we have that $|\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)| \leq min\{(2n-3)(p+1)2^{p+3}, 4(2n-3)^2(p+1)\}$.

□

Note that $|\Gamma_{p-ECR}(T)| \in \omega(\binom{n-3}{p}2^p)$ and recall from Chapter 1 that $\Gamma_{TBR}(T) \in \Theta(n^3)$. Thus, Theorem 5 implies that $|\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)| \in o(min\{|\Gamma_{p-ECR}(T), \Gamma_{TBR}(T)|\})$. In other words, the size of the intersection of the p-ECR and TBR neighborhoods is asymptotically smaller than either neighborhood.

The following is the analogous result for the p-sECR operation. The proof is similar to that of Theorem 5.

Lemma 2 *Let T be an unrooted binary leaf-labeled tree on n leaves. Then, for any p,*

$$|\Gamma_{p-sECR}(T) \cap \Gamma_{TBR}(T)| \leq min\{(2n-3)(p+1)2^{p+2}, 4(2n-3)^2(p+1)\}.$$

The following result relates the above two results by comparing $\Gamma_{p-sECR} \cap \Gamma_{TBR}$ with $\Gamma_{p-ECR} \cap \Gamma_{TBR}$.

Lemma 3 *For any $p > 1$, there exists an unrooted binary leaf-labeled tree T such that $\Gamma_{p-sECR}(T) \cap \Gamma_{TBR}(T)$ is a proper subset of $\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)$.*

Further, for any tree T and for any $p \geq 1$, $\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T) \subseteq \Gamma_{(p+1)-sECR}(T) \cap \Gamma_{TBR}(T)$.

Proof: In Figure 2.1 the tree T' is in $|\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)|$, as evinced by the bold edges in T that correspond to bipartitions in $C(T) - C(T')$. However, $T' \notin |\Gamma_{p-sECR}(T) \cap \Gamma_{TBR}(T)|$ since the p bold edges do not form a subtree. This proves the first part of the lemma.

For the second part, let T' be a tree in $\Gamma_{p-ECR}(T) \cap \Gamma_{TBR}(T)$. Then, as observed in Theorem 5, the edges corresponding to bipartitions in $C(T) - C(T')$ lie on a path and all except at most one edge (the edge that is broken for the TBR move) in the path are in $C(T) - C(T')$ (i.e., the corresponding bipartitions are in $C(T) - C(T')$). There are at most p bipartitions in $C(T) - C(T')$ and hence the length of the path is at most $p+1$. The tree T' can be obtained by contracting all edges on the path and subsequently introducing the edges corresponding to bipartitions in $C(T') - C(T)$ through refinements, while retaining the edge that gets broken for the TBR move. This corresponds to a $(p+1)$-sECR operation on T, and hence $T' \in \Gamma_{(p+1)-sECR}(T) \cap \Gamma_{TBR}(T)$. This completes the proof of the second part of the lemma. □

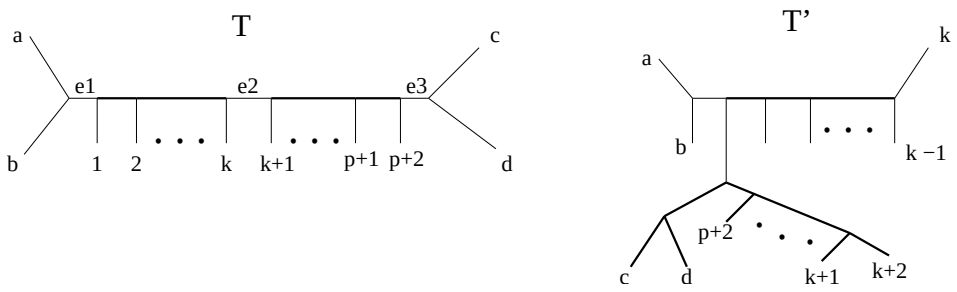


Figure 2.1: Trees T and T' are TBR neighbors; The TBR move deletes edge $e2$ in T and introduces an edge bifurcating edges $e1$ and $e3$. Tree T' is a p-ECR neighbor, but not a p-sECR neighbor of T. The p-ECR move contracts the bold edges in T and introduces the bold edges in T'.

2.1.2 Diameter of the p-ECR Search Space

My results prove asymptotically tight bounds for the diameter of the tree-space induced by the p-ECR operation (as a function of p). As mentioned earlier, it was shown in [LTZ96] that $\Delta(G_{NNI}) \in \Theta(n \log n)$, and it was shown in [AS01] that $\Delta(G_{TBR}) \in \Theta(n)$. The NNI operation is just the 1-

ECR operation. Hence the diameter of the 1-ECR operation is in $\Theta(n \log n)$. The diameter of the $(n-3)$-ECR operation is, of course, 1. As observed earlier, obtaining the diameter as a function of p might give us a way to pick the right range of values of p to use in a search, based on the diameter.

We prove the upperbound for the p-sECR move, and the lower bound for the p-ECR move. Since p-ECR subsumes p-sECR, the upperbound and lowerbound then hold for both the moves.

Theorem 6 $\Delta(G_{(2p-2)-sECR}) \in O(\frac{n \log n}{p \log p})$.

Proof: Let C be the sorted "caterpillar" tree for the set of leaf labels $\{1, 2, \ldots, n\}$ (Figure 2.2). We will show that for any unrooted binary leaf-labeled tree T, $\delta_{(2p-2)-sECR}(T, C) \leq \frac{n}{p} \log_p n + O(\frac{n}{p})$.

We will first show that the number of $(2p-2)$-sECR steps needed to convert a complete binary tree on n leaves to a caterpillar C is at most $\frac{n}{p} \log_p n$. We then show that any tree can be transformed into a complete binary tree in $O(\frac{n}{p})$ steps, each step being a $(2p-2)$-sECR move. The above two results would then imply that any tree can be converted to C in at most $\frac{n}{p} \log_p n + O(\frac{n}{p})$ steps. A complete binary tree and the sorted caterpillar tree for the set of leaves labeled from 1 through 7 are shown in Figure 2.2.

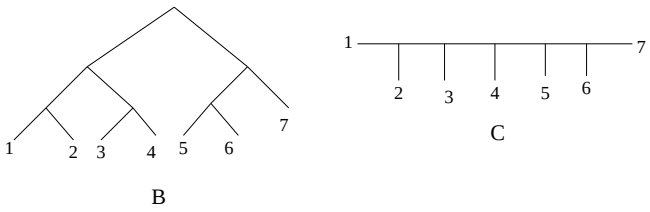


Figure 2.2: B is a complete rooted binary tree on seven leaves. Note that in general the leaves need not be in sorted order in the tree. C is the sorted caterpillar tree for the same set of leaves.

Converting an arbitrary complete binary tree to a sorted caterpillar tree. The procedure is recursive, and is illustrated in Figures 2.3 and 2.4. Let B be the complete binary tree on n leaves. Let $B_1, B_2, B_3, \ldots B_p$ be the subtrees of B at a depth of $\log_2 p$. Since B is a complete binary tree, so are subtrees B_1 through B_p. Recursively convert each of the p subtrees to a sorted caterpillar tree,

producing the *bc-tree* (binary-cum-caterpillar tree) B'. The subtrees of B' at a depth $\log_2 p$ are now sorted caterpillar trees C_1 through C_p (see Figure 2.3).

The following process is illustrated in Figure 2.4: consider the first p leaves in the sorted order. These p leaves can be "pulled" up to the root by contracting only $2p-2$ edges (this is because the caterpillar trees C_1 through C_p are sorted). The contraction of these $2p-2$ edges make the root of the tree unresolved, with each of the p leaves now a descendant of the root. To complete the $(2p-2)$-sECR operation, we have to refine the root, and when we refine the root the p leaves can be transferred to "above" (see Figure 2.4) the root in sorted order. The next $(2p-2)$-sECR move will transfer the next p leaves in the sorted order to above the root. In this manner we can obtain a C from B' in $\frac{n}{p}$ $(2p-2)$-sECR moves. This gives us the following recursive equation for the number of moves required to convert B to C. Let $S(n)$ denote the number of $(2p-2)$-sECR steps required to convert an n-leaf complete binary tree to the corresponding sorted caterpillar tree. Then,

$$S(n) \leq pS(\frac{n}{p}) + \frac{n}{p}.$$

Solving the recurrence yields us $S(n) \leq \frac{n}{p}\log_p n$.

Converting any tree to a complete binary tree with p-sECR moves. We now show that any tree can be transformed into a complete binary tree in $O(\frac{n}{p})$ steps, each step being a p-sECR move. This will then give us the desired result. Note that these transformations concern only the tree topology and not the leaf labels.

In a caterpillar we define the *terminal leaves* to be the two pairs of leaves at each end of the caterpillar; the remaining leaves are the *internal leaves*. The path connecting the two pairs of terminal leaves is the *spine* of the caterpillar. We define a q-caterpillar as a caterpillar in which each internal leaf is replaced by a *q-spoke*, which is a caterpillar with $q-2$ internal leaves and one pair of terminal leaves (see Figure 2.5). The last spoke in the q-caterpillar (one that is adjacent to the parent of one of the two terminal pairs of leaves) is a q'-spoke for some $q' \leq q$.

Let T be an unrooted binary tree. It can be shown by induction on the number of leaves that any binary tree with at least four leaves contains at least two pairs of leaves that are siblings. We fix two such pairs of sibling leaves in T and we call the unique path connecting their parents x_1 and x_2 the spine of T. Note that the spine thus depends on the choice of the siblings. We fix one of the parents, say x_1, and relative to x_1 we define the *potential* $\phi(T) = 2 \cdot n_1 + n_2$, where n_1 is the number of successive q-spokes in T starting with the vertex adjacent to x_1 on the spine, and n_2 is the number of successive subtrees with at least q leaves rooted at vertices on the spine following the initial n_1 q-spokes.

Consider the transformation of an arbitrary binary tree T into a q-caterpillar using p-sECR moves, where $q = \lfloor p/2 \rfloor$. The initial potential of T is non-negative and final potential of the transformed tree is $2 \cdot \lfloor (n-4)/q \rfloor$. We now describe a method that transforms T into a q-caterpillar using p-sECR moves that increase the potential of the transformed tree in each step by at least one. Thus this method transforms T into a q-caterpillar in $O(\frac{n}{p})$ moves.

Our method will apply a p-sECR move by contracting edges starting with the edges in the subtree rooted at the vertex v on the spine, which is adjacent to the last vertex that is a root of one of the n_1 q-spokes already constructed (if $n_1 = 0$ then this is the vertex adjacent to x_1 on the spine). The resulting contracted subtree S on p internal edges will have $p+3$ external edges incident on it, of which two are on the spine and $p+1$ are within subtrees rooted at one or more vertices on the spine. If S can be refined to form a new successive q-spoke, then the potential increases by at least one. Otherwise, at least q external edges end in subtrees, each of which contain at least two leaves, which implies that S can be refined into two subtrees, each containing at least q leaves. Further, since S could not be refined into a q-spoke, the subtree rooted at v was not of size between q and $2q$, so one of the two subtrees formed is a new one, resulting in an increase in potential of at least one. Hence T can be transformed into a q-caterpillar in $O(\frac{n}{p})$ p-sECR moves.

By reversing the above strategy the q-caterpillar can be transformed into any binary tree, including a complete binary tree, in the same number of moves. Hence any binary tree can be

transformed into a complete binary tree in $O(\frac{n}{p})$ steps, each step being a p-sECR move. □

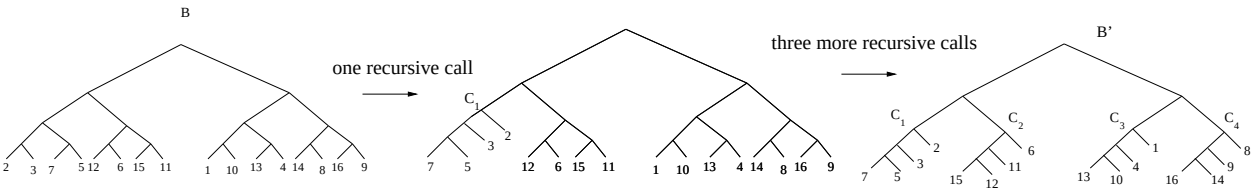


Figure 2.3: Converting a complete binary tree B to a *bc*-tree B'

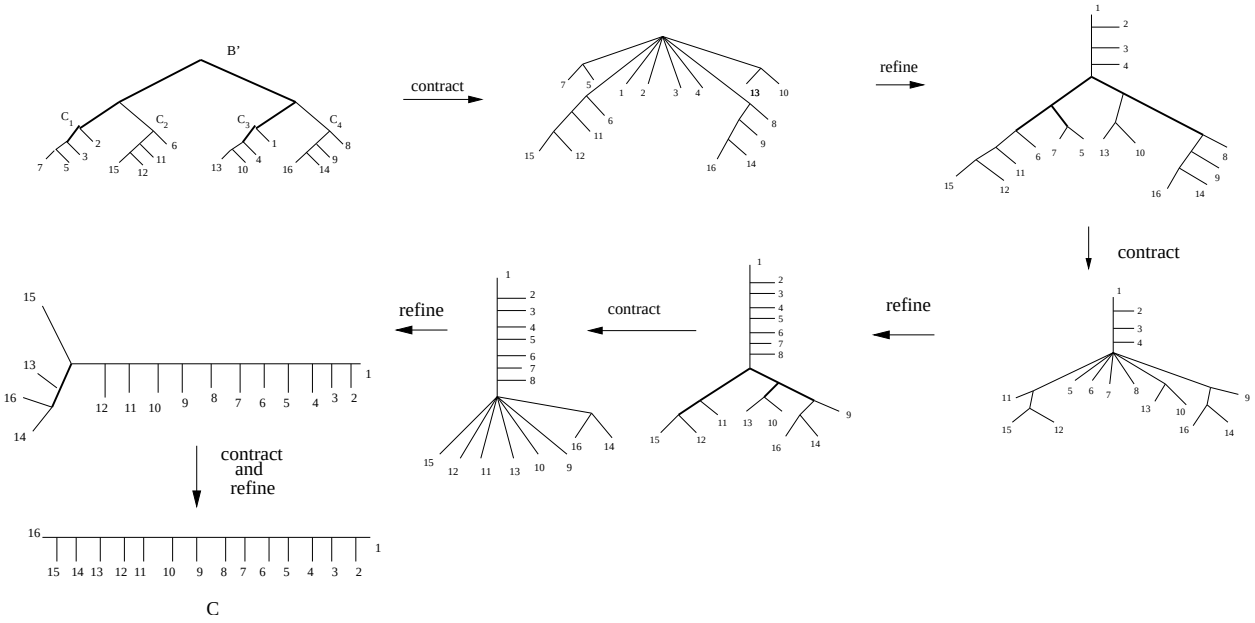


Figure 2.4: Converting a *bc*-tree B' to a sorted caterpillar tree C with 6-ECR moves. At every contraction the bold edges are contracted.

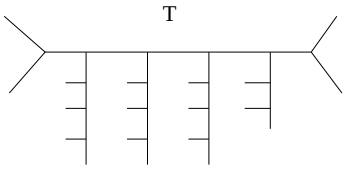


Figure 2.5: Tree T is a 4-caterpillar. The leaf-labels are not shown since they are not relevant in the transformation of a tree to a q-caterpillar.

The following theorem establishes a lower bound on the p-ECR diameter.

Theorem 7 $\Delta(G_{p-ECR}) \geq \frac{n\log_2 n - o(n\log n)}{8p\log_2 p + O(p)}$, *for all* $p > 1$.

Proof: Let T be an unrooted binary tree on n labeled leaves, and let T' be a tree such that $\delta_{p-ECR}(T,T') = 1$. It can be established that $\delta_{NNI}(T,T') \leq 2p\log_2 p + O(p)$, for $p > 1$.

Without loss of generality, assume that $|C(T) - C(T')| = p$, and let X be the p-ECR operation that transforms T to T'. If the edges corresponding to bipartitions in $C(T) - C(T')$ form a connected subtree of T, say S, let the corresponding connected subtree in T' be S'. Then S and S' may be considered to form a tree with $p+3$ leaves each, and S can be transformed into S' using at most $2p\log_2 p + O(p)$ NNI moves, as was shown in [LTZ96].

If the edges corresponding to bipartitions in $C(T) - C(T')$ form a forest of k connected subtrees, then it can be shown that there exist some k ECR operations X_1 through X_k that act on disjoint sets of bipartitions such that X is equivalent to performing the k operations X_1 through X_k one after the other. Let X_i be a p_i-ECR operation. We have $\sum_{i=1}^{k} p_i = p$. Let T^i be the tree obtained by the application of X_i on T^{i-1} (thus, $T = T^0$ and $T' = T^k$). Then, $\delta_{NNI}(T^{i-1}, T^i) \leq 2p_i\log_2 p_i + O(p_i)$. Hence, $\delta_{NNI}(T,T') \leq 2\sum_{i=1}^{k} p_i \log p_i + O(p)$. The summation on the right-hand side is less than $p\log_2 p$ for all $p > 1$ and $k > 1$, and hence $\delta_{NNI}(T,T') \leq 2p\log_2 p + O(p)$, for $p > 1$.

From the results in [LTZ96, AS01], $\Delta(G_{NNI}) \geq \frac{n\log_2 n - o(n\log n)}{4}$. This, together with the fact that any p-ECR move can be emulated by at most $2p\log_2 p + O(p)$ NNI moves, gives us the desired result. □

As observed earlier, since p-ECR is strictly more general than p-sECR for $p > 1$, $\Delta(G_{p-sECR}) < \Delta(G_{p-ECR})$ for $p > 1$. For $p = 1$, a 1-ECR move is also a 1-sECR move. Thus, we have:

Corollary 1 $\Delta(G_{p-ECR})$ *and* $\Delta(G_{p-sECR})$ *are both in* $\Theta(\frac{n\log n}{p\log p})$.

2.2 Efficient Hill-Climbing with 2-ECR

The problem of computing an optimal 2-ECR neighbor under the parsimony criterion arises during steepest descent hill-climbing heuristic for maximum parsimony, where at each step one chooses the neighbor with the least parsimony score to be the next candidate tree. In this section we present a fast algorithm for computing an optimal 2-ECR neighbor under parsimony. The technique used to obtain the fast algorithm is general and has been used to obtain fast algorithms for the optimal neighbor problem under the NNI, SPR and TBR moves as well (see [Swo86, Gol96]). We now define the Optimal 2-ECR Neighbor problem.

Definition 18 *Optimal* 2*-ECR Neighbor*

Input *An unrooted binary tree T on n leaves, each bijectively leaf-labeled by a set S of sequences of length k over an alphabet of size r.*

Output*: An unrooted binary tree T' on n leaves, each bijectively labelled by the same set S, such that T' has the minimum MP score among all such trees t for which $\delta_{2-ECR}(T,t) = 1$.*

Henceforth, we will call such a tree an *optimal* 2*-ECR-neighbor* of T. The *Optimal TBR-neighbor*, *Optimal SPR-neighbor* and *Optimal NNI-neighbor* problems are defined similarly. Note also that there can be more than one optimal neighbor, and that in general the objective is to find one optimal neighbor.

We observe that a brute-force algorithm for the above problem would take $\Theta(n^3rk)$ time, since there are $\Theta(n^2)$ 2-ECR neighbors for any tree, and computing the parsimony score of each tree using Fitch's algorithm 1.3 would take $\Theta(nrk)$ time. We will obtain a $\Theta(n^2rk)$ time algorithm for the Optimal 2-ECR neighbor problem which will return all the optimal neighbors.

2.2.1 The Three Labels Technique

As was observed in Section 1.6.1, an *NNI* move can be thought of as a 1-ECR move. So, not surprisingly, we use a fast algorithm for computing optimal NNI-neighbors in our algorithm for computing optimal 2-ECR-neighbors. A brute-force optimal NNI neighbors algorithm would run in $\Theta(n^2rk)$ time, but our algorithm runs in $\Theta(nrk)$ time.

The way we obtain a speed-up over the brute-force techniques for each of the problems we address is by performing a preprocessing step in which we assign three labels to each node in the tree.

In order to understand why we do this preprocessing step, consider an NNI move across an edge, say (u,v), in a given tree T. Let W and X be the rooted subtrees below u, and let Y and Z be the rooted subtrees below v. The NNI move will, for example, involve swapping W with Y. Let the resulting tree be T'. Supposing that we have the parsimony scores and optimal state assignments for the rooted subtrees W, X, Y and Z, the parsimony score of T' can be computed in $\Theta(rk)$ time, thus: we can subdivide edge (u,v) and root T' at the newly created node, which we shall call x. This produces a binary tree rooted at x, with subtrees off x rooted at u and v. The parsimony score of the subtree of T' rooted at u depends just on the parsimony scores and optimal state assignments of X and Y, and can be computed from them in $\Theta(rk)$ time, as in Fitch's algorithm. Similarly, the parsimony score of the subtree of T' rooted at v can be computed in $\Theta(rk)$ time. Finally, the parsimony score of T' (rooted at x) can be computed in $O(rk)$ time from the parsimony scores and optimal state assignments of the subtrees rooted at u and v.

The above observations suggest that a preprocessing stage that computes the parsimony score and the optimal state assignments for every rooted subtree will let us compute the parsimony score of each NNI neighbor in $\Theta(rk)$ time. The scores and the associated labels of a subtree would be stored at the internal node that is its root. Each internal node is a root of three such subtrees (since the tree is binary), and hence three such labels have to be computed for each internal node (this is illustrated in Figure 2.6). Hence, this preprocessing step is called the *Three-Way Labels*

algorithm. The brute-force way of performing this preprocessing step would take $\Theta(n^2rk)$ time, but this stage can be completed in $\Theta(nrk)$ time, by using a dynamic programming technique, the details of which we omit. We have:

Lemma 4 *The Three-Way Labels algorithm takes $O(nrk)$ time, where n is the number of leaves in the tree T, and each leaf is labeled by a sequence of length k over an alphabet of size r.*

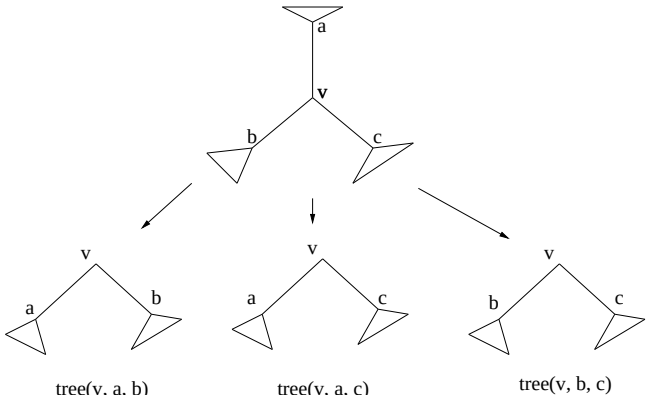


Figure 2.6: Internal node *v* and the three subtrees associated with it.

As we saw previously, the preprocessing stage would let us compute the parsimony score of each NNI-neighbor in $\Theta(rk)$ time. Since there are $2n-6$ NNI-neighbors, the optimal NNI neighbors can be identified in $\Theta(nrk)$ time after the preprocessing step. To summarize,

Theorem 8 *We can solve the Optimal NNI-Neighbors Problem in $\Theta(nrk)$ time.*

In the following section, we show how to use the optimal NNI neighbor algorithm to compute the optimal 2-ECR neighbors in $O(n^2rk)$ time.

2.2.2 Computing an optimal 2-ECR neighbor

We now show how to compute an optimal 2-ECR neighbor of an unrooted binary tree on *n* labeled leaves, each labeled by a sequence of length *k* over an alphabet of size *r*, in $\Theta(n^2rk)$ time, thus spending only $\Theta(rk)$ time per neighbor.

A 2-ECR move on such a tree T is specified by (i) the two edges e_1, e_2 to be contracted, and (ii) the refinement of the resulting contracted tree into an unrooted binary tree that differs from T.

Our algorithm will handle the following two cases separately.

1. The edges e_1 and e_2 are not adjacent to each other.

2. The edges e_1 and e_2 are adjacent to each other.

We now show how to handle case (1). We first prove that in this case any 2-ECR move can be "simulated" by two successive NNI moves.

Observation 3 *Let T^* be an unrooted leaf-labelled tree on n leaves that has exactly p unresolved nodes, each of degree 4. Then, the number of ways to refine T^* is exactly 3^p.*

Proof: There are three ways to resolve each of the p unresolved nodes. □

Lemma 5 *Let T be an unrooted binary tree on n leaves. The number of trees (not including T) that can be obtained from T by performing one p-ECR move involving a given set S of p internal edges, no two of which are adjacent, is exactly $3^p - 1$.*

Proof: A p-ECR move involving such a set S of p edges first generates a tree T^* with exactly p unresolved nodes, each of degree four, and then refines T^*. From Observation 3, the number of ways to refine T^* is 3^p. However, one of the refinements gives back T, and must be discounted. □

Thus there are exactly eight trees that can be obtained by performing a 2-ECR move on a tree T involving two given non-adjacent edges e_1 and e_2. Let us denote by $N(e_1, e_2)$ the set of new trees that can be obtained by performing a 2-ECR move involving edges e_1 and e_2 in T. As mentioned earlier, any tree that can be obtained by an NNI move just on e_1, by an NNI move just on e_2, or by an NNI move on e_1 followed by an NNI move on e_2, will be in $N(e_1, e_2)$, but the latter

may contain trees that cannot be obtained by sequentially performing NNI on e_1 and e_2. For any edge e, let $N(e)$ denote the set of new trees formed by performing NNI on e, and let $N(e_1) \cdot N(e_2)$ be the set of new trees formed by performing NNI on e_2 for each tree in $N(e_1)$.

Corollary 2 *If e_1 and e_2 do not share a vertex, then $N(e_1, e_2) = N(e_1) \cup N(e_2) \cup N(e_1) \cdot N(e_2)$.*

Proof: As observed earlier, each of $N(e_1)$, $N(e_2)$ and $N(e_1) \cdot N(e_2)$ is a subset of $N(e_1, e_2)$. Also, these sets are pairwise disjoint. But $|N(e_1)| + |N(e_2)| + |N(e_1) \cdot N(e_2)| = 2 + 2 + 4 = 8 = |N(e_1, e_2)|$ by Lemma 5. □

We now show how we deal with case 1 (non-adjacent edges) and case 2 (adjacent edges).

- **Case 1.** Corollary 2 means that a 2-ECR move involving two separated edges e_1 and e_2 can be simulated by two successive NNI moves involving the contraction and refinement of e_1 and e_2. To compute the optimal 2-ECR neighbor of T in this case, we do the following:

  1. For each tree $T' \in \mathcal{N}_{NNI}$, compute the best NNI-neighbor of T' . This can be accomplished in $\Theta(nrk)$ time (from Theorem 8).
  2. There are $\Theta(n)$ trees in $\mathcal{N}_{NNI}$. Hence, the set of optimal 2-ECR neighbors can be computed in $\Theta(n^2rk)$ time, when the two edges are contracted and refined one after another.

- **Case 2.** We observe that there are only $O(n)$ pairs of adjacent edges. We can process each such pair in $O(nrk)$ time using the brute-force method, for a total of $O(n^2rk)$ time. Here we can again use the three-way labeling technique from Section 2.2.1 to obtain a factor of n improvement in running time over the brute-force method as follows.

  The contraction of two adjacent edges creates a single unresolved node of degree five. The number of ways of refining this degree five unresolved node is $5 \times 3 = 15$. Each of the 15 refinements involves only a rearrangement of the five rooted subtrees that span out of the

unresolved node around the two new edges that result from the refinement. However we can compute the optimal labels at the root of all such subtrees in $\Theta(nrk)$ time in a preprocessing step as in Section 2.2.1. Hence for each 2-ECR involving two adjacent edges, we can compute the optimal neighbor in $\Theta(rk)$ time. There are only $\Theta(n)$ pairs of adjacent edges. Hence, in this case we can compute the best 2-ECR neighbor in $\Theta(nrk)$ time.

The overall optimal 2-ECR neighbors will be those with the best score, computed as per Case 1 (non-adjacent edges) and Case 2 (adjacent edges).

Theorem 9 *Let T be an unrooted binary tree on n leaves, each labeled by a sequence of length k over an alphabet of size r. Then the optimal 2-ECR neighbors of T can be computed in $\Theta(n^2rk)$ time.*

The use of the 3-Way-Labels algorithm, finding optimal 2-ECR neighbors is new, but similar techniques have been presented before (see [Swo86, Gol94, Gol96]).

2.2.3 Random Sampling from the *p*-ECR and the *p*-sECR Neighborhoods

The use of MCMC (Markov Chain Monte Carlo) algorithms in phylogenetic reconstruction is of increasing interest in the research and user community [HRNB01, HR01, LS99]. In this section, therefore, we address the problems of generating a *p*-ECR neighbor and a *p*-sECR neighbor uniformly at random. We assume that the trees are binary, a condition satisfied by all intermediate trees in any current MCMC, MP or ML heuristic [HR01, Swo96, GG03, BHD$^+$02].

Basic Strategy. Before we begin, we formally define the random *p*-ECR neighbor problem:

Definition 19 *Random p-ECR Neighbor*

Input *An unrooted binary tree T on n leaves, each bijectively leaf-labeled by a set S.*

Output*: An unrooted binary tree T' on n leaves, each bijectively labelled by the same set S, chosen uniformly at random from $\Gamma_{p-ECR}(T)$, the p-ECR neighborhood of T.*

The random p-sECR neighbor problem is defined in a similar manner.

We approach both the problems in a similar manner, in two phases:

- **Phase 1.** Choose at random the edges to be contracted.
- **Phase 2.** A apply a random move on the chosen edges.

It turns out that Phase 1 is considerably simpler for a random p-ECR move than for a random p-sECR move. Further, Phase 2 for the p-ECR move can be reduced to a series of applications of p'-sECR Phases 2, for suitable values of p'. Hence, we first describe our random p-sECR neighbor algorithm and then the random p-ECR neighbor algorithm.

A Random p-sECR Neighbor

As described above, we first choose a p-subtree uniformly at random, and then apply a p-sECR move at random on the chosen subtree.

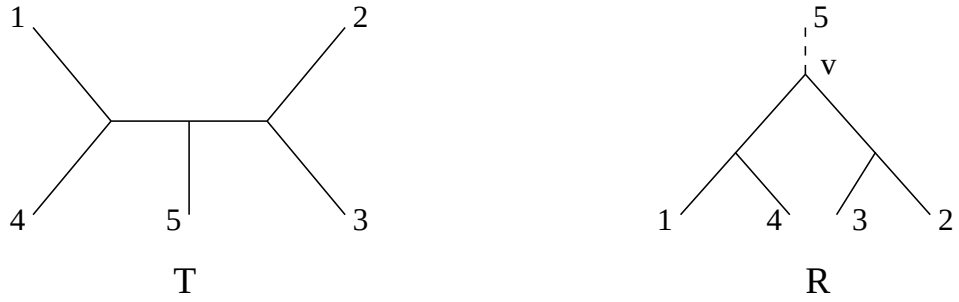


Figure 2.7: The unrooted tree T has been rooted at the leaf labeled 5 to give the rooted tree R. Tree R is considered to have four leaves, with the fifth leaf being implicit, as indicated by the dotted line attaching 5 to the tree R. Node v is the root of tree R.

Choosing a p-subtree Uniformly at Random. Let T be the input tree on n leaves. We first root tree T on the highest labeled leaf, as shown in Figure 2.7 to obtain tree R. Note that this does not change the p-sECR neighborhood, since no internal edges have been deleted or introduced in obtaining R from T. Let $V(R)$ be the set of all nodes in R, and let R_v be the maximal subtree of R

rooted at a node v. Let $A[p,v]$ denote the number of p-subtrees rooted at v. The first phase of our algorithm proceeds as follows:

1. Root the input tree T to obtain tree R.
2. Choose an internal node v from R with probability $\frac{A[p,v]}{\sum_{w \in V(R)} A[p,w]}$.
3. Choose a p-subtree uniformly at random from among all p-subtrees rooted at v, i.e., with probability $\frac{1}{A[p,v]}$.

Note that for any p-subtree X with root v, the probability that X is chosen is $\frac{A[p,v]}{\sum_{w \in V(R)} A[p,w]} \times \frac{1}{A[p,v]} = \frac{1}{\sum_{w \in V(R)} A[p,w]}$. Thus, the above procedure chooses p-subtrees uniformly at random, provided steps 2 and 3 are performed correctly.

We adopt the following convention regarding A: we assume $A[0,w] = 0$ if w is a leaf and 1 otherwise. With this convention, we state and prove a recurrence relation satisfied by A:

Lemma 6 *Let $A[k,w]$ be the number of k-subtrees rooted at node w of a given rooted tree. Then, for all $k \geq 0$ and $w \in V(R)$, $A[k,w] = B[k,w]$ where,*

$$\begin{aligned} B[k,w] &= 0 \quad \textit{for any leaf } w, \textit{ for } k \geq 0 \\ &= 1 \quad \textit{for } k = 0, \textit{ for any non-leaf } w \\ &= B[k-1,w_l] + B[k-1,w_r] + \sum_{i=1}^{k-1} B[i-1,w_l] B[k-i-1,w_r] \quad \textit{for } k > 0, \textit{ for any non-leaf } w. \end{aligned}$$

Here, w_l and w_r are the left and right children of w respectively.

Proof: Let O be a post-ordering of the $V(R)$, and let Z^+ be the set of non-negative integers. Define a partial order $\lhd$ on $\mathcal{P} = Z^+ \times V(R)$ as follows: $(k_1,w_1) \lhd (k_2,w_2)$ if and only if w_1 is a proper descendant of w_2 and $k_1 < k_2$. The proof is by induction on $\mathcal{P}$ partially ordered by $\lhd$. The base case consists of pairs (k,w) such that w is a leaf or $k = 0$. Our recurrence correctly sets $B[k,w] = 0$

for $k > 0$ where w is a leaf. For the rest of the base case pairs (k, w), our recurrence is correct by our convention that $A[0, w] = 0$ if w is a leaf and 1 otherwise. Now assume by way of induction that for some pair $(k, w) \in \mathcal{P}$ such that $k > 0$ and w not a leaf, $A[x, u] = B[x, u]$ for all $(x, u) \lhd (k, w)$. We will now show that $A[k, w] = B[k, w]$. Let w_l and w_r be the left and right children of w.

