Phylogenetic Trees and Their Analysis

Eric Ford

Graduate Center, City University of New York

Recommended Citation
https://academicworks.cuny.edu/gc_etds/37
Phylogenetic Trees and Their Analysis

by

ERIC FORD

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2014
This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy

Katherine St. John

Date

Chair of Examining Committee

Robert Haralick

Date

Executive Officer, Dept. of Computer Science

Ward Wheeler
Amotz Bar-Noy
Daniel Gusfield

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK
ABSTRACT

Phylogenetic Trees and Their Analysis

by

Eric Ford

Advisor: Katherine St. John

Determining the best possible evolutionary history, the lowest-cost phylogenetic tree, to fit a given set of taxa and character sequences using maximum parsimony is an active area of research due to its underlying importance in understanding biological processes. As several steps in this process are NP-Hard when using popular, biologically-motivated optimality criteria, significant amounts of resources are dedicated to both heuristics and to making exact methods more computationally tractable. We examine both phylogenetic data and the structure of the search space in order to suggest methods to reduce the number of possible trees that must be examined to find an exact solution for any given set of taxa and associated character data. Our work on four related problems combines theoretical insight with empirical study to improve searching of the tree space. First, we show that there is a Hamiltonian path through tree space for the most common tree metrics, answering Bryant’s Challenge for the minimal such path. We next examine the topology of the search space under various metrics, showing that some metrics have local maxima and minima even with perfect data, while some others do not. We further characterize conditions for which sequences simulated under the Jukes-Cantor model of evolution yield well-behaved search spaces. Next, we reduce the search space needed for an exact solution by splitting the set of characters into mutually-incompatible subsets of compatible characters, building trees based on the perfect phylogenies implied by these sets, and then searching in the neighborhoods of these trees. We validate this work empirically. Finally, we compare two approaches to the generalized tree alignment problem, or GTAP: Sequence alignment followed by tree search vs. Direct Optimization, on both biological and simulated data.
Acknowledgements

Many, many thanks to my advisors, Ward Wheeler and especially Katherine St. John, without whose guidance this document would never have been written. I would also like to thank my co-authors on these papers, the aforementioned Drs. Wheeler and St. John, as well as Kevaughn Gordon and Ellen Urheim, both of whom contributed many significant insights into the problems, when I wasn’t preventing them from doing so.
# Contents

Acknowledgements ....................................................... v

Contents ......................................................................... vi

List of Tables .................................................................... vii

List of Figures ..................................................................... vii

1 Introduction .................................................................... 1
   1.1 Computational Complexity ........................................ 5
   1.2 Tree Definitions and Structure ..................................... 6
   1.3 Tree Metrics ............................................................ 14
   1.4 Agreement Forests ................................................... 21
   1.5 Tree Space ............................................................... 23
   1.6 The Generalized Tree Alignment Problem ................. 25
   1.7 Models of Evolution ................................................ 28

2 Minimal Paths through Tree Space .................................. 29
   2.1 Introduction ............................................................ 29
   2.2 Background ............................................................ 31
   2.3 Main results ............................................................ 33
   2.4 Conclusion ............................................................. 42
   2.5 Acknowledgments .................................................. 43

3 Hill-Climbing Search of Tree Space ............................... 44
   3.1 Introduction ............................................................ 44
   3.2 Local and Global Optima for Perfect Data .................. 47
   3.3 Empirical Results .................................................... 48
   3.4 Discussion ............................................................. 50
   3.5 Future Work .......................................................... 50
   3.6 Acknowledgments .................................................. 51

4 Landscapes of Tree Space ............................................... 52
   4.1 Background ............................................................ 57
   4.2 Materials ............................................................... 59
4.3 Methods .............................................................. 61
4.4 Results ............................................................. 65
4.5 Discussion ........................................................ 69
4.6 Proofs of Results ............................................... 72
4.7 Acknowledgments .............................................. 74

5 Efficient Search in Tree Space 75
  5.1 Introduction .................................................... 75
  5.2 Software tools ............................................... 76
  5.3 Materials ....................................................... 78
  5.4 Methods ......................................................... 78
  5.5 Results .......................................................... 81
  5.6 Conclusion ..................................................... 88
  5.7 Future Work .................................................. 88

Bibliography ........................................................ 89
List of Tables

4.1 Reductions in sizes of tree spaces for TreeBASE data ................................................. 68
5.1 MAFFT settings for various data sets ............................................................................. 80
5.2 Alignment results ......................................................................................................... 83
5.3 Comparative alignment results ..................................................................................... 85
5.3 Comparative alignment results ..................................................................................... 86
5.3 Comparative alignment results ..................................................................................... 87

List of Figures

1.1 Phylogenetic tree ........................................................................................................... 2
1.2 A perfect phylogeny ....................................................................................................... 4
1.3 NNI operation .............................................................................................................. 16
1.4 SPR and TBR operations ............................................................................................ 18
1.5 p-ECR ......................................................................................................................... 21
2.1 The NNI operation ....................................................................................................... 31
2.2 Hamiltonian path construction ..................................................................................... 33
2.3 Expansion of a tree ....................................................................................................... 35
2.4 Base Case I and three walks from Lemma 2.3.3 ........................................................... 38
3.1 Local minima in NNI and SPR spaces for eight taxa ...................................................... 47
3.2 λ versus average number of optima for 8-taxon SPR-space ........................................ 49
4.1 Minimal trees ............................................................................................................... 55
4.2 Relaxed RF .................................................................................................................. 58
4.3 Upper bound on RF ...................................................................................................... 63
4.4 Walking Metasiro tree space ....................................................................................... 66
1 Introduction

For biologists, the necessity of creating, analyzing, and evaluating evolutionary or phylogenetic trees has presented itself at least since the time of Darwin [23, 43]. If species are considered to have evolved from common ancestors, then the relationships between extant (and for that matter extinct) species could perhaps best be represented by a graph in the form of a tree, where vertices represent species or taxa (the leaves being extant species and interior vertices putative ancestral or extinct ones) and the edges between vertices represent parent-child relationships, much as a family tree represents the lines of descent of a specific group of individuals, albeit in this case a group of individuals reproducing asexually. As these trees represent phylogenies, they are referred to as phylogenetic trees.