- If w_l and w_r are both leaves, $A[k, w] = 0$, and $B[k, w] = 0$ since we set $B[x, u] = 0$ for all leaves u irrespective of x.

- If w_l is a leaf while w_r is not, then $B[k, w] = B[k-1, w_r]$. Now, if w_l is a leaf, then all k-subtrees rooted at w have all their edges to the right of w. Each such k-subtree consists of the edge (w, w_r) together with a $(k-1)$-subtree rooted at w_r. Thus, $A[k, w] = A[k-1, w_r]$, which equals $B[k-1, w_r]$ by the induction hypothesis. Note that our convention that $A[0, w_r] = 1$ ensures that $A[1, w] = 1$. Thus, we have $A[k, w] = B[k, w]$. The argument is similar when w_r is a leaf while w_l is not.

- If neither w_l nor w_r is a leaf, then, let $A[k, x, w]$ be the number of k-subtrees rooted at w that have x edges to the left of w. Then, $A[k, w] = \sum_{x=0}^{k} A[k, x, w]$. Note that $A[k, x, w] = A[x-1, w_l] \times A[k-x-1, w_r]$ for $0 < x < k$. This is because each k-subtree rooted at w with x edges to the left is made of a $(x-1)$-subtree rooted at w_l and a $(k-x-1)$-subtree rooted at w_r along with the two edges (w, w_l) and (w, w_r). If $x = 0$, then $A[k, x, w] = A[k-1, w_r]$, since each such k-subtree is made of a $(k-1)$-subtree rooted at w_r and the edge (w, w_r). Similarly, $A[k, k, w] = A[k-1, w_l]$. Thus, $A[k, w] = A[k-1, w_l] + A[k-1, w_r] + \sum_{i=1}^{k-1} A[i-1, w_l] A[k-i-1, w_r]$, which equals $B[k, w]$ by the induction hypothesis.

□

The above recurrence suggests an obvious bottom-up algorithm for choosing a p-subtree uniformly at random from a rooted subtree R. The algorithm is given in pseudocode below. The main routine RAND-SUBTREE takes p and R as input and returns a p-subtree of R chosen uni-

formly at random. This routine invokes two subroutines: COMPUTE-NUM-SUBTREES computes the values $B[k,w]$ for $0 \leq k \leq$ and for all $w \in V(R)$ using the above recurrence relation. The routine RAND-SUBTREE-GIVEN-ROOT returns a uniform random p-subtree rooted at a specified node v.

COMPUTE-NUM-SUBTREES(p, $R = (V(R), E(R))$) {

/* Compute the number of p-subtrees in the rooted tree R */

Compute a post-order O of the nodes in $V(R)$

foreach (leaf l)

for ($i = 0$ to p)

$B[i,l] \leftarrow 0$

foreach (internal node w in post-order O)

/* w_l is the left child of w, and w_r the right child */

$B[0,w] \leftarrow 1$

for ($i = 1$ to p)

$B[i,w] \leftarrow B[i-1,w_l] + B[i-1,w_r] + \sum_{j=1}^{i-1} B[j-1,w_l]B[i-j-1,w_r]$

}

RAND-SUBTREE-GIVEN-ROOT(p, $R = (V(R), E(R))$, v) {

/* Choose a p-subtree rooted at v uniformly at random. */

$H \leftarrow \emptyset$ /*H will hold the edges of the subtree */

if (v is a leaf)

return H

/* Let v_l be the left child of v, and let v_r be the right child */

$Q[0] \leftarrow \frac{B[p-1,v_r]}{B[p,v]}$

$Q[p] \leftarrow \frac{B[p-1,v_l]}{B[p,v]}$

for ($i = 1$ to $p-1$)

$Q[i] \leftarrow \frac{B[i-1,v_l]B[p-i-1,v_r]}{B[p,v]}$

$k \leftarrow$ a random variable from $[0,p]$ with distribution Q

/* k will be the number of subtree edges in the left subtree */

if $(k = 0)$

$H \leftarrow H \cup \{(v,v_r)\} \cup$ RAND-SUBTREE-GIVEN-ROOT($p-1$, R, v_r)

if $(k = p)$

$H \leftarrow H \cup \{(v,v_l)\} \cup$ RAND-SUBTREE-GIVEN-ROOT($p-1$, R, v_l)

if $(0 < k < p)$

$H \leftarrow H \cup \{(v,v_r),(v,v_l)\}$

$H \leftarrow H \cup$ RAND-SUBTREE-GIVEN-ROOT($k-1$, R, v_r)

$H \leftarrow H \cup$ RAND-SUBTREE-GIVEN-ROOT($p-k-1$, R, v_r)

return H

}

RAND-SUBTREE(p, $R = (V(R),E(R))$) {

/* Choose a p-subtree of R uniformly at random */

$B \leftarrow$ COMPUTE-NUM-SUBTREES(p, R)

$sum \leftarrow 0$

foreach (internal node w)

$sum \leftarrow sum + B[p,w]$

$v \leftarrow$ an internal node chosen with probability $\frac{B[p,v]}{sum}$

$S \leftarrow$ RAND-SUBTREE-GIVEN-ROOT(p, R, v)

return S

}

Lemma 7 *After COMPUTE-NUM-SUBTREES terminates, $B[k,w]$ holds the number of k-subtrees rooted at w.*

Proof: COMPUTE-NUM-SUBTREES simply calculates $B[k,w]$ according to the Lemma 6, and its correctness follows from the correctness of Lemma 6. □

Lemma 8 *The routine RAND-SUBTREE-GIVEN-ROOT(p, R, v) correctly chooses a p-subtree rooted at v uniformly at random, where p is a positive integer, R is a rooted subtree and v is an internal node in R.*

Proof: Let O be a post-order on $V(R)$. The proof is by induction on the nodes of $V(R)$ ordered by O. For the base case, RAND-SUBTREE-GIVEN-ROOT correctly returns an empty subtree for all leaves. Now assume that for a node v, RAND-SUBTREE-GIVEN-ROOT(k, R, u) correctly returns a k-subtree rooted at u with probability $1/B[k,u]$ for all $k < p$ and for all proper descendants u of v. Let v_l and v_r be the left and right children of v respectively. Let H be the subtree returned by the call to RAND-SUBTREE-GIVEN-ROOT(p, R, v). Note that H is always a p-subtree. There are three cases to consider, based on the number of edges k of H to the left of v:

- $k = 0$ with probability $\frac{B[p-1,v_r]}{B[p,v]}$. The p-subtree H consists of the edge (v,v_r) and the $(p-1)$-subtree obtained through a call to RAND-SUBTREE-GIVEN-ROOT($p-1$, R, v_r). Thus, by the induction hypothesis, the probability with which H is chosen is $\frac{B[p-1,v_r]}{B[p,v]}\frac{1}{B[p-1,v_r]} = \frac{1}{B[p,v]}$.
- When $k = p$, a similar analysis shows that H is returned with probability $\frac{1}{B[p,v]}$.
- k equals i for some $0 < i < p$ with probability $\frac{B[i-1,v_l]B[p-i-1,v_r]}{B[p,v]}$. Here H consists of the edges (v,v_l), (v,v_r), and an $(i-1)$-subtree and a $(p-i-1)$-subtree chosen through calls to RAND-SUBTREE-GIVEN-ROOT. Thus, by the induction hypothesis, the probability that H is chosen is $\frac{1}{B[p,v]}$ in this case too.

□

Theorem 10 *The routine RAND-SUBTREE(p, R) correctly chooses a p-subtree of R uniformly at random, where p is a positive integer and R is a rooted subtree.*

Proof: The proof follows from Lemmas 7 and 8. □

Running Time of RAND-SUBTREE. The subroutine COMPUTE-NUM-SUBTREES takes $\Theta(np^2)$ time. The subroutine RAND-SUBTREE-GIVEN-ROOT takes $\Theta(p^2)$ time: if $T(p)$ is the time taken by the subroutine for choosing a p-subtree rooted at v, then $T(p) = T(i-1) + T(p-i-1) + \Theta(p)$ for some i satisfying $0 < i < p$, or $T(p) = T(p-1) + \Theta(p)$. The recurrence is satisfied by $T(p) \in O(p^2)$. Thus, the total time taken by RAND-SUBTREE is $\Theta(np^2)$.

The time taken by COMPUTE-NUM-SUBTREES for generating a p-subtree depends on n, which makes it very expensive in the context of a local move. We note that the dependency on n comes from the fact that B is a $(2n-3) \times (p+1)$ array: for two nodes u and v and for some positive integer k, $B[k,u]$ could be potentially different from $B[k,v]$: for example, if both children of u are leaves, $B[k,u] = 0$ for all $k > 0$, but if neither of v's children are leaves, then $B[1,v] = 2$. Let v_l and v_r be the children of v, and let u_l and u_r be the children of u. Clearly, if $B[i,u_l] = B[i,v_l]$ and $B[i,u_r] = B[i,v_r]$ for all $0 < l < p$, then $B[k,u] = B[k,v]$ for all $0 \leq k \leq p$. This condition for equality can be propagated down the tree through nodes u_l and v_l on the one hand and nodes v_l and v_r on the other hand. In general, if the shortest path leading from u to a leaf and the shortest path leading from v to a leaf both have at least k internal edges, then $B[k,u] = B[k,v]$. If this condition is met for all pairs of nodes u and v and for all $k \geq 0$, then the array B can be parameterized on just p, and consequently the time to compute it can be brought down to $\Theta(p^2)$. In reality the condition is not always met, since it is tantamount to assuming that the tree is infinite, or equivalently that there are no leaves. However, in practice we can still approximate B well by using this "infinite-tree" assumption: we approximate $B[k,v]$ by $C[k]$ where $C[k]$ is given by:

$$\begin{aligned} C[0] &= 1, \text{ and} \\ C[k] &= 2C[k-1] + \sum_{i=1}^{k-1} C[i-1]C[k-i-1] \text{ for } k \geq 0. \end{aligned}$$

In fact, for all nodes v and for all $k \geq 0$, $B[k,v] \leq C[k]$. With this approximation, COMPUTE-NUM-SUBTREES runs in $\Theta(p^2)$ time. Further, it has to be run only once at the start of a sequence of p-sECR moves, since C is independent of the tree structure. However, since the approximation of B by C is based on the "infinite-tree" assumption, it might happen that in a call to RAND-SUBTREES-GIVEN-ROOT(k, R, v), k might exceed the number of internal edges below v. We can check this by computing the number of internal edges under each node, and return no k-subtree if k exceeds this number. Computing the number of internal edges takes $O(n)$ time initially, but this information can be updated in $O(p)$ time after a p-sECR move.

With this modification, a call to RAND-SUBTREE for choosing a p-subtree runs in $O(p \log p)$ time after a $\Theta(p^2)$ preprocessing step for COMPUTE-NUM-SUBTREES: in the algorithm RAND-SUBTREE-GIVEN-ROOT, the probability distribution Q of the random variable k, the number edges to be chosen from the left subtree, can be pre-computed in $\Theta(p)$ time, and sampling k from the distribution Q can be performed in $\Theta(\log p)$ time. If $T(p)$ is the time taken now by RAND-SUBTREE-GIVEN-ROOT for choosing a p-subtree rooted at v, then $T(p) = T(i-1) + T(p-i-1) + \Theta(\log p)$ for some i satisfying $0 < i < p$, or $T(p) = T(p-1) + \Theta(\log p)$. For any $T(p)$ satisfying the recurrence, $T(p) \in O(p \log p)$.

Note that while we may now fail to return any p-subtree, if a p-subtree is returned, it is a uniform random subtree: each failed attempt can be construed as generating a virtual random p-subtree from the non-existent infinite tree, and each generated random p-subtree, whether virtual or real is now generated with the same probability $\frac{1}{C[p]N}$, where N is the number of internal nodes. Thus, each real p-subtree has the same probability of being chosen as any other real p-subtree.

In practice, we run the faster version of RAND-SUBTREE until it succeeds in returning a p-subtree up to a maximum of $\frac{np}{10 \log p}$ times. If all the $\frac{np}{10 \log p}$ attempts fail, we run the more expensive $\Theta(np^2)$ algorithm to generate the p-subtree. Thus, if one of the first $\frac{np}{10 \log p}$ attempts succeed, we are at least 10 times as fast as the more expensive RAND-SUBTREE algorithm.

Applying a p-sECR move Uniformly at Random on a Given p-Subtree. Our algorithm is

based on a correspondence between binary trees on f leaves and permutations of integers from 0 through $2f-3$, i.e., a permutation on $[0, 2f-3]$ [DH98]. For every permutation on $[0, 2f-3]$ there corresponds a binary tree on f leaves. The algorithm PERM-TREE, described below in pseudocode, generates a binary tree from a given permutation. The algorithm is illustrated in Figure 2.8.

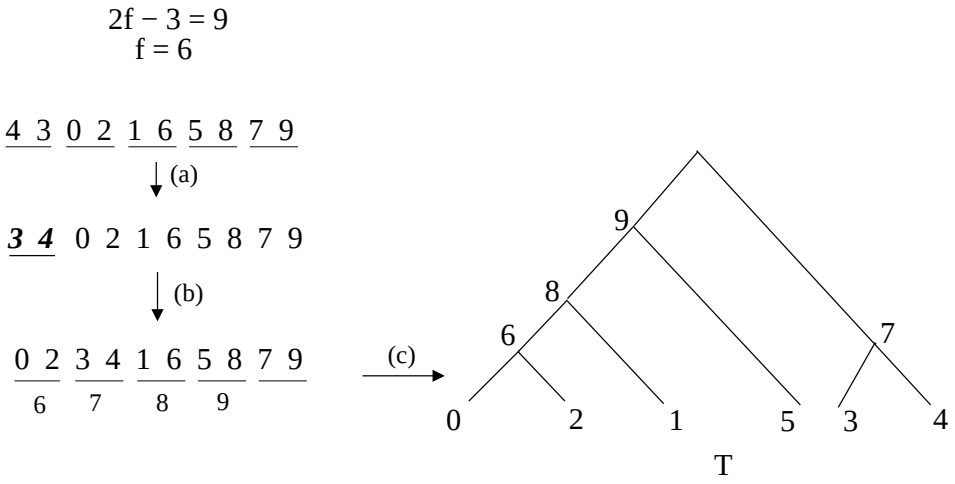


Figure 2.8: Transforming a permutation on $[0, 2f-3]$ to a rooted binary tree on f leaves: here, $f = 6$. A permutation always has an even number of elements, $2f-2$, and successive elements are grouped into pairs. A permutation is first transformed into a *canonical* permutation, in steps (a) and (b). Step (a) sorts each *pair* of elements in ascending order. In our example, all pairs except $(4\ 3)$ are already sorted to begin with. Step (b) retains the pairings, but sorts the pairs in ascending order according to the larger element in each pair, producing the canonical permutation. Step (c) produces a rooted binary tree from the canonical permutation. In the permutation, elements 0 through $f-1=5$ correspond to leaves, and elements $f=6$ through $2f-3$ correspond to ancestral nodes. The pairs in the permutation correspond to siblings in the tree. The parent of the first pair of siblings in the permutation is labeled $f=6$, the parent of the next pair of siblings is $f+1=7$, and so on. The parent of the last pair of siblings is the root of the binary tree. The label of the parent of each pair in the canonical permutation is shown under the pair. Note the correspondence between the canonical permutation and the tree: 6 is the parent of leaves 0 and 2, 7 is the parent of leaves 3 and 4, 8 is the parent of leaf 1 and the ancestral node 6, and 9 is the parent of leaf 5 and the ancestral node 8.

PERM-TREE($A[0 \ldots 2f-3]$) {

/* A holds a permutation of $2f-3$. */

1 **for** (i in $\{0,\ 2,\ 4, \ldots, 2f-4\}$)

2 **if** ($A[i] > A[i+1]$) swap $A[i]$ and $A[i+1]$

3 Re-order A to get A^* such that

4 for all $j \geq 0$, $A[2j] = A^*[2j^*]$ and $A[2j+1] = A^*[2j^*+1]$ for some $j^* \geq 0$

5 $A^*[1] < A^*[3] < A^*[5] < \ldots < A^*[2f-3]$

$T \leftarrow \emptyset$ /* T will hold the edges of the tree */

for ($i = 0$ to $2f-2$) create nodes with label i

$ilabel \leftarrow f$

for (i in $\{0,\ 2,\ 4, \ldots, 2f-4\}$)

$T \leftarrow T \cup \{(ilabel, A^*[i]), (ilabel, A^*[i+1])\}$

$ilabel \leftarrow ilabel + 1$

return T

}

Note that lines 1 through 5 of PERM-TREE apply a re-ordering to the input permutation to produce what we call a *canonical* permutation. In fact $2^{f-1}(f-1)!$ permutations correspond to the same canonical permutation. Hence, the number of canonical permutations equals $\frac{(2f-2)!}{2^{f-1}(f-1)!}$, which equals the number of rooted binary trees on f leaves. This number, usually denoted $(2f-3)!!$, is the the product of all odd numbers from 1 through $2f-3$. Further, it can be shown that there exists a one-one map between canonical permutations on $[0, 2f-3]$ and binary trees on f leaves [DH98].

Our strategy for applying a p-sECR move is as follows:

- First label the leaves of the selected p-subtree from 0 through $p+1$. Note that a p-subtree has $p+2$ leaves.
- Generate an integer z uniformly at random in the range $[0, (2(p+2)-3)!!) - 1]$.
- Generate a unique permutation π_z corresponding to z, using function GENERATE-PERM

described below. The function GENERATE-PERM will ensure that permutations for two different integers z_1 and z_2 do not have the same canonical permutation.

- We then apply PERM-TREE on π_z to obtain a new p-subtree which will replace the old p-subtree.

We now present function GENERATE-PERM:

GENERATE-PERM(z, f) {

/* z is an integer such that $0 \leq z \leq (2f-3)!! - 1$

for ($i = 0$ to $2f-3$) $A[i] \leftarrow i$

/* A holds the identity permutation */

$dividend \leftarrow z$

$k \leftarrow f$

$divisor \leftarrow (2k-3)!!$

$j \leftarrow 0$

for ($i = 0$ to $f-2$) $quotient[i] \leftarrow 0$

while ($dividend > 0$) **do**

$quotient[j] \leftarrow \lfloor \frac{dividend}{divisor} \rfloor$

$remainder \leftarrow dividend - quotient[j] * divisor$

$dividend \leftarrow remainder$

$k \leftarrow k-1$

$divisor \leftarrow (2k-3)!!$

$j \leftarrow j+1$

1 **for** ($i = 0$ to $f-2$)

2 swap $A[2i+1]$ and $A[2i+1+quotient[i]]$

3 **return** A

}

Lemma 9 *Let z_1 and z_2 be two integers in the range $[0,(2f-3)!!-1]$. Let A_1 = GENERATE-PERM(z_1, f) and A_2 = GENERATE-PERM(z_2, f). Then, there exists j in the interval $[0,f-2]$ such that $A_1[2j]=A_2[2j]$ but $A_1[2j+1]\neq A_2[2j+1]$. Further, the canonical permutations of A_1 and A_2 are different.*

Proof: Let Q_1 and Q_2 be the two *quotient* arrays generated by GENERATE-PERM corresponding to z_1 and z_2 respectively. Then $z_1=\sum_{j=0}^{f-2}Q_1[j]*(2f-5-2j)!!$ and $z_2=\sum_{j=0}^{f-2}Q_2[j]*(2f-5-2j)!!$. Hence, $z_1=z_2$ if and only if $Q_1[j]=Q_2[j]$ for all j such that $0\leq j\leq n-2$. Now, the lines marked 1, 2 and 3 permute array A that is initialized with the identity permutation based on the *quotient* array. Now, let Q_1 and Q_2 be two different quotient arrays. Let A_1 and A_2 be the permutations corresponding to Q_1 and Q_2. There exists a minimum index $j\geq 0$ such that $Q_1[j]\neq Q_2[j]$. Now consider the **for** loop of lines 1, 2 and 3 when the index i equals j. At this point, $A_1[k]=A_2[k]$ for all $0\leq k\leq 2n-3$. In particular $A_1[2j]=A_2[2j]$. Now $A_1[2j+1]$ is swapped with $A_1[2j+1+Q_1[j]]$ and $A_2[2j+1]$ is swapped with $A_2[2j+1+Q_2[j]]$. Hence in the final permutations $A_1[2j+1]\neq A_2[2j+1]$. This completes our proof. This also implies that the canonical permutations of the final A_1 and A_2 are different: in the canonical permutation of A_1, the successor of $A_1[2j]$, which equals $A_2[2j]$, is $A_1[2j+1]$. However, in the canonical permutation of A_2, the successor of $A_1[2j]$ is $A_2[2j+1]$. □

Thus, each permutation generated by GENERATE-PERM corresponds to a different binary tree.

Running Time of the Random *p*-sECR Neighbor Algorithms. Both PERM-TREE and GENERATE-PERM run in $\Theta(f)$ time. Hence, the time taken to apply a random p-sECR move on

a previously selected p-subtree is $\Theta(p)$ time. The total time taken to generate a random p-sECR neighbor of an unrooted binary tree on n leaves is, therefore $\Theta(p^2+p) = \Theta(p^2)$ if the faster version of COMPUTE-NUM-SUBTREES succeeds, and is $\Theta(np^2+p) = \Theta(np^2)$ otherwise.

We now present our random p-ECR neighbor algorithm.

A Random p-ECR Neighbor

Choosing p edges Uniformly at Random. Let T be the input tree on n leaves. There are $n-3$ internal edges in T. We repeatedly pick an internal edge uniformly at random with replacement from the set of all internal edges until we have p distinct internal edges. Let X be the number of steps before we have chosen p distinct internal edges. Then, the expected value of X, $E(X)$, equals $(n-3)\sum_{i=0}^{p-1} \frac{1}{n-3-i}$. Thus, $E(X) \leq \frac{(n-3)p}{n-3-p}$. If p is independent of n, then $E(X) \in O(p)$.

Applying a Uniform Random p-ECR Move. Let the p internal edges chosen as above form r connected subtrees. Let p_i be the number of edges in the i^{th} connected subtree. We apply a p-ECR move on the p-chosen edges by applying a p_i-sECR move uniformly at random on the i^{th} connected subtree, for $1 \leq i \leq r$. If m_i is the number of ECR neighbors of T that can be obtained by applying a p_i-sECR move on the i^{th} connected subtree, then the number of p-ECR neighbors T that can be obtained by contracting the p chosen internal edges equals $\Pi_{i=1}^{r} m_i$. Since each p_i-ECR move is independent of others, our procedure ensures that each p-ECR neighbor is chosen with probability $\frac{1}{\Pi_{i=1}^{r} m_i}$.

Running Time of the Random p-ECR Neighbor Algorithm. As noted in the the previous section, the time taken to apply a random p-sECR move on a previously selected p-subtree is $\Theta(p)$. If p is independent of n, the expected time taken to select a set of p internal edges uniformly at random is in $O(p)$. Thus, a random p-ECR neighbor of an unrooted binary tree on n leaves can be generated in expected $O(p)$ time.

We summarize the results in this section in the following theorem:

Theorem 11 *Given an unrooted binary tree on n leaves:*

- *A random p-sECR move can be generated in $O(np^2)$ time. It can be generated in $\Theta(p^2)$ time after a $\Theta(n+p^2)$ pre-processing step, if the faster version of the procedure COMPUTE-NUM-SUBTREES succeeds.*

- *A random p-ECR move can be generated in expected $\Theta(p)$ time.*

2.3 Structural Properties of the p-ECR Move

In this section, we define the notion of an *irreducible* or elementary p-ECR operation, and describe a method to construct, for any two given trees, a sequence of elementary or irreducible ECR operations that transforms one tree to another.

We first introduce some terminology and notation. Let T be an unrooted leaf-labeled tree. Let X and Y be two ECR operations on T. We will say X equals Y if performing X on T results in the same tree as the one obtained by performing Y on T. For two *ECR* operations X and Y, we will let $Y \circ X$ be the following sequence of two *ECR* operations: X on T, followed by Y on the tree that results from performing X on T.

Definition 20 *Reducible p-ECR operation*

Let T be an unrooted leaf-labeled tree. Let X be a p-ECR operation on T. X is said to be reducible if there exists a p_1-ECR operation X_1 and a p_2-ECR operation X_2 such that $X = X_2 \circ X_1$ and $p = p_1 + p_2$.

The concepts of reducibility and irreducibility of ECR operations are illustrated in Figure 2.9.

In this section we address the *Irreducible Decomposition* problem, which we define as follows: given two binary trees T and T' such that $RF(T,T') = 2p$, (*RF* denotes the Robinson-Foulds metric, see Section 1.5) decompose the p-ECR operation X that separates T and T' such that $X = X_k \circ X_{k-1} \circ \ldots \circ X_1$, each X_i is an irreducible p_i-ECR operation, and $\sum_{i=1}^{k} p_i = p$.

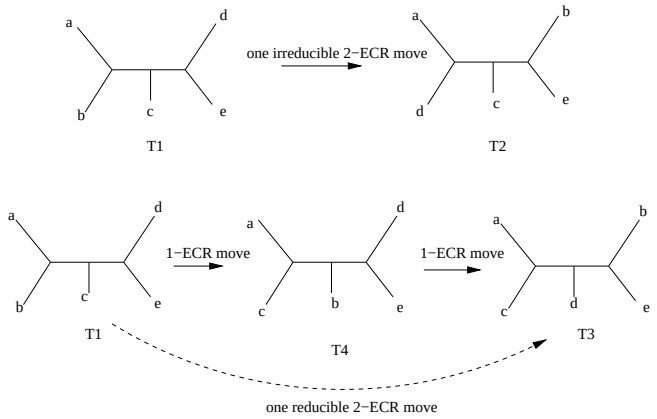


Figure 2.9: Trees T1 and T2 are one irreducible 2-ECR move away, and trees T1 and T3 are one reducible 2-ECR move away.

2.3.1 Irreducibility and Elementary Bipartite Graphs

We begin with a definition:

Definition 21 *Bipartition (or edge) compatibility:*

A set of bipartitions B is said to be compatible *if and only if* $B \subseteq C(T)$ *for some tree* T.

Lemma 10 (From Buneman [Bun71]) *A set of bipartitions is compatible iff any two bipartitions in the set are pairwise compatible. Furthermore, two bipartitions* $A = A_1 : A_2$ *and* $B = B_1 : B_2$ *are compatible iff at least one of the four sets* $A_1 \cap B_1$, $A_1 \cap B_2$, $A_2 \cap B_1$ *and* $A_2 \cap B_2$ *is empty.*

Observe that there cannot be more than $2n-3$ edges in a phylogenetic tree with n leaves, since there are no internal nodes of degree two (in general we use n to denote the number of leaves). This gives us the following:

Corollary 3 *The maximum cardinality of any set of compatible bipartitions of a set of n elements is* $2n-3$.

We now define a graph, which we call the *incompatibility* graph, defined by two leaf-labeled trees.

Definition 22 *Incompatibility Graph*

Let T and T' be two unrooted leaf-labeled trees. The incompatibility graph G between T and T' is defined as follows: G is a bipartite graph with vertex set $U \cup V$, where $U = C(T) - C(T')$, $V = C(T') - C(T)$ [*], and edge set $\{(u,v) : u \in U,\ v \in V,\ u \text{ and } v \text{ incompatible}\}$.

An elementary bipartite graph is one where every edge is in some perfect matching [LP86]. Suppose T and T' are two trees such that $\delta_{p-ECR}(T,T') = 1$; then, we show that the p-ECR move that separates them is irreducible if and only if the incompatibility graph induced by the two trees is elementary. We start with the following lemma.

Lemma 11 *Let G be the incompatibility graph between two unrooted binary leaf-labeled trees. Then G has a perfect matching.*

Proof: We show that the incompatibility graph G satisfies the following two properties: (1) For every subset S of V, $|\Gamma(S)| \geq |S|$, and (2) For every subset R of U, $|\Gamma(R)| \geq |R|$. Our result will then follow from Hall's matching theorem [Hal35]. Let $q = |C(T) - C(T')| = |C(T') - C(T)|$. We first show that (1) holds. Let S be a subset of V. Note then that the set of bipartitions $A = (U - \Gamma(S)) \cup S \cup (C(T) \cap C(T'))$ is compatible. Now if $|\Gamma(S)| < |S|$, then the set A contains more than $2n-3$ bipartitions that are pairwise compatible, since $|(U - \Gamma(S)) \cup S| > q$ and $|C(T) \cap C(T')| = 2n-3-q$. But this is a contradiction by Corollary 3. Similarly, we can show that (2) holds. □

Theorem 12 *Let X be a p-ECR move that transforms an unrooted leaf-labeled binary tree T to T'. Let $G = (U,V,E)$ be the incompatibility graph between T and T'. Then, X is irreducible if and only if G is elementary.*

Proof: We show that X is irreducible if and only if there is no proper subset S of V such that $|\Gamma(S)| = |S|$. This is equivalent to showing that G is elementary, since we have already established

[*]Note that the definition here is almost the same as the definition of the incompatibility graph appearing in [PW96], where U and V were $C(T)$ and $C(T')$ respectively. Our definition has the effect of removing isolated vertices from the incompatibility graph.

that G always contains a perfect matching. We show that X is reducible if and only if there is a proper subset S of V such that $|\Gamma(S)| = |S|$, which is equivalent to showing that G is not elementary, since we have established that G always contains a perfect matching.

Suppose that X is reducible and is equivalent to $X_2 \circ X_1$. Then the set of bipartitions S that results from the refinement phase of X_1 satisfies the condition that $|\Gamma(S)| = |S|$. If not there is at least one bipartition in S that is incompatible with a bipartition in $U - \Gamma(S)$. But this makes carrying out X_1 on T impossible, and hence $|S| = |\Gamma(S)|$.

Conversely, if there is a set $S \subset V$ that satisfies $|S| = |\Gamma(S)|$, then the contraction of the bipartitions in $\Gamma(S)$ and the creation of bipartitions in S can be "scheduled" before the other contractions and refinements in X, and that makes X reducible. □

Using the above characterization, we now show that we can check efficiently if a p-ECR move is irreducible. This also means that we can compute, for any given p-ECR move, its irreducible decomposition.

Theorem 13 *Let X be a p-ECR move that can be performed on an unrooted binary leaf-labeled tree on n leaves. Then, in $O(n+p^2)$ time, we can determine if X is reducible, and we can compute an irreducible decomposition of X.*

Proof: Let $G = (U, V, E)$ be the incompatibility graph corresponding to the p-ECR move X. The graph G can be constructed in $O(n+p^2)$ time as follows: The sets U and V can be computed in $O(n)$ time, while calculating the RF distance between T and T' [Day85]. Once U and V have been determined, E can be calculated in $O(n+p^2)$ time, since for each bipartition in U, we can identify all bipartitions in V incompatible with it in $O(p)$ time.

Once we have G, we use the method in [LP86], Section 4.3, to decompose G into maximal vertex-disjoint components such that the subgraph of G induced by each component is elementary, as follows: we compute a perfect matching M in G (which is guaranteed to exist by Lemma 11) and then compute an associated directed graph, say H. The graph H is computed from G by first

orienting all the edges uniformly towards either U or V, and then identifying the vertices matched by M.

The directed graph H is strongly connected if and only if G is elementary. This is due to the following reason: each edge of G not in M is in some perfect matching if and only if the corresponding directed edge in H is in a directed cycle. Hence, G is elementary if and only if every edge not in M is also in a perfect matching, by definition. Every edge in H is in a cycle if and only if H is strongly connected. This proves our claim.

If G is not elementary, then H can be decomposed into strongly connected components, say C_1 through C_k, with component C_i representing an elementary subgraph of G induced by the sets of vertices (S_i, T_i), with $S_i \subset U$ and $T_i \subset V$. Without loss of generality, let $C_1, C_2, \ldots, C_k$ be the topologically sorted order of the strongly connected components. Then, this represents an ordering of the corresponding elementary subgraphs of G, as follows: if $(u,v) \in E$ such that $u \in S_i$ and $v \in T_j$, then $i \leq j$ (assuming without loss of generality that all edges were oriented towards V while creating H from G). Hence, if we let the induced subgraph (S_i, T_i) stand for the i^{th} ECR operation X_i, it is assured that X_i can be performed once operations X_1 through X_{i-1} have been performed. This is because the incompatibilities of the bipartitions created by X_i (i.e., those in T_i) are with bipartitions in components S_j with $j < i$, and these bipartitions would have been eliminated by the ECR moves from X_1 through X_{i-1}. The outcome of the sequence of operations X_1 through X_k is X.

The decomposition of H into strongly connected components can be computed in $O(p^2)$ time. Hence the corresponding irreducible decomposition of X can be computed in $O(p^2)$ time, after an $O(n+p^2)$ time taken to compute the graphs G and H. □

Commutable p-ECR Moves. A p-ECR operation X is said to be separable if and only if there are two ECR moves X_1 and X_2 such that $X = X_2 \circ X_1 = X_1 \circ X_2$. The ECR moves X_1 and X_2 are then said to be commutable. The following is a necessary and sufficient condition for separability of a p-ECR operation:

Lemma 12 *Let X be an ECR move executable on an unrooted leaf-labeled binary tree. The incompatibility graph induced by X is not connected if and only if X is separable. In particular, the incompatibility graph has k components, for $k > 1$ if there exist k commutable ECR moves X_1 through X_k such that $X = X_1 \circ X_2 \circ \ldots \circ X_k$.*

Proof: Let X transform an unrooted binary tree T to an unrooted binary tree T'. Let $G = (U, V, E)$ be the incompatibility graph of X. Let A equal $C(T) \cap C(T')$.

We first show that if the G is not connected, then X is separable. Let $G = (U, V, E)$ be the incompatibility graph of X. Let $G_i = (U_i, V_i, E_i)$ be the i^{th} connected component of G. Let p_i be $|U_i|$. First, note that $|U_i| = |V_i|$ for all i. If not, assume without loss of generality that $|U_i| < |V_i|$. Then, since the graphs G_i are not connected, $D = A \cup (U - U_i) \cup V_i$ forms a set of compatible edges. However, $|D| > 2n - 3$ which contradicts Corollary 3. Thus, $p_i = |U_i| = |V_i|$ for all i. Now, $A \cup (U - U_1) \cup V_1)$ is a compatible set of bipartitions, since there are no edges between V_1 and $U - U_1$. This holds in general for $W_i = A \cup (U - \cup_{i=1}^{j} U_i) \cup (\cup_{i=1}^{j} V_i)$, for $0 \leq j \leq k$. Let T_i be the unrooted binary tree whose set of non-trivial bipartitions is W_i. Note that $T_0 = T$ and $T_k = T'$. Then, there exists a p_i-ECR move X_i that transforms T_{i-1} into T_i for $1 \leq i \leq k$. Note that the incompatibility graph of X_i is G_i. Thus $X = X_1 \circ X_2 \circ \ldots \circ X_k$. Our ordering of the components G_i is arbitrary, and hence the ECR moves X_i are mutually commutable. Thus, X is a separable ECR move.