Traditionally, for the purpose of determining evolutionary relationships, biologists represented species as groups (hereafter sequences) of morphological characteristics. In the simplest case these could be binary, having two discrete states (“has feathers”/“does not have feathers”), but they may be higher order, as in the case of four-state nucleotide data (“A”, “C”, “G”, “T”), or even continuous (i.e. femur length). With the availability of gene and protein sequencing, these trees can now be built using molecular characters. This new data can be used either in place of or alongside the more traditional characters, and offers several benefits. First, the information is discrete rather than continuous and therefore
more easily quantified; second, more distantly related species can be more easily compared, as genes are often strongly conserved over evolutionary time; and third, it presents a wealth of information. However, being able to (possibly more reliably) compare more species also means that the data available for interpretation in a phylogenetic context increases by several orders of magnitude, making the task of finding the optimal evolutionary relationship for that group of organisms a significantly more difficult procedure. Not only does the task of comparing sequences grow as the sequences grow in length, but as more taxa can be compared, the number of comparisons needed grows quickly. Specifically, for any group of $n$ taxa there are $(2n − 5)!! = (2n − 5)(2n − 7) \cdots (5)(3)(1)$ possible phylogenetic trees [64], meaning that the number of possible trees grows exponentially on the number of taxa.

If a phylogeny is a presumptive ancestry with each taxon is defined by its character sequence, then putative ancestors are internal vertices. These vertices, in turn, may be assigned character sequences, and evolutionary change is represented by changes in the characters from node to node. This change could be traced from the root (the most distant putative ancestor) to the leaves (whichever taxa are being compared). In a "good" phylogeny,
characters are assumed to have changed as few times as possible on the tree. This means two things: First, if a character is in a certain state on the vertex labels in one section of the tree, the character in the same position ought not exist in the same state on the labels of vertices in an unconnected part of the tree. That is, it is unlikely that a given character state will have evolved twice in different lineages. Second, if a character changes from state A to state B along one path from the root, then none of the descendants of the node labeled with a B ought to have that character in state A. This is a biologically reasonable approach, as it would be unlikely for complex macroscopic structures such as feathers to have evolved multiple times in history, as would happen if the same character appeared in different parts of the phylogeny (although analogous structures, like bat wings and bird wings might). Likewise, it would be unlikely for evolution to reverse itself, and an evolved structure to revert to an earlier form, retreading the same path it traveled.

On the other hand, if one considers molecular characters, those are less reasonable assumptions. It is not at all unlikely—given the large number of characters in a molecular sequence and the lack of complexity of those characters—that a single nucleotide or amino acid might appear as the same character for taxa in multiple non-contiguous locations in the tree, or that that single character might revert to a previous state. Nonetheless, one could assume that the number of times that these situations might occur would be outweighed by the number of times they do not occur in the phylogeny.

This assumption of maximum parsimony (essentially a rephrasing of Occam’s Razor) also matches the hypothesis behind the construction of a phylogeny: Taxa with similar characters are grouped in the phylogeny because the character similarity makes the taxa similar, and similar taxa (or putative taxa) are closely related.
Figure 1.2: Left: A perfect phylogeny. Each character state evolves at most once on the tree. Right: A non-perfect phylogeny. Some of the characters have evolved multiple times on the same tree. As an example, both $t_3$ and $v_4$ have a “C” in the first position, whereas their respective parent nodes, $\rho$ and $v_1$, have other characters in that position, thus “C” must have evolved twice independently on the tree.

As noted above, finding an exact solution to the problem of determining the most optimal phylogenetic tree for a group of taxa grows exponentially more difficult as the number of taxa increases. The only known ways to guarantee the exact solution are to check every tree or to do a branch-and-bound search. Although the situation will improve as computers grow more efficient and computer memory gets cheaper, it is currently only feasible to find exact solutions to this problem for sets of at most 15 to 18 taxa (≈ 8 trillion to ≈ 190 quadrillion trees). Given this, much research effort is focused on finding good, but inexact, solutions using heuristics. Defining “good” in this context is difficult, as the best tree remains unknown, but some information is known even before the search begins.

The length of the character sequences and the number of states for each character are known, and the cost of a perfect phylogeny can be determined from this information. If there are $k$ characters, each with $c$ states, and each character changes to each possible state at most once, then the total change on the tree is $k(c - 1)$. 
As only inexact solutions are available for even reasonably small sets of taxa, search is used. In this case, all the possible trees for \( n \) taxa constitute a space that must be searched, which we will refer to as tree space. Which trees are nearer or farther from each other is determined by the similarity of the trees, and tree space might have hills and valleys, whose heights and depths are determined by the scores of the trees found there. However, while much effort has been focused on heuristic solutions, these heuristics have largely been borrowed from the field of search, without giving a great deal of consideration to the underlying search space. We posit that a better understanding of tree space will allow better search, and an attempt to determine the complexity of tree space will be the unifying theme of this thesis.

1.1 Computational Complexity

In general, the problems dealt with in the following chapters are considered difficult and do not lend themselves to exact solutions. They fall into a class of problems referred to as \textit{NP-Hard} [36]. Before we discuss those problems, therefore, it is worth a quick introduction to computational complexity. (See [22] for more details.)

While some problems can be computed quickly (e.g. sorting a list of numbers or determining the parsimony score of a tree topology under a fixed character sequence), other problems are quite difficult computationally. To classify the difficulty, upper bounds on the running time are used. If \( n \) is the number (or length) of the inputs, then problems that, in the worst case, can be solved in time proportional to \( n^k \), for a fixed \( k \), are part of the class of polynomial-time problems (“P-time” or “P”). Similarly, if a problem can be solved in time proportional to \( 2^n \), then it is part of the class of exponential-time problems (“EXP-time” or
“EXP”). By definition, $P \subset EXP$. There is another class of problems, the class of problems with non-deterministic polynomial-time algorithmic solutions ("NP-time" or "NP"). This third set of problems are all those for which, if a solution is given, the correctness of the solution can be checked in polynomial time. Examples of problems in NP, some of which are discussed in the following chapters, include finding the optimal maximum parsimony tree [33], calculating the NNI, SPR, or TBR distance between two trees [2, 10, 13, 57], finding a proof for a high school geometry problem (formally called Frege proof systems), or calculating the shortest tour for a salesman to visit a set of cities (see [22, 36] for more examples). NP lies between the first two classes described (i.e. $P \subseteq NP \subseteq EXP$). An important challenge (with a million dollar prize for the solution) is whether $NP$ is different from $P$ [21].

The difficulty of some problems can be parametrized. For example, for several metrics, finding the distance between trees has been show to be easy for small distances. A good example of this is Tree Bisection and Reconnection, or TBR (which will be discussed in depth below): if two trees are distance 1, then that can be determined very quickly. Trees of distance $\leq 2$ take a bit longer (proportional to $n^2$ time) and those of distance $\leq k$ can be determined in time proportional to $n^k$, where $n$ is the number of taxa [2]. Formally:

**Definition 1.1.1.** [26]: A parameterized problem is called fixed-parameter tractable if it admits a solving algorithm whose running time on input instance $(I, k)$ is $f(k) \cdot |I|$, where $f$ is an arbitrary function depending only on $k$.

### 1.2 Tree Definitions and Structure

We have begun a discussion of trees, but a formal definition is necessary. Trees are a subset of graphs.
Definition 1.2.1. A graph $G = (V,E)$ is a collection of vertices, $v \in V$ connected by edges $e \in E$.

Definition 1.2.2. The valency or degree of a node is the number of nodes to which it is connected.

Definition 1.2.3. The diameter of a graph $G$, $\Delta(G)$, is the maximum distance between two nodes in a graph.

Formally, as defined by Robinson [64],

Definition 1.2.4. A (binary) phylogenetic tree is a non-cyclic connected graph $T$ on collection of labeled nodes $L$ (the taxa) and unlabeled interior vertices. The labeled nodes form the leaves of the tree and therefore have valency one, and each interior vertex has a valency three.

This interior valency of three yields binary trees and has a biological motivation—two related species are considered to have one common ancestor, so each interior vertex has two children and one parent, and trees with a higher valency represent uncertainty about evolutionary history. An interior valency of three is practical as well, as it creates regularity in the tree structure, allowing easier mathematical proofs and algorithm design. While Robinson concentrated on binary—or fully resolved—trees, trees with nodes of higher valency ("polytomies") are also studied. In general, trees can be rooted or unrooted. (If rooted, it is often useful to think of “hanging” a leaf $t_0$ from the root. $t_0$ can then be treated as an additional taxon, and the root will have valency consistent with the other interior vertices.)
We, too, focus on binary trees. Given that $|L| = n$, an unrooted tree $T$ has $(n - 2)$ interior vertices, $(n - 3)$ interior edges and a maximum diameter of $(n - 1)$ [64]. These numbers differ for rooted trees.

For the purposes of some algorithms the leaves of a tree are often grouped into subtrees, of which the smallest subtrees consist of single pairs of leaves.

**Definition 1.2.5.** A sibling pair, or cherry, is a pair of leaves whose incident edges share a common vertex.

Each vertex $v$ has a label $s = \chi_1\chi_2 \cdots \chi_k$, which is a string of length $k$ on alphabet $C$ with $|C| = r$ (the size or cardinality of $C$ is $r$). This label is a list of the states of each character for each taxon, where a character is a characteristic of the given taxon (“has wings”, “has fur”, etc.), so for instance in traditional cladistics $C$ might be a binary list [true, false], as it might be for single nucleotide polymorphism (SNP) data, whereas if the taxa are being compared using nucleotide data, $C = \{A, C, G, T\}$. Formally, given a set $X$,

**Definition 1.2.6.** [72] A character on $X$ is a function $\chi$ mapping non-empty $X' \subseteq X$ into a set $C$ of character states.

As there are no cycles in a tree, one can consider the edges in a phylogenetic tree to be splits of the leaves into two sets. Each edge thus sorts its ancestor nodes from its child nodes.

**Definition 1.2.7.** [41] Given binary characters, in the perfect phylogeny, each interior edge corresponds to one character in the [aligned] character sequences and the edges sort the vertices by character state.
That is, in a perfect phylogeny, if one were to trace a path down from the root to a given vertex, the values of the characters at that vertex would be enumerated by the edges traced. This sorting also implies that in a perfect phylogeny on binary characters the length of each label—the number of characters—is \((n - 3)\), as that is the number of interior edges, and any additional characters are extraneous. However, in the case of non-binary characters each character corresponds to multiple edges, with each \(r\)-state character defining \(r - 1\) edges in the tree.

Given a set of labeled vertices and associated character strings, it may or may not be possible to form a perfect phylogeny. It can be determined in \(O(nk)\) time whether a perfect phylogeny exists [41]. Also, it has been shown [41, 50] that building a perfect phylogenetic tree from a set of labeled taxa is fixed-parameter tractable, with the parameter being the number of character states.

**Optimality Criteria for Trees**

In the case that no perfect phylogeny exists, some optimality criterion is defined, allowing the trees to be scored, and a most optimal tree is found. However, finding the most optimal tree for a given set of taxa is a version of the Steiner tree problem and has been shown to be NP-Complete [33] even when the characters are binary (see discussion below). Therefore, heuristic methods of solving the problem are suggested.

Before discussing heuristic solutions to the problem of finding the most optimal tree, the method of scoring the trees must be defined. In general, this is done using parsimony or maximum likelihood [77].

In maximum likelihood, each edge in a tree is given a weight or a length. The length of a
given edge is a measure of the amount of change, that is, what proportion of the characters change, across that edge. The lengths further interact with the character sequences according to some model of evolution, which varies according to the assumptions of the researchers.

Given a model, a probability can then be computed for a tree with edge lengths, \( \text{pr}(D|T) \), the probability of the data \textit{given that tree}. The most optimal tree is then the tree that maximizes \( \text{pr}(D|T) \). A further discussion of maximum likelihood can be found in [72, 77, 82].

We will concentrate on maximum parsimony optimality criterion, in which each tree is given a parsimony score based on summing the Hamming distances of all adjacent vertices, where the Hamming distance is simply a count of the number of position-by-position differences between two strings of characters. For the purpose of computing the total sum, the internal vertices of the tree must labeled, in addition to the leaves. An example of a labeling algorithm by Fitch is examined below.

The maximum parsimony criterion seeks the tree with the minimal number of changes needed to explain a given character sequence. Formally:

**Definition 1.2.8.** [72]: For a graph \( G = (V, E) \) and a function \( f \) on \( V \), the changing set of \( f \) is the subset \( \text{Ch}(f) = \{\{u,v\} \in E : f(u) \neq f(v)\} \) of the edges of \( G \). The changing number of \( f \), denoted \( \text{ch}(f) \) is the cardinality of \( \text{Ch}(f) \).

**Definition 1.2.9.** [72]: An \( X \)-tree, or semi-labeled tree, is an ordered pair \( (T; \phi) \), where \( T \) is a tree with vertex set \( V \) and \( \phi : X \rightarrow V \) is a map with the property that, for each \( v \in V \) of degree at most two, \( v \in \phi(V) \).

For our purposes, then, an \( X \)-tree is a tree with leaves labeled by characters.

**Definition 1.2.10.** [72]: Let \( \chi : X' \rightarrow C \) be a character on \( X \) and let \( T \) be an \( X \)-tree. An extension of \( \chi \) to \( T \) is a function \( \bar{\chi} : V(T) \rightarrow C \) which is identical to \( \chi \) on \( X' \). The
parsimony score, \( l(\chi, T) \) of \( \chi \) on \( T \) is the minimum value of \( ch(\bar{\chi}) \) over all extensions \( \bar{\chi} \) of \( \chi \) to \( T \).

Informally, if we have a character assigned to leaves of a tree, we can extend that character to label the internal nodes of the tree. If that extension minimizes the changing number of the initial character, then it is called a minimum extension. Using the above notation, the parsimony score for sequence of character, \( C = (\chi_1, \chi_2, \ldots, \chi_k) \) is

\[
l(C, T) = \sum_{i=1}^{k} l(\chi_i, T)
\]

The parsimony score is also referred to as the length of the tree.