We now show that if X is separable, G is not connected. $X = X_1 \circ X_2 \circ \ldots \circ X_k$, where the X_i are mutually commutable ECR moves. Let T_j be the tree obtained by applying $X_1 \circ \ldots \circ X_j$ on tree T. Let G_i be the incompatibility graph of X_i. Note that the graphs G_i are mutually disjoint. If not, assume without loss of generality (due to the commutability of X_i) that G_j and G_{j+1} are not disjoint, for some $j < k$. Assume also that there exists an edge between V_j and U_{j+1}. However, this violates the assumption that there exists an intermediate tree T_j. For the application of X_j on T_{j-1} is impossible since it would lead to a conflict with the edges in U_{j+1}. Thus, the assumption that G_i are not mutually disjoint leads to a contradiction. □

However, there is a necessary and sufficient condition for separability of X that can be verified without computing the incompatibility graph, as described by the following theorem:

Theorem 14 *Let T be an unrooted leaf-labeled tree. Let X be a p-ECR operation on T that would result in a tree T'. Then, there exist k mutually commutable ECR operations X_1, X_2, $\ldots X_k$ such that $X = X_k \circ X_{k-1} \circ \ldots \circ X_1$ if and only if the edges corresponding to bipartitions in $C(T) - C(T')$ constitute a forest with k connected trees.*

Proof: We show that X is separable if and only if the edges corresponding to the bipartitions in $C(T) - C(T')$ do not form a connected subtree. The desired result would then follow.

We first prove that X is separable if the edges represented by $C(T) - C(T')$ do not form a connected subtree. Let the edges represented by $C(T) - C(T')$ form two disjoint subtrees S_1 and S_2 in T. For clarity of presentation, we assume that S_1 and S_2 are connected subtrees of T. This assumption, however, is not crucial to our argument. Let f be an edge in T such that S_1 and S_2 lie on opposite sides of f. Contracting all the edges of S_1 would result in an unresolved node v_1. Let L_1 be the set of all leaves that can be reached from v_1 without going through f. Similarly, let v_2 be the unresolved node that would be created by contracting all the edges of S_2, and let L_2 be the set of all leaves that can be reached from v_2 without going through f. Note that $L_1 \cap L_2 = \emptyset$. Now, let e_1 be any edge created by refining v_1, and let e_2 be any edge created by refining v_2. Let $A_1|A_2$ be the bipartition induced by e_1 on the set of all leaves of T, and let $B_1|B_2$ be the bipartition induced by e_2. At least one of A_1 and A_2, say A_1, is a subset of L_1. Similarly, at least one of B_1 and B_2, say B_1 is a subset of L_2. Hence, $A_1 \cap B_1 = \emptyset$. This proves that e_1 and e_2 are compatible edges. Since the choice of e_1 and e_2 was arbitrary, this shows that in executing X we could first contract all edges of S_1 and fully refine the resulting unresolved node, and then contract all the edges of S_2 and fully refine the resulting unresolved node. This, in turn, shows that X is separable.

We now prove the other direction of our theorem. Let $U = C(T) - C(T')$ and $V = C(T') - C(T)$. We prove the following: let u_1, u_2 be two bipartitions in U, corresponding to two edges e_1

and e_2 adjacent in T. Let v_1 and v_2 be two bipartitions in V such that, $(u_1, v_1) \in E$, $(u_2, v_2) \in E$, $(u_1, v_2) \notin E$, and $(u_2, v_1) \notin E$. We show that u_1, u_2, v_1 and v_2 can be reached from one another in G. This would imply that if the edges in U form a single subtree in T, then G is connected.

We now prove the above claim. Let u_1 be the bipartition $P : P'$ and let u_2 be $P \cup Y : P' - Y$, for some $Y \subset P'$ (this entails no loss of generality since u_1 and u_2 are compatible). Similarly, let v_1 and v_2 be $Q : Q'$ and $Q \cup Z : Q' - Z$ respectively, for some $Z \subset Q'$. Since u_1 and v_2 are incompatible, we have $P \cap (Q' - Z) = \emptyset$ (it can be verified that the other three pairwise intersection cannot be empty) from Lemma 10. Similarly, we have $Q \cap (P' - Y) = \emptyset$.

Now, since the tree T is binary, and the edges e_1 and e_2 are adjacent, we have that $Y : Y'$ (where Y' is the complement of Y) is a bipartition in T, and the corresponding edge is adjacent to both e_1 and e_2. We show that $Y : Y'$ is incompatible with both v_1 and v_2, thus showing that u_1, u_2, v_1, v_2 are reachable from each other.

Now, $Q \cap (P' - Y) = \emptyset$ and $Q \cap P' \neq \emptyset$ implies that $Q \cap P' \subseteq Y$. Now, we show that $Y \nsubseteq Q \cap P'$. Suppose, to the contrary, that $Y \subseteq Q \cap P'$. Then, $Y \subseteq Q$. This, combined with the fact that $(Q' - Z) \subseteq P$, means that $(Q' - Z) \subseteq (P' - Y)$. However, this contradicts $(Q' - Z) \cap (P \cup Y) \neq \emptyset$, and hence $Y \nsubseteq Q \cap P'$. Thus, we have $(Q \cap P') \subset Y$. We now show that $Y : Y'$ is incompatible with both v_1 and v_2, thus completing our proof.

1. $Y \cap Q \neq \emptyset$, since as we already saw, $Q \cap P' \subset Y$. Also, $Y \nsubseteq Q$. Hence, $Y \cap Q' \neq \emptyset$. Moreover, $Q \nsubseteq Y$ (since $Q \nsubseteq P'$), and hence $Q \cap Y' \neq \emptyset$. Similarly, $Q' \cap Y' \neq \emptyset$. Thus, we have that $Y : Y'$ is incompatible with v_1.

2. $Y \cap (Q \cup Z) \neq \emptyset$, since $Y \cap Q \neq \emptyset$. Now, since $(Q' - Z) \cap P = \emptyset$ and $(Q' - Z) \cap (P \cup Y) \neq \emptyset$, we have $Y \cap (Q' - Z) \neq \emptyset$. This means that $Y' \cap (Q \cup Z) \neq \emptyset$ and $Y' \cap (Q' - Z) \neq \emptyset$ as well. Thus, we have that $Y : Y'$ is incompatible with v_2.

This completes our proof. □

Existence of Irreducible p-ECR Moves We establish that an irreducible p-ECR move can be constructed for every set of p connected edges in any tree through explicit construction:

Theorem 15 *Let T be any unrooted binary tree with p internal edges, for some $p \geq 1$. Then there is a tree T' with $RF(T,T') = 2p$ such that the incompatibility graph between T and T' is elementary.*

Proof: The proof relies on constructing two trees T and T' such that the incompatibility graph between them contains a Hamiltonian cycle, and hence is elementary.

Consider any cyclic tour through all the leaves of of T that visits each internal edge exactly twice. There exists at least one such tour: root the tree at any leaf and perform an in-order traversal of the rooted tree. Let the leaves of T be x_1 through x_n, numbered based on their order of appearance in some such cyclic traversal that visits each internal edge exactly twice. Note that $n \geq 4$, since $p \geq 4$. Let $e_1,\ e_2, \ldots,\ e_r$ be the internal edges of T, named in the order in which they are visited for the first time in the above cyclic traversal of the leaves ($r = n-3$, but that is not of consequence in our proof).

For convenience, we will use the following notation: $[x_i, x_j]$ will denote the set of consecutively labeled leaves from x_i through x_j, if $i \leq j$. If $i > j$, then $[x_i, x_j] = [x_i, x_n] \cup [x_1, x_j]$.

Our circular numbering of the leaves leads to the following observation: if $\pi_j = J_1|J_2$ is the bipartition induced by the internal edge e_j, then there exist integers k, l such that $1 \leq k < l < n$ and $J_1 = [x_k, x_l]$ and $J_2 = [x_{l+1}, x_{k-1}]$ or $[x_{l+1}, x_n]$ (the latter case will hold when $k = 1$).

We construct a tree T' from T by permuting T's leaf labels as follows: each leaf x_i is relabeled x_{i-1}, and the leaf x_1 is relabeled x_n. Thus, T' is on the same set of leaves as T.

We claim that T and T' are separated by one irreducible p-ECR move. Our strategy is as follows: we first show that $RF(T,T') = p$, and exhibit two perfect matchings in the incompatibility graph between T and T'. The existence of two such perfect matchings is sufficient to show that the incompatibility graph has a Hamiltonian cycle and is thus elementary.

Let $G = (U, V, E)$ be the incompatibility graph where $U = C(T) - C(T')$ and $V = C(T) - C(T')$. Then,

- We have that $RF(T, T') = p$, and that G contains a perfect matching. Consider an internal edge e_j in T. Let $\pi_j = J_1|J_2$ be the bipartition induced by e_j. Then, as observed above, $J_1 = [x_k, x_l]$ and $J_2 = [x_{l+1}, x_{k-1}]$ for some $1 \leq k < l < n$. For clarity of presentation, we assume that $k > 2$. Because we just permuted the leaves of T to construct T', there exists an internal edge e'_j in T' that corresponds to e_j. The correspondence is as follows: e'_j induces a bipartition $\pi'_j = J'_1|J'_2$, such that $J'_1 = [x_{k-1}, x_{l-1}]$ and $J'_2 = [x_l, x_{k-2}]$. Then, π_j is incompatible with π'_j. To see this, observe that $x_k \in J_1 \cap J'_1$, $x_{k-1} \in J_2 \cap J'_1$, $x_l \in J_1 \cap J'_2$, and $x_{l+1} \in J_2 \cap J'_2$. Thus, π_j cannot be induced by any internal edge of T'. Since e_j was arbitrarily chosen, T' has no internal edges that induce any bipartition induced by an internal edge of T. Similarly, no internal edge of T induces any bipartition induced by an internal edge of T'. Thus, $RF(T, T') = p$, and we have exhibited a perfect matching in G.

- Now consider the above edge e_j in T, and the edge e'_{j+1} in T'. Let $B_1|B_2$ be the bipartition induced by e'_{j+1}. Then, due to our circular numbering of the leaves, $B_1 = [x_{k-1}, x_{l-1}] \cup [x_m, x_{k-2}]$ and $B_2 = [x_l, x_{m-1}]$ for some m such that $m \geq l + 2$. Now, e_j is incompatible with e'_{j+1}, for the following reason: $x_k \in J_1 \cap B_1$, $x_{l+1} \in J_2 \cap B_2$, $x_l \in J_1 \cap B_2$, and $x_{k-1} \in J_2 \cap B_1$. The above argument holds when $j < r$, but the argument for $j = r$, to show that e_r is incompatible with e'_1, is the same except for differences in leaf-label indices. Thus, we have demonstrated another perfect matching in G.

The following is a Hamiltonian cycle in G: $e'_1, e_1, e'_2, e_2, \ldots, e_{r-1}, e'_r, e_r, e'_1$. The presence of a Hamiltonian cycle in G implies that G is elementary, since the cycle by itself is a minimally elementary bipartite subgraph of G. This completes our proof. □

Chapter 3

Factors Affecting Maximum Likelihood Heuristic Searches

3.1 Introduction

Maximum likelihood is an NP-hard problem [CT05b], and is also one of the most important methods of phylogenetic reconstruction. Maximum likelihood is increasingly being preferred over even maximum parsimony as a phylogenetic reconstruction method. While both MP and ML are NP-hard problems as was observed in Chapter 1, it is "easier" to obtain a good MP solution in a reasonable amount of time. This is because unlike MP, ML is not a purely combinatorial optimization problem, and involves the optimization of many real-valued parameters of the model of evolution and the real-valued branch lengths of a tree. The MP score of a tree can be computed in time linear in the number of taxa and the length of the sequences using Fitch's algorithm, described in Chapter 1. But even computing the ML score of a tree can take many hours, as we will see later in this chapter. Even then, there is no guarantee that the obtained score is really the maximum.

Software for obtaining good MP solutions for datasets with thousands of taxa in the matter of days have been available for a few years now [RMWW04, Gol99]. But good software for ML

heuristic optimization have not been available until very recently. In fact, Moret et al. find in [MW03] that *MrBayes*, a software intended not for ML optimization but for Bayesian analysis, was the best ML heuristic. Similarly, the study in [BHD+02] finds that a parsimony heuristic, the *ratchet* ([Nix99]) is a good ML heuristic, better than even the ML heuristics studied in the same paper.

Recently, however, three new ML heuristic software have been developed: PHYML [GG03], RAxML [SOL05] and GARLI [Zwi06]. In this chapter, we experimentally evaluate the relative performances of these ML-specific software, along with that of the ML heuristic implemented in the "traditional" software PAUP [Swo96]. We also study the Bayesian analysis software, MrBayes, and the ratchet heuristic treated as ML heuristics.

The Objectives of our Experiments. Our experiments are designed to answer the questions listed below. In our experiments we allow each software a maximum of 24 hours of optimization.

- Which of the various ML heuristics obtain trees with the highest (best) maximum likelihood scores after 24 hours of optimization?
- Which of the various ML heuristics, PAUP, PHYML, RAxML, MrBayes and GARLI, recovers the most topological accurate trees after 24 hours of optimization? Also, how do the three new software compare against PAUP?
- All local-search heuristics require a *starting tree*, the starting point of a heuristic's search of the tree space. Do starting trees have an impact on the final outcome of the search, or are ML heuristics indifferent to the starting point?
- As observed earlier calculating the ML score of a fixed tree is a hard problem in itself. Which ML software calculates the ML score of a fixed tree topology the best?

Our rationale for allowing a maximum of only 24 hours of optimization is as follows: PHYML, RAxML and GARLI complete their heuristic optimization in less than 24 hours on all

our datasets, real and simulated. The actual running time of PHYML and GARLI can be seen in Table 3.9, and all PHYML optimizations reach completion in times comparable to those of RAxML and GARLI. Further, we ran PAUP for 60 days on two of our three real datasets to see if a longer analysis affects our conclusions about the relative performance of methods significantly. A similar optimization on the third real dataset could not be completed in time for this dissertation. We find that while PAUP finds a tree with a numerically higher ML score if run for 60 days, the improvements in ML scores are not statistically significant. These results are presented and discussed in greater detail in Section 3.3. Moreover, MrBayes was included in our study since an earlier study from 2003 had found it to be the best ML heuristic [MW03], and our results from 24-hour optimizations are sufficient to show that that conclusion no longer holds.

Our results show that the new ML heuristics PHYML, RAxML and GARLI vastly outperform the PAUP ML heuristic, and the parsimony ratchet used as an ML heuristic. The Bayesian analysis software, MrBayes, often outperforms PHYML, but not RAxML or GARLI on real datasets. Our results also show that PAUP is the best heuristic for estimating the optimal model parameters and branch lengths for, and computing the ML score of, a given tree topology.

While our experiments were designed primarily with the previously listed objectives in mind, two other interesting observations emerged from our experiments:

- Our results suggest that the difference in performance between PAUP and the newer ML heuristics stems mainly from the ability of the new methods to quickly obtain crude approximations of the ML scores of the intermediate trees. We discuss this in Section 3.5.

- While the new methods obtain final trees with higher ML scores, for the most part the differences in ML scores are not statistically significant. Moreover, there was up to 25% topological difference between tree topologies that were statistically indistinguishable. This raises questions about the significance of the improvements achieved by the new ML heuristics, and the use of ML as a phylogenetic estimation criterion. We briefly discuss this issue in

Section 3.3, and leave a thorough investigation for future research.

The rest of this chapter is organized as follows: Section 3.2 describes the experimental methodology and the datasets used. Section 3.3 presents the results of our experiments on real datasets, and Section 3.4 presents the results of our experiments on simulated datasets. In Section 3.5 we discuss mechanisms used by various heuristics for computing optimal branch lengths of intermediate trees, and provide some evidence that more time spent optimizing the branch lengths of intermediate trees during a search may be counterproductive from the point of view of obtaining final trees with high ML scores. Section 3.6 concludes this chapter with a summary of our results and observations.

3.2 Experimental Methodology

In this section, we describe the datasets used in our experiments, and our experimental methodology. Our experiments have the following general outline. Each stage in the outline will be elaborated upon later in this section.

Outline of our Experiments.

- **Starting Trees.** For each dataset we obtain up to three different starting trees.
- **Scoring the Starting Trees.** Each starting tree is then scored, i.e., has its ML score calculated, with PAUP. This results in the "optimal" branch lengths for the tree topology and the "optimal" model parameters being calculated.
- **Topology and Branch Length Optimization.** Each software is then provided each of the starting trees, and is allowed a maximum of 24 hours for optimizing the tree topology and branch lengths, while keeping as many model parameters as possible fixed at the values computed previously by PAUP. The reason for this is as follows: while ML is a probabilistic model-based optimization method, the true interest lies only in the tree topologies that any

ML method produces. Hence, our intention is to evaluate the heuristics with respect to their ability to effectively search the tree space.

- **Evaluation of the Final Trees.** The final trees for the real datasets are evaluated on the basis of their ML scores. However, we do not just numerically compare their ML scores, instead, we assess the statistical significance of the differences in ML scores by applying the Shimodaira-Hasegawa test [SH99]. The final trees resulting from the simulated datasets are evaluated on the basis of their differences in ML scores as well as their topological accuracy. For the simulated datasets, the topological accuracy of the final trees can be computed since the true tree, the model tree, is known.

We perform two sets of experiments: first, we perform all the experiments described above with real biological datasets. Our second set of experiments attempt to validate the major conclusions of our results from the first experiments using simulated datasets.

3.2.1 Datasets

We now describe the datasets used in our experiments. We used both real biological datasets, and simulated datasets derived from the biological datasets. We first list our real biological datasets, and then describe how we generate our simulated datasets.

Real Datasets. We used the following real datasets:

- A set of 228 aligned DNA sequences of a group of Angiosperm plants, each of length 4811. This dataset was first analyzed in [SSN+97], and later in [BHD+02]. We call this the *Angio* dataset.

- A set of 162 aligned nuclear ribosomal DNA sequences from the genus Helianthus (Sunflower), each of length 1296 [TSL06]. We call this the *Helia* dataset.

- A set of 57 aligned nuclear ribosomal DNA sequences from the genus Helianthus, each of length 238. We call this the *ETS* dataset, since the sequence is from the *external transcribed spacer* region of nuclear ribosomal DNA. This dataset and the Helia dataset were both sequenced and aligned by Ruth Timme [TSL06].

- A set of 96 aligned chloroplast DNA sequences from the Linaceae (flax) family of plants, each of length 429. We call this the *Linum* dataset.

- A set of 25 aligned chloroplast DNA sequences from the Linaceae family of plants, each of length 481. We call this the *Lina* dataset. This dataset and the above Linum dataset were sequenced and aligned by Joshua McDill [MRKS06].

- A set of 682 aligned small subunit rRNA sequences, consisting of 678 species of *Nematodes* (species of microscopic multicellular worms) and 4 outgroups, each of length 1808. This dataset has been assembled as part of the *Nematode Tree of Life* project [NEM]. This dataset will be called the *Nematodes* dataset.

- A set of 500 aligned *rbcL* DNA sequences each of length 1398, also known as the *rbcL500* dataset. This dataset was first published in [CSO+93]. This dataset will be called the rbcL dataset.

Simulated Datasets. We generated simulated datasets corresponding to the three largest real datasets, Angio, Nematodes and rbcL, in the following manner: first, we performed extensive ML optimizations on the three real datasets with our various ML heuristics. We thus obtained trees with high ML scores, along with the "optimal" model parameters and branch lengths, for each of three real datasets. We then used the trees which then had the highest ML scores as model trees for our simulations. Corresponding to each real dataset, we simulated 10 datasets according to the GTR+G+I model using the program *Seq-Gen* [RG97] on the model tree. A more detailed description of how our model trees were obtained follows:

- For the Angio dataset, we used a tree that was provided to us by Derrick Zwickl, which at the time had the highest known ML score for the Angio dataset. The branch lengths and model parameters were also supplied along with the tree. This tree was obtained by running GARLI from a random starting tree.

- For the Nematodes dataset, we used the tree obtained by running RAxML on a ratchet starting tree. The parameters and branch lengths on this tree were estimated by PAUP. Later in this section, we explain how we obtain our starting trees, including the ratchet trees.

- For the rbcL dataset, we used the final tree from a RAxML analysis starting from a ratchet tree. The model parameters and branch lengths were estimated on the tree by PAUP.

All the model trees had the highest-known ML score for their datasets at the time the simulations were performed. We subsequently obtained trees with higher ML scores than the model trees, but the crucial point is that the model trees are still very good ML trees, and are thus likely to be realistic.

Software. We used the following versions of each software. Each version was the latest version at the times the experiments were conducted.

- PAUP version 4, beta 10.

- RAxML version V.

- PHYML version 2.4.3

- MrBayes version 3.0, beta 4.

- GARLI version 0.93

Starting Trees. We used two starting trees for each dataset: a neighbor-joining tree, and an MP-ratchet tree. We now describe in detail how we obtain each of these starting trees.

- **The Neighbor Joining (NJ) tree.** The NJ starting tree for each dataset was obtained using PAUP, with the following command: dset distance=gtr rates=gamma basefreq=empirical; nj brlens=no breakties=random tieseed= int(10000*rand()); The command means that the distance matrix used was a *GTR* distance model, the substitution rates were assumed to be from a gamma distribution, and the empirical estimates are to be used for initial base frequency estimates.

- **The MP Ratchet tree.** The MP ratchet tree (henceforth simply called the ratchet tree) for each dataset was the obtained by running the "standard" ratchet heuristic (see [Nix99]), implemented in PAUP by Tiffani Williams, for approximately one hour (hence, the number of iterations run was different for each dataset). The standard ratchet heuristic, briefly, is as follows: In the zero-th iteration an initial tree is obtained by performing a TBR search with the input dataset until one local optimum tree is reached. Each subsequent iteration is structured thus: starting from the tree saved from the previous iteration, a hill-climbing search is performed on a modified dataset where a random 25% of the sites are assigned double the weight of the rest of the sites, and from the resulting local optimum tree a hill-climbing search is performed with the original dataset. The resulting tree is the saved tree for the current iteration. The above procedure is carried out for a specified number of iterations.

Apart from the NJ and ratchet starting trees, we also explored the use of random starting trees. However, as the discussion in Section 3.2.3 shows, the performances with random starting trees are never better, and sometimes significantly worse, than the performances with the other two starting trees. Hence, we use only NJ and ratchet starting trees in our experiments.

Scoring the Starting Trees. The starting trees were scored with PAUP under the GTR+G+I model. The choice of PAUP to score the starting trees (and later, final trees) will be explained in Section 3.2.2. The model, GTR+G+I, was chosen taking into consideration the size of the datasets. This is the most general time-reversible model, and is the preferred model among systematists for datasets

with hundreds of taxa [Zwi06]. The following parameters are estimated when a tree is scored under the GTR+G+I model.

- the branch lengths,
- the six site-substitution rate parameters (one for each pair of bases),
- the equilibrium frequencies of the bases,
- the proportion of invariant sites, and
- the shape parameter of the gamma distribution.

The number of rate categories in the discrete approximation of the gamma distribution was set to the default value of 4 while scoring the trees.

Tree Topology and Branch Length Optimization. During this phase, each dataset is subjected to heuristic optimization with the various heuristics under consideration, with each starting tree. For the real datasets, all the six heuristics, PAUP, PHYML, RAxML, GARLI, MrBayes and Ratchet, were run. We found that the ratchet heuristic's performance was poor (see Section 3.3 for a discussion), and hence eliminated it from consideration for the simulations.

During the search, as far as possible, the model parameters were kept fixed at the values. However, not all the software allow fixing all the model parameters: RAxML does not support fixing any parameter, and PHYML does not support fixing the base frequencies.

Though it appears that the experimental settings favor RAxML by allowing it it to optimize all parameters while keeping model parameters fixed in other software, such differences in the settings do not matter:

- For PAUP, our way of running it, estimating the model parameters first on a fixed starting tree and keeping the estimated model parameters fixed while optimizing tree topology and branch lengths, is the best way of running it [SAJS05]. In general, the authors of [SAJS05]

advocate a successive approximations approach where the tree resulting from one iteration of tree-topology optimization is scored and then used as the initial tree for the next iteration. However, in 24 hours on large datasets it is not possible to complete even one iteration of hill-climbing optimization in PAUP. Hence for large datasets with a 24-hour time limit, the successive approximation scheme boils down to our approach.

- Allowing PHYML to estimate model parameters simultaneously while estimating tree topology and branch lengths does not seem to improve the ML score of the best final tree, as Table 3.6 that appears in Section 3.3 shows.

To recapitulate, the following are the differences in the way model parameters were handled during topology and branch-length optimization in each software.

- **PAUP.** All model parameters were fixed.
- **MrBayes.** All model parameters were fixed.
- **PHYML.** All model parameters except the equilibrium base frequencies were fixed.
- **RAxML.** The standard distribution does not allow fixing any model parameters. Hence, RAxML was run in a default manner.
- **GARLI.** All model parameters were fixed.

Each software was allowed a maximum of 24 hours for optimization. It has been suggested that GARLI be run five times for each initial settings [Zwi06], since it is a randomized algorithm. Hence, the reported data from GARLI analyses are averaged over five runs. All the other software were run once for each dataset with each starting tree.

Two versions of PAUP were run: one with the number of rate categories in the discrete approximation of the gamma distribution set to a default 4, and one with the number of rate categories set to 10. This was done for the following reason: with 10 rate categories, PAUP obtains a

more accurate estimate of the ML score of a tree. PAUP exhaustively calculates the ML scores of intermediate trees, and hence, with 10 rate categories, PAUP would spend more time scoring each intermediate tree. Running PAUP twice, with the number of rate categories set to 10 and 4, would enable us to explore the trade off between accuracy of the ML scores of the intermediate trees and the time spent obtaining these scores.

The ratchet heuristic was run as follows: we first calculated, for each dataset, the number of ratchet iterations that correspond to 24 hours. This was done by extrapolating the time taken by ratchet for a small number of iterations. Then, for each dataset, the ratchet was run for the calculated number of iterations, and the tree output at the end of the last iteration was taken to be the ratchet final tree.

The Final Trees. The heuristics GARLI, RAxML, PHYML and the ratchet report back one "best" final tree. The software PAUP does not complete its hill-climbing search in 24 hours, but was run so that it writes out the best current tree at the end of the 24-hour period. MrBayes outputs not one final tree, but the posterior probability distribution on a set of trees. A tree with the highest posterior probability was taken to be the MrBayes final tree. The maximum likelihood scores of all final trees were computed with PAUP, with the number of rate categories set to 10. This was necessitated because we ran PAUP with both 4 and 10 rate categories during the topology and branch-length optimization phase. All final trees are still scored in an uniform manner, and hence the ML scores of the final trees remain comparable. We do not measure the topological accuracy of the final trees obtained for the real datasets, as the true tree for these datasets cannot be known. However, for each dataset, we do assess the statistical significance of the differences in ML scores among the pool of trees consisting of the initial trees and the final trees. We use the Shimodaira-Hasegawa statistical test, implemented in PAUP [SH99]. We present the results of the statistical test, along with an explanation of the test, in Section 3.3. For the simulated datasets, the model tree is the true tree, and we compute the topological accuracy of the final trees by computing the RF distance to the model trees.

3.2.2 Calculating the ML Score of a Fixed Tree Topology

There are two measures of the quality of the result of an ML heuristic search: the maximum likelihood score of the final tree, and the topological accuracy of the final tree. However, as observed earlier, calculating the maximum likelihood score of a tree involves not a clear-cut combinatorial optimization, but an iterative numerical optimization. Moreover, when the GTR+G+I model is assumed, discrete approximations of the continuous gamma distribution have to be employed. The likelihood score of a tree computed by different software even for a fixed topology, branch lengths and model parameters can be different, depending on how the gamma distribution is approximated. For example, for a fixed tree topology, branch lengths and model parameters for the Angio dataset, PAUP computed a likelihood score of $e^{-121209.68226}$, whereas PHYML computed a likelihood score of $e^{-120925.58250}$. The tree topology, branch lengths and model parameters were obtained by performing a PHYML heuristic ML optimization with an NJ starting tree. The above instance makes it clear that if the the results from different software are to be comparable, it is necessary to score all the final trees with the same software.

To this end, we first compared PAUP and PHYML on their ability to optimize the ML score of a given tree for five of our smallest real biological datasets: Angio, Lina, Linum, Helia and ETS. We did not include the other three software in this comparison for the following reasons: MrBayes is not an ML heuristic, but a Bayesian analysis software, and hence it cannot calculate the ML score of a given tree. The GARLI manual itself recommends that the final trees be scored using PAUP [Zwi06]. The version of RAxML used in our study was RAxML-V, which was the latest version at the time of the study [SOL05]. RAxML-V computes the ML scores of trees, and indeed performs the entire heuristic search, under a different model of variation of rates of evolution across different sites [Sta06]. The author of RAxML-V, Alexandros Stamatakis, strongly recommends that trees not be compared based on ML scores computed by RAxML-V [Sta06]. Thus, we compared only PHYML and PAUP.

For each dataset, we first computed an MP ratchet tree [Nix99]. The ratchet heuristic was

run for 200 iterations. The resulting tree was then scored using PHYML and PAUP, under the model of evolution suggested by the program ModelTest [PC98]. ModelTest suggested GTR+G+I model for the Angio dataset, and GTR+G for all the other datasets except the ETS dataset, for which the suggested model was HKY85+G. See Chapter 1 for a brief discussion of probabilistic models of evolution. Table 3.1 shows the results of our comparison. Note that the likelihood scores are typically very small fractions, and hence it is conventional to report the natural logarithm of the likelihood score, denoted *lnL*, or its negative -*lnL*.

	PAUP -lnL	PHYML -lnL
Angio	121176.45774	121450.384
ETS	916.38580	916.392035
Helia	12604.79916	12608.83973
Lina	1553.67286	1553.83973
Linum	2154.94922	2155.25683

Table 3.1: **Calculation of the -lnL score of a fixed tree: PAUP vs. PHYML**

As can be seen from the table, PAUP obtains a lower -*lnL* score, and hence a higher ML score in every case. Hence, we conclude that PAUP optimization is the best way to calculate the ML score of a fixed tree topology. We calculate the ML scores of the starting trees and the final trees in the rest of our experiments using PAUP.

3.2.3 The Use of Random Starting Trees

A random tree refers to a tree chosen uniformly at random from among all possible tree topologies. We did not use them in our experiments, for the following reasons:

- Earlier experiments with the smaller datasets, Lina, Linum and ETS indicated that the final trees obtained with a random tree were far worse than trees obtained with the NJ and ratchet

starting trees for PAUP and PHYML, and are never better than other final trees for RAxML. This is illustrated in Figure 3.1.

- Derrick Zwickl, the author of GARLI suggested that sometimes GARLI does better with a random starting tree than with a ratchet starting tree (personal communication). However, for the three largest real datasets, GARLI's performance with a random starting tree is relatively inferior to its performance with the other two starting trees, as seen from Figure 3.2. Only for the Angio dataset is the final tree with the highest ML score obtained with a random starting tree. The random starting trees were not generated and scored by PAUP, but GARLI itself. For each dataset and for each starting tree, GARLI was run independently five times, as recommended by Derrick Zwickl [Zwi06], and the ML scores reported are averaged over the five runs.

- We did not use random starting trees with MrBayes for the following reason: recall that we score all our starting trees with PAUP, estimating the model parameters that are to be kept fixed during topology and branch-length optimization. Calculating the ML score of random starting trees proved to be a daunting task for the two largest datasets, Nematodes and rbcL. A random tree on these datasets could not be scored even after days of analysis using PAUP. Since the tree-topology and branch-length optimization itself is limited to 24 hours, the results obtained with a starting tree that has been scored after many days with PAUP would not be very meaningful. Alternatively, the random starting trees could have been generated and have had their ML scores calculated with GARLI, but this is not how MrBayes is run in practice.

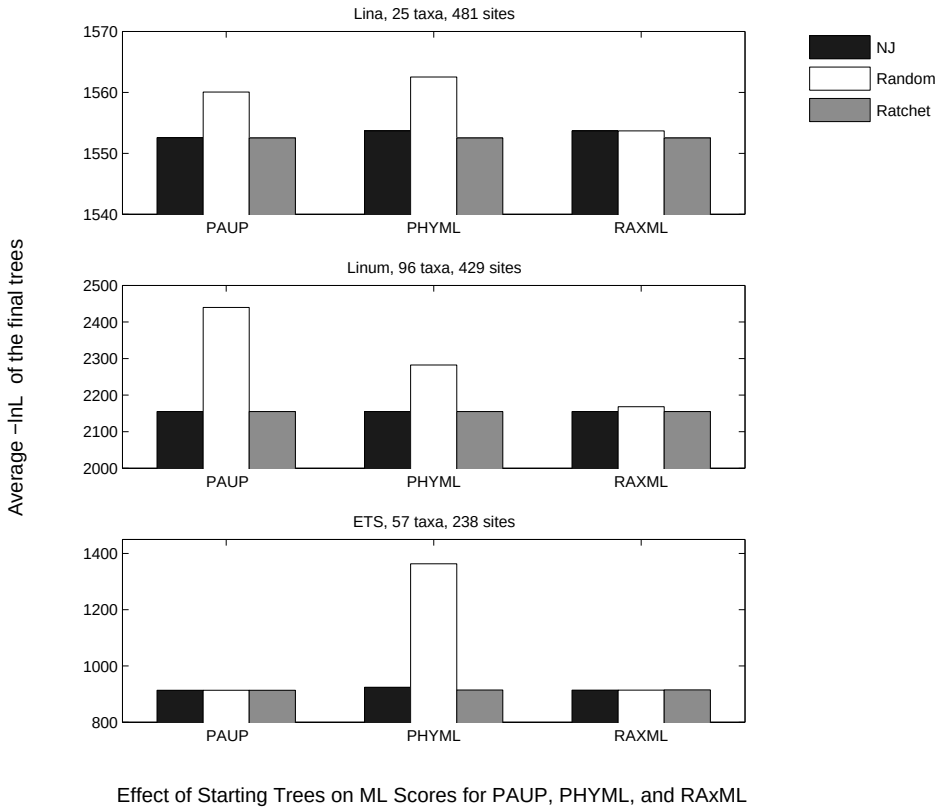


Figure 3.1: ML scores of the final trees for the Lina, Linum and ETS datasets: the performances of PAUP and PHYML are much worse with a random starting tree than with the other two starting trees, and the performance of RAxML with a random starting tree is never better than its performance with the other two starting trees.