Definition 1.2.11. Given a character labeling of leaves \( \phi : X \rightarrow V \) of an \( X \)-tree \( T \), if \( T \) is a perfect phylogeny, then the character set \( X \) is a compatible set.

In this unweighted parsimony, the characters are considered to be independent, and any change from one character to another is equally likely. (If certain character state mutations were considered to have specific probabilities then a weighted parsimony or maximum-likelihood model would be used [77].) In maximum parsimony, the “best” tree for any set of taxa is the most-parsimonious tree—the tree with the lowest parsimony score.

Recall that in a perfect phylogeny each interior edge corresponds to a character state. In this case, then, each edge bipartitions (or sorts) the tree by a given character state, so that each edge contributes one to the total Hamming distance. That is, for each internal edge, the two incident vertices differ on the character corresponding to that edge, and only on that character. Thus the parsimony score for a perfect phylogeny on binary characters is \((n - 3)\), and this represents the lowest possible score for a set of taxa.
Finding the most-parsimonious labeling of a tree *with a given topology* is a polynomial time problem, and was shown to be so by Fitch [31]. Fitch’s algorithm consists of first traversing the tree upward from the leaves, making decisions regarding character states of each node based upon its child nodes, and preserving ambiguities. Once the root has been reached, the tree is once again traversed, but in the opposite direction, eliminating the saved ambiguities based upon a model of least-evolution—that is to say, eliminating the ambiguities using the assumption that a character will change as few times as possible during evolution.

A common measure of how well a tree explains a character sequence is the consistency index:

**Definition 1.2.12.** [54]: The consistency index (CI) of a set of characters and tree is the ratio \( m/s \), where \( m \) is the minimal score of the characters possible on any tree and \( s \) is the actual score of the tree.

Although finding the most-parsimonious tree for a given topology is polynomial, Foulds and Graham showed [33] that the problem of finding the most-parsimonious tree for a given set of taxa is NP-Complete. They recast the problem as a Steiner tree problem, which involves finding the minimum-length tree given a set of vertices. This is similar to a minimum spanning tree, except that additional vertices (Steiner points) may be added. In the case of an evolutionary tree, the original vertices are the taxa at the leaves, the edge weights are determined by the Hamming distances between vertices, so should all be one, and the Steiner points are the posited ancestral species. In their paper, Foulds and Graham defined the Steiner minimal tree for phylogeny (SPP) as the Steiner minimal tree where the edge weights are the Hamming distances of the connected nodes with binary characters. They
then showed that finding the SPP is NP-complete by reducing the Exact Three Cover (X3C) problem to the SPP.

The X3C problem asks, for a given set $I_m$, $|I_m| = 3q$, whether there exist $\{F_1, F_2, \ldots, F_m\}$, $F_i \subseteq I_m$, $|F_i| = 3$ such that $F_i \subseteq \{1, 2, \ldots, 3m\} \equiv I_m$. (That is, are there disjunct subsets of $I$, each of size three, the union of which is equal to $I$?) To show equivalence, a tree was created and the vertices were given a specific labeling so that the Hamming distance (i.e. the weight or length) of each edge was one and chains of vertices ran between Steiner points—which represented the three-sets—and the leaves, which represented elements of the set $I_m$. Foulds and Graham then proved that if there is an X3C of the set in question the Steiner minimum tree of the set is a specific length, thereby showing an equivalence between the two problems. Interestingly, while the length of the labels used is quite long, the Hamming distance can be computed in linear time on the number of taxa. Also, the cardinality of the characters in the proof was only two, with the implication being that finding a solution on a character set with such a low cardinality was the easiest version of the Steiner tree problem for phylogenetic trees. As all edge weights are equal, the Hamming distance can be calculated easily, and unweighted maximum parsimony is considered the simplest optimality criterion.

As previously mentioned, maximum-likelihood can also be used to grade every possible evolutionary tree. In maximum likelihood, a model of evolutionary change is posited. A model may take into consideration the likelihood of specific character-character changes, the length of evolutionary time, base mutation rates, and so on. Because it takes biological criteria into account, maximum likelihood can be more consistent and its estimates often have a lower variance than other models of change [77]. However, because maximum likelihood
takes into consideration edge lengths in addition to vertex labels, it increases the complexity
of any operations or solutions.

The first attempt to build a tree based on maximum-likelihood seems to have been by
Fitch and Margoliash in 1967 [32]. Using amino acids as the characters and variations of
cytochrome-c as their taxa labelings, they reconstructed an evolutionary tree of 20 organisms
from the animal and fungi kingdoms. They based the presumed mutation rates between the
various cytochromes on the number of nucleotide changes needed to mutate one amino acid
into another. By using this criterion to group pairs of organisms (or implied ancestral
organisms) they were able to construct an evolutionary tree with edge lengths based on the
evolutionary distances of the proteins. Forty most-likely trees were created and the lowest-
scoring of those was presented in their paper. They found that their best tree was highly
similar to the “classical” phylogenetic tree created from pre-genetic biological data.

1.3 Tree Metrics

Tree metrics generally serve two purposes. First, as we have discussed, there are several
methods for determining the optimality of a given tree; unfortunately these methods often
return incompatible trees on the same data. In fact the same method can sometimes return
different answers on subsequent runs on the same data. It is therefore necessary to be able
to compare different posited trees in order to determine their similarity or dissimilarity. This
allows various optimality criteria to be compared, contrasted and grouped.

Second, as even the simplest method of determining a phylogeny is NP-Complete, for
data sets over 20 or so taxa, brute force and exhaustive search solutions are usually set aside
in favor of heuristic algorithms. These generally involve various hill-climbing algorithms
on the space of possible trees for a given set of taxa. In order to move through that space in an orderly manner, operations must be used both to move from one tree to another. In addition, in order to use local search algorithms such as hill-climbing or simulated annealing that compare adjacent data points, operations are used to define “neighbors.” Also, once a search space is defined for a given metric or operation, the properties of that space can be investigated in order to perform efficient searches.

Following Robinson [64], on any given set of taxa $L$, where $|L| = n$, there exists a set $\mathbb{G}_n$ of possible unrooted trees $G$. $\mathbb{G}_n$ is the search space in which the optimal tree must be found. Robinson showed that there are $(2n - 5)!!$ trees in $\mathbb{G}_n$, meaning that $|\mathbb{G}_n|$ grows exponentially as $n$ increases.

**Definition 1.3.1.** $\mathcal{T}_m = (\mathbb{G}_n, m)$ or tree space is a graph with vertices labeled by $G \in \mathbb{G}_n$. The edges of this graph connect vertices which are “next to” each other, where “next to” is defined using some operation $m$ on a tree $G_x$ which converts $G_x$ into $G_y$. If $G_x$ can be converted to $G_y$ using just one operation, then an edge is drawn in $\mathcal{T}_m$ connecting the vertices $G_x$ and $G_y$.