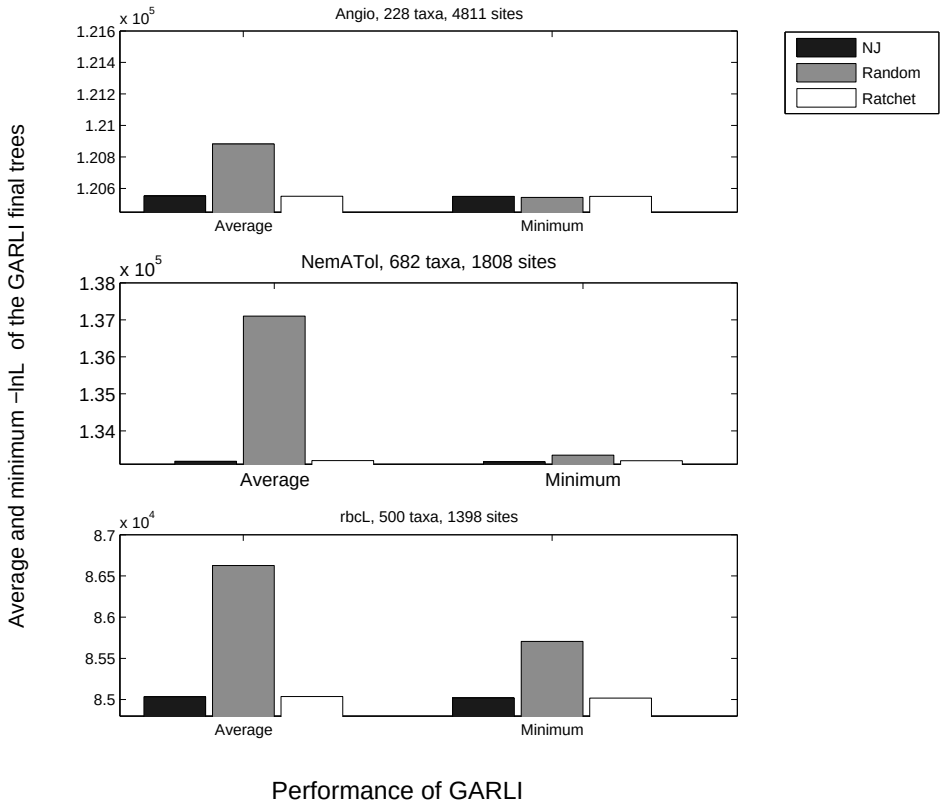


Figure 3.2: GARLI's average performance with random starting trees is poor. But in one instance, for the Angio dataset, a final tree obtained with a random starting tree achieves the best -lnL score of 120543.71465 among all the final trees: the next best score is 120549.67754, achieved with a ratchet or an NJ starting tree. However, for the Nematodes and rbcL datasets, performance with random starting trees is inferior to performances with the NJ starting trees.

3.3 Experimental Results on Real Datasets

The two main objectives of our experiments, as detailed at the beginning of this chapter, were to determine the relative performance of the ML heuristics and the impact of starting trees on the outcome of the heuristic searches. In this section on real datasets, the measure of the quality of search heuristics and the starting trees will be the ML score of the final trees they produce. The true

evolutionary tree cannot be known for real datasets, and hence it is not meaningful to talk about the topological accuracy of the final trees. Topological accuracy of the final trees will be discussed when we present our results on simulated datasets, in Section 3.4.

Relative Performance of ML-specific Heuristics. Among the software intended as ML heuristics, PHYML, RAxML and GARLI are the three best methods. Among PHYML, RAxML and GARLI, RAxML and GARLI are better than PHYML. The performance of the traditional heuristic, PAUP, while comparable to that of PHYML, is not comparable to those of RAxML and GARLI. This assessment is supported by Figures 3.3, 3.4 and 3.5.

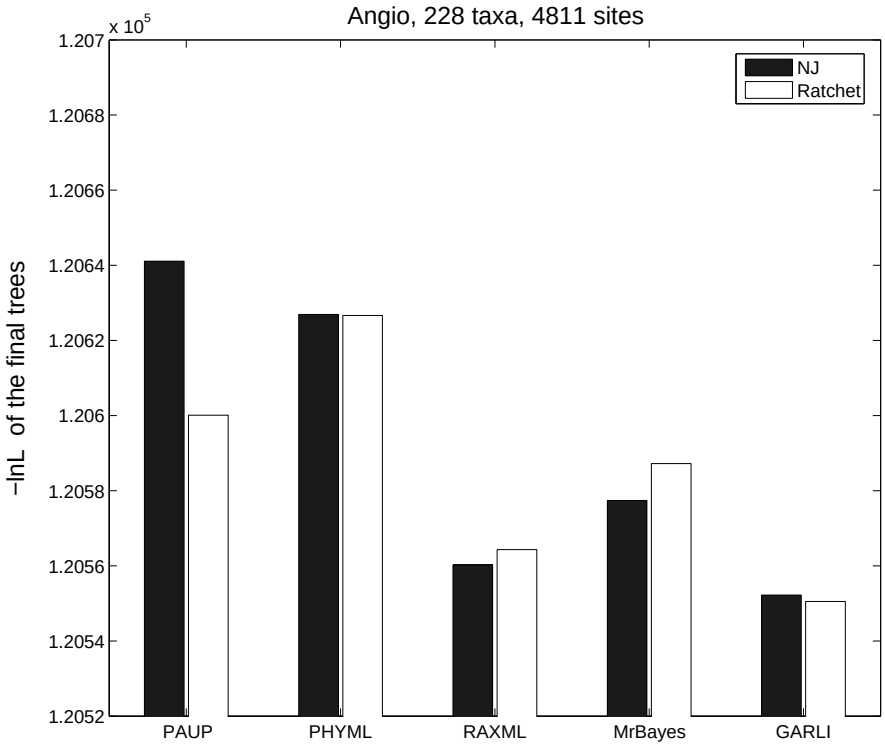


Figure 3.3: The plot shows the -lnL scores of the final trees from PAUP, PHYML, MrBayes, RaxML and GARLI optimizations for the Angio dataset. The figures for GARLI are the averages from five independent runs.

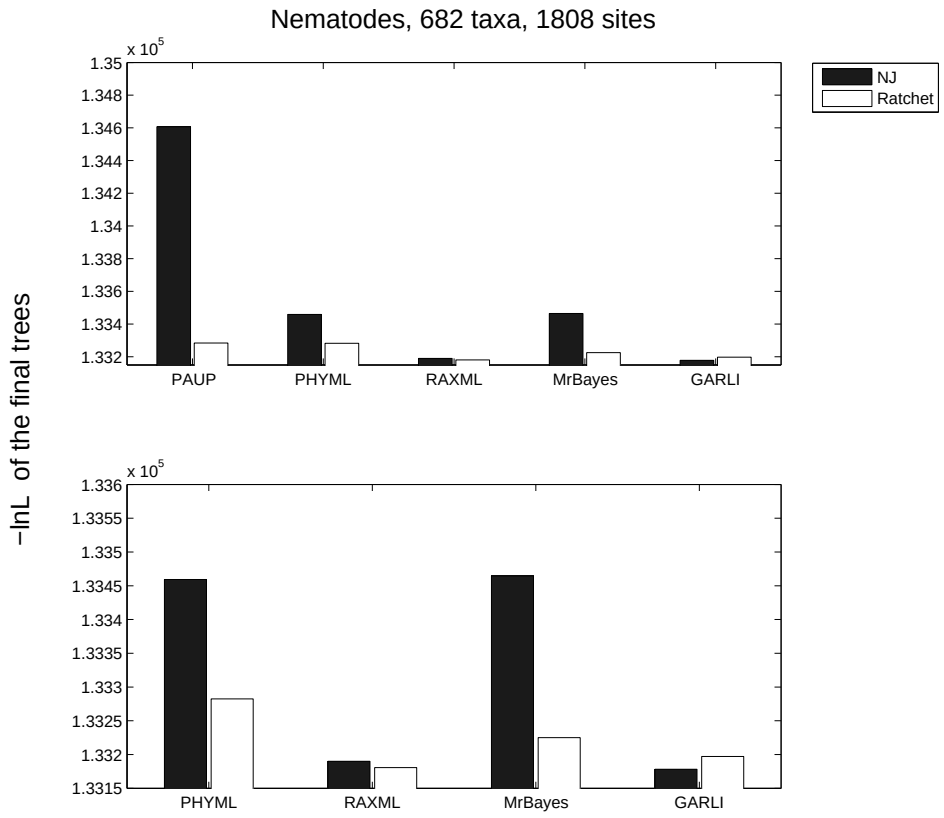


Figure 3.4: The first plot shows the -lnL scores of the final trees from PAUP, PHYML, RAxML, MrBayes and GARLI optimizations for the Nematodes dataset. The second plot presents the same data, but excludes results from PAUP for better visualization. The figures for GARLI are the averages from five independent runs.

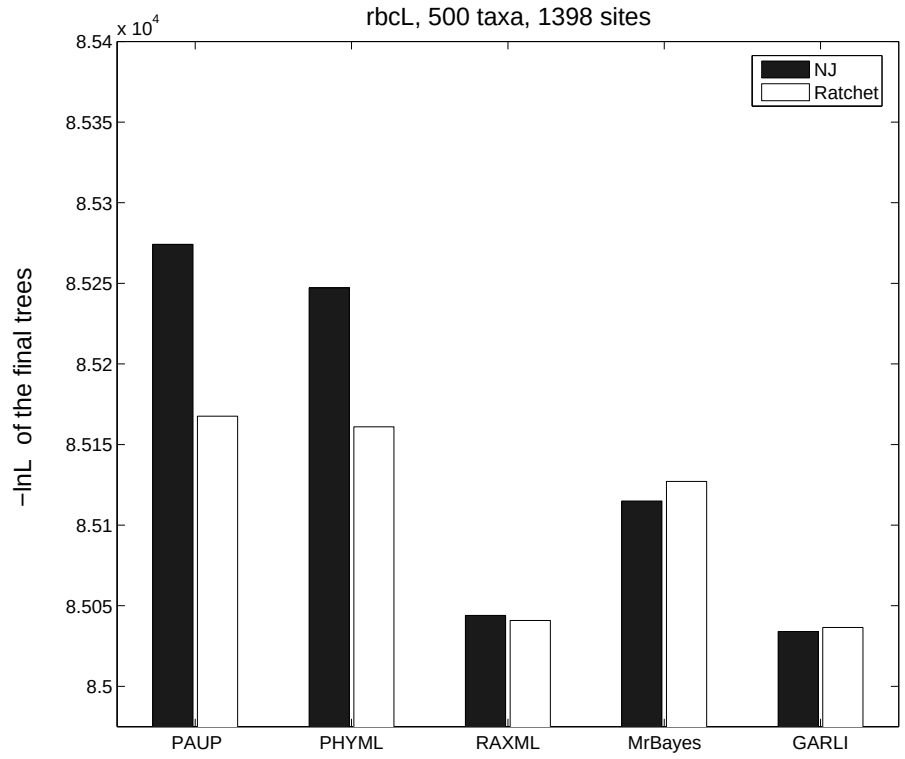


Figure 3.5: The plot shows the -lnL scores of the final trees from PAUP, PHYML, MrBayes, RaxML and GARLI optimizations for the rbcL dataset. The figures for GARLI are the averages from five independent runs.

The Impact of the Starting Trees. The data presented show that starting trees do have an impact on the outcome of the search, but the relationship between the quality of the starting trees and the quality of the final trees is not always straightforward.

- The ML scores of the starting trees improve in this order: random, NJ, ratchet. This is seen from Tables 3.2, 3.3 and 3.4.

- PAUP and PHYML benefit from being provided good starting points. For PAUP and PHYML, the ML score of the final tree is strictly correlated with that of the ML score of the initial tree they were provided. This can be observed in Figures 3.3, 3.4 and 3.5. Significantly, PAUP

and PHYML are also the software that perform poorly with random starting trees.

- For RAxML, MrBayes and GARLI, the starting trees do have an impact, but the manner in which they affect the outcome is not always straightforward. Sometimes the final tree obtained with an NJ starting tree has a higher ML score than the tree obtained with a ratchet starting tree. See, for instance, Figures 3.3 and 3.4. It can be seen from the first of these figures that RAxML obtains better final trees in terms of ML scores with an NJ starting tree for the Angio dataset. The second figure shows that GARLI obtains a final tree with a higher ML score with an NJ starting tree than with a ratchet starting tree for the Nematodes datasets. Similarly, it can be seen from Figure 3.3 that MrBayes performs better with an NJ starting tree than with a ratchet starting tree.

	-lnL	Time (hr:min:sec)	
		to construct	to score
NJ	121414.69990	00:00:01	00:38:30
Ratchet	121189.78026	01:00:00	01:14:02

Table 3.2: **-lnL Scores of Angio Starting Trees**

	-lnL	Time (hr:min:sec)	
		to construct	to score
NJ	136190.64441	00:00:01	01:23:07
Ratchet	133840.69739	01:00:00	03:55:31

Table 3.3: **-lnL Scores of Nematodes Starting Trees**

	-lnL	Time (hr:min:sec)	
		to construct	to score
NJ	86232.11381	00:00:01	00:30:38
Ratchet	85645.25561	01:00:00	00:30:20

Table 3.4: **-lnL Scores of rbcL Starting Trees**

Performance of MrBayes. While MrBayes is not intended to be an maximum likelihood optimization tool, it is widely used as such [MW03]. In fact, [MW03] reports that MrBayes was the best ML optimization tool at that time. Our study shows that MrBayes obtains, after 24 hours, trees that are surprisingly good considering the fact that MrBayes is not an ML heuristic. In fact, it can be seen from Figures 3.3, 3.4 and 3.5 that MrBayes often outperforms PHYML on real datasets. However, RAxML and GARLI still outperform MrBayes consistently.

The Performance of MP Ratchet as an ML Heuristic. We ran the ratchet heuristic on each dataset for 24 hours and computed the ML score of the resulting tree with PAUP. For the real datasets, the MP ratchet tree was far worse than the best ML tree obtained by running an ML software, as can be seen from Table 3.5. This represents a vast improvement in the performance of ML-specific heuristics since the publication of the study [BHD+02] when the ratchet outperformed ML-specific heuristics.

Dataset	ML score of MP-Ratchet tree	ML score of the best ML heuristic
	-lnL	-ln L
Angio	120707.99704	120560.24587
Nematodes	133456.72390	133180.62977
rbcL	85228.07207	85040.82773

Table 3.5: **Performance of MP-Ratchet as an ML heuristic for real datasets**

Performance of PHYML when allowed to estimate the model. Allowing PHYML to estimate model parameters simultaneously while estimating tree topology and branch lengths does not improve upon the the ML score of the overall best final tree, as Table 3.6 shows. In the table, the first PHYML column shows the PAUP-calculated ML scores of the final trees obtained when all model parameters are estimated during the PHYML optimization. The second column shows the PAUP-calculated ML scores of the final trees when all the model parameters are kept fixed during PHYML optimization. In one of the three datasets PHYML recovers a tree with a slightly worse -lnL score when it is allowed to estimate the model parameters than when the model parameters are fixed. In the other two datasets, the improvements in -lnL scores PHYML obtains by estimating model parameters is very slight.

Dataset	PHYML (estimating model parameters) -lnL	PHYML (fixing model parameters) -lnL
Angio	120625.36073	120623.12021
Nematodes	133302.99467	133305.93587
rbcL	85160.42069	85160.99522

Table 3.6: **Two Ways of Running PHYML**

Statistical Significance of the Differences in ML Scores. As observed earlier, some final trees have relatively higher ML scores than others. However, this does not mean that the differences in ML scores are necessarily significant. In fact, our results show that *most* final trees are statistically indistinguishable from the highest-scoring tree, based on the Shimodaira-Hasegawa test [SH99]. The test results are presented in Table 3.7. We explain the test and the results in detail below.

In maximum likelihood phylogenetic estimation from a dataset of sequences, the evolution of the input dataset is assumed to be a random process. In estimating a tree along with the model parameters and branch lengths, we are attempting to estimate the parameters of the random process

that generated the sequences. However, note that the input dataset is not necessarily the only product of the random process. Rather, the specific input dataset is just one instance of the possibly many products of the true random process. Hence, if a tree T_1 has a higher ML score than a tree T_2 for a dataset x, it could be just due to chance. It is possible that for a different dataset y generated by the same random process that generated x, T_2 could have a higher ML score than T_1. Thus, we would like to *statistically* test if T_1 is better than T_2 under ML, not by measuring the difference in their ML scores on just one dataset. Intuitively, T_1 would be a better ML tree than T_2 if it has a higher ML score on a large fraction of independently sampled datasets all generated by the same random evolutionary process as the original dataset. For real datasets, a large number of such independent samples can be generated by *bootstrap sampling* from the original dataset [Efr79]. Given a data matrix with k columns, a bootstrap sample or a *replicate* is generated by sampling k columns from the original data matrix at random, independently and with replacement.

The Shimodaira-Hasegawa (SH) test for the significance of differences in ML scores among a set of trees T_1 through T_m on a dataset x adapts the above ideas to test the statistical significance of the ML scores among a *set* of trees. The test works as follows (our description follows [Fel03]):

1. For each tree T_i, its log-likelihood score on the dataset x, L_i, is calculated. Let the highest log-likelihood score achieved by any tree on x be H, and let tree T^* be the tree that achieves it.

2. A large number, say b, of bootstrap replicates x_1 through x_b are generated from x.

3. For each tree T_i and each replicate x_j, the log-likelihood score L_{ij} of tree T_i on replicate x_j is calculated. Further the difference between L_{ij} and the highest log-likelihood score achieved for that replicate, denoted S_{ij} is calculated. If the highest log-likelihood score for replicate j is H_j, then $S_{ij} = H_j - L_{ij}$.

4. For each tree T_i, the fraction of bootstrap replicates where $S_{ij} > H - L_i$ is calculated. This fraction is the *p-value* associated with tree T_i in the SH test.

This p-value associated with tree T_i approximates the probability that we observe a difference in log-likelihood score of $H - L_i$ between T_i and T^* if the *null hypothesis* that T_i and T^* are indistinguishable under ML is true. Customarily, the null hypothesis is rejected is the p-value is less than 0.05. See [BNB96] for a discussion of hypothesis testing and p-values.

Thus, under the SH test, all trees with a p-value greater than 0.05 are considered statistically indistinguishable under ML. The results of our experiments are presented in Table 3.7. For each dataset, the following trees were included in the SH test:

- Two initial trees, NJ and ratchet.
- Two final trees each from PHYML, MrBayes and PAUP run with 4 and 10 rate categories. The two final trees correspond to the NJ and ratchet initial trees.
- Three final trees from RAxML, since RaxML was run with a random starting tree as well.
- Fifteen final trees from GARLI, since GARLI was run five times independently with three different starting trees, NJ, random and ratchet.
- A PHYML final tree whose starting tree itself was a RAxML final tree from a ratchet starting tree.
- Finally, for the Angio and rbcL datasets, we ran PAUP for 60 days starting from a tree that had the highest known ML score at that time. For the Angio dataset, this tree was the same as the model tree used for simulations (see Section 3.2 for a description of the various model trees). For the rbcL dataset, this tree was obtained by running PHYML from the final tree of a RAxML optimization starting from a ratchet tree (as in the previous bullet point). For the Angio and rbcL datasets, this tree proved to be the tree with the highest ML score. Such a tree for the Nematodes dataset could not be obtained in time to be included in our statistical tests.

Thus, the pool of trees in our test includes the non-random starting trees, and all the final trees from PAUP, PHYML, RAxML, GARLI and MrBayes. Two more trees were obtained by first running PHYML on the tree that had the highest known ML score at that time, and subsequently running PAUP for two months on the then best ML tree (recall that this last tree was not obtained for the Nematodes dataset). These last two trees were obtained to make sure that the results aren't influenced by the 24-hour constraint of our experiments. The RAxML and GARLI trees from random starting trees were included because (a) RAxML performs well with random starting trees, even if it never improves upon other starting trees, as seen from Figure 3.1, and (b) while on an average GARLI's performance with random starting trees is relatively poor, GARLI with a random starting tree returns the best known tree for at least one dataset, as seen from Figure 3.2.

As can be seen from Table 3.7, out of the 30 or 31 trees under consideration, only for about 5-8 trees is the difference between their ML scores and that of the highest scoring tree statistically significant. In other words, for three-quarters of the trees there is a probability of at least 0.05 that they are as good as the highest scoring tree under ML, even when the observed difference in ML score on the original dataset is hundreds of log likelihoods. Further, as noted in Table 3.7, at least one such tree is about 25% topologically different from the highest scoring tree for each of the datasets. That is, there is high topological variance among trees that are currently statistically indistinguishable under ML. The high topological difference between statistically indistinguishable trees calls into question the desirability of ML as phylogenetic inference criterion, since for a phylogenetic inference method, it is desirable that the tree topologies it favors are not very dissimilar. On the other hand, if we accept ML as a phylogenetic inference method, we must conclude that the existing ML heuristics are not satisfactorily solving ML. In other words, it is possible that trees with much higher ML scores which are topologically similar among themselves exist that can be statistically distinguished under ML from the currently high-scoring tree topologies, but that the current ML heuristics are not obtaining such high-scoring tree topologies. In any event, our results indicate that ML optimization is anything but a solved problem, in spite of the impressive

improvements made by the newer ML software PHYML, RaxML and GARLI.

	Angio	Nematodes	rbcL
# trees	31	30	31
# trees with $p < 0.05$	5	8	7
highest ln L	-120536.56254	-133166.04199	-85016.75318
Method achieving highest ln L	PAUP, 60-day analysis	GARLI, NJ starting tree	PAUP 60-day analysis
Δ_{-lnl}	104.54201	344.74307	257.45157
Δ_{RF}	24.44	27.39	27.36

Table 3.7: The results of the SH test of the significance of differences in ML scores among a pool of trees for the three datasets. The row labeled Δ_{-lnl} shows the maximum difference in the negative log-likelihood scores between the highest scoring tree and a tree with $p > 0.05$. The row labeled Δ_{RF} shows the maximum normalized RF distance from the highest scoring tree to a tree with $p > 0.05$.

3.4 Experimental Results on Simulated Datasets

With the experiments on the real datasets, we were able to compare the various ML heuristics on their ability to search for trees with higher ML scores. Experiments on the simulated datasets allow us to compare the topological accuracy of the ML heuristics as well, apart from their ability to search for better ML trees. Recall that our experimental methodology for simulated datasets is identical to that for the real datasets, except for the following two differences:

- Since the performance of the ratchet as an ML heuristic was poor compared to the other heuristics (Section 3.3, Table 3.5), we eliminate ratchet from our studies.
- With real datasets, GARLI was run five times independently with each starting tree. But with

simulations, GARLI was run just once for each simulated dataset and with each starting tree. This was done so as to avoid the proliferation of experimental runs. We have 10 simulated datasets corresponding to each real dataset. For each heuristic, including GARLI, the ML scores and topological accuracies presented are averaged over the 10 simulated datasets.

We present the results of our experiments on simulated datasets below.

Relative Performance of ML-specific Heuristics. The new heuristics, PHYML and RAxML and GARLI, outperform PAUP but only marginally. PAUP's performance is indistinguishable from that of others for the simulated Angio datasets with the NJ or the ratchet starting trees (Figure 3.6): the final lnL scores obtained by PAUP are just 3 less than those obtained by RAxML and GARLI and are 2 higher than those obtained by PHYML, and the topological error rates of all methods are within 1% of one another. PAUP's performance is similarly indistinguishable from that of the others for the simulated Nematodes datasets too, when a ratchet starting tree is used (Figure 3.7): the differences in lnL scores are less than 10, and the differences in error rates are less than 2%. Only for the simulated rbcL datasets do the newer ML methods achieve a much higher final lnL scores with both the starting trees, with the difference being about 300. But even here, the difference in topological error rates is only about 2% (Figure 3.8). Nevertheless, PAUP almost never outperforms the newer heuristics. The only exception is that PAUP's final trees for the simulated Angio datasets achieve very slightly better topological accuracies than the final trees from the other methods. But even here, the improvement is less than 0.3%. The performances of the newer heuristics are nearly identical across all the simulated datasets, except that the performance of PHYML is slightly worse than that of RAxML and GARLI with NJ starting trees. The above observations are supported by Figures 3.6, 3.7 and 3.8.

The Impact of the Starting Trees. Again as observed with the real datasets, starting trees generally do have an impact on the outcome of the search, but the relationship between the quality of the starting trees and that of the final trees is not the same for all methods or across all methods. The ratchet starting trees are more topologically accurate than the NJ starting trees, as seen

from Table 3.8. However, it is not the case that the ratchet starting trees result in better final trees always. For the simulated Angio datasets, the starting trees do not seem to have any significant impact on the outcome of the search, as seen from Figure 3.6. The ML score or the topological accuracy of the final trees produced by RAxML do not seem to depend on the starting tree, for any dataset. Starting trees seem to have the greatest impact on PAUP, which for the Nematodes and the rbcL datasets produces final trees with higher ML scores and accuracies with ratchet starting trees. The impact of starting trees on PHYML is similar to that on PAUP, but the difference between final trees obtained with ratchet and NJ starting trees is smaller with PHYML than with PAUP. MrBayes performs marginally better with NJ starting trees than with ratchet starting trees, when there is a difference in performance between the two starting trees. GARLI's performance with the two starting trees is nearly identical for the Angio and the rbcL datasets. On the Nematodes datasets, GARLI performs better with the NJ starting trees. These observations are supported by Figures 3.6, 3.7 and 3.8.

The Topological Accuracy of the Final Trees. The significant observation here is that for two of the three datasets, the final trees have high topological error: for the simulated rbcL datasets, the error ranges from about 8% to 11%, and for the simulated Nematodes datasets, the error ranges from about 11% to 18%. These observations are supported by Figures 3.7 and 3.8. It is, however, desirable that the error is well below 10%, as with the simulated Angio datasets (Figure 3.6).

We hypothesized that the high error rates might be because the optimization is performed under the wrong model. To test this, we ran the following experiment: for the simulated Nematodes datasets, we ran GARLI and PHYML with the model parameters fixed to their true values, and compared the outcomes with the earlier outcomes shown in Figure 3.7. The comparison is shown in Figure 3.9. The figure shows that the topological error of the final trees produced by PHYML and GARLI worsen in some cases, and in any event are always above 10%. This suggests that the high error rate may not be due to erroneous assumptions about the model. That is, even when provided the the correct model, the current ML methods are not able to obtain accurate final trees.

This could be due to the limitations of the methods because of which the final trees are far from being optimal. But again, as in Section 3.3, this also may point to the limitation of ML as a phylogenetic estimation criterion. We leave these questions open for future research.

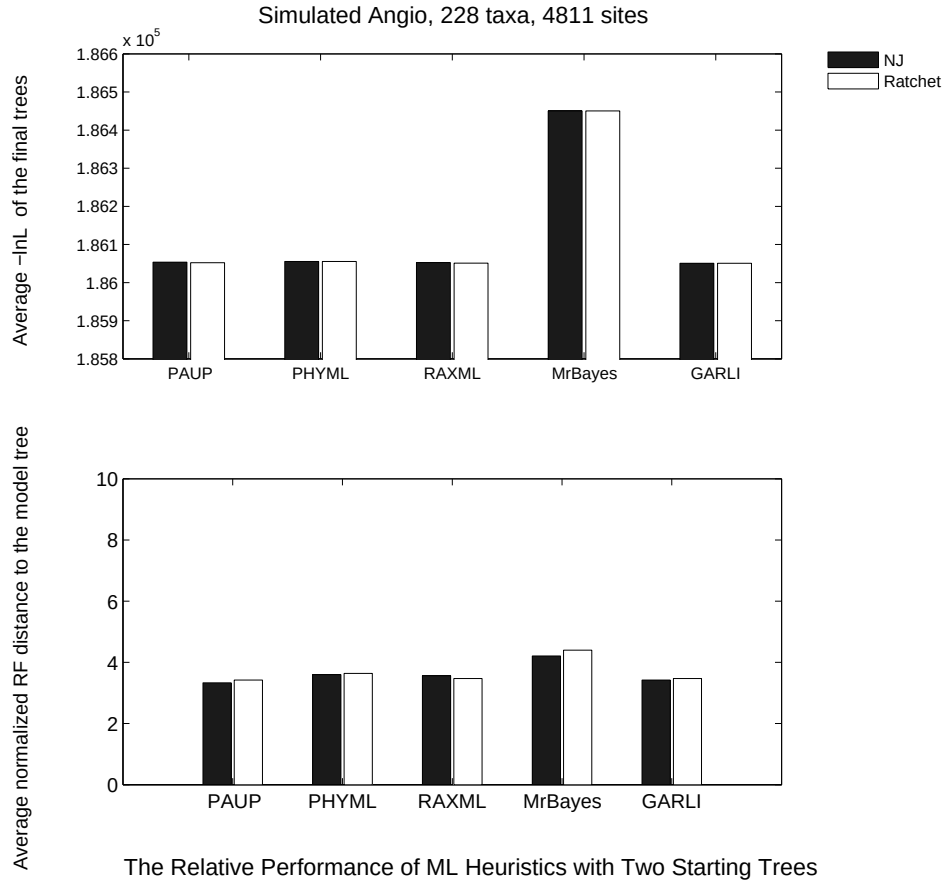


Figure 3.6: The plots show the results of topology and branch-length optimization for the simulated Angio datasets. The first plot shows the average -lnL scores of the final trees obtained with different software with NJ and ratchet starting trees. The second plot shows the topological accuracy of the final trees.

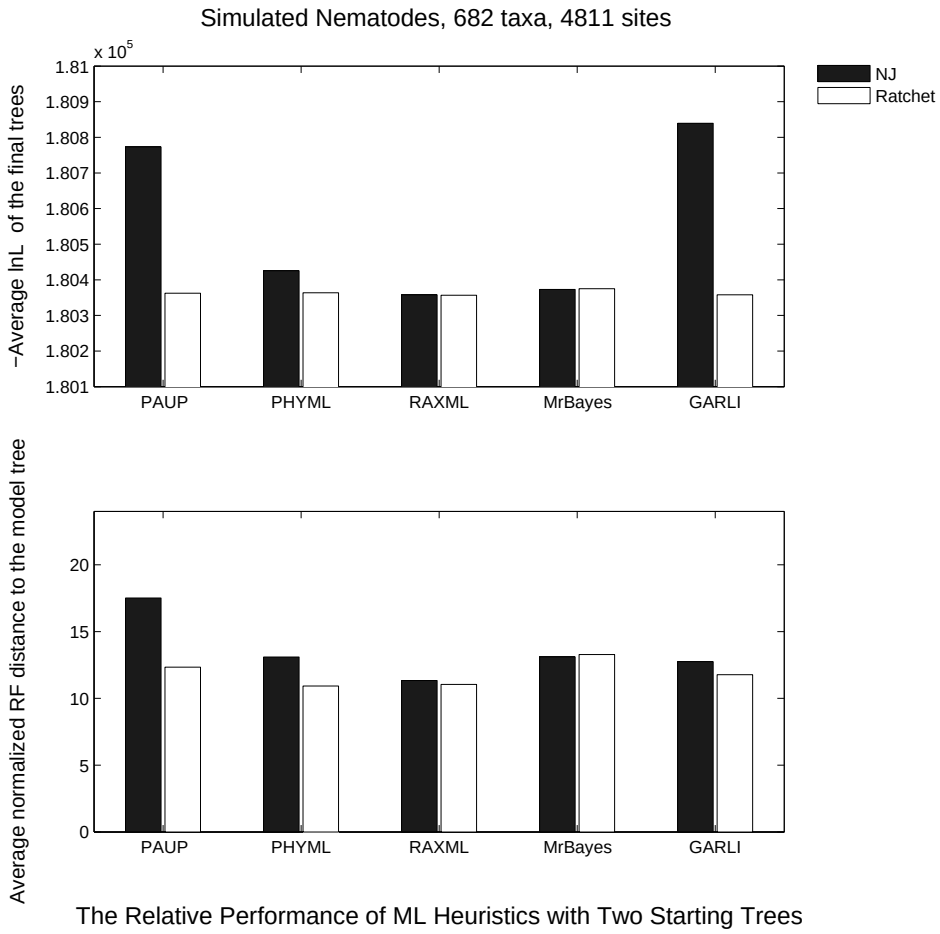


Figure 3.7: The plots show the results of topology and branch-length optimization for the simulated Nematodes datasets. The first plot shows the average -lnL scores of the final trees obtained with different software with NJ and ratchet starting trees. The second plot shows the topological accuracy of the final trees.