**Definition 1.3.2.** Let $d$ be a discrete tree metric. the 1-neighborhood (or simply neighborhood) of $T$ is the set of all trees $T_m$ where $d(T, T_m) \leq 1$.

In 1981, Robinson and Foulds described a metric for unweighted trees (of any valency), subsequently named the Robinson-Foulds (RF) distance [65]. The RF distance measures the dissimilarity between two trees by determining the number of edges with which the two trees differ. Formally,

**Definition 1.3.3.** Given two trees $T_1, T_2 \in \mathbb{G}_n$, the Robinson-Foulds distance, $RF(T_1, T_2)$, is the minimal number of contractions and refinements necessary to convert $T_1$ to $T_2$. 

Figure 1.3: NNI operation. There are two possible NNI operations a given edge, and the two subsequent trees in turn differ from each other by one NNI move.

Given a set of graphs \([G_1, G_2, \ldots, G_n] \in \mathbb{G}\) with \(n\) leaves labeled by \(X\), then \(T_{RF}\) is the graph of \(\mathbb{G}\) with vertices labeled by \(G\) and edges connecting vertices which are “neighbors”—distance one—under RF.

Robinson and Foulds showed that, given two trees \(T_1\) and \(T_2\) with \(n\) labeled leaves, \(RF(T_1, T_2) \leq 2n - 6\), thus \(\Delta(T_{RF}) \leq 2n - 6\); and Day [25] found an algorithm to compute the RF distance in linear time.

**Definition 1.3.4.** [2] A forced contraction is an operation on a tree whereby a vertex \(v_i\) of valency two is removed along with its incident edges. The vertices which were previously adjacent to \(v_i\) are then connected by a new edge.

The Robinson-Foulds metric is used due to its linear running time, but other metrics are also popular. They include three metrics: NNI, SPR and TBR, which we will now discuss.

**Definition 1.3.5.** [2] The Nearest Neighbor Interchange (NNI) move swaps any two subtrees connected to opposite ends of an edge (see Figure 1.3).
For any given edge there are two possible NNI moves, thus given a tree $T_1$, two trees, $T_2$ and $T_3$, are reachable from $T_1$ by NNI operations on a single edge edge of $T_1$. Each of the trees $T_1, T_2, T_3$ is reachable from each of the others via one NNI move (see Figure 1.3).

Robinson [65] showed that $\mathcal{T}_{\text{NNI}}$ (that is, tree space under NNI) is connected. He also found the diameter of $\mathcal{T}_{\text{NNI}}$,

$$\Delta \mathcal{T}_{\text{NNI}} \leq 1/2(n - 2)(n - 3)$$

where $n$ is the number of taxa. This result was improved [57] to

$$\frac{n - 2}{4} \log \left( \frac{2\sqrt{2}}{3e} (n - 2) \right) \leq \Delta \mathcal{T}_{\text{NNI}} \leq n \log_2 n + O(n).$$

Robinson also showed that the 1-neighborhood of $T \in \mathcal{T}_{\text{NNI}}$ is $2n - 6$, and if $w_2$ contains all the trees reachable from $T$ with two NNI operations, then

$$2n^2 - 10n + 8 \leq w_2 \leq 2n^2 - 8n$$

if $n$ is even, and

$$2n^2 - 10n + 8 \leq w_2 \leq 2n^2 - 8n - 2$$

if $n$ is odd.

DasGupta, et al., expanding on the work of Li, et al. [57], found that computing the NNI distance between two labeled trees is NP-Complete [24]. They did this by showing a reduction from the X3C problem.

NNI operations are computationally simple, so finding the NNI neighbors of a tree $T_i$ is straightforward, but finding the NNI distance, or shortest path between two trees, is difficult due to the large number of possible paths and the structure of the space. This is also the case for the following operations.

Two other operations which are related to NNI are Subtree Prune and Regraft and Tree Bisection and Reconnection.
Figure 1.4: SPR and TBR operations. In SPR, the pruned subtree is reconnected to any vertex in the tree by the same vertex in the subtree, whereas in TBR it is reconnected using any vertex in the subtree.

**Definition 1.3.6.** [2] A Subtree Prune and Regraft (SPR) operation on a tree $T$ involves pruning a subtree $t$ by cutting one edge $e$ from $T$, where $e$ is incident on $t$ at vertex $v_t$ and incident on $T-t$ at $v_T$. $T_{new}$ is then created by reconnecting $t$ to $T-t$ with a new edge incident on $v_t$ and a new vertex in $v'_T$ which is created to bisect some edge in $T-t$ (see Figure 1.4). In addition, $v_T$ is removed by a forced contraction. This preserves the valency of $T_{new}$. The SPR distance ($d_{SPR}$) between two trees is the minimal number of SPR operations that transforms the first tree into the second.

**Definition 1.3.7.** [2] A Tree Bisection and Reconnection (TBR) operation on a tree $T$ involves pruning a subtree $t$ by removing an edge $e$ from $T$, where $e$ is incident on $t$ at vertex $v_t$ and incident on $T-t$ at $v_T$. $t$ is then reconnected to $T-t$ to create $T_{new}$ in the following manner. First an edge in $T-t$ is bisected with a new vertex $v'_T$. $v'_T$ is then connected by an edge to a new vertex $v'_t$ which is created to bisect some edge in $t$ (see Figure 1.4). In addition, $v_T$ and $v_t$ are removed by forced contractions. This preserves the valency of $T_{new}$. The TBR distance ($d_{TBR}$) between two trees is the minimal number of TBR operations that transforms the first tree into the second.
The SPR distance is especially compelling biologically because it models horizontal gene transfer and recombination events [2].

Interestingly, all NNI operations can be viewed as SPR operations, and all SPR operations as TBR operations [58], thus

\[ NNI \subseteq SPR \subseteq TBR. \]  

(1.1)

This is obvious for SPR and TBR, since SPR is just a special case of TBR where the new edge is always added at a specific vertex in \( t \) (the one from which the original edge was removed). Likewise, an NNI operation can be considered as an SPR operation where the pruned subtree is regrafted to an edge incident on the edge created by the forced contraction which removed the extraneous vertex from \( T - t \). As one can “move” through tree space using these operations, they will occasionally be referred to as “moves” as well as “operations” in this text.

As \( T_{NNI} \) is connected [64], this subset relation implies that \( T_{SPR} \) and \( T_{TBR} \) are also connected, thus search algorithms can be used on all of these spaces. It also means that the 1-neighborhood of a vertex, \( v \) under NNI is properly contained in the 1-neighborhood of \( v \) under SPR, which is, in turn, properly contained in the 1-neighborhood of \( v \) under TBR. Furthermore, it implies that one can “travel” farther within tree space with the same number of moves using TBR than SPR, and farther again with the same number of moves using SPR than NNI. This has important implications for hill-climbing searches, which can get trapped in local optima (see discussion below).