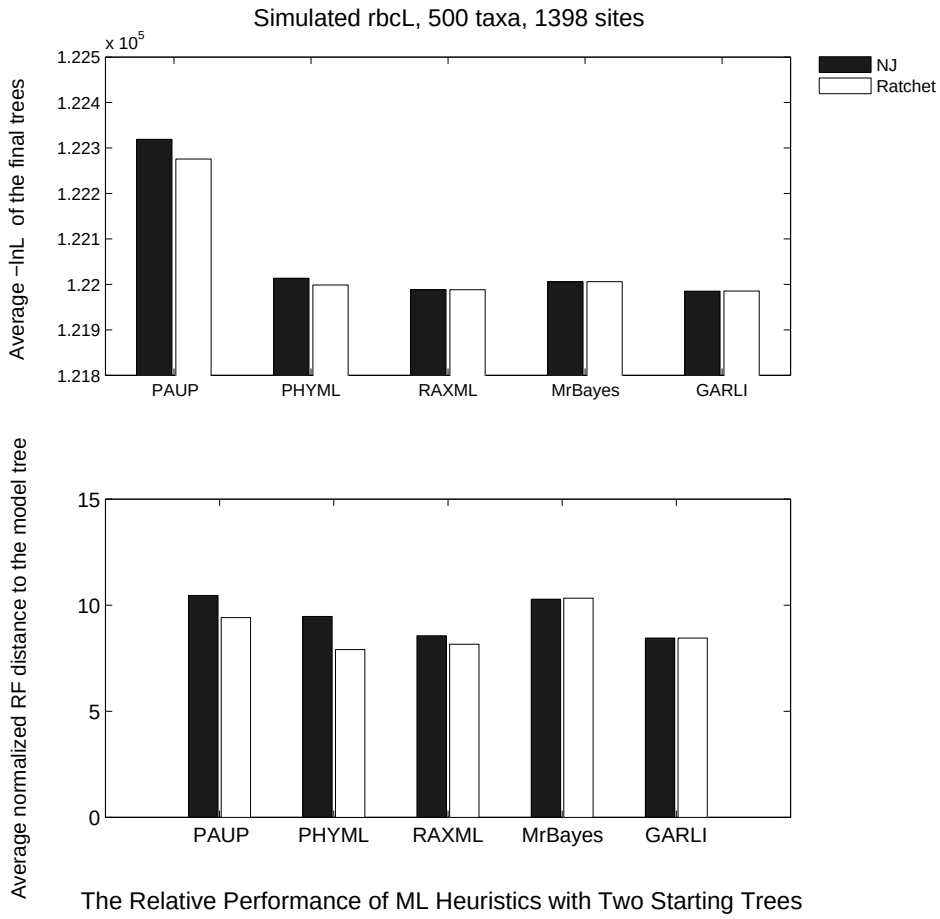


Figure 3.8: The plots show the results of topology and branch-length optimization for the simulated rbcL datasets. The first plot shows the average -lnL scores of the final trees obtained with different software with NJ and ratchet starting trees. The second plot shows the topological accuracy of the final trees.

	NJ	Ratchet
Angio	16.93	10.27
Nematodes	26.54	13.78
rbcL	19.72	12.72

Table 3.8: The topological accuracies of the initial trees: the figures are the average of the normalized RF distances between the 10 initial trees of each kind and the model tree. The ratchet trees are more accurate across all the datasets.

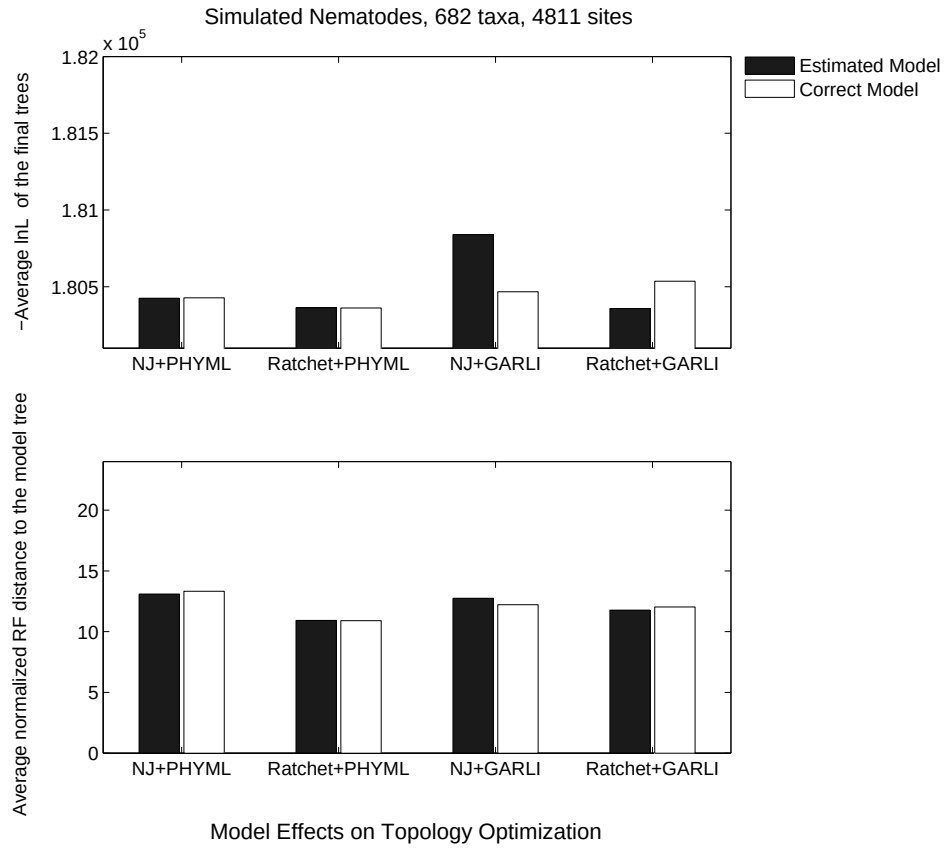


Figure 3.9: The plots show the results of topology and branch-length optimization with PHYML and GARLI for the simulated Nematodes datasets, once with fixed estimated model parameters, and once with fixed correct model parameters. The correct model seems to have very little impact on either the -lnL scores or the topological accuracies. The topological error of the methods does not fall below 10% even when the correct model is used.

3.5 Evaluation of Intermediate Trees in the ML Heuristics

Another factor that seems to have a significant impact on the quality and speed of the search is the heuristic's approach to evaluating the intermediate trees.

As noted in Section 3.2, PAUP is the best method at evaluating the ML score of a given tree topology. However, PAUP also exhaustively evaluates the ML scores of the intermediate trees

that it encounters during heuristic searches. In contrast, the new "second-generation" ML software PHYML, RAxML and GARLI settle for a quick and crude evaluation of the ML scores of the intermediate trees. This allows them to explore a much larger space of tree topologies in a given amount of time.

In this section, we briefly describe the features of the search heuristic that are relevant to our discussion: how each software generates and evaluates intermediate trees during a search.

In **PAUP**, new candidate trees are generated through the application of an NNI, SPR or TBR reconnection move on the current candidate tree (refer to Chapter 2 for the definitions of these moves). The default tree-rearrangement move is TBR, which we use in our experiments. After each tree-rearrangement move, PAUP carries out a combined heuristic optimization of the lengths of all branches in the tree so as to maximize the likelihood score of the new candidate tree.

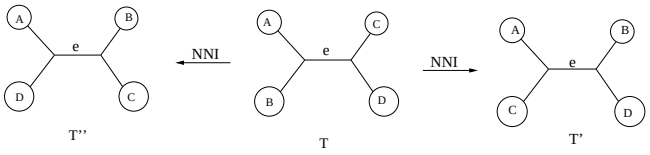


Figure 3.10: Tree T can be transformed into either T' or T'' with one NNI move

PHYML follows the following procedure to generate and evaluate a new candidate tree: an NNI move on an internal edge e can generate one of two NNI neighbors, as seen in Figure 3.10, reproduced here from Chapter 1. Let the current length of an internal edge e be denoted l_e. PHYML first heuristically computes the optimum length l_e^* of each internal branch e so as to maximize the ML score of the current tree while keeping all other branch lengths fixed at the original value. It then evaluates each possible NNI move as follows: the optimal length l_f^* of the new branch f created as a result of the NNI move is estimated, keeping all other branch lengths fixed, so as to maximize the likelihood score of the NNI neighbor. The likelihood scores of the NNI neighbors thus computed are used to rank the NNI moves. Now a new candidate tree is generated as follows:

- The fraction λ of the best NNI moves are applied. This fraction is initially set to a high value, and in the default PHYML implementation, it is set to 0.75. The best NNI move is always applied irrespective of the value of λ. When an NNI move is applied so as to create a new branch f, the new branch gets the optimal length l_f^* computed earlier.

- For those edges e for which an NNI move does not improve upon the ML score of the current tree, the length of the edge is modified to $l_e + \lambda(l_e^* - l_e)$.

The likelihood score of the new candidate tree is computed, and if it doesn't improve upon the current score, the whole procedure is repeated with λ divided by two. The application of the λ fraction of the best moves, and the division of λ by two if the score does not improve, is repeated until the likelihood score of the new candidate tree betters that of the current tree. Note that even if λ approaches zero, the best NNI move is always applied, and this ensures that the likelihood score of the current tree is improved upon, as long as there is at least one NNI move that improves the likelihood score. Note also that PHYML never carries out combined heuristic optimization of all branch lengths in a tree, as PAUP does. When a new candidate tree is obtained by applying a set of NNI moves on the current tree, it is by no means certain that the assigned branch lengths are optimal for the new candidate tree. Each assigned length l_f^* or l_e^* is optimal only if all the rest of the lengths are held at the values in the current tree. For the new candidate tree, only the likelihood score, and not the maximum likelihood score over all possible branch lengths, is computed with these assigned branch lengths.

The software **RAxML**, follows a different strategy to generate and evaluate new candidate trees from the current tree. RAxML uses the SPR move to generate new trees, as follows:

- Recall that in an SPR move, a subtree is pruned from the current tree by deleting an edge e, and a new tree is generated by re-attaching the subtree at a different edge f. The re-attachment bifurcates the edge f into two edges in the new tree, f_1 and f_2. RAxML evaluates an approximate likelihood score of the new tree by only optimizing the branch lengths of

e, f_1 and f_2. RAxML then collects the top 20 SPR neighbors based on this approximate likelihood score.

- A global branch-length optimization is performed on the top 20 trees to estimate their ML scores more accurately. The top-scoring tree is chosen as the new candidate tree.

RAxML also reduces the number of trees it evaluates before it generates a new tree as follows: initially, the SPR moves are constrained so that the number of edges between the old point of attachment of the pruned subtree and its new point of attachment is limited to a specified value. However, if this results in the current tree being a local optimum, this limit is increased by one, up to a maximum of 20, and whole process is repeated with the new limit. If the current tree proves to be a local optimum, even with the maximum limit, the search is ended. Note that RAxML does not perform a global branch-length optimization on all trees considered during the search.

In **MrBayes**, the *local move* from [LS99] is predominantly used as the topology-changing move. The *local* move in fact may or may not change the topology, but when it does it is an NNI move. However, MrBayes is a Bayesian analysis software, and at no point during a run is a global branch-length optimization on the current tree performed.

In **GARLI**, new candidate trees are generated from old candidate trees using an NNI or an SPR move [Zwi06]. The most commonly employed move is the SPR move constrained so that the point of re-attachment of the pruned subtree is within a small radius of the point where the subtree was originally attached. Here, the radius refers to the number of edges between the original point of attachment and the point of re-attachment. After the SPR move, an exhaustive branch length optimization is not performed. Instead, branch lengths in a small radius around the point of re-attachment are iteratively optimized until the ML score of the tree improves by more than the a prescribed threshold. If the radius of optimization is too small to improve the ML score, the radius is increased and the branch lengths are re-optimized. Thus, GARLI again avoids exhaustive

re-optimization of the branch lengths of intermediate trees.

Search Space Explored by PAUP, PHYML and GARLI. We now make the above discussion concrete, by presenting some evidence that the newer ML methods PHYML and GARLI explore a larger tree space in a given amount of time than PAUP. For PAUP, for each of our experiments with real datasets, we report the number of TBR rearrangement moves *attempted* by PAUP over the course of 24 hours. The number of successful TBR attempts is not reported by PAUP. We ran PAUP in two different ways for each dataset with each starting tree: once with the number of rate categories in the discrete approximation of the Gamma distribution set to 4 and once with the number of rate categories set to 10. With 10 rate categories, the ML scores of the intermediate trees in the search are better estimated, at the expense of spending more time per tree during the search. For PHYML and GARLI, we report the number of *successful* topological re-arrangements performed during the course of the optimization. While comparing our reported figures for PAUP, PHYML and GARLI, it is important to keep in mind that the number of successful moves, i.e., the moves that increase the ML score, is typically far fewer than the number of attempted moves: for an NNI move, for example, there are $\Theta(n)$ neighbors of a tree, where n is the number of taxa. In the initial stages of a heuristic, a large fraction of attempted moves would be successful, and during the later stages, only a small fraction of attempted moves are successful, until the heuristic reaches a local optimum, when no move will be successful. Thus, on an average if a heuristic explores half of its neighbors before it finds a neighbor with a higher ML score, there would be roughly about $n/2$ attempted moves for every successful move. For PAUP, in fact, this ratio would be much higher, since it uses the TBR move under which each tree has $\Theta(n^3)$ neighbors.

The data are presented in Table 3.9. Similar data for RAxML are not available since the program does not report them. The figures reported for GARLI are averaged over 5 independent runs.

- The number of tree-rearrangement operations *attempted* by PAUP over 24 hours number only in the thousands. Significantly, PAUP explores fewer trees with 10 rate categories, and

returns a tree with a lower ML score at the end of 24 hours. This can be seen in Figure 3.11.

- PHYML makes hundreds of *successful* topological moves. As noted above, typically only a very small fraction of attempted moves are actually successful. Moreover, PHYML reaches a local optimum in just a few hours, as seen from Table 3.9. In contrast, PAUP doesn't reach a local optimum even after 24 hours.

- GARLI makes thousands of successful topological moves. However, since GARLI implements a genetic algorithm, it accepts a topological move with some probability even if it doesn't immediately improve the ML score. Thus, the actual number of topological moves made by GARLI that increase the ML score is likely smaller than the number reported, but is still likely to be in the thousands. Moreover, GARLI completes the search in a few hours, sometimes even faster than PHYML, as seen in Table 3.9. This suggests that GARLI is excellent at exploring the tree space.

Thus, the vast difference between the performance of PAUP on the one hand and that of PHYML, RAxML and GARLI on the other hand appears to stem partly from how large a space of trees that a method is able to explore in a given amount of time.

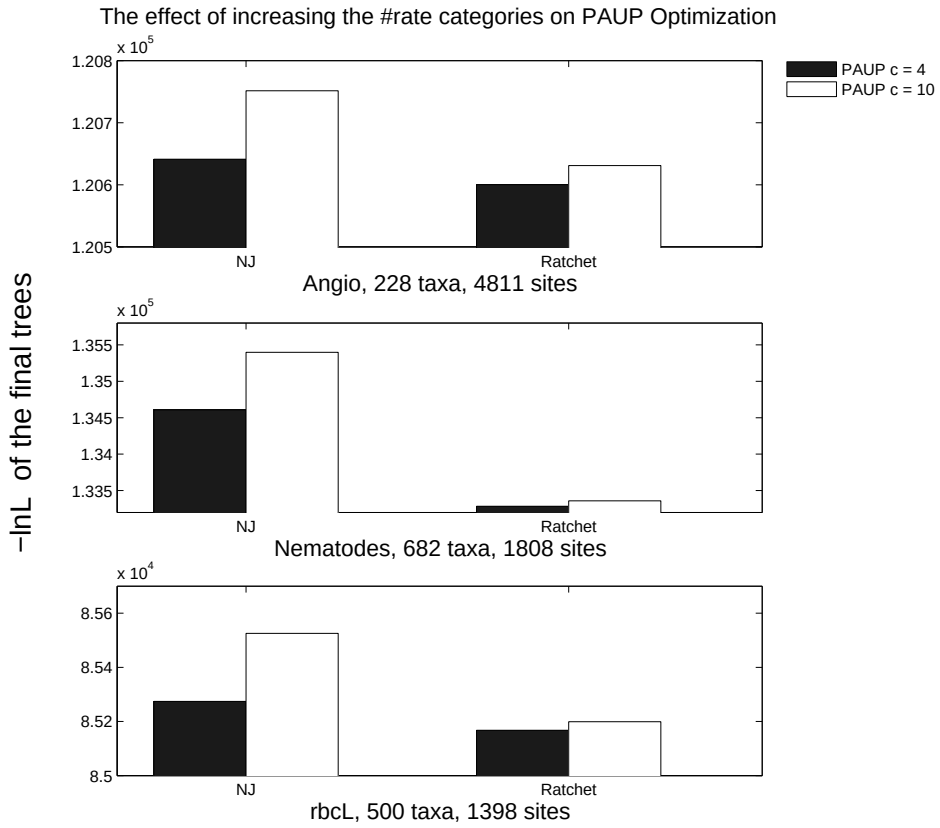


Figure 3.11: The plots show the results of PAUP topology and branch-length optimizations with two different rate categories. "NJ" and "Ratchet" refer to the starting trees for the optimizations. One optimization was carried out with the number of rate categories set to 4 and one with the number of rate categories set to 10. PAUP's performance with 10 rate categories is worse than its performance with 4 rate categories.

	PAUP c = 4	PAUP c = 10	PHYML		GARLI	
	# moves tried	# moves tried	# moves taken	time	# moves taken	time
Angio, NJ	5167	1169	101	3:13:53	96183.8	3:15:33
Ratchet	4948	1196	90	3:57:54	87540.4	2:24:03
Nematodes, NJ	3144	916	462	6:50:50	248289.2	4:01:03
Ratchet	3713	1014	145	5:37:20	153273.4	2:35:03
rbcL, NJ	7829	1730	302	3:11:15	179646.2	1:40:54
Ratchet	6371	1745	94	1:40:01	141646.8	1:23:50

Table 3.9: Size of the search spaces explored by PAUP, PHYML and GARLI, for three different datasets, from NJ and ratchet starting trees. The running time is given in hours, minutes and seconds. PAUP was run for 24 hours for each dataset and with each starting tree.

3.6 Conclusions

We assessed the relative performance of ML heuristics on large real and simulated datasets. Our experiments show that the new ML-specific software PHYML, RAxML and GARLI are significantly better than the traditional software PAUP in obtaining trees with a high ML score over a period of 24 hours. The Bayesian analysis software MrBayes often outperforms PHYML, but not RAxML or GARLI, on real datasets. This represents an improvement in the state of the art of ML heuristics, considering that it was reported in the year 2003, in [MW03], that MrBayes was the best ML heuristic. The performance of a maximum parsimony heuristic, the ratchet, was far worse than those of the ML-specific heuristics. Again, this is evidence of the drastic improvement

in the performance of ML-specific heuristics over what was reported in [BHD+02], where it was reported that the ratchet was a competitive ML heuristic. We hypothesize that the improvement in the new ML heuristics is due the manner in which intermediate trees are generated and evaluated. PAUP spends a lot of time evaluating the ML score of intermediate trees, whereas the new methods PHYML, RAxML and GARLI evaluate the intermediate trees quickly, settling for crude approximations. This enables them to explore a larger search space, and also reduces their dependency on good starting trees to obtain good final trees. Of the three new methods, our data show that RAxML and GARLI are better than PHYML.

While dramatic improvements have been achieved in searching for better ML trees on large real datasets, our results indicate that the higher ML scores achieved are, in most cases, statistically insignificant. This could be interpreted in two ways: either the newer ML methods, in spite of their vast improvements, do not produce final trees statistically distinguishable from the final trees produced by earlier heuristics, or ML is limited as a phylogenetic inference criterion. The high topological error of the final trees obtained for two of the simulated datasets also supports the above interpretations. We leave the resolution of the questions raised here for future research.

Chapter 4

The Maximum Compatible Tree Consensus Method

Recall that in the maximum compatible tree consensus method, we seek to delete a minimum-size subset of the taxa so that the input trees restricted to the remaining set of taxa share a common refinement. As discussed in the Introduction (Chapter 1, Section 1.7), the MCT and the related MAST consensus methods return more informative consensus trees than the strict and majority consensus methods when the input trees differ just in the placement of a few rogue taxa. However, maximum agreement subtrees are often not large enough to be interesting (David Swofford, personal communication). The problem is that requiring complete agreement turns out to be too stringent a requirement, due to which "short edges", like those with a low bootstrap value, pose a problem for MAST. In MCT, we relax the agreement requirement and only demand that the trees on the reduced set of taxa share a common refinement.

In this chapter, we present a polynomial time algorithm for the MCT problem when the trees have a bounded degree. Our algorithm shows that the MCT problem is fixed parameter tractable when the trees have bounded degree [DF99]. We then present two approximation algorithms for the complement of the MCT problem, the CMCS problem defined in the Introduction in Section

1.7.

The chapter is organized as follows: in Section 4.1 we present a fixed-parameter tractable algorithm for the MCT problem for k trees. In Section 4.2, we describe a 4-approximation algorithm and a 3-approximation algorithm for the complement of the MCT problem, the CMCS problem. The 4-approximation algorithm is faster than the 3-approximation algorithm: it runs in $O(kn^2)$ time as opposed to the $O(kn^3)$ running time of the latter algorithm, where k is the number of trees and n is the number of leaves in each tree.

4.1 Fixed Parameter Tractable Algorithm for MCT

It was shown by Hein et al. in [HJWZ96] that the MCT problem can be solved in polynomial time when the trees have bounded degree. The results detailed here were obtained independently.

My results for the MCT problem include algorithms for rooted as well as unrooted trees. For rooted trees, the algorithm runs in time $O(2^{4d}n^2)$ for two trees on n leaves each, d being the degree bound. When there are k trees on n leaves each, the algorithm runs in $O(2^{2kd}n^k)$ time. The algorithm implies that the two-tree MCT problem is fixed-parameter tractable (the parameter being the degree bound). For unrooted trees, the algorithm runs in $O(2^{4d}n^3)$ time for two trees and in $O(2^{2kd}n^{k+1})$ time for k trees. Note that for unrooted trees, the degree bound is assumed to be $d+1$, and not d. The reason is that for unrooted binary trees, the degree of each internal node is three, whereas for rooted binary trees, the degree of each internal node is conventionally considered to be two, discounting the edge that connects a node to its parent.

The algorithm we present is a dynamic programming algorithm and is an extension of the earlier dynamic programming algorithm for the two-tree MAST problem in [SW93], which we described in the Introduction in Section 1.7. We begin by observing the following relation between a maximum compatible subset (MCS) and a maximum agreement subset:

Relation Between MCS and MAST. We begin by observing the following:

Lemma 13 *Let T_1 and T_2 be two unrooted leaf-labelled trees. Let X be an MCS of T and T'. Then there exists a binary tree T_1' refining T_1 and a binary tree T_2' refining T_2 such that X is a MAST of T_1' and T_2'.*

Proof: Since X is a compatible subset of taxa for the pair of trees T_1 and T_2, there is a common refinement T^* of $T_1|X$ and $T_2|X$. Hence we can refine T_1, obtaining T_1', and refine T_2, obtaining T_2', so that T_1' restricted to X yields T^*, and similarly T_2' restricted to X also yields T^*. Then X is an agreement subset of T_1' and T_2'. □

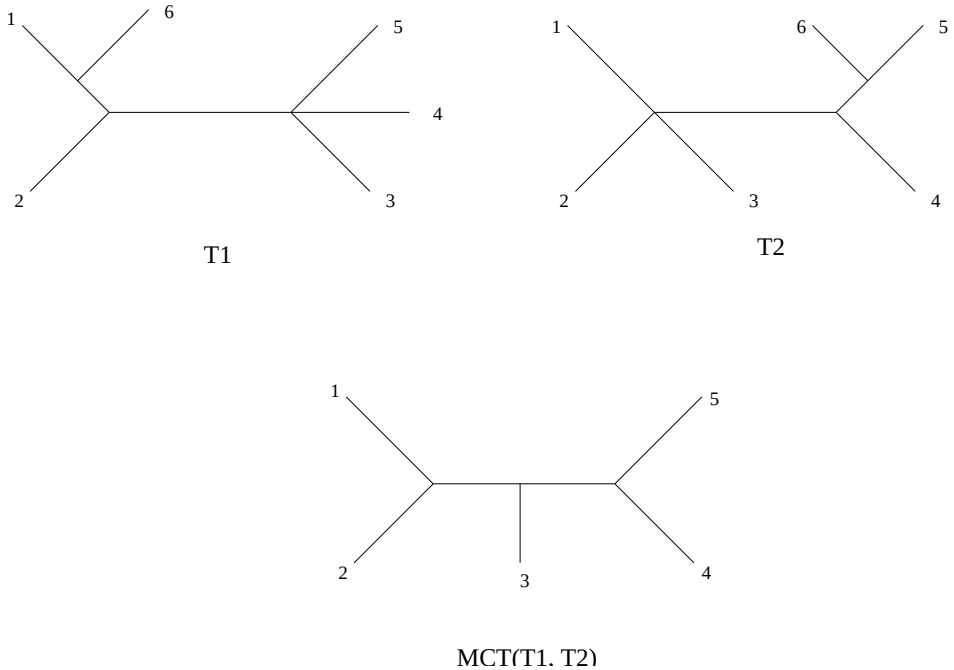


Figure 4.1: The MCT of Trees T1 and T2

The above observation is illustrated in Figures 4.1 and 4.2.

This observation suggests an obvious algorithm for computing an MCS of two trees: for each way of refining the two trees into a binary tree, compute a MAST. However, this algorithm is computationally expensive, since the number of binary refinements of an n leaf tree with maximum degree d can be $O(4^{nd})$. Hence this brute force algorithm will not be acceptably fast.

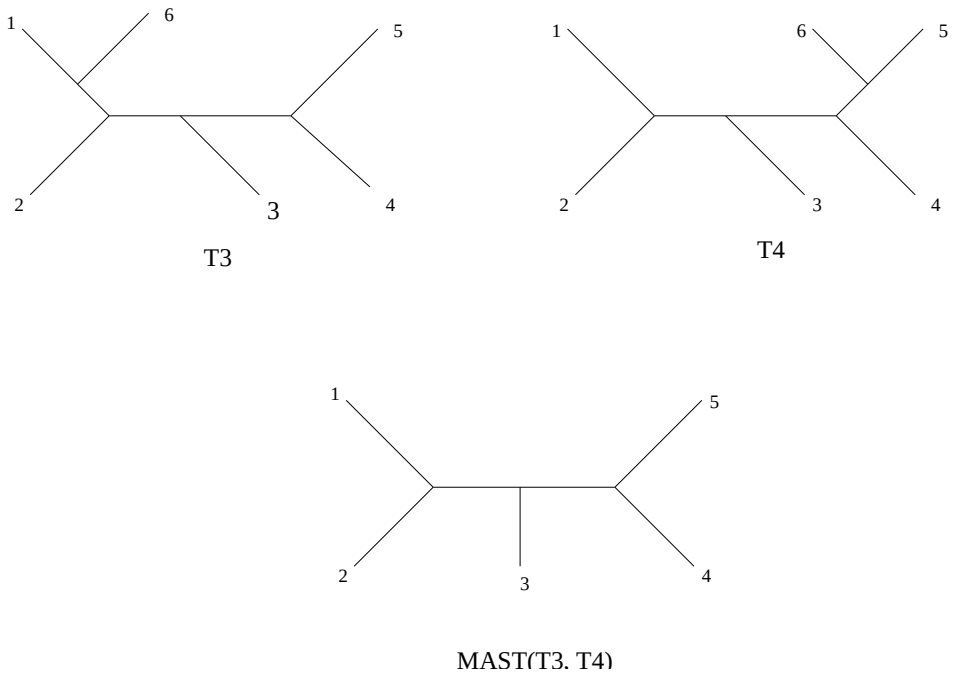


Figure 4.2: The MAST of Trees T3 and T4

4.1.1 Computing an MCS of two rooted trees

We now describe the dynamic programming algorithm for the maximum compatible set (MCS) of two rooted trees, both with degree bounded by d (by this we mean that every node has at most d children). As for the MAST problem, here too the algorithm computes the cardinality of an MCS, but can be easily modified to compute an MCS itself. This algorithm can be extended to produce a dynamic programming algorithm for computing an MCS of two unrooted trees, by computing an MCS of each of the n pairs of rooted trees (obtained by rooting the unrooted trees at each of the n leaves). Furthermore, we also show how to extend the algorithm to handle k rooted or unrooted trees.

The basic set of problems we need to compute must include the computation of an MCS of subtrees T_v and T'_w, for every pair of nodes v and w. We will also need to include the computation of MCS's of other pairs of trees, but begin our discussion with these MCS calculations.

Let T and T' be two rooted trees, and let v and w denote nodes in T and T' respectively. As

in the MAST algorithm, let $c(v)$ and $c(w)$ be the set of children of v and w respectively. Let X be the set of leaves involved in an MCS T^* of T_v and T'_w. Note that $T|X$ and $T'|X$ will only include those children of v and w which have some element(s) of X below them. Let A be the children of v included in $T|X$ and B be the children of w included in $T'|X$. (Note that X defines the sets A and B.)

Note also that any MCS of T and T' actually defines an agreement subset of some binary refinement of T and some binary refinement of T' (Lemma 1). Hence, T^* defines a binary refinement at the node v if $|A| > 1$, and a binary refinement at the node w if $|B| > 1$. In these cases, T^* defines a partition of the nodes in A into two sets, and a partition of the nodes in B into two sets.

There are four cases to consider:

1. $|A| = |B| = 1$, i.e., $A = \{v_i\}$ and $B = \{w_j\}$ for some $v_i \in c(v)$ and $w_j \in c(w)$. In this case, any MCS of T_v and T'_w is an MCS of T_{v_i} and T'_{w_j}.

2. $|A| = 1$ and $|B| > 1$, i.e., any MCS of T_v and T'_w is an MCS of T_{v_i} and T'_w for some $v_i \in c(v)$.

3. $|A| > 1$ and $|B| = 1$, i.e., any MCS of T_v and T'_w is an MCS of T_v and T'_{w_j} for some $w_j \in c(w)$.

4. $|A| > 1$ and $|B| > 1$.

The analysis of the fourth case is somewhat complicated, and is the reason that we need additional subproblems. Recall that T^* defines a bipartition of A into $(A', A - A')$ and B into $(B', B - B')$. Further, recall that T^* is a binary tree with two subtrees off the root; we call these subtrees T_1 and T_2. It can then be observed that T_1 is an MCS of the subtree of T_v obtained by restricting to the nodes below $A - A'$ and the subtree of T'_w obtained by restricting to the nodes below $B - B'$. Similarly, T_2 is an MCS of the subtree of T_v obtained by restricting to the nodes below A' and the subtree of T'_w obtained by restricting to the nodes below B'. Hence we need to define additional subproblems as follows. For each $A' \subset A$ define the tree $T(v, A')$ to be subtree of T_v obtained by deleting all the children of v (and their descendents) not in A'. Similarly define

the tree $T'(w,B')$ to be the subtree of T'_w obtained by deleting all the children of w (and their descendents) not in B'. The construction of tree $T(v,A')$ is illustrated in Figure 4.3. Now define $MCS(v,A',w,B')$ to be the size of an MCS of $T(v,A')$ and $T'(w,B')$ *. From the above discussion it follows that:

$MCS(v,A',w,B')$ is the maximum of:

- $max\{MCS(v,A',w_j,Children(w_j)) : w_j$ a child of $w\}$,
- $max\{MCS(v_i,Children(v_i),w,B') : v_i$ a child of $v\}$,
- $max\{MCS(v,A'',w,B'')+MCS(v,A'-A'',w,B'-B'') : A''$ and B'' are non-empty proper subsets of A' and B', respectively. $\}$.

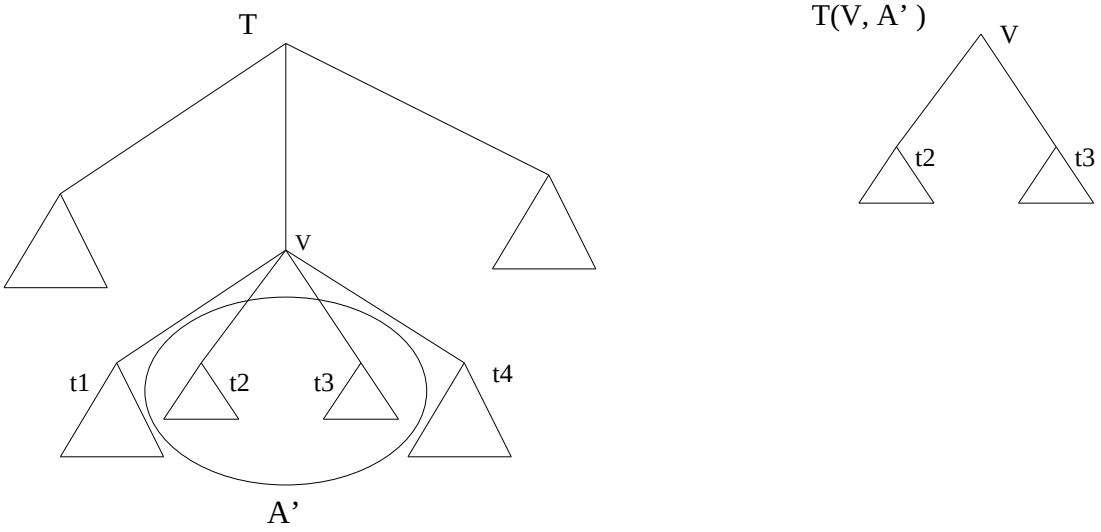


Figure 4.3: The tree T, the set A' and the tree T(v, A')

The computation of these subproblems follows the obvious partial ordering, in which $MCS(v,A,w,B)$ must follow $MCS(v',A',w',B')$ if both of the following conditions holds:

- v lies above v' or $[v=v'$ and $A'\subseteq A]$.

*The size of a tree is taken to mean the number of leaves in the tree.

- w lies above w' or [$w = w'$ and $B' \subseteq B$].

The base cases, in which v is a leaf or w is a leaf, are easy to compute, and equal 1 or 0. For example, if v is a leaf then necessarily $A = \emptyset$, and so $MCS(v, \emptyset, w, B) = 1$ if $v \in T'(w, B)$, and 0 otherwise.

Running time analysis There are $O(2^d n)$ trees $T(v, A)$, and hence $O(2^{2d} n^2)$ subproblems. The computation of $MCS(v, A, w, B)$ involves computing the maximum of $2d + 2^{2d}$ values, and hence takes $O(2^{2d})$ time. Hence the running time is $O(2^{4d} n^2)$.

4.1.2 Algorithm for the MCS problem of k rooted trees with bounded degree

We now show how to extend the analysis to k rooted trees. In this case, the subproblems are $2k$-tuples of the form $MCS(v_1, A_1, v_2, A_2, \ldots, v_k, A_k)$ where v_i is a node in T_i and $A_i \subset Children(v_i)$. Hence there are $O(2^{kd} n^k)$ subproblems. Computing each subproblem involves taking the maximum of $O(kd + 2^{kd})$ values. Hence the running time for the algorithm is $O(2^{2kd} n^k)$ time.