Bordewich and Semple [13] noted that the NNI and TBR operations are identical on rooted and unrooted trees, if the root is not included in the pruned subtree. To clarify, in the NNI operation there is no pruned subtree, and in the TBR operation either of the two
trees created after the pruning can be the subtree. Both operations are (or can be made to be) symmetric with regard to the root. However, for the SPR operation the rooted and unrooted versions are not identical if there is not a pendant vertex hanging from the root. In this case, because the pruned subtree must be reattached using a specific vertex, the operation is not symmetric about the root.

**Theorem 1.3.1.** [13] Given two rooted trees $T_1$ and $T_2$, and two unrooted trees $T'_1$ and $T'_2$ created from $T_1$ and $T_2$ by hanging pendant edges from the root,

\[
d_{\text{NNI}}(T_1, T_2) = d_{\text{NNI}}(T'_1, T'_2) \quad \text{(1.2)}
\]

\[
d_{\text{TBR}}(T_1, T_2) = d_{\text{TBR}}(T'_1, T'_2) \quad \text{(1.3)}
\]

\[
d_{\text{TBR}}(T'_1, T'_2) \leq d_{\text{uSPR}}(T'_1, T'_2) \leq d_{\text{rSPR}}(T_1, T_2) \leq d_{\text{NNI}}(T_1, T_2) \quad \text{(1.4)}
\]

Moreover, each of the inequalities in (1.4) can be strict.

Regarding the diameter of tree space $\Delta \mathcal{T}$, Eq. (1.1) also implies that

\[
\Delta \mathcal{T}_{\text{NNI}} \geq \Delta \mathcal{T}_{\text{SPR}} \geq \Delta \mathcal{T}_{\text{TBR}}
\]

and in fact it was shown [2] that

\[
n/2 - o(n) < \Delta \mathcal{T}_{\text{SPR}} \leq n - 3 \quad \text{and}
\]

\[
n/4 - o(n) < \Delta \mathcal{T}_{\text{TBR}} \leq n - 3.
\]

Allen and Steel found [2] that the size of the 1-neighborhood is $2(n-3)(2n-7)$ under SPR and is bounded above by $(2n-3)(n-3)^2$ under TBR. These neighborhoods are significantly larger than the size of the 1-neighborhood under NNI (recall that it is $2n - 6$), thus the topology of the SPR and TBR tree spaces are significantly different than that of the NNI.
Figure 1.5: $p$-ECR (here 2-ECR). In $p$-ECR, $p$ edges are contracted and the tree is then refined by adding $p$ edges to any unresolved vertices.

The fact that the tree spaces are the same size while the neighborhoods are much larger means that the SPR and TBR spaces are significantly more interconnected, and it is much quicker to move from one side of the space to the other than with NNI operations. In fact, TBR and SPR searches are significantly more exhaustive at each step, and therefore often preferred. However, even TBR searches can get stuck in local optima. This has lead to the creation of other transformation operations on trees. Ganapathy, et al. introduced one such operation, $p$-ECR, in two papers [34, 35] in 2004.

**Definition 1.3.8.** A $p$-ECR operation involves first contracting $p$ adjacent edges in a graph. The graph is then refined by adding $p$ edges to any unresolved vertices (vertices with valency greater than 3). (See Figure 1.5)

Note that NNI is equivalent to 1-ECR. Ganapathy, et al. [35] found that

$$\Delta T_{p-ECR} = \Theta\left(\frac{n \log n}{p \log p}\right),$$

and that the $p$-ECR neighborhood is $\Omega(n^{p2^p})$.

### 1.4 Agreement Forests

Hein, et al. [44] introduced the maximum agreement forest (MAF) and showed a polynomial time reduction from X3C to MAF, which has been a useful tool in proving that the SPR
Definition 1.4.1. \cite{44} Given two trees, $T_1$ and $T_2$ on leaves $L$, an agreement forest is a set of trees $F = T_3, T_4, \ldots, T_n$ that can be obtained from both $T_1$ and $T_2$ by deleting $q$ edges from each tree and using forced contractions to eliminate any vertices of valency less than three from $T_i$.

Definition 1.4.2. \cite{44} A maximum agreement forest (MAF) is an agreement forest $F$ for $T_1$ and $T_2$ such that $|F|$ is minimal.

Allen and Steel used MAFs to show that determining $d_{TBR}(T_1, T_2)$, where $T_1$ and $T_2$ are two unrooted trees on the same leaf set, $L$, is NP-hard. They did this by first proving that the MAF for $T_1$ and $T_2$ is one more than their TBR distance. They defined $m(T_1, T_2) = (|MAF(T_1, T_2)| - 1)$ and showed that $m(T_1, T_2)$ is a metric. Then, using induction on $k = d_{TBR}(T_1, T_2)$, they then proved that $d_{TBR}(T_1, T_2) \geq m(T_1, T_2)$. Next, they used induction on $m(T_1, T_2)$ to show that $d_{TBR}(T_1, T_2) \leq m(T_1, T_2)$. By combining this finding with results from \cite{44}, they showed that computing the TBR distance is NP-hard.

Using MAFs, Hein et al. had shown by reduction to X3C that computing the SPR distance is NP-hard. However, Allen and Steel found a rooted counter-example to Hein et al.’s argument (this is related to the inequality between rSPR and uSPR noted in Eq. (1.4)). They then pointed out that the same argument could instead be used to show that the TBR distance is NP-hard. They then went on to show that the TBR distance is fixed-parameter tractable, where the parameter to fix is the maximum distance between the trees.

Following up on this finding, and using similar arguments, Bordewich and Semple found that computing the rSPR (rooted SPR) distance is also NP-complete \cite{13}. They also showed that, analogously to the TBR distance, computing the rSPR distance is fixed-parameter tractable.
tractable. This left open the question of whether computing the uSPR distance is NP-hard. It was shown to be so by Hickey, et al. [47].

Given these results and the necessity of determining the distance between two trees, approximation algorithms become an important tool. A good example of an approximation for $d_{rSPR}$ was given by Bonet et al. [11]. They did this by creating an approximate MAF for $T_1$ and $T_2$, where the size of the agreement forest they found is not larger than five times the size of the MAF. In addition, they were able to show that their algorithm could run in linear time. In a follow-up, Bordewich et al. [12] found a polynomial-time 3-approximation for the $d_{rSPR}$.

1.5 Tree Space

As discussed above, determining the optimal phylogeny for a given set of taxa is computationally difficult [33] and is most often posited as a search problem. The focus of our work on is not on discovering or elaborating improved algorithmic means of carrying out this task, rather to better explore and illuminate the search space, both to improve existing search strategies and in the hope that the increased knowledge of the space, itself, might allow more efficient methods to be discovered.