4.1.3 Extension to unrooted trees.

For each of the algorithms described, extending the algorithm to handle unrooted trees involves rooting each of the trees at each of the n leaves (for each leaf all the trees are rooted at that leaf), and then finding the best rooting. This is based on the observation that given a leaf l, the size of an MCS of the trees rooted at l is the maximum size of a compatible set for the unrooted trees that includes l (while rooting the trees at the leaf l, l itself must be excluded from the trees, and hence the size of a maximum compatible set for the unrooted trees that includes l will be actually one more than the size of an MCS of the trees rooted at l). Since there are n leaves, this multiplies the running time by n. Hence,

Theorem 16 *We can compute an MCS of k unrooted trees in which each tree has degree at most* $d+1$ *in* $O(2^{2kd}n^{k+1})$ *time.*

4.2 Approximation Algorithms

In this section we present approximate algorithms for the CMCS problem. The approximation algorithm is obtained by reducing the CMCS problem in an approximation preserving manner to the *Hitting Set* problem defined on a collection of subsets of a ground set. The hitting set problem is NP-hard. However, it is amenable to a q-approximation algorithm when the cardinality of all sets in the collection of subsets is q, and in this case the problem is called the q-hitting set problem. The best-known hitting set problem is *Vertex Cover*, which is the 2-hitting set problem. For more on hitting set and vertex cover problems, refer to [JG79].

4.2.1 Basics

In this section we present the basic definitions and lemmas that motivate a brute force algorithm that forms the basis for our $O(k^2n^2)$ 4-approximation algorithm for the CMCS problem. Our theory rests upon tying together notions of compatibility of trees, bipartitions and *quartet trees* which are trees on four leaves.

The following relationship between tree compatibility and bipartition compatibility is folklore.

Theorem 17 *A set* $\mathcal{T}$ *of trees is compatible if and only if the set* $\bigcup_{T\in\mathcal{T}} C(T)$ *is compatible.*

Hence, if we wish to determine if a set of trees is compatible, we need only determine whether the set of bipartitions of all the trees in the set is compatible. This can be determined in $O(kn)$ time [Gus97, War94], where $\mathcal{T}$ has k trees, each on the same n leaves.

Our 4-approximation algorithm operates by eliminating taxa from the set of trees which cause the set of bipartitions to be incompatible. Compatibility between bipartitions can be tested

by applying Lemma 10 from Chapter 2: a set of bipartitions is compatible iff any two bipartitions in the set are are pairwise compatible. Furthermore, two bipartitions $A = A_1 : A_2$ and $B = B_1 : B_2$ are compatible iff at least one of the four sets $A_1 \cap B_1$, $A_1 \cap B_2$, $A_2 \cap B_1$ and $A_2 \cap B_2$ is empty.

Definition 23 *(Quartets and Quartet Trees).*

A quartet *is a set of four leaves, and a* quartet tree *is an unrooted tree on four leaves that does not have any internal node of degree two. We denote the quartet tree induced by a tree T on a quartet q by $T|q$. We will call q the* basis *of the quartet tree $T|q$. Two quartet trees on the same quartet are said to be* incompatible *if both are binary trees and they differ.*

Corollary 4 *A set $\mathcal{T}$ of trees is incompatible if and only if there exists a quartet q of leaves and a pair of trees T_1 and T_2 in $\mathcal{T}$ such that $T_1|q$ and $T_2|q$ are incompatible quartet trees.*

Proof: If the set is incompatible, then there is a pair of bipartitions $A_1 : A_2$ and $B_1 : B_2$ (coming from trees T and T', respectively) and leaves a, b, c, d such that $a \in A_1 \cap B_1, b \in A_1 \cap B_2, c \in A_2 \cap B_1$, and $d \in A_2 \cap B_2$. Hence, T induces $ab|cd$ and T' induces $ac|bd$, so that $T|\{a,b,c,d\}$ and $T'|\{a,b,c,d\}$ are incompatible. The converse is trivial. □

We now introduce the Hitting Set problem.

Definition 24 *(Hitting Set).*

Let $\mathcal{X} \subseteq 2^A$ be a collection of subsets of a ground set A. A hitting set *for $\mathcal{X}$ is a subset $V \subset A$ such that for all $X \in \mathcal{X}$ $\exists a \in V$ such that $a \in X$.*

The best known version of the Hitting Set problem is the *Vertex Cover* problem: given a graph $G = (V, E)$, find a minimum subset $A \subset V$ so that every edge in E has at least one of its endpoints in A. This problem is NP-Hard but can be 2-approximated in a simple algorithm which greedily accumulates a *maximal* matching E_0 within E (so that no two edges within E_0 share an endpoint) [JG79]. Then the vertex set $A = \{v : \exists w \in V : (v, w) \in E_0\}$ is a vertex cover for G which

is at most twice the size of an optimal vertex cover. Thus, the NP-hard Vertex Cover problem can be 2-approximated in polynomial time.

More generally, the Hitting Set problem can be p-approximated in polynomial time using the same greedy technique, where $p = max\{|S| : S \in \mathcal{X}\}$.

Our algorithms are based upon a connection between the Hitting Set problem and the CMCS problem. We begin by defining the set Q_{inc}:

Definition 25 *(Q_{inc}): Let $\mathcal{T}$ be a set of trees leaf-labelled by the same set S, and let Q_{inc} denote the set of all quartets on which the trees in $\mathcal{T}$ are incompatible.*

The following lemma explains the connection between the Hitting Set problem and the CMCS problem.

Lemma 14 *Let $\mathcal{T}$ be a set of trees on a set of leaves S. Then, H is a hitting set for Q_{inc} if and only if $S - H$ is a compatible set for $\mathcal{T}$.*

Proof: Let H be a hitting set for Q_{inc}. Hence, for every $q \in Q_{inc}$, there is at least one $s \in H$ such that $s \in q$. Now consider any pair of trees T_1 and T_2, when restricted to $S - H$. By Corollary 4 they are compatible, since they have no quartets on which they are incompatible. Hence the set of trees $\mathcal{T}$ restricted to $S - H$ is compatible. For the converse, if $X \subseteq S$ is such that all trees in $\mathcal{T}$ are compatible when restricted to $S - X$, then there are no incompatible quartets left; hence X is a hitting set for Q_{inc}. □

Thus, for us Q_{inc} is the set $\mathcal{X}$ in the definition of the Hitting Set problem. Hence, as argued before, we can 4-approximate the CMCS problem in polynomial time:

Theorem 18 *Let Q be a maximal collection of pairwise disjoint quartets drawn from Q_{inc}. Let $S_Q = \bigcup_{q \in Q} q$. Then, $\mathcal{T}$ is compatible on $S - S_Q$, and $|S_Q| \leq 4|H^{opt}|$, where H^{opt} is an optimal solution to the CMCS problem on input $\mathcal{T}$.*

(The proof follows from the previous lemma.)

As in all Hitting Set problems, this suggests a straightforward algorithm to obtain a 4-approximation for the CMCS problem. That is, compute Q_{inc}, and then find a maximal collection of pairwise disjoint quartets within this set:

APPROX-HITTING-SET(Q_{inc})

$H \leftarrow \emptyset$

$\mathcal{P} \leftarrow Q_{inc}$

while $\mathcal{P} \neq \emptyset$

    pick an arbitrary set $q \in \mathcal{P}$

    $H \leftarrow H \cup q$

    remove every $X \in \mathcal{P}$ such that $q \cap X \neq \emptyset$

return H

The above procedure would give us a hitting set for Q_{inc} that is at most four times the size of an optimal hitting set. However, the above outlined procedure depends on explicitly enumerating all the incompatible quartets, and this is expensive. (There are $O(n^4)$ such quartets, and determining if a set of k trees is incompatible on a given quartet takes $O(nk)$ time.) This brute force approach for finding a 4-approximation to the hitting set problem is therefore not practical, as it uses $O(kn^5)$ time.

4.2.2 The Efficient 4-Approximation Algorithm

Outline of the Algorithm. Note that the input to the CMCS problem is a set of k unrooted trees, which *implicitly* codes for the set Q_{inc}. The approach we use for obtaining a faster 4-approximation algorithm takes advantage of this implicit representation of Q_{inc}, by never explicitly calculating Q_{inc}.

The key to an efficient algorithm is efficient identification of incompatible quartets. In our algorithm we do not generate all the incompatible quartets - rather, we identify incompatible quartets through (on the fly) identification of incompatible *bipartitions*. That is, we show that we can efficiently identify a pair of incompatible bipartitions, and from that pair of bipartitions we can then efficiently obtain an incompatible quartet, so that in $O(k^2n^2)$ time we have obtained a hitting set of size at most four times the optimal hitting set. This is the basic idea of our algorithm. Here we describe this basic idea just on two trees, noting that the algorithm for k trees operates by repeatedly examining pairs of trees, finding incompatible quartets, and deleting each of the leaves in the incompatible quartets from all the trees in the input.

MODIFIED-APPROX-HITTING-SET(T_1, T_2)

```
        /* T1 and T2 are two trees on the set of leaves S */
1       H ← ∅; S' ← S
2       Mark all edges in T1 as BAD
3       let e be the first BAD edge
4       while there are BAD edges
5               if e is compatible with all edges in T2
6                       mark e as GOOD
7                       e ← next BAD edge, if there is one
8               else
9                       find an edge e' in T2 that e and e' are incompatible
10                      let q be an incompatible basis as in Corollary 4
11                      S' ← S' − q
12                      H ← H ∪ q
13                      remove leaves in q from T1 and T2
                        (but do not restrict T1 and T2 to S')
14      return H
```

It can be seen easily that MODIFIED-APPROX-HITTING-SET is correct. Note that since the while loop in line 4 terminates only when there are no BAD edges, all the edges remaining in T_1 at the end are compatible with the edges in T_2. Hence, by Lemma 10 from Chapter 2, $T_1|(S-H)$ is compatible with $T_2|(S-H)$. Hence $S-H$ is indeed a compatible set for $\{T_1, T_2\}$, and thus, H is a hitting set for the collection of all incompatible bases for $\{T_1, T_2\}$. Moreover, since H is the union of *disjoint* bases (each of which is of cardinality four), $|H| \leq 4|H^{opt}|$, where $S-H^{opt}$ is an MCS for $\{T_1, T_2\}$.

Note that the while loop terminates after $O(n)$ iterations, since in each iteration it eliminates four leaves or marks an edge as GOOD and there are $O(n)$ edges in T_1. Hence, if we ensure that each iteration can be completed in $O(n)$ time, we would have an $O(n^2)$ time algorithm. In the next section we will describe how this running time can be achieved.

Achieving $O(n^2)$ running time for the CMCS of two trees

From the outline of the algorithm in the previous section, it can be observed that the key to achieving the $O(n^2)$ time bound is identifying an edge $e \in E(T_1)$ in $O(n)$ time such that e is either compatible with all the edges in T_2 or is incompatible with some edge in T_2. We will now show how the above objective can be achieved.

We modify and extend MODIFIED-APPROX-HITTING-SET thus:

1. Pick a taxon arbitrarily, and root both the trees on this taxon. Let the rooted trees be T_1' and T_2'.
2. For each node in T_1' maintain a sorted list of all leaves below it. We denote the set of leaves below node v as L_v. We maintain the sorted set as an indexed (doubly) linked list, to facilitate fast deletions. Such a data structure suffices since we will only delete from the list and never insert anything.

3. Let (v,w) be an edge in T_1' with w below v. For each node x in T_2', compute two quantities n_x and n_x' defined as follows: $n_x = |L_x \cap L_w|$ and $n_x' = |L_x \cap (S' - L_w)|$. Here, S' is the *current* set of leaves in T_1' and T_2'.

The following lemma allows us to determine if an edge in T_1 is compatible with all other edges in T_2 or not.

Lemma 15 *Let (v,w) be an edge in T_1 and let (y,x) be an edge in T_2. Without loss of generality assume that v is the parent of w in T_1' and that y is the parent of x in T_2'. Then (v,w) is incompatible with (y,x) iff $0 < n_x < |L_w|$ and $0 < n_x' < |S' - L_w|$.*

Proof: The proof follows from the following four observations.

1. $n_x > 0$ iff there exists a leaf $l \in L_x$ in T_2' that is in L_w in T_1'. In other words, if and only if $L_x \cap L_w \neq \emptyset$.
2. $n_x < |L_w|$ iff there exists a leaf $l \in L_w$ that is *not* in L_x. In other words, iff $L_w \cap (S' - L_x) \neq \emptyset$.
3. $n_x' > 0$ iff there is a leaf that is not in L_w but is in L_x. In other words, iff $(S' - L_w) \cap L_x \neq \emptyset$.
4. $n_x' < |S' - L_w|$ iff there is a leaf that is neither in L_x nor in L_w. In other words, iff $(S' - L_x) \cap (S' - L_w) \neq \emptyset$.

□

We can compute n_x and n_x' for all nodes x in T_2' in $O(n)$ time by doing the computations bottom up. We then inspect all edges in T_2' to see if they are compatible with the edge (v,w) (which is in T_1'). This can again be accomplished in $O(n)$ time since there are at most $O(n)$ edges. Hence, we can identify if an edge (v,w) is GOOD or not in $O(n)$ time. In case we determine it to be GOOD, we do no further processing. In case we find that the edge is incompatible with an edge (y,x) we have to do some further processing.

1. We compute a basis q by computing $L_w \cap L_x$, $L_w \cap (S' - L_x)$, $(S' - L_w) \cap L_x$ and $(S' - L_w) \cap (S' - L_x)$ and picking one element from each set. Each of the four intersection computations can be carried out in $O(n)$ time since we maintain the sets sorted.

2. We remove the leaves in q from T_1 and T_2 (but we do not eliminate nodes of degree two that may be created due to the removal of q) and hence we have to update the sorted set of leaves maintained in each node. A deletion from this set can be carried out in $O(1)$ time since the set is indexed and doubly linked. Hence, the time taken for the removal of q from the two trees is in $O(n)$.

The above discussion shows that both the *if* and *else* clause in the *while* loop in our algorithm can be accomplished in $O(n)$ time. We also noted that the loop terminates in $O(n)$ iterations. Hence the loop takes $O(n^2)$ time. The rest of the algorithm takes $O(n)$ time. Thus, the entire algorithm can be made to run in $O(n^2)$ time. The extension to k trees is straightforward, as we just greedily compute incompatible quartet trees, and delete the leaves from each of the trees in the input. Hence we have the following theorem.

Theorem 19 *Given a set $\mathcal{T}$ of k unrooted trees on a set of leaves S, let $S - H^{opt}$ be a maximum compatible set for $\mathcal{T}$. We can compute a compatible set $S - H$ such that $|H| \leq 4|H^{opt}|$ in $O(k^2n^2)$ time.*

4.2.3 A slower algorithm with a better approximation ratio

In this section we describe how to modify the previously described algorithm to achieve a 3-approximation ratio, albeit using more time. The algorithm operates by approximating the solution to the rooted version of the CMCS problem, but must apply it to n subproblems, where the i^{th} subproblem considers all the trees in $\mathcal{T}$ rooted at leaf i. By taking the minimum of all the solutions obtained, we can compute the overall solution with the desired approximation bound. (Later we give an example of two unrooted trees which shows why it does not suffice to just look at a single rooting.) Although we can show that each individual rooted problem can be solved again

in $O(k^2n^2)$ problem, because we look at n subproblems this approach is a factor of $O(n)$ slower than the 4-approximation algorithm we showed earlier. Hence, this is a slower algorithm, but it achieves a better guaranteed approximation ratio than our first algorithm.

We begin by making some definitions.

Definition 26 *(Triplet Tree): A* triplet tree *is a rooted tree on three leaves that does not have any internal node of degree two. We use the notation $T|q$ to denote the triplet tree induced by a rooted tree T on a set $q = \{a,b,c\}$ of three leaves. The star triplet tree (where all three leaves are children of the root) is denoted by (a,b,c), whereas $((a,b),c)$ (or its equivalent form, $(c,(a,b))$) denotes the triplet tree in which a and b are siblings, and have a least common ancestor below the root. We will call q the* basis *of the triplet tree $T|q$.*

Definition 27 *(Triplet Compatibility): Two triplet trees, each on the same leaf set, are said to be* compatible *if at least one of them is a star or they are the same. Otherwise they are said to be* incompatible.

As in unrooted trees, we can talk about the subtree of a rooted tree induced by a set of leaves. Here, however, we do not suppress a node of degree two if it is the root. Corresponding to the set Q_{inc} of incompatible quartet trees, we can define the set $Trip_{inc}$ of incompatible triplet trees:

Definition 28 *($Trip_{inc}$): Given a set $\mathcal{T}$ of rooted trees, each leaf-labelled by the same set S of leaves, we define the set $Trip_{inc} = \{\{a,b,c\} \subseteq S : \exists \{A,B\} \subseteq \mathcal{T} : A|\{a,b,c\}$ and $B|\{a,b,c\}$ induce incompatible triplet trees$\}$.*

Now let A be an unrooted tree on the set S of leaves $S = \{1,2,\ldots,n\}$, and let $i \in S$ be arbitrary.

Definition 29 *(A^i): A^i refers to the rooted tree on the leaf set $S - \{i\}$, obtained by rooting A at the leaf i, and deleting i and its incident edge.*

As before, we let L_v represent the leaves below the node v in a tree (whose identity will be known in context).

We now show through the following lemma that the incompatibility question for unrooted trees can be rephrased in terms of incompatibility of induced triplets, an idea crucial to improving the approximation ratio. We state the lemma in terms of two trees, but the result obviously extends to any number of trees.

Lemma 16 *Let A and B be two trees on leaf set S, and let $i \in S$ be a fixed leaf. Then A^i and B^i induce no incompatible triplet trees if and only if the unrooted trees A and B are compatible.*

Proof: First we prove that if the rooted pair of trees A^i and B^i have a pair of incompatible triplets, then A and B are incompatible. Suppose that the set of three leaves $\{a,b,c\}$ is an incompatible triplet, so that A^i induces the triplet tree $(a,(b,c))$ and B^i induces the triplet tree $((a,b),c)$. Then we can see that on $\{a,b,c,i\}$, tree A induces the split $\{a,i\}|\{b,c\}$ whereas tree B induces the split $\{a,b\}|\{c,i\}$. Hence, A and B are incompatible.

Suppose now that A and B are incompatible trees. Hence, there is a set $q = \{a,b,c,d\}$ on which the two trees induce two different splits. Recall that i is an arbitrary fixed leaf. We will show no matter how i is chosen, we can select three leaves from q so that A^i and B^i induce incompatible triplet trees for that subset. Let A induce $\{a,b\}|\{c,d\}$ and B' induce $\{a,c\}|\{b,d\}$ on q. We first address the case where $i \in \{a,b,c,d\}$. In this case, without loss of generality assume $i = d$. Then A^i $(= A^d)$ induces $((a,b),c)$ on leaf set a,b,c, whereas B^i $(= B^d)$ induces $((a,c),b)$. Hence, A^i and B^i induce incompatible triplet trees. Now we consider the case where $i \notin \{a,b,c,d\}$. Consider the restriction of the rooted trees A^i and B^i to the leaves a,b,c,d. In each of these rooted trees there is at least one sibling pair of leaves. Since A induces the quartet $ab|cd$, the only possible sibling pairs in $A^i|\{a,b,c,d\}$ are a,b and c,d. Assume, without loss of generality, that a,b are siblings in $A^i|\{a,b,c,d\}$. Similarly, the only possible sibling pairs in $B^i|\{a,b,c,d\}$ are a,c and b,d. Again, without loss of generality, assume a,c are siblings in $B^i|\{a,b,c,d\}$. Then on the set a,b,c of leaves

the rooted trees A^i and B^i differ: $A^i|\{a,b,c\} = ((a,b),c)$ whereas $B^i|\{a,b,c\} = ((a,c),b)$. Hence, a,b,c induces incompatible quartet trees in A^i and B^i. □

We now define the maximum compatible subset of rooted trees problem and its complement:

Definition 30 *Maximum Compatible Subset of Rooted Trees (MCSR)*

- **Input:** *Set $\mathcal{T}$ of rooted trees, each leaf-labelled by the same set L of leaves.*
- **Output:** *A maximum cardinality subset $L' \subseteq L$ so that the trees in $\mathcal{T}$ restricted to the leaves in L' do not induce any incompatible triplet trees.*

Definition 31 *Complement of the Maximum Compatible Subset of Rooted Trees (MCSR)*

- **Input:** *Set $\mathcal{T}$ of rooted trees, each leaf-labelled by the same set L of leaves.*
- **Output:** *A minimum cardinality subset $X \subseteq L$ so that the trees in $\mathcal{T}$ restricted to the leaves in $L - X$ do not induce any incompatible triplet trees.*

The next observation follows directly from Lemma 16, and we state it without proof:

Observation 4 *Let $\mathcal{T}$ be a set of unrooted trees leaf-labelled by the same set S, and assume that $|S| \geq 1$. Let $i \in S$ be arbitrary. For every subset $S_0 \subset S - \{i\}$, if S_0 is a feasible solution to the MCSR problem on set $\{T^i : T \in \mathcal{T}\}$ of trees, then $S_0 \cup \{i\}$ is a feasible solution to the MCS problem on $\mathcal{T}$.*

This suggests the following general strategy for the MCS problem:

- For each $i \in S$, let $L(i)$ denote the MCSR of the set of rooted trees $\{T^i : T \in \mathcal{T}\}$.
- Let i^* be selected so that $|L(i^*)|$ is maximum among $\{|L(i)| : 1 \leq i \leq n = |S|\}$. Return $L(i^*) \cup \{i^*\}$.

Comments:

(1) Observation 4 implies that this algorithm exactly solves the MCS problem, but that it requires n iterations of an exact solution to the MCSR problem. However, a little arithmetic and Observation 4 also imply that an r-approximation algorithm for the CMCSR problem would yield an r-approximation algorithm for the CMCS problem. Hence, although CMCSR is also NP-hard, if we can approximate CMCSR we can use that algorithm to approximate CMCS, and with the same guaranteed performance ratio. This is the approach we will take.

(2) Note also that we cannot simply pick a single leaf and root the trees at this leaf, and use the solution obtained for the resultant rooted trees. This approach can be arbitrarily bad. Consider the two caterpillar trees T and T' on leaf set $1, 2, 3, \ldots, 100, x$, as shown in Figure 4.4. These two trees clearly have a very large compatible subset: namely, the set $1, 2, \ldots, 100$ of leaves. However, if we root T and T' at leaf x, then they have very small compatible subsets as rooted trees. Similarly, arbitrary rootings of the trees T and T' will greatly affect the size of the compatible subset of the leaves that is obtained.

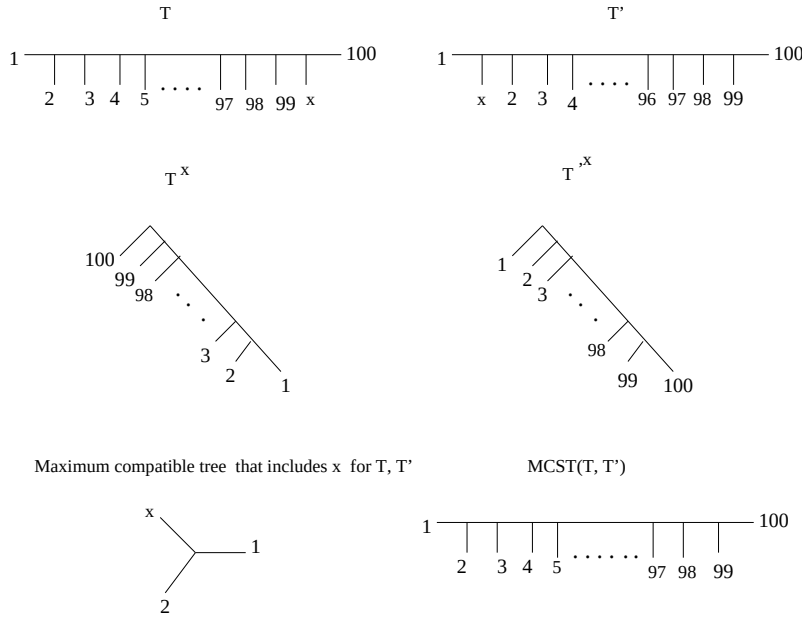


Figure 4.4: Why we cannot root trees arbitrarily.

Approximating the CMCSR problem: We now show how we can 3-approximate the complement of the MCSR problem. The technique is essentially the same as for the complement of the MCS problem, but rather than 4-approximating the Hitting Set for Q_{inc} we 3-approximate the Hitting Set for $Trip_{inc}$. Here, too, we can use the same techniques we used in the unrooted version (the CMCS problem) in order to get an $O(n^2)$ algorithm for the CMCSR of two trees. Since we do this for each possible leaf at which to root each of the trees, this gives us an $O(n^3)$ 3-approximation algorithm for the CMCSR problem on two trees, and hence, an $O(k^2n^3)$ 3-approximation algorithm for k trees.

We now show how we do this. Let A and B be two unrooted trees on leaf set L, and root each at leaf i thus producing A^i and B^i. Let $L' = L - \{i\}$. As before, let L_x denote the leaves below node x. The only variant is how we calculate incompatible triplets "on the fly". As before, we look for "bad" edges. Suppose we find a bad edge (v,w) in A^i that is incompatible with edge (x,y) in B^i. As before, we identify one leaf from each of the four sets $L_w \cap L_y$, $L_w \cap (L' - L_y)$ and $(L' - L_w) \cap L_y$ and $(L' - L_y) \cap (L' - L_w)$. Let these leaves be a, b, c and d respectively. Earlier, we eliminated all the four leaves, whereas now we eliminate only a, b and c from A^i and B^i leaving d untouched.

Observe that the set $\{a,b,c\}$ can not be part of any compatible set for the rooted trees A^i and B^i since when restricted to $\{a,b,c\}$, A^i induces $((a,b),c)$ and B^i induces $(a,(b,c))$. Hence, at least one leaf from $\{a,b,c\}$ has to be eliminated. Moreover observe that $\{a,b,c,i\}$ forms an incompatible quartet for the unrooted trees A and B. Hence, all incompatible induced triplets can be identified in the above manner. Hence we have the following theorem:

Theorem 20 *Given a set $\mathcal{T}$ of k unrooted trees on a set of leaves L, let $L - H^{opt}(\mathcal{T})$ be a maximum compatible set for the trees. We can compute a compatible set $L - H$ such that $|H| \leq 3|H^{opt}|$ in $O(k^2n^3)$ time.*

Chapter 5

Pattern-Identification in Biogeography

5.1 Introduction

Biogeography is the study of the geographic distribution of organisms [BL98, CKP03]. Biogeographers seek to understand ecological processes (e.g., climatic stability and effect of area) that influence the distribution of living organism over short periods of time, and to uncover events occurring in the distant past (e.g., continental drift, glaciation, and evolution) which have resulted in the geographic distribution observed today. Historical biogeography is the study of the geographic distribution of organisms in the light of their evolutionary history. One of the main tools of historical biogeographic inference is the comparison of phylogenetic trees of different groups of organisms that share their geographic distributions, in order to detect common patterns. However, until very recently comparisons have largely been made visually. In my work, I formalize the comparison and pattern identification problems, develop efficient algorithms to detect common patterns, and develop distance metrics so as to compare two patterns.

5.1.1 Historical Biogeography

One of the ways of understanding the geographic distribution of species is by studying the evolutionary history of the species, and this forms the basis for the discipline of historical biogeography [Bro81, CKP03, EO05, Jac04b].

Historical biogeographers generally assume that the current geographic distribution of organisms is a result of the following past events: (a) an event that splits an area into two or more distinct parts, known as geographic vicariance (b) extinction of species in an area, and (c) dispersal of organisms from one area to another. Historical biogeographical inference, then, aims to reconstruct these past events, and usually takes one of the following two forms:

- **Direct Inference.** In direct inference methods, a branching history of areas, called an *area cladogram*, is inferred from the phylogeny of organisms living in the areas. Brooks parsimony analysis (BPA) [Bro81], Assumptions 0, 1 and 2 [vVZK99] and Page's reconciliation maps [Pag94] are examples of this approach.

- **Indirect Inference.** Here, phylogenies of different groups of organisms which share their geographic distributions are compared. Common patterns observed in the different phylogenies are taken to be evidence of common past geological or climatic events that influenced the geographic distribution of species [Jac04b, LR05]. The contributions of our paper are towards indirect historical biogeographic inference. We formalize the notion of comparison of phylogenies of co-distributed groups of organisms, and develop algorithms and metrics in order to compare such phylogenies.

We now look at direct inference and indirect inference in detail.

Direct Inference

Brooks parsimony analysis and Assumptions 0, 1 and 2 take as input a phylogeny of the organisms whose geographic distribution is to be understood, and the geographic distribution of the organ-

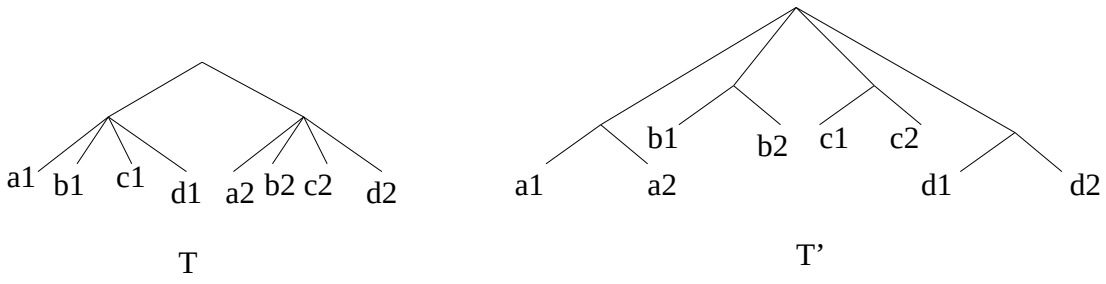


Figure 5.1: Two hypothetical phylogenies on eight taxa on four islands (a, b, c, d) with two ecological zones each (1 and 2).

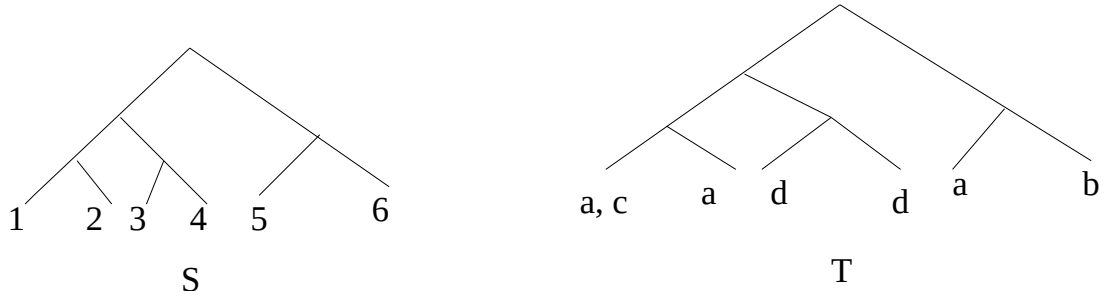


Figure 5.2: A phylogeny S and its associated area cladogram T, assuming taxon 1 appears in area c; 2 appears in area a; 3 appears in area d; 4 appears in area d; 5 appears in area a; and 6 appears in area b.

isms. Figure 5.1 depicts two hypothetical phylogenies and geographic distributions. The output of these methods is a *branching history of the areas* that support the organisms. As a first step, these methods construct a *general area cladogram* by replacing the taxon label of the leaf with the label of the area in which the taxon is found; see Figure 5.2. Note that some taxa may occur in more than one area (called *widespread taxa*) and there may be many taxa endemic to one area (called *redundant taxa*). This in turns translates to many leaves with the same area label or leaves with more than one area label in the general area cladogram. Hence, the area cladograms constructed as above do not, and cannot, represent a history of the areas. Consequently, the direct inference methods further process the general area cladograms to produce a branching history of areas where each leaf is labeled with a unique area, called *resolved area cladograms.*

Brooks Parsimony Analysis produces a resolved area cladogram as follows: each node, including

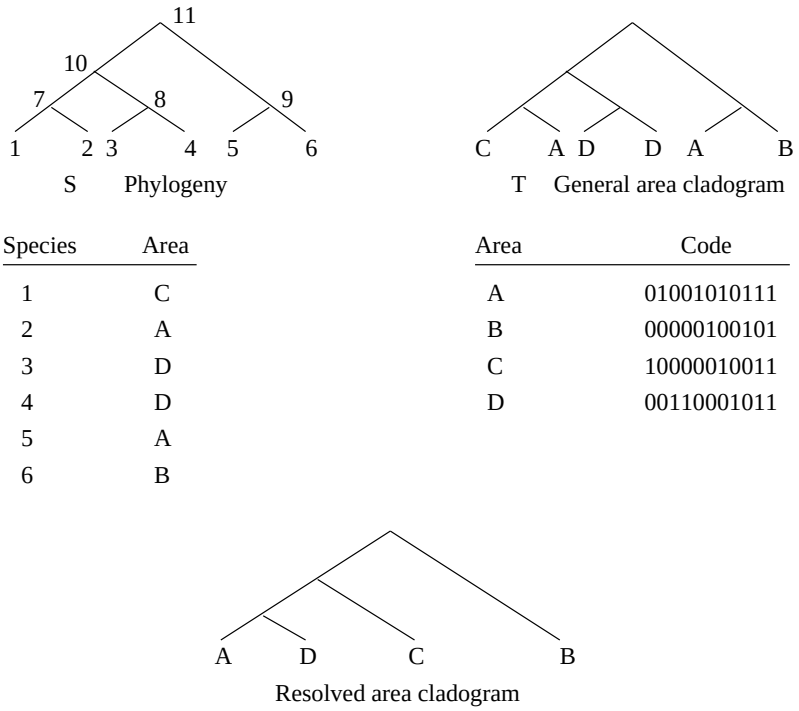


Figure 5.3: Brooks parsimony analysis: A general area cladogram T is derived from the phylogeny S and the geographic distribution. The areas are then coded as binary strings and subjected to a maximum parsimony analysis. The resolved area cladogram is a most parsimonious tree for the set of strings that encode the four areas.

the leaves, in the general area cladogram is given a number. In a general area cladogram with n leaves, there will be a total of $2n-1$ nodes. Each area is then represented as a binary string of length $2n-1$. The i^{th} bit of the string for an area p is 1 if node i is an ancestor of any occurrence of the area p in the general cladogram. This process is illustrated in Figure 5.3. The set of binary strings representing areas is then subjected to a *maximum parsimony analysis* to produce a resolved area cladogram. Thus, Brooks parsimony analysis reconstructs a purely vicariance based history of the areas.

Page's Reconciliation Maps combine the branching histories of two associated entities into one summary of historical association between the two entities. The associated entities can be hosts and parasites, organisms and genes, or, as in our case, areas and organisms that live in them. The need for reconciliation arises when the branching histories of the associated entities are incon-

gruent. The simplest hypothesis about the co-evolution of two associated entities (such as areas and organisms) is that a vicariance event in one entity corresponds to a speciation event in the other entity; incongruence arises when this hypothesis is violated. Reconciliation maps explain incongruence in terms of vicariance-independent speciation of organisms and extinct or uncollected species lineages; see Figure 5.4 (from [Pag94]) for an illustration of this. Reconciliation maps thus invoke vicariance and extinction in order to understand the geographic distribution of species.

Assumptions 0**,** 1 **and** 2 are again methods that produce resolved area cladograms from general area cladograms. In *A*0, vicariance is the only *a priori* hypothesis used to explain the geographic distribution of species. In *A*1 vicariance and extinction are the *a priori* hypotheses, and in *A*2, vicariance, extinction and dispersal are the *a priori* hypotheses. However, it is contentious how exactly these assumptions must be applied [vVZK99].

Indirect Inference

The fundamental idea behind indirect inference is that *a consistent pattern* observed in the phylogenies of species from different genera in the same geographic area will imply stronger evidence for the particular hypotheses suggested by the pattern. As an example of this approach, consider a group of islands, each containing multiple ecological zones (for example, each island can contain coastal and mountain ecological zones). Suppose our goal is to understand the observed geographic distribution of species on the islands. One hypothesis about the distribution, called *inter-island colonization*, is that species dispersed from each ecological zone in each island to similar zones in other islands and then differentiated. Another hypothesis, called *adaptive radiation* is that dispersal *between* islands happened first followed by dispersal to the different ecological zones and differentiation into many species [JEOH00]. The crucial idea is that we might be able to infer which of the above two hypotheses is responsible for the observed distribution: inter-island colonization is suggested by taxa on different islands but the same ecological zone forming a monophyletic group (i.e., a “clade”, or rooted subtree), and adaptive radiation is suggested if

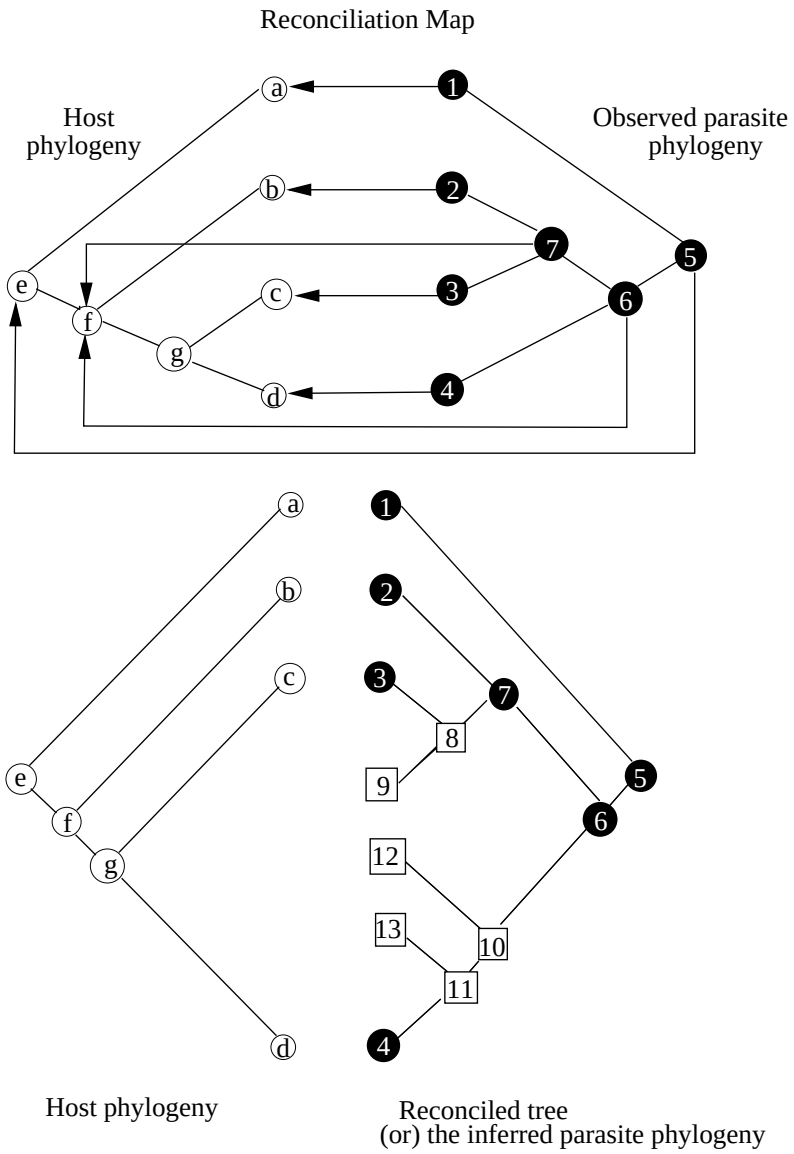


Figure 5.4: Reconciliation Maps: A host phylogeny, H, and a parasite phylogeny, P are incongruent. The parasites 1, 2, 3 and 4 live in hosts a, b, c and d respectively. The nodes 6 and 7 of the parasite phylogeny are both mapped to node f of the host phylogeny. This represents a speciation event of the parasite inside the host species f without an accompanying host speciation. The ancestors of the two resulting parasitic lineages are labeled 7 and 10 in the reconciled tree; Both the parasite lineages then follow the host lineage f's speciation pattern, leading to a parasite phylogeny shown as the reconciled tree. The dissimilarity between this inferred parasite phylogeny and the observed parasite phylogeny on top is explained by postulating the extinct or unsampled parasite lineages 9, 12 and 13.

species on the same island in different ecological zones form a monophyletic group. For example, in Figure 5.1, T suggests dispersal, while T' suggests adaptive radiation.

In practice, common patterns are identified between general area cladograms derived from the phylogenies of the different groups of co-distributed species. Until very recently, such common patterns have been identified by visual observation [Jac04b, Jac04a, CLW95]. Recently, Lapointe *et al.* in [LR05] identify common patterns among area cladograms by applying the *maximum agreement subtree (MAST)* method originally developed for phylogenies [FG85]. A maximum agreement subtree between two rooted trees is obtained by deleting a minimum number of leaves from either tree so that on the remaining set of leaves the trees are identical (i.e., isomorphic). However, before the the application of the MAST algorithm, the authors of [LR05] obtain area cladograms using a distant-based approach, as follows: (a) first, pairwise distances between areas are computed, where the distance between two areas $A1$ and $A2$ is the average distance between a species in $A1$ and a species in $A2$. The distance between species is the distance between sequence that represents the species. (b) Then, a neighbor-joining ([SN87]) tree for the areas is computed based on the calculated distances between the areas. The problem with this approach is the calculated distance between the areas does not capture all the evolutionary history of the species in the areas. Further, neighbor-joining is not the best method for obtaining phylogenies; most realistic phylogenies are computed using either maximum parsimony or maximum likelihood [SOWH96, Fel03].

In our work, we take a different approach to constructing and comparing area cladograms. We construct area cladograms from phylogenies by replacing the species labels of the leaves with the corresponding area labels. We then formulate the *maximum agreement area cladogram* (MAAC) problem analogous to the MAST problem for phylogenies. The MAAC problem is, in fact, a generalization of the MAST problem. We then show that the Steel-Warnow algorithm for MAST from [SW93] does in fact apply to the MAAC (maximum agreement area cladogram) problem as well without modification. However, in general care must be exercised before adapting any

MAST algorithm for the MAAC problem.

Comparing Area Cladograms. Apart from identifying common patterns among area cladograms, it is of interest to quantify the difference between an observed area cladogram and a hypothesized area cladogram. Earlier work on comparing area cladograms has included pruning the cladograms until the two cladograms agree on the remaining leaves (see [Ros78]), and using similarity metrics such as the bipartition metric (see Introduction, Chapter 1, 1.2.1) and the *triplets* metric between rooted area cladograms [Pag88]. However, all these methods apply only to resolved area cladograms. In this chapter, we develop distance metrics to compare general area cladograms.

5.1.2 Our Contributions

Our contributions are two-fold: we develop both metrics and algorithmic results for comparing area cladograms. More specifically,

- We show that the equivalence between the edge-contract-and-refine metric ("RF-distance") and the bipartition metric that holds for phylogenies *does not hold* for area cladograms. More specifically, we show that the bipartition metric, when extended to area cladograms, is not a metric. For the edge-contract-and-refine edit distance between two area cladograms we present a simple, but worst-case exponential-time algorithm. This edit distance can compare only area cladograms that are on the same number of leaves, and when each area labels the same number of leaves in both area cladograms (Section 5.2).

- We define another metric, the *MAAC* distance metric, for comparing two rooted area cladograms, which is based on the size of the largest common pruned subtree between the two area cladograms. The MAAC distance metric can compare two arbitrary trees that are not necessarily on the same number of leaves, which is particularly useful when comparing area cladograms (Section 5.2).

- We present a polynomial-time algorithm for computing a MAAC of two rooted area clado-

grams. The algorithm is a dynamic programming algorithm that runs in $O(n^{2.5}\log n)$ time, where n is the maximum number of leaves in either cladogram. We also describe a linear-time algorithm to decide if two area cladograms are identical (Section 5.3.2).

5.2 Distance Measures Between Area Cladograms

In this section, we develop distance metrics for the set of area cladograms. We first show that the bipartition distance between two different area cladograms can be zero, and hence the bipartition distance is not a metric on area cladograms, and in particular cannot be used as a test of isomorphism. While the bipartition metric for phylogenies does not extend to area cladograms, the contract-and-refine edit distance still defines a metric since it is an edit distance. We present an algorithm to compute the edge contract-and-refine edit distance between area cladograms. This algorithm is efficient if there are only a few occurrences of widespread taxa, but it is exponential-time in general. For phylogenies this edit distance is the Robinson-Foulds (RF) distance and it equals the bipartition distance (See the Introduction: Chapter 1, Section 1.2.1). The RF distance between two phylogenetic trees on n leaves can be computed efficiently, in $\Theta(n)$ time [Day85].

Finally in this section, we define the notion of a *Maximum Agreement Area Cladogram (MAAC)* of a collection of area cladograms, which is roughly a largest pruned subtree of all trees in the collection, and we propose a distance metric for comparing area cladograms based on the MAAC, and argue that this is a more appropriate metric for area cladograms than the contract-and-refine edit distance.

5.2.1 The Character Encoding Cannot Distinguish Between Area Cladograms

We first formally define an area cladogram:

Definition 32 *An area cladogram is a rooted or unrooted tree whose leaves are labeled with areas. Thus an area cladogram T is a triplet (t, A, M) where t is its unlabeled topology, A is the set of labels and M an onto map from the set of leaves of t to A.*

The map M can map many leaves to the same label, and may also map single leaves to many labels. It will not be always necessary in this paper to explicitly refer to the triplet (t, A, M) of an area cladogram T. The triplet will be left out of the notation where unnecessary.

We now define the *extended character encoding* of an area cladogram.

Definition 33 *Let T be an area cladogram. The multi-set $\{\pi_e : e \in E(T)\}$ is called the* extended character encoding *of T, and will be denoted by $C(T)$. Here π_e denotes the bipartition of the multi-set of leaf labels induced by the edge e.*

Contrary to our experience with phylogenetic trees where the mapping between leaves and labels is 1-1, it is possible for two area cladograms T_1 and T_2 to satisfy $C(T_1) = C(T_2)$ and yet not be isomorphic. We exhibit such a pair of trees in Figure 5.5.

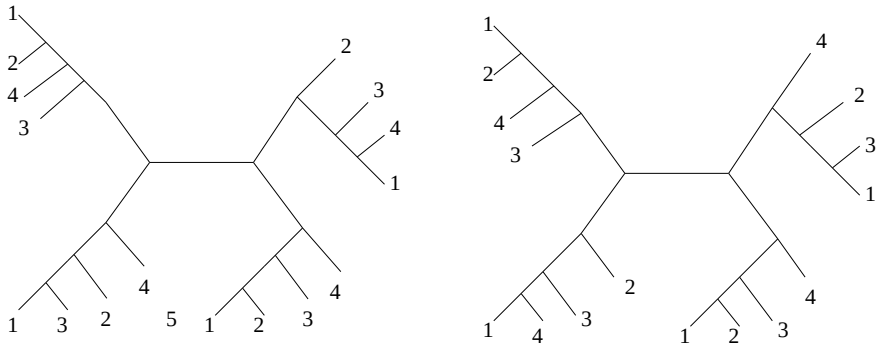


Figure 5.5: Two different binary area cladograms that induce the same multi-set of partitions

5.2.2 The Edge-Contract-and-Refine Distance Metric for Area Cladograms

Though the character-encoding distance fails to extend to area cladograms, the RF distance, being an edit distance, can be extended to unrooted area cladograms to provide a distance metric.

Definition 34 *Robinson-Foulds Distance Between Unrooted Area Cladograms*

The Robinson-Foulds distance *between two unrooted area cladograms T_1 and T_2 is defined to be the number of contractions and refinements necessary to transform T_1 to T_2 (or equivalently, T_2 to T_1).*

Handling Widespread Taxa. Taxa endemic (resident) to more than one area would result in cladograms with leaves labeled by many areas. Our definition of the Robinson-Foulds distance applies to such cladograms as well: if a leaf is labeled with a set of areas, we can consider that set of areas to be the unique label for that leaf. Thus, throughout the rest of this section, we will assume that each area cladogram leaf has just one area label.

Notation. We let n_1 and n_2 be the number of leaves in trees T_1 and T_2, respectively, and we let $n = max\{n_1, n_2\}$. We let $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ be the set of areas with which the leaves of T_1 and T_2 are labeled, and we let $\pi_{i,j}$, $j = 1,2$, be the number of leaves in tree T_j which are labeled with a_i; hence $\sum_{i=1}^{k} \pi_{i,j} = n_j$ for $j = 1,2$. Our analysis is parameterized on the numbers $\pi_{i,j}$.

Note that if $\pi_{i,1} \neq \pi_{i,2}$ for some i then there is no sequence of contractions and refinements that can transform T_1 into T_2; in such cases we define $RF(T_1, T_2) = \infty$. So throughout the rest of *this* section, we will assume that a given pair of cladograms T_j, T_k will have $\pi_{i,j} = \pi_{i,k}$ for all i and hence $n_j = n_k$. We therefore will set n to denote the number of leaves in each of the cladograms, and π_i to be the number of leaves labeled with area a_i.

As shown in Section 5.2.1, the RF distance may not be equal to the extended-character-encoding distance for area cladograms (see Definition 33). However, we can relate the RF distance between two area cladograms to the RF distance between two associated phylogenies, as we now show. We begin with some definitions.

Definition 35 *Full Differentiation of an Area Cladogram*

Let $T = (t, A, M)$ be an unrooted area cladogram, with the unrooted topology t, set of labels

A and the map M from the leaves of t to A. Then, a full differentiation *of T is a leaf-labeled tree* $T^* = (t, A^*, M^*)$ *such that* M^* *is one-one.*

In other words, T^* has the same topology as T, but has its leaves labeled uniquely. Therefore, $A \neq A^*$ is possible.

Definition 36 *Consistent Full Differentiations*

Let $T_1 = (t_1, A, M_1)$ *and* $T_2 = (t_2, A, M_2)$ *be two unrooted area cladograms with the same set A of leaf labels, and let* $T_1^* = (t_1, A^*, M_1^*)$ *and* $T_2^* = (t_2, A^*, M_2^*)$ *be full differentiations of* T_1 *and* T_2 *respectively.* T_1^* *and* T_2^* *are* consistent full differentiations *if, for each label* $l \in A$*, the set of labels assigned to leaves in* T_1^* *that were labelled l in* T_1 *is identical to the set of labels assigned to leaves in* T_2^* *that were labelled l in* T_2*. Mathematically, this is:* $\forall l \in A, \{M_1^*(x) : M_1(x) = l\} = \{M_2^*(x) : M_2(x) = l\}$.

Theorem 21 *Let* T_1 *and* T_2 *be two unrooted area cladograms. Then* $RF(T_1, T_2) = \min\{RF(T_1^*, T_2^*) : T_1^*$ *and* T_2^* *are mutually consistent full differentiations of* T_1 *and* T_2*, respectively*$\}$.

Proof: Let S_1 and S_2 be two mutually consistent full differentiations of T_1 and T_2 such that $RF(S_1, S_2)$ is minimum. We will show that $RF(T_1, T_2) = RF(S_1, S_2)$.

We first show that $RF(T_1, T_2) \leq RF(S_1, S_2)$ by induction on the RF distance between S_1 and S_2. We begin with a simple observation: if S_1 and S_2 are isomorphic, then T_1 and T_2 are also isomorphic. To see this, let g_i be the isomorphism from T_i to S_i, for $i = 1, 2$, and let f be the isomorphism between S_1 and S_2. We define f' from T_1 to T_2 implicitly by $g_2(f'(u)) = f(g_1(u))$. This mapping f' is an isomorphism, since S_1 and S_2 are consistent.

We now continue with our proof. Suppose $RF(S_1, S_2) = 1$, and assume without loss of generality that S_2 is obtained by contracting an edge (u, v) in S_1 to a single vertex w in S_2. Then, there is a mapping f between the vertices of S_1 and S_2 such that $f(u) = f(v) = w$, and for any pair of vertices x and y in S_1 such that $\{x, y\} \neq \{u, v\}$, x and y are adjacent if and only if $f(x)$ and

$f(y)$ are adjacent. The mapping f also preserves leaf labels. Hence, an analogous mapping f' can be defined between the vertices of T_1 and T_2 that preserves leaf-labels (this is possible because S_1 and S_2 are consistent). Hence, $RF(T_1,T_2) \leq 1$. Suppose that $RF(S_1,S_2) = k+1$. Then, there is a phylogeny S_3 such that $RF(S_1,S_3) = 1$ and $RF(S_3,S_2) = k$. Assume that it takes a contraction to convert S_1 to S_3 (the claim can be proven in a very similar manner when it takes a refinement). Then, it can be shown that there is an area cladogram T_3 such that S_3 is a full differentiation of T_3 consistent with S_1. Since S_1 and S_3 are consistent and S_1 and S_2 are consistent, S_2 and S_3 are consistent as full differentiations of T_2 and T_3. Hence, we can conclude by induction that $RF(T_1,T_3) \leq 1$ and $RF(T_3,T_2) \leq k$. Hence, we have $RF(T_1,T_2) \leq k+1$.

Similarly, there exist consistent full differentiations X_1 and X_2 of T_1 and T_2, respectively, such that $RF(X_1,X_2) \leq RF(T_1,T_2)$. Let $RF(T_1,T_2) = 1$, and assume without loss of generality that $T2$ is obtained from $T1$ by contracting an edge (u,v) to a single vertex w. Then, there exists a mapping g from T_1 to T_2 such that (a) $g(u) = g(v) = w$, (b) for any pair of vertices x and y in T_1 such that $\{x,y\} \neq \{u,v\}$, x and y are adjacent if and only if $f(x)$ and $f(y)$ are adjacent, and (c) g preserves leaf labels. Now, let X_1 be a full differentiation of T_1. Let X_2 be a full differentiation of T_2 such that the label of a leaf $z \in T_2$ is the label assigned to $g^{-1}(z)$ in the full differentiation X_1. Since g preserves leaf labels, X_1 and X_2 are consistent. Thus, X_1 and X_2 are two consistent full differentiations of T_1 and T_2 such that $RF(X_1,X_2) \leq 1$, Even when $RF(T_1,T_2) > 1$, it can be shown similarly that there exist consistent full differentiations X_1 and X_2 of T_1 and T_2 such that $RF(X_1,X_2) \leq RF(T_1,T_2)$. It follows that $RF(S_1,S_2) \leq RF(T_1,T_2)$ since we assumed that the S_1 and S_2 minimize the RF distance between two consistent full differentiations of T_1 and T_2. Hence, we have $RF(T_1,T_2) = RF(S_1,S_2)$, and this completes our proof. □

Note that the RF distance between two cladograms T_1 and T_2 is at most the RF distance between *any* consistent full differentiations of T_1 and T_2. Hence this provides a linear time method for obtaining an upper bound on the RF distance between two area cladograms T_1 and T_2: we first

compute two mutually consistent full differentiations, and then compute their RF distance. We can compute two mutually consistent full differentiations of two area cladograms in linear time, and since the second step also can be performed in linear time [Day85], this is a linear time algorithm. Similarly, by Theorem 21 we can compute the RF distance between two area cladograms T_1 and T_2 by computing the RF distance between all the possible consistent full differentiations of T_1 and T_2, and choosing the minimum.

Theorem 22 *Let T_1 and T_2 be two unrooted area cladograms on n leaves on the same set of areas. For each area a_i appearing at the leaves of T_1 and T_2, let both trees have π_i leaves labeled with area a_i. Then, the RF distance between T_1 and T_2 can be calculated in $\Theta(n\Pi_{i=1}^{k}(\pi_i)!)$ time.*

Proof: The number of different consistent full differentiations of A_1 and A_2 is $\Pi_{i=1}^{k}(\pi_i)!$. Each such differentiation can be obtained in $O(n)$ time. Computing the RF distance between two consistent full differentiations takes $\Theta(n)$ time [Day85]. □

5.2.3 The MAAC Distance Metric Between Area Cladograms

In this section we define the problem of computing the largest common pruned subtree of two rooted area cladograms and describe a distance metric based on the size of a largest common pruned subtree. We call a largest common pruned subtree a Maximum Agreement Area Cladogram (MAAC); thus the MAAC is analogous to the maximum agreement subtree (MAST) of two phylogenies.

Let T be an area cladogram on a set L of leaves. Recall from Chapter 1, Section 1.7 that the *restriction* of T to a set of leaves L' is the cladogram obtained by deleting leaves in the set $L - L'$ from T and then suppressing internal nodes of degree two (except the root, if there is one).

Definition 37 *Maximum Agreement Area Cladogram (MAAC) and MAAC distance*

Let $\{T_1, T_2, \ldots, T_k\}$ be a set of rooted area cladograms, with L_i the leaf set of tree T_i, for $i = 1, 2, \ldots, k$. Let $\lambda_1 \subseteq L_1$ through $\lambda_k \subseteq L_k$ be sets of leaves of maximum cardinality such that the respective restrictions of the trees $T_1, \ldots, T_k$ to the sets $\lambda_1 \ldots \lambda_k$ are all isomorphic, with the isomorphisms preserving leaf labels. A restriction of any tree T_i to such a subset of leaves λ_i is a maximum agreement area cladogram *(MAAC) for the cladograms T_1 through T_k. The size of the MAAC is defined to be the number of leaves in the maximum agreement area cladogram, and is denoted by $size_{maac}(T_1, T_2, \ldots, T_k)$.*

The above definition is illustrated in Figure 5.6. The MAAC distance *between two trees T_1 and T_2 is $d_M(T_1, T_2) = max(n_1, n_2) - size_{maac}(T_1, T_2)$, where n_1 and n_2 are the number of leaves in T_1 and T_2 respectively.*

The MAAC distance can be viewed as a generalization of the maximum agreement subtree metric for phylogenies [GKK94], which for two phylogenies on the same set of n labeled leaves was defined as $n - size_{mast}$ where $size_{mast}$ is the size of a maximum agreement subset of the two phylogenies.

Handling Widespread Taxa. For comparing cladograms using maximum agreement area cladograms, leaves labeled by more than one area can be treated thus: each leaf labeled by a group of areas can be split into many separate leaves (all having the same parent), each of which is labeled by a single unique area from the group of areas.

We now show that the MAAC distance defines a metric on the set of area cladograms.

Theorem 23 *The MAAC distance d_M is a metric on the set of all area cladograms.*

Proof: We begin with a simple observation about the MAAC distance. Let T_1 and T_2 be area cladograms. It is clear that $d_M(T_1, T_2) = 0$ if and only if T_1 and T_2 are isomorphic, and that $d_M(T_1, T_2) = d_M(T_1, T_2)$. Hence, all we need to do is to prove that d_M satisfies the triangle inequality.

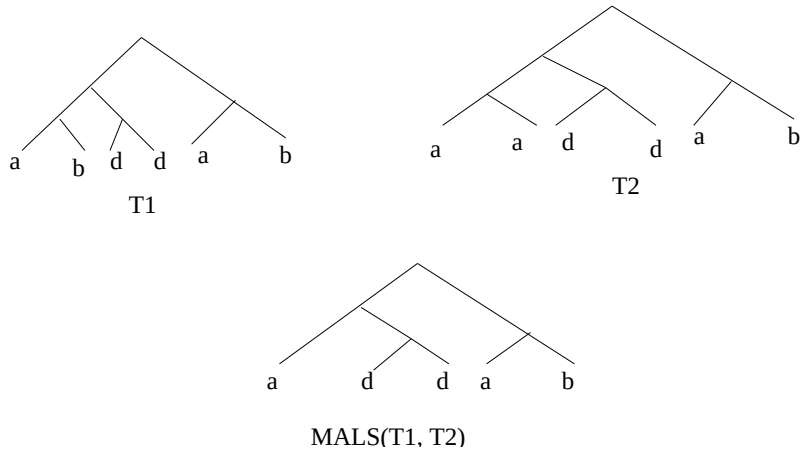


Figure 5.6: Two area cladograms T1 and T2, and their MAAC

So let $T_1, T_2,$ and T_3 be three area cladograms with $n_1, n_2,$ and n_3 leaves, respectively. We have to show that $d_M(T_1,T_2) + d_M(T_2,T_3) \geq d_M(T_1,T_3)$. We begin by defining some notation.

Let M_{ij} be the set of leaves in a MAAC of T_i and T_j and $m_{ij} = |M_{ij}|$. We also let $n_{ij} = max\{n_i, n_j\}$. Let $d_{ij} = d_M(T_i,T_j)$ (i.e., d_{ij} is the MAAC distance between T_i and T_j), so that $d_{ij} = n_{ij} - m_{ij}$. Let $m_{123} = |M_{12} \cap M_{23} \cap M_{13}|$, and let $m'_{ij} = m_{ij} - m_{123}$, so that $m_{ij} = m'_{ij} + m_{123}$.

We have:

$$\begin{aligned}
d_{12} + d_{23} &= \max(n_1,n_2) - m_{12} + \max(n_2,n_3) - m_{23} \\
&= \underline{\max(n_1,n_2) + \max(n_2,n_3)} - (\underline{m'_{12} + m_{123} + m'_{23}} + m_{123}) \\
&\geq \max(n_1,n_2,n_3) + n_2 - (n_2 + m_{123}) \\
&\geq \max(n_1,n_2,n_3) - m_{123} \\
&\geq \max(n_1,n_3) - m_{13} \;=\; d_{13}
\end{aligned}$$

□

Note that twice the MAAC distance between two cladograms is an upper bound on the

number of insertions and deletions of leaves necessary to transform one of the cladograms to the other.

An important feature of the MAAC definition is that *we do not require that all the trees in the given set contain the same number of leaves, or that they be labeled with the same set of areas, or even that they be consistent*. Thus the MAAC distance metric is a more versatile metric for area cladograms than the Robinson-Foulds distance. Further, as we show in the next section, the MAAC of two trees can be computed in polynomial time, in contrast to the result in Theorem 22 for the RF distance.

5.3 Algorithm for the Maximum Agreement Area Cladogram Problem

In this section we present an algorithm for the MAAC of two area cladograms. In Section 5.3.1 we present a basic dynamic programming algorithm, which is based on an algorithm for the MAST problem given in [SW93]. For the problem of determining if two area cladograms are isomorphic, we present a linear-time algorithm in Section 5.3.2.

In more recent work not included in this dissertation, we present a refined version of this algorithm that is more efficient when the number of leaves with any given label is not too large, and show that the problem of computing the MAAC of k trees if NP-hard, even if all trees are binary [GGJ+06].

5.3.1 Basic Dynamic Programming Algorithm for MAAC

In this section, we describe an algorithm for computing a MAAC of two given rooted area cladograms. This is a dynamic programming algorithm and is an adaptation to MAAC of the first polynomial-time algorithm for the phylogenetic rooted MAST algorithm presented by Steel and

Warnow [SW93], described in the Introduction, Chapter 1, Section 1.7. The algorithm is essentially the same as the Steel-Warnow algorithm, but the reason why the algorithm works for MAAC is different from why it works for MAST. Hence, we first describe the recursive structure of the MAAC solutions here, pointing out along the way, the adjustment that needs to made in order to argue that the Steel-Warnow algorithm is a correct MAAC algorithm. We will then present the MAAC algorithm in pseudocode, and analyze its running time.

The Basic Recursion in MAAC. In our description, we let $MAAC(T, T')$ denote a maximum agreement cladogram of the leaves of T and T'. We describe the algorithm for the case where T and T' are binary; extending this to the case where T and T' are not binary is straightforward.

Let T and T' be two given binary rooted area cladograms. Let v be a node in T, and denote by T_v the subtree of T rooted at v. Similarly denote by T'_w the subtree of T' rooted at a node w in T'. Let v_1 and v_2 be the two children of node v, and let w_1 and w_2 be the two children of w. The dynamic programming algorithm for MAAC operates by computing $MAAC(T_v, T'_w)$ for all pairs of nodes (v, w) in $V(T) \times V(T')$ "bottom-up". We now show how to reduce computing $MAAC(T_v, T'_w)$ to computing a small number of smaller MAAC computations $MAAC(S, S')$ where S and S' are subtrees of T_v and T'_w respectively, with at least one of them being a proper subtree.

To begin with, $MAAC(T_v, T'_w)$ is easy to compute when either v or w is a leaf. Therefore, in the following discussion, we assume neither v nor w is a leaf.

Let T^* be a MAAC of T_v and T'_w. Then there exist homeomorphisms mapping T^* to a rooted subtree of T_v and to a rooted subtree of T'_w. In fact, because T and T' may contain more than one leaf with the same label, T^* might be homeomorphically mapped to more than one rooted subtree of T_v and T'_w; however, this cannot happen if there is only one leaf with any given label.

Let p be the (not necessarily proper) *farthest* descendant of v such that the root of T^* is mapped to p. Similarly let q be the farthest descendant of w in T' such that that the root of T^* is mapped to w. Then, $MAAC(T_v, T'_w)$ is in fact equal to $MAAC(T_p, T'_q)$. In general, there may be many descendants x of v and y of w such that the root of T^* is mapped to x and y, since T and T'

are area cladograms and not phylogenies. Letting p and q be the farthest descendants is the crucial adjustment required to prove the correctness of the Steel-Warnow MAST algorithm from [SW93] as a MAAC algorithm.

The vertex p may be actually v or it might be a descendant of v. Similarly q may be w or some descendant of w. Based on the location of p and q, we have the following cases.

1. *Vertex p is a proper descendent of v.* In this case, T_p is a proper subtree of T_v, and $MAAC(T_v, T'_w)$ equals $MAAC(T_p, T'_w)$. Since p is a proper descendant of v, $MAAC(T_p, T'_w)$ either equals $MAAC(T_{v_1}, T'_w)$ or $MAAC(T_{v_2}, T'_w)$.

2. *Vertex q is a proper descendent of w.* In this case, $MAAC(T_v, T'_w)$ equals $MAAC(T_v, T'_q)$. Since q is a proper descendant of w, $MAAC(T_v, T'_q)$ either equals $MAAC(T_v, T'_{w_1})$ or $MAAC(T_v, T'_{w_2})$.

3. *Vertex p equals v and vertex q equals w.* Let T_1^* and T_2^* be the subtrees of the root of the MAAC T^*. Then, T_1^* is homeomorphic to a subtree of T_{v_1} (or to a subtree of T_{v_2}; there is no loss of generality in assuming that it is homeomorphic to a subtree of T_{v_1}). Similarly, T_2^* is homeomorphic to a subtree of T_{v_2}. It cannot be homeomorphic to a subtree of T_{v_1}, since then T^* would be homeomorphic to a subtree of T_{v_1}, contradicting the assumption that there is no proper descendent x of v such that root of T^* is mapped to x. Note that the assumption that p is the farthest descendent of v such that the root of T^* is mapped to p is crucial here. Arguing similarly, we can conclude that T_1^* and T_2^* are homeomorphic to subtrees of T'_{w_1} and T'_{w_2} respectively. Now, since T^* is a MAAC, we can conclude that T_1^* is a MAAC of T_{v_1} and T'_{w_1}, and that T_2^* is a MAAC of T_{v_2} and T'_{w_2}. So in this case we have reduced computing $MAAC(T_v, T'_w)$ to computing $MAAC(T_{v_1}, T'_{w_1})$ and $MAAC(T_{v_2}, T'_{w_2})$ and then taking their union.

The above discussion suggests a straightforward dynamic programming algorithm: we do not know which of the above three cases is true, but we do know that one of them is true. Hence we solve the subproblems corresponding to all the three cases and then choose the largest solution.

Note that the algorithm described above is the same as the MAST algorithm from [SW93], but the reason it is correct for MAAC is somewhat different from the reason it is correct for MAST.

We now describe this MAAC algorithm in pseudocode, but before we do so, we introduce some notation.

Notation For a node v in T_1 or T_2, let $c(v)$ denote the set of children of v, and let $A(v)$ denote the set of all labels of leaves that descend from v. For each pair of nodes $v \in T_1$ and $w \in T_2$, we let $G_{v,w}$ be a weighted complete bipartite graph with bipartition $(c(v), c(w))$ where the weight of the edge $(x,y) \in G_{v,w}$ is the number of leaves in $MAAC(T_x, T_y)$. We denote by $MWBM(G_{v,w})$ the maximum weighted bipartite matching of $G_{v,w}$. We let $V(T)$ be the set of all nodes of the tree T. In the pseudocode, the subroutine DIAG corresponds to the first two cases in our discussion of the MAST dynamic program, and the subroutine MATCH corresponds to the third case.