The topology of tree space was first investigated by Maddison [58]. Maddison defined an island as a group of trees with scores less than some number $L$, all of which can be reached from each other via some operation (in his paper, NNI, SPR and TBR) without visiting a tree with score greater than $L$. He used real data and found that islands existed for each of the three operations. Within an island, the trees might be at a distance greater than one apart, and for a given $L$, there were several islands, each separated from the others by
paths through “valleys” of several trees with higher scores. In essence, these islands are local minima, so Maddison’s empirical study showed that all three popular rearrangement operations can get stuck in local minima on real data. He also found that trees within islands differed from each other less than trees in different islands, so that there may be several topologies of parsimonious trees for a given set of taxa.

Kirkup and Kim [53], investigated tree space under NNI, SPR and TBR. Using real data, they ran heuristic hill-climbing searches on tree space using increasing numbers of taxa and found that as the number of taxa increased so did local optima under all three operations. However, the increase was markedly stronger for NNI, so that it ceased to be useful for searching at a fairly low number of taxa. They also looked at the attraction basins for each local minimum. As the number of taxa increased, the size of the attraction basins decreased, implying that there are more local minima and those minima are increasing spread out over the search space. Using Robinson-Foulds as a metric, and as with Maddison, they found that the topologies of the trees in disparate minima were significant, and again especially so for NNI. In general, they recommended against using NNI for heuristic searches, but found that SPR and TBR are largely comparable.

In their papers mentioned above, Ganapathy, et al. [34, 35] introduced the $p$-ECR operation and compared it to TBR. They showed that for every $n$ and every $1 < p < n - 3$, there are trees whose $p$-ECR distance is less than their TBR distance, and vice versa [34]. In fact the overlap was computed to be $O(\min(n2^p, n^2p))$ They also compared TBR neighborhoods with 2-ECR neighborhoods and found them to be significantly different, suggesting that more efficient searches might be performed by interchanging TBR and $p$-ECR moves.

Since search methods based on subtree transfer operations (NNI, SPR, TBR and $p$-ECR) are subject to getting stuck in local optima, it has often been the case that researchers will
end up with several different trees all with similar scores. When this occurs, the trees can either be given confidence intervals or some sort of consensus tree can be created. Felsenstein [30] proposed the bootstrap to estimate a confidence interval. Bootstrapping involves resampling the data in order to infer the variability of the distribution. If the function yielding some parameter on the data is too difficult to assign a standard error to, or the distribution of the data is unknown, then the data can be repeatedly resampled and the function repeatedly run. These repeated runs give estimates of the parameter, yielding a distribution of the parameter, which in turn can be used to give the parameter a confidence level.

Felsenstein suggested bootstrapping on the characters (i.e. for each run leaving some characters out and sampling some others multiple times). In this case the phylogeny would be estimated a number of times and the resulting trees compared to give a confidence interval for a given topology or to give a majority-rule consensus tree. This is computationally intensive. Additionally, the character data itself must be carefully vetted to insure that they are independent, as non-independent characters will result in over-confidence.

1.6 The Generalized Tree Alignment Problem

Given a set of taxa and associated non-aligned sequence data, the generalized tree alignment problem (GTAP) is to find a phylogenetic tree (and associated vertex sequences) with the lowest alignment cost for that data under maximum parsimony. As noted above, finding the most-parsimonious tree for a set of taxa and sequences is NP-Hard, and exact solutions for $|n| > 12$ are, as yet, intractable. Solutions have therefore relied on various heuristic search schemes. The search has, in turn, often relied on sequence alignment as a preliminary step.
This is especially so when the optimality criterion for the trees is maximum parsimony as defined above, which relies on the Hamming distances of character sequences, and necessitates those sequences being the same length. However, other schemes are available, such as Direct Optimization, which we will discuss below.

Building and scoring trees with parsimony requires that the sequences be comparable. As the sequences from different taxa are usually not the same length, one option is to align the sequences prior to their being assigned to the leaves of a tree. Doing this alignment allows for Hamming distances (or similar metrics) between the sequences to be used in scoring the trees. In alignment, gaps are inserted in order to arrange the characters in the sequences so that similar or identical characters occur in the same positions in different sequences. There are essentially two options for this process: informal alignment, or aligning by eye, which we will set aside as having ill-defined or idiosyncratic alignment criteria; and multiple sequence alignment, or aligning the sequence prior to tree building.

As an alternative to alignment followed by search, in Direct Optimization the sequences may be assigned to a tree and search and alignment carried out simultaneously. As discussed below, in Direct Optimization an alignment is not created, but an implied alignment may be generated once the optimal tree is found.

Although the specifics vary (see software descriptions below for more detail), in general multiple sequence alignment consists of multiple iterations of the following two-step process: First, pairwise distances of all the sequences are computed, and these distances are then used to construct a guide tree. This guide tree may or may not be refined upon further iterations of the process. Progressive alignments are built using this tree as a guide (hence “guide tree”), with passes both from the taxa to the root of the tree, and in the opposite direction.
Once the final alignment is created, a search is conducted using the aligned sequences as the leaf labelings for the generated phylogeny.

Direct Optimization [79], in contrast, never creates an alignment. Rather, at every step it conducts tree search, and the resulting trees are scored by making a pass on the tree from the leaves to the root, creating hypothetical ancestors for each set of sibling vertices and scoring the tree in the process. For scoring purposes, the ancestor sequence of each pair of nodes is found using a variant of the Needleman-Wunsch algorithm [60]. Note that this means the lengths of the sequences may differ from node to node in the tree. If desired, an implied alignment may be constructed [80], by conducting a second pass (from root to leaves). As the sequence at each internal node of the tree is generated based on its daughter nodes, this alignment is specific to the tree on which it is constructed, and may not match the implied alignment from even a very similar tree.

Despite there being many alignment schemes and several software packages available, there is no decided criterion for the “best” alignment or aligner. Rather, each aligner uses its own criteria. There are several criteria available, including computational time complexity, actual running time, pairwise matching, the length of the aligned sequences, and the length of the subsequent trees. As the goal of GTAP is to find the most-parsimonious relation between the taxa, for this paper we will use tree length as the scoring criterion for alignment. In other words, the alignments themselves will only be artifacts created in attempts to find parsimonious trees.
1.7 Models of Evolution

In 1969 Jukes and Cantor described a model of evolution, the Jukes-Cantor model [49]. In this model, character evolution from root to leaves is a Markov chain, thus the change along any edge relies only on the state of the parent node. In particular, Jukes and Cantor assumed that all bases occur with equal probability and that all substitutions from one base to another also occur with equal probability. Thus, along any edge with length $\lambda$, the chance of a mutation at any given site is $\lambda$, and the chance of a character changing to any particular character state is $\lambda/4$ (as there are four character states). I.e. there is a $\lambda/4$ chance that the character at site $x$ does not change across any given edge, and a $3\lambda/4$ chance that it does change.