MATCH(v,w)

1 Construct $G_{v,w}$

2 Construct $E_0 = \text{MWBM}(G_{v,w})$

3 Let $E_0 = \{(v_1,w_1),(v_2,w_2),\ldots,(v_k,w_k)\}$

4 Construct tree M with root s such that $\text{MAAC}(T_{v_i}, T_{w_i})$ is the i^{th} child of s

5 **return** M

DIAG(v,w)

1 $t_1 \leftarrow$ largest $\text{MAAC}(T_v, T_x)$ such that $x \in c(w)$

2 $t_2 \leftarrow$ largest $\text{MAAC}(T_y, T_w)$ such that $y \in c(v)$

3 **return** the larger of t_1 and t_2

ALGORITHM $\text{MAAC}(T_1, T_2)$

1 Let O be an ordering of $V(T_1) \times V(T_2)$

2 such that if (v_1, w_1) is before (v_2, w_2),

3 then v_1 is not an ancestor of v_2 and w_1 is not an ancestor of w_2.

4 **for** (v, w) in increasing order of O

5 **do if** v or w is a leaf

6 **then** $\text{MAAC}(T_v, T_w) \leftarrow$ a node with label in $S = A(v) \cap A(w)$ if $S \neq \emptyset$; else $\emptyset$

7 **else** $\text{MAAC}(T_v, T_w) \leftarrow$ larger of $\text{MATCH}(v, w)$ and $\text{DIAG}(v, w)$

8 **return** $\text{MAAC}(T_{r_1}, T_{r_2})$; r_1 is the root of T_1 and r_2 is the root of T_2.

The Running Time of the MAAC Algorithm. The running time is the same as that of the MAST algorithm. Thus, the MAAC algorithm runs in $O(n^2 d^{2.5} \log d)$ time for trees on n leaves with degrees bounded by d, and in $O(n^{2.5} \log n)$ time in general. Recall that a running-time analysis of the Steel-Warnow MAST algorithm was presented in Chapter 4, Section 4.1.

5.3.2 Testing Isomorphism Between Two Rooted Area Cladograms

The MAAC distance metric between area cladograms gives us a polynomial-time algorithm for testing isomorphism: we apply the maximum agreement area cladogram algorithm from the previous section to compute the MAAC distance between the two area cladograms, and we conclude that the two cladograms are isomorphic if and only if the distance is zero. However, we can do better: we present a fast algorithm for testing isomorphism between area cladograms without computing the MAAC distance between the cladograms. The algorithm is adapted from the algorithm for testing rooted tree isomorphism from [AHU74].

The input to the algorithm consists of two rooted area cladograms T_1 and T_2 on n leaves (if the number of leaves is different, then clearly they are not isomorphic). We assume that the leaves are labeled with integers from 1 through n, not all distinct. The algorithm is based on assigning an integer $index(u)$ to each node u in the tree. When the node u is a leaf, the index is just its label. The algorithm is as follows:

1. Compute the *height*, the maximum distance between the root and a leaf, of the two trees. If the heights are not the same, then the trees are not isomorphic, otherwise, let the height be h.

2. Based on the height, assign level numbers to the nodes of the trees. The level number of a node at a distance of d from the root is set to be $h-d$.

3. For each leaf u at level 0, set $index[u]$ to be the leaf-label.

4. For each level i, in order, we compute $index[v]$ for each node v at level i as follows.

   - We define an ordered list of the indices of the children of the node v, sorted in ascending order. If v is a leaf, then its tuple consists of just its label. Let L_i be the list of tuples of nodes at level i in T_1. Let L_i' be the corresponding list for T_2. Now lexicographically sort L_i and L_i' to obtain S_i and S_i' respectively.
   - Set $index[v]$ to be the *rank* of v's tuple in the sorted list S_i. The ranks start from 1, and all identical tuples receive the same rank. Indices for vertices in T_2 are assigned similarly.
   - If S_i and S_i' are not identical, then declare T_1 and T_2 to be non-isomorphic and quit.

5. If the roots of T_1 and T_2 are assigned the same index, then the trees are isomorphic, otherwise not.

Proof of Correctness: We first show that if the algorithm declares two trees to be isomorphic, then they are indeed so. The proof is by induction on the number of levels in the trees. Suppose there is only one level, then the trees have only one leaf each. If the algorithm declares the trees to be isomorphic, then the leaves have the same label, and hence they are indeed isomorphic. Inductively assume that the algorithm correctly tests isomorphism of trees that have up to k levels. Suppose T_1 and T_2 have $k+1$ levels each. If the algorithm declares T_1 and T_2 to be isomorphic, then the tuples assigned to the roots of T_1 and T_2 are identical, which means that for each node of T_1 at level k,

there is a node at level k in T_2 that is assigned the same index, and vice versa. From the induction hypothesis, subtrees at level k that are assigned identical indices are isomorphic. Hence, for each subtree of T_1 at level k, there is a level k subtree isomorphic to it in T_2, and vice versa. This implies that T_1 and T_2 are isomorphic themselves. It can be proved similarly by induction that if T_1 and T_2 are isomorphic, the algorithm declares them so. This completes our proof.

Running Time. The running time of the above algorithm for testing isomorphism is $O(n)$, where n is the number of leaves in the input trees (see [AHU74]).

Chapter 6

Conclusion

Reconstructing the phylogenetic tree of all life on earth is one of the grandest challenges in all of biology [ToL06]. However, the historical evidence in the form of fossil records with which to reconstruct this evolutionary tree is scant and incomplete [Fel03, ST04, Dzi05]. This has motivated molecular phylogenetic estimation, where we attempt to infer past evolutionary history solely from currently observed variations in biomolecular sequences like DNA and RNA [Fel03]. Molecular data is becoming increasingly abundant, but molecular phylogenetic estimation still faces many conceptual and computational difficulties, and addressing some of these difficulties was the focus of this dissertation.

We first addressed the problem of efficient phylogenetic estimation under the maximum parsimony or the maximum likelihood criteria, and efficient Bayesian phylogenetic estimation. Maximum parsimony and maximum likelihood phylogenetic estimations are NP-hard problems [Fel03, CT05a, CT05b], and local-search heuristics are currently the most successful methods in MP and ML phylogenetic inference. In this dissertation, we proposed and analyzed a new local move, the p-Edge-Contract-and-Refine (p-ECR) move, and showed that the move has many promising theoretical properties that suggest that it can combine well with the currently used TBR move in local searches. We also developed efficient algorithms for generating a random p-ECR

neighbor of a tree, and for evaluating all 2-ECR neighbors under the maximum parsimony criterion. These algorithms will allow the *p*-ECR move to be incorporated efficiently in local search heuristics. The 2-ECR move has been implemented by David Swofford in a version of PAUP yet to be released in public. Further work is necessary to implement *p*-ECR in newer ML software like RAxML, PHYML and GARLI and evaluate its performance. We also investigated the performance of some current ML heuristics on real and simulated datasets. Our results show that much progress has been made in ML heuristic optimization: the newer software PHYML, RAxML and GARLI obtain trees with higher ML scores in a shorter amount of time than the traditional ML heuristic PAUP, or the heuristics MrBayes and the ratchet which were considered the best a few years ago. However, much progress remains to be made: our experiments with real datasets show that tree topologies with higher ML scores obtained by the newer ML heuristics are statistically indistinguishable from tree topologies obtained by the older heuristics such as PAUP, and that there is a high topological difference between trees that are currently statistically indistinguishable. Our experiments with simulated datasets show that two of the newer ML methods, PHYML and GARLI, return trees with high topological error even when they are allowed to optimize under the correct model of evolution. These observations, reported in Chapter 3, suggest that either ML is not a desirable method of phylogenetic inference, or that the current ML heuristics, in spite of their improvements, are still solving ML inadequately. In either case, our results strongly indicate that much work remains to be done.

A problem with phylogenetic inference based on an optimization criterion like MP or ML is that there may not be one unique optimal phylogenetic tree. Biologists increasingly want to estimate phylogenies of thousands of species, and with such large datasets, it is almost certain that there will be many trees with identical or near-identical optimal scores, be it MP or ML. We are faced with a similar problem with Bayesian phylogenetic analysis, where even from the outset the goal of the analysis is to estimate a probability distribution on the set of all trees, and not just one tree. Thus, the question of how to interpret and summarize the results of a phylogenetic analysis

is very important, and in this dissertation, we addressed the problem of building consensus trees to succinctly summarize information in a large collection of trees that may result from a phylogenetic analysis. We developed algorithms for the maximum compatible subset (MCT) consensus method, and approximation algorithms for computing a complement of the maximum compatible subset. The need to build informative consensus trees in the presence of rogue taxa and short edges motivated the MCT method. Further research is necessary to implement and experimentally validate the MCT method by comparing it with other consensus methods like the strict consensus, the majority consensus and the maximum agreement subset.

Building consensus trees is also important in the context of historical biogeographical inference, where identifying patterns common to phylogenies of organisms that share their geographic distribution is an important tool in inferring the evolution of the geographic range of the species along with their biological evolution. In this dissertation, we defined the maximum agreement area cladogram (MAAC) method to identify such common patterns, and developed an algorithm to compute a MAAC of two area cladograms, and to decide if two area cladograms are identical.

Phylogeny is an area of study that is at once both fundamental and practical. Understanding the evolutionary history of life on earth is a fundamental scientific problem, and phylogenetic analyses are also becoming central to some of our most pressing practical concerns: namely, the conservation of biodiversity, and the tracking of the evolution of contagious diseases like influenza ([PGB05, Erw91, MF98, BCS$^+$99]. Thus, there is no need to overstate the importance of doing phylogenetic analyses well. However, phylogenetic inference still faces many conceptual difficulties, from assembling the the input data to the interpretation of the final results:

- The input to a molecular phylogenetic analysis is a set of aligned biomolecular sequences. However, aligning a set of sequences, called multiple sequence alignment, is itself far from being a solved problem. The essential problem is this: when aligning a set of sequences into a matrix, we are asserting that all the nucleotides in a column evolved from the same ancestor. Thus, some knowledge of the phylogenetic history of the sequences is required in

order to align the sequences and infer their history. In order to overcome this chicken-and-egg problem, multiple sequence alignment and phylogenetic inference must be ideally performed simultaneously. However, until recently the multiple sequence alignment algorithms either operated without the knowledge of the phylogeny, or used crudely approximate phylogenies [DKM98]. Recently, there have been efforts to perform multiple sequence alignment and phylogenetic inference simultaneously, in the form of the generalized tree alignment approach ([Gir99, Knu03, PAD00]), but a lot of further research is necessary in order to construct reliable multiple sequence alignments quickly.

- Even assuming an accurate multiple sequence alignment, reconstructing a phylogeny for the alignment faces a lot of uncertainties: the model of evolution assumed by a reconstruction method may not be correct, and even if the model is correct, phylogenetic reconstruction is computationally intensive, and solving problems like MP and ML exactly may not be feasible. The GTR+G+I model of evolution, currently the most popular model makes several crucial assumptions such as: (a) different sites evolve independently and identically, (b) the parameters of the model remain constant across time, and across different parts of the evolutionary tree. In reality, these assumptions may be violated [GY94, GY06, HH93]. Thus, there is much scope for improving phylogenetic inference by designing better models of evolution.

- There has been a growing interest in *supertree* methods as a way to reconstruct large-scale phylogenies. Supertree phylogenies are constructed by combining phylogenies of possibly overlapping subsets of taxa, [BE04a], and are motivated when the datasets used to construct the phylogenies on the subsets of taxa are disparate. For this reason, supertrees may play a big role in assembling the entire tree of life. However, there has been much debate within the systematics community about supertree approaches [BE04b], such as: (a) how do supertree approaches compare against the *supermatrix* approaches where the different datasets on the

subsets of taxa are combined into one big dataset for the whole group? and (b) does a supertree represent the evolutionary history of the whole group of taxa, or is it just a summary of the individual input trees on the subsets of the whole group of taxa? Much work is necessary in order to resolve these issues satisfactorily.

- Evolution is not always tree-like. Events such as hybridization and transfer of genetic material from one species to another means that the evolutionary history of life is likely to be *reticulated* [NTWJ05], and reconstructing reticulate evolution poses its own problems.

Thus, it is clear that the open problems in phylogeny are far more numerous and difficult than the solved problems, and we hope for significant progress in solving these problems in the future.

Bibliography

[ACJ03] N. Amenta, F. Clark, and K. St. John. A Linear-Time Majority Tree. In *Proceedings of the Third International Workshop on Algorithms in Bioinformatics (WABI)*, pages 216–227, 2003.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, 1974.

[AK97] A. Amir and D. Keselman. Maximum Agreement Subtrees in a Set of Evolutionary Trees: Metrics and Efficient Algorithms. *SIAM Journal of Computing*, 26(6):1656–1669, 1997.

[Ald95] D. Aldous. *Random Discrete Structures*, chapter entitled "Probablity Distributions on Cladograms", pages 1–18. Springer, 1995.

[Ald01] D. Aldous. Stochastic Models and Descriptive Statistics for Phylogenetic Trees: from Yule to Today. *Statistical Science*, 16:23–34, 2001.

[AS01] B. Allen and M. Steel. Subtree Transfer Operations and Their Induced metrics on Evolutionary Trees. *Annals of Combinatorics*, 5:1–15, 2001.

[BCS+99] R. M. Bush, C. A. Caprara, K. Subbarao, N. J. Cox, and W. M. Fitch. Predicting the Evolution of Human Influenza Virus A. *Science*, 286:1921–1925, 1999.

[BE04a] O. R. P. Bininda-Emonds. *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[BE04b] O. R. P. Bininda-Emonds. The Evolution of Supertrees. *Trends in Ecology and Evolution*, 19(6):315–322, 2004.

[BHD+02] M. J. Brauer, M. T. Holder, L. A. Dries, D. J. Zwickl, P. O. Lewis, and D. M. Hillis. Genetic Algorithms and Parallel Processing in Maximum-Likelihood Inference. *Molecular Biology and Evolution*, 19(10):1717–1726, 2002.

[BL98] J. H. Brown and M. V. Lomolino. *Biogeography*. Sinauer Associates, Sunderland, Massachusetts, second edition, 1998.

[BNB96] R. Bartoszynski and M. Niewiadomska-Bugaj. *Probability and Statistical Inference*. John Wiley & Sons, Inc., 1996.

[Bro81] D. R. Brooks. Hennig's Parasitological Method: A Proposed Solution. *Systematic Zoology*, 30:229–249, 1981.

[BS04] M. Bordewich and C. Semple. On The Computational Complexity of the Rooted Subtree Prune and Regraft Distance. *Annals of Combinatorics*, 8:409–423, 2004.

[BSWY98] M. Bonet, M. Steel, T. Warnow, and S. Yooseph. Better Methods for Solving Parsimony and Compatibility. *Journal of Computational Biology*, 5(3):409–422, 1998.

[Bun71] P. Buneman. The Recovery of Trees from Measures of Dissimilarity. *Mathematics in the Archaelogical and Historical Sciences*, pages 387–395, 1971.

[BW01] J. J. Bull and H. A. Wichman. Applied Evolution. *Annual Review of Ecology and Systematics*, 32:183–217, 2001.

[Cha96a] J. Chang. Full Reconstruction of Markov Models on Evolutionary Trees: Identifiability and Consistency. *Mathematical Biosciences*, 137:51–73, 1996.

[Cha96b] J. Chang. Inconsistency of Evolutionary Tree Topology Reconstruction Methods When Substitution Rates Vary Across Characters. *Mathematical Biosciences*, 134:189–215, 1996.

[CKP03] J. V. Crisci, L. Katinas, and P. Posadas. *Historical Biogeography: An Introduction*. Harvard University Press, Cambridge, Massachusetts, 2003.

[CLW95] M. Crisp, H. P. Linder, and P. Weston. Cladistic Biogeography of Plants in Australia and New Guinea: Congruent Pattern Reveals Two Endemic Tropical Tracts. *Systematic Biology*, 44(4):457–473, 1995.

[CSO+93] M. Chase, D. Soltis, R. Olmstead, D. Morgan, D. Les, B. Mishler, M. Duvall, R. Price, H. Hillis, Y.-L. Qiu, A. Cron, J Retig, E. Conti, J. Palmer, J. Manhart, K. Systma, H. Michaels, W. Kress, K. Karol, W. Clark, M. Hedrn, B. Gaut, R. Jansen, K.-J. Kim, C. Wimpe, J. Smith, G. Furnier, S. Strauss, Q.-Y. Xiang, G. Plunkett, P. Soltis, S. Swenson, S. Williams, P. Gadek, C. Quinn, L. Eguiarte, E. Golenberg, G. Jr, S. Graham, S. Barrett, S. Dayanandan, and V. Albert. Phylogenetics of Seed plants: An Analysis of Nucleotide Sequences from the Plastid Gene *rbcL*. *Annals of the Missouri Botanical Garden*, 80:528–580, 1993.

[CT05a] B. Chor and T. Tuller. Finding the Maximum Likelihood Tree is Hard. In *Proceedings of the 9th Annual International Symposium on Research in Computational Biology (RECOMB 2005)*, 2005.

[CT05b] B. Chor and T. Tuller. Maximum Likelihood of Evolutionary Trees: Hardness and Approximation. *Bioinformatics*, 21:97–506, 2005.

[Day85] W. H. E. Day. Optimal Algorithms for Comparing Trees with Labeled Leaves. *Journal of Classification*, 2:7–28, 1985.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[DH98] P. W. Diaconis and S. P. Holmes. Matchings and Phylogenetic Trees. *Proceedings of the National Academy of Sciences*, 95:14600–14602, 1998.

[DHJ+97] B. Dasgupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On the Distances Between Phylogenetic Trees. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 427–436. ACM-SIAM, 1997.

[DKM98] R. Durbin, S. R. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabalistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[Dzi05] J. Dzik. The Chronophyletic Approach: Stratophenetics Facing an Incomplete Fossil Record. *Special Papers in Palaeontology*, 73:159–183, 2005.

[Efr79] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *Annals of Statistics*, 7:1–26, 1979.

[EO05] B. C. Emerson and P. Oromi. Diversification of the Forest Beetle Genus Tarphius on the Canary Islands, and the Evolutionary Origins of Island Endemics. *Evolution*, 59(3):586–598, 2005.

[Erw91] T. L. Erwin. An Evolutionary Basis for Conservation Strategies. *Science*, 253(5021):750–752, 1991.

[FCPT95] M. Farach-Colton, T. M. Przytycka, and M. Thorup. On the Agreement of Many Trees. *Information Processing Letters*, 55:297–301, 1995.

[Fel81] J. Felsenstein. Evolutionary Trees for DNA Sequences: A Maximum Likelihood Approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[Fel03] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates Inc., 2003.

[FG85] C. R. Finden and A. D. Gordon. Obtaining Common Pruned Trees. *Journal of Classification*, 2:255–276, 1985.

[Fit71] W. Fitch. Toward Defining Course of Evolution: Minimum Change for a Specified Tree Topology. *Systematic Zoology*, 20:406–416, 1971.

[GG03] S. Guindon and O. Gascuel. A Simple, Fast and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Systematic Biology*, 52(5):696–704, 2003.

[GGJ+06] G. Ganapathy, B. Goodson, R. Jansen, H. Le, V. Ramachandran, and T. Warnow. Pattern Identification in Biogeography. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2006. To appear.

[Gir99] G. Giribet. Exploring the Behavior of POY, a Program for Direct Optimization of Molecular Data. *Cladistics*, 15:415–428, 1999.

[GKK94] W. D. Goddard, E. Kubicka, and G. Kubicki. The Agreement Metric for Labeled Binary Trees. *Mathematical Biosciences*, 123:215–226, 1994.

[Gol94] P. A. Goloboff. Character Optimization and Calculation of Tree Lengths. *Cladistics*, 9:433–436, 1994.

[Gol96] P. A. Goloboff. Methods for Faster Parsimony Analysis. *Cladistics*, 12:199–220, 1996.

[Gol99] P. A. Goloboff. Analyzing Large Datasets in Reasonable Times: Solutions for Composite Optima. *Cladistics*, 15:415–428, 1999.

[GRW03] G. Ganapathy, V. Ramachandran, and T. Warnow. Better Hill-Climbing Seaches for Parsimony. In *Proceedings of the Third International Workshop on Algorithms in Bioinformatics (WABI)*, pages 245–258, 2003.

[GRW04] G. Ganapathy, V. Ramachandran, and T. Warnow. On Contract-and-Refine-Transformations Between Phylogenetic Trees. In *Proceedings of the Fifteenth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 893–902, 2004.

[GT89] H. Gabow and R. R. Tarjan. Faster Scaling Algorithms for Network Problems. *SIAM Journal of Computing*, 18(5):1013–1036, 1989.

[Gus97] D. Gusfield. *Algorithms on Strings Trees and Sequences*. Cambridge University Press, 1997.

[GW01] G. Ganapathy and T. Warnow. Finding a Maximum Compatible Tree for a Bounded Number of Trees with Bounded Degree Is Solvable in Polynomial Time. In *Proceedings of the First International Workshop on Algorithms in Bioinformatics (WABI)*, pages 156–163, 2001.

[GW02] G. Ganapathy and T. Warnow. Approximating the Complement of Maximum Compatible Subset of Leaves of *k* Trees. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 122–134, 2002.

[GY94] N. Goldman and Z. Yang. A Codon-based Model of Nucleotide Substitution for Protein-coding DNA Sequences. *Molecular Biology and Evolution*, 11:725–736, 1994.

[GY06] N. Goldman and Z. Yang. Heterotachy and Tree Building: A Case Study with Plastids and Eubacteria. *Molecular Biology and Evolution*, 23(1):40–45, 2006.

[Hal35] P. Hall. On Representatives of Subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.

[HH93] M. Hasegawa and T. Hashimoto. Ribosomal RNA Trees Misleading? *Nature*, 361:23, 1993.

[HJWZ96] J. Hein, T. Jiang, L. Wang, and K. Zhang. On the Complexity of Comparing Evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, 1996.

[HR01] J. P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian Inference of Phylogenetic Trees. *Bioinformatics*, 17(8):754–755, 2001.

[HRNB01] J. P. Huelsenbeck, F. Ronquist, R. Nielsen, and J. P. Bollback. Bayesian Inference of Phylogeny and its Impact on Evolutionary Biology. *Science*, 294:2310–2314, 2001.

[HS96] A. Hamel and M. A. Steel. Finding a Maximum Compatible Tree is NP-hard for Sequences and Trees. *Applied Mathematics Letters*, 9(2):55–60, 1996.

[Jac04a] A. P. Jackson. Cophylogeny of the Ficus Microcosm. *Biological Review*, 79(4):751–768, 2004.

[Jac04b] A. P. Jackson. Phylogeny and Biogeography of the Malagasy and Australasian Rainbowfishes (Teleostei : Melanotaenioidei): Gondwanan Vicariance and Evolution in Freshwater. *Molecular Phylogenetics and Evolution*, 33(3):719–734, 2004.

[JC69] T. H. Jukes and C. Cantor. *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, 1969.

[JEOH00] C. Juan, B. C. Emerson, P. Oromi, and G. M. Hewitt. Colonization and Diversification: Towards a Phylogeographic Synthesis for the Canary Islands. *Trends in Ecology and Evolution*, 15(3):104–109, 2000.

[JG79] D. S. Johnson and M. R. Garey. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[KKTM05] K. Katoh, K. Kuma, H. Toh, and T. Miyata. MAFFT Version 5: Improvement in Accuracy of Multiple Sequence Alignment. *Nucleic Acids Research*, 33(2):511–518, 2005.

[KMKM02] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: A Novel Method for Rapid Multiple Sequence Alignment Based on Fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.

[Knu03] B. Knudsen. Optimal Multiple Parsimony Alignment with Affine Gap Cost Using a Phylogenetic Tree. In *Proceedings of the Third International Workshop on Algorithms in Bioinformatics (WABI)*, pages 433–446, 2003.

[LP86] L. Lovasz and M. D. Plummer. *Matching Theory*. North-Holland Publishing Company, 1986.

[LR05] F. Lapointe and L. Rissler. Congruence, Consensus and Comparative Phylogeography of Codistributed Species in California. *The American Naturalist*, 166(2):290–299, 2005.

[LS99] B. Larget and D. L. Simon. Markov Chain Monte Carlo Algorithms for the Bayesian Analysis of Phylogenetic Trees. *Molecular Biology and Evolution*, 16:277–283, 1999.

[LTZ96] M. Li, J. Tromp, and L. Zhang. On the Nearest Neighbour Interchange Distance Between Evolutionary Trees. *Journal of Theoretical Biology*, 182:463–467, 1996.

[Mad91] D. R. Maddison. The Discovery and Importance of Multiple Islands of Most Parsimonious Trees. *Systematic Zoology*, 43(3):315–328, 1991.

[Mau96] B. Mau. Bayesian Phylogenetic Inference via Markov Chain Monte Carlo Methods. 1996. Ph. D. Dissertation, University of Wisconsin, Madison.

[MF98] C. Moritz and D. Faith. Comparative Phylogeography and the Identification of Genetically Divergent Areas for Conservation. *Molecular Ecology*, 7:419–429, 1998.

[MRKS06] J. R. McDill, M. Repplinger, J. Kaderit, and B. B. Simpson. The Phylogeny of the Temperate Linaceae. *American Journal of Botany*, 2006. in preparation.

[MW03] B. Moret and T. Williams. An Investigation of Phylogenetic Likelihood Methods. In *Proceedings of the Third IEEE Symposium on Bioinformatics and Bioengineering*, pages 79–83, 2003.

[NEM] The Nematode Branch of the Assembling the Tree of Life Project: NemATOL. http://nematol.unh.edu.

[Nix99] K. C. Nixon. Parsimony Ratchet: A New Method for Rapid Parsimony Analysis. *Cladistics*, 15(4):407–414, 1999.

[NMR+02] L. Nakhleh, B. Moret, U. Roshan, K. St. John, J. Sun, and T. Warnow. The Accuracy of Fast Phylogenetic Methods for Large Datasets. In *Proceedings of the Seventh Pacific Symposium on Bioinformatics*, pages 211–222, 2002.

[NRJ+01] L. Nakhleh, U. Roshan, K. St. John, J. Sun, and T. Warnow. The Performance of Phylogenetic Methods on Trees of Bounded Diameter. In *Proceedings of the First*

International Workshop on Algorithms in Bioinformatics (WABI2001), pages 214–226, 2001.

[NTWJ05] L. Nakhleh, C. R. Linder T. Warnow, and K. St. John. Reconstructing Reticulate Evolution in Species: Theory and Practice. *Journal of Computational Biology*, 12(6–7):796–811, 2005.

[PAD00] D. R. Powell, L. Allison, and T. I. Dix. Fast, Optimal Alignment of Three Sequences Using Linear Gap Costs. *Journal of Theoretical Biology*, 207:325–336, 2000.

[Pag88] R. D. M. Page. Quantitative Cladistic Biogeograpy: Constructing and Comparing Area Cladograms. *Systematic Zoology*, 37:254–270, 1988.

[Pag94] R. D. M. Page. Maps Between Trees and Cladistic Analysis of Historical Associations Among Genes. *Systematic Biology*, 43(1):58–77, 1994.

[PC98] D. Posada and K. A. Crandall. Testing the Model of DNA Substitution. *Bioinformatics*, 14(9):817–818, 1998.

[PGB05] A. Purvis, J. L. Gittleman, and T. Brooks, editors. *Phylogeny and Conservation*. Cambridge University Press, 2005.

[PW96] C. A. Phillips and T. Warnow. The Asymmetric Median Tree: A New Model for Building Consensus Trees. *Discrete Applied Mathematics*, 71:311–335, 1996.

[RF81] D. F. Robinson and L. R. Foulds. Comparison of Phylogenetic Trees. *Mathematical Biosciences*, 53:131–147, 1981.

[RG97] A. Rambaut and N. Grassly. Seq-Gen: An Application for the Monte Carlo Simulation of DNA Sequence Evolution along Phylogenetic Trees. *Computer Applications in the Biosciences*, 13:235–238, 1997.

[RMWW04] U. Roshan, B. M. E. Moret, T. L. Williams, and T. Warnow. Rec-I-DCM3: A Fast Algorithmic Technique for Computing Large Phylogenetic Trees. In *Proceedings of the IEEE Computational Systems Bioinformatics Conference (CSB2004)*, 2004.

[Rob71] D. F. Robinson. Comparison of Labeled Trees with Valency Three. *Journal of Combinatorial Theory*, 11:105–119, 1971.

[Ros78] D. E. Rosen. Vicariant Patterns and Historical Explanation in Biogeograpy. *Systematic Zoology*, 27:159–188, 1978.

[SAH94] D. Sankoff, Y. Abel, and J. Hein. A Tree, A Window, A Hill, Generalization of Nearest Neighbor Interchange in Phylogenetic Optimization. *Journal of Classification*, 11:209–232, 1994.

[SAJS05] J. Sullivan, Z. Abdo, P. Joyce, and D. L. Swofford. Evaluating the Performance of a Successive-Approximations Approach to Parameter Optimization in Maximum-Likelihood Phylogeney Estimation. *Molecular Biology and Evolution*, 22(6):1386–1392, 2005.

[SH99] H. Shimodaira and M. Hasegawa. Multiple Comparison of Log-Likelihoods With Applications to Phylogenetic Inference. *Molecular Biology and Evolution*, 16:1114–1116, 1999.

[SN87] N. Satou and M. Nei. The Neighbor Joining Method: A New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[SOL05] A. Stamatakis, M. Ott, and T. Ludwig. RAxML-OMP: An Efficient Program for Phylogenetic Inference on SMPs. In *Proceedings of the 8th Internationational Conference on Parallel Computing Technologies*, pages 288–302, 2005.

[SOWH96] D. L. Swofford, G. J. Olson, P. J. Waddell, and D. M. Hillis. *Molecular Systematics*, chapter entitled "Phylogenetic Inference", pages 407–425. Sinauer Associates, Sunderland, Massachusetts, second edition, 1996.

[SS73] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy*. W. H. Freeman and Company, San Francisco, 1973.

[SS03] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.

[SSN+97] D. E. Soltis, P. S. Soltis, D. L. Nickrent, L. A. Johnson, W. J. Hahn, S. B. Hoot, J. A. Sweere, R. K. Kuzoff, K. A. Kron, M. W. Chase, S. M. Swenson, E. A. Zimmer, S. M. Chaw, L. J. Gillespie, W. J. Kress, and K. J. Sytsma. Angiosperm Phylogeny Inferred from 18S ribosomal DNA sequences. *Annals of the Missouri Botanical Garden*, 84:1–49, 1997.

[ST97] M. Steel and C. Tuffley. Links Between Maximum Likelihood and Maximum Parsimony Under a Simple Model of Site Substitution. *Bulletin of Mathematical Biology*, 59:581–607, 1997.

[ST04] F. Santini and J. C. Tyler. The Importance of Even Incomplete Fossil Taxa in Reconstructing the Phylogenetic Relationships of the Tetraodontiforms (Acanthomorpha: Pisces). *Integrative and Comparative Biology*, 44(5):349–357, 2004.

[Sta06] A. Stamatakis. Phylogenetic Models of Rate Heterogeneity: A High-Performance Computing Perspective. In *Proceedings of the 20th Internationational Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006.

[Ste02] M. Steel. *Molecular Systematics and Evolution: Theory and Practice*, chapter entitled "Some Statistical Aspects of the Maximum Parsimony Method", pages 125–140. Birkhauser, 2002.

[SW93] M. Steel and T. Warnow. Kaikoura Tree Theorems: Computing the Maximum Agreement Subtree. *Information Processing Letters*, 48:77–82, 1993.

[Swo86] D. L. Swofford. *Studies in Numerical Cladistics: Phylogentic Inference Under the Principle of Maximum Parsimony*. PhD thesis, University of Illinois at Urbana-Champaign, 1986.

[Swo96] D. L. Swofford. *PAUP*. Phylogeny Analysis Using Parsimony (* and Other Methods)*. Sinauer Associates, Sunderland, Massachusetts, 1996. Version 4.

[THG94] 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*, 22:4673–4680, 1994.

[ToL06] The Tree of Life Web Project, 1996–2006. http://tolweb.org.

[TSL06] R. E. Timme, B. B. Simpson, and C. R. Linder. Phylogenetic Resolution in Helianthus (Asteraceae) Using the External Transcribed Spacer Region. *American Journal of Botany*, 2006. in preparation.

[vVZK99] M. G. P. van Veller, M. Zandee, and D. J. Kornet. Two Requirements for Obtaining Common Patterns Under Different Assumptions in Vicariance Biogeography. *Cladistics*, 15:393–405, 1999.

[War94] T. Warnow. Tree Compatibility and Inferring Evolutionary History. *Journal of Algorithms*, 16:388–407, 1994.

[Yan94] Z. Yang. Maximum Likelihood Phylogenetic Estimation from DNA Sequences When Substitution Rates Differ over Sites: Approximate Methods. *Journal of Molecular Evolution*, 39:306–314, 1994.

[Zwi06] D. J. Zwickl. *GARLI: Genetic Algorithm for Rapid Likelihood Inference*. 2006. url: http:///www.zo.utexas.edu/faculty/antisense/Garli.html.

Vita

Ganeshkumar Ganapathysaravanabavan was born in Trivandrum, in the Indian state of Kerala on November 16, 1977, the son of Geetha Sankaran and Saravanabavan Ganapathy. He grew up in Tiruchendur, in the neighboring state of Tamilnadu, and attended Kamalavati Higher Secondary School until he was 12, when his family moved to Tiruchy. In Tiruchy, he attended Sri Akilandeswari Vidyalaya until his tenth grade, and graduated high school from R. S. Krishnan Higher Secondary School in 1995. He moved to the northern state of Rajasthan for his college education, where he received his Bachelor's degree in Computer Science from the Birla Institute of Technology and Science, located in Pilani, a rural college town not unlike College Station, Texas. For a year, from 1999 to 2000, he was a software engineer at Sasken, a multimedia and broadband technology firm in Bangalore, India. In spring 2000, he was admitted to the Ph.D. program in Computer Sciences at the University of Texas, and moved to Austin from Bangalore in August 2000.

Permanent Address: #208, 407 SFS, Yelahanka New Town

Bangalore, India, 5600064

This dissertation was typeset with LaTeX 2ε* by the author.

*LaTeX 2ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.