We deal largely with unweighted trees, in which the edge lengths are not brought into consideration in any analyses. However, in dealing with models of evolution, and when simulating data, we use weighted trees.

**Definition 1.7.1.** A weighted tree is a tree on which each edge has been assigned some length or weight.

In evolutionary terms, the edge weight are a measure of how much change occurs between sequences at nodes subtending an edge. For our purposes the edge weights determine how much change occurs in a sequence as it is evolved across an edge when creating simulated data.
2 Minimal Paths through Tree Space

Hamiltonian Walks of Phylogenetic Tree Spaces

Abstract

We answer Bryant’s Combinatorial Challenge on minimal walks of phylogenetic tree space under the nearest neighbor interchange (NNI) metric. We show that the shortest path through the NNI-tree space of \( n \)-leaf trees is Hamiltonian for all \( n \). That is, there is a minimal path that visits all binary trees exactly once, under NNI moves.

2.1 Introduction

Phylogenetic trees depict the evolutionary relationships within a set of taxa, represented as leaf labels [72]. The trees may be rooted—in which case they illustrate the ancestry of the taxa—or unrooted. In this paper, we look at unrooted phylogenies.

Finding the tree that best fits the data, where the data is a set of taxa and ordered characters, is a central goal of evolutionary biology. However, the number of possible trees grows as an exponential function of the number of taxa, and finding the optimal tree under the criteria most used by biologists is NP-hard [33, 66]. Due to the size of the search space,
exhaustive search is often not possible, so heuristic search is often used to discover the best tree. To systematically traverse the space, it is necessary that it be arranged in some manner. A common arrangement is to link trees that are a single move apart under some tree rearrangement operation; the resulting graph is often called a tree space.

Focusing on trees that differ by a single nearest neighbor interchange (NNI) move, David Bryant asked for the length of the shortest walk that visits all trees in NNI tree space [15]. It is known that using metrics that yield more neighbors than NNI (namely subtree prune and regraft (SPR) and tree bisection and reconnection (TBR)) have the shortest walk possible: a Hamiltonian path [16]. Previous to our current work, the best known NNI-walk of all binary trees visited every tree at most twice [16].

We show that, for all $n$, a Hamiltonian path exists on the space of all binary trees on $n$ leaves under the NNI metric, settling Bryant’s challenge. We follow the strategy of previous work in expanding Hamiltonian paths on $n$-leaf trees to the space of all binary trees on $(n+1)$ leaves [16]. This idea does not work directly for NNI-walks but can be employed with a subtle twist. Instead of developing walks on the expansion of a single $n$-leaf tree, we look at all trees that can be created from subsequent triples of $n$-leaf trees on the Hamiltonian path for the smaller space. Using the Hamiltonian path of the smaller space as a “backbone,” we can then “glue” together the unions of the expansions to form a Hamiltonian path on the $(n+1)$-leaf trees. Since every NNI move can be simulated by an SPR or TBR move, this paper provides an alternative proof for the existence of Hamiltonian paths for the SPR and TBR tree spaces.
Figure 2.1: The left side shows NNI transformations. To transform $T_1$ into $T_2$ or vice-versa, interchange the subtree $B$ with the subtree containing the leaf $l_{n+1}$. To transform $T_2$ into $T_3$ or vice-versa, either interchange the subtree containing the leaf $l_{n+1}$ with the subtree containing $C$ and $D$ or interchange the subtree containing $A$ with the subtree containing $B$. $T_1$, $T_2$ and $T_3$ are all neighbors in NNI tree space. The right side illustrates how a path in NNI tree space will be represented in this paper. Note that the top series of moves is equivalent to moving from $T_1$ to $T_2$ to $T_3$, as in the left side of the figure. The bottom right represents the same moves with the curved edge representing the path of the $(n+1)^{st}$ leaf, $l_{n+1}$.

\hspace*{1cm}

2.2 Background

We briefly define binary phylogenetic trees and the associated terms used in this chapter. For a more detailed treatment, refer to the introductory chapter or see Semple and Steel [72].

Phylogenetic trees depict evolutionary relationships between taxa placed at the leaves. Trees can be rooted, in which case they illustrate the ancestry of the taxa, or unrooted. We look at unrooted binary phylogenetic trees (hereafter referred to as trees).

We will use a well-known fact about the number of unrooted binary trees:

**Lemma 2.2.1.** [64]: For $n$ taxa, there are $(2n - 5)!! = (2n - 5)(2n - 3)\cdots5\cdot3\cdot1$ possible unrooted trees.

Note that for all $n \geq 4$, $(2n - 5)!!$ is divisible by three. We will use this characteristic of
tree space to partition paths of $n$-leaf trees into triples. We will also examine pairs of leaves, or cherries.

Bryant’s challenge focuses on Nearest Neighbor Interchange (NNI). Other popular metrics include Subtree Prune Regraft (SPR) and Tree Bisection and Reconnection (TBR) [2].

We note that the NNI operation is symmetric in that any NNI tree rearrangement operation can be reversed. These moves define neighborhoods on the space.

An $n$-leaf tree has $n - 3$ internal edges. Using an NNI move, a new tree can be formed by swapping one of the four subtrees on opposite sides of the internal edge (see Figure 2.1). Only two of these swaps will produce new trees, and thus every $n$-leaf tree has $2(n - 3)$ neighbors in NNI tree space.

The challenge on which we focus is phrased in terms of “walks,” we will use this term interchangeably with the common term from graph theory: “paths.”

**Definition 2.2.1.** [15]: An NNI-walk is a sequence $T_1, T_2, \ldots, T_k$ of unrooted binary phylogenetic trees where each consecutive pair of trees differ by a single NNI move.

**Definition 2.2.2.** [22]: A Hamiltonian path in a graph is a simple path that visits every node exactly once. This path can be represented as an ordered set of nodes, $v_1, v_2, \ldots, v_n$, where $v_i$ is connected to $v_{i+1}$ by an edge.

Determining whether an arbitrary graph has a Hamiltonian path is NP-hard [22]. However, for many classes of graphs (for example, complete graphs), Hamiltonicity can be determined easily in polynomial time.
2.3 Main results

We prove that a Hamiltonian path exists through the set of $n$-leaf trees under the NNI metric, for all $n$. The proof constructs a Hamiltonian path of the $(n+1)$-leaf tree space from a Hamiltonian path of the $n$-leaf tree space (see Figure 2.2). This is done by taking subsequent triples from the path of the $n$-leaf tree space and constructing a path through all $(n+1)$-leaf trees that be created from those three trees (formally defined as the “expansion” of trees, below). Since every $(n+1)$-leaf tree belongs to exactly one such expansion of a triple, linking the paths of the expansions yields a path that visits every $(n+1)$-leaf tree exactly once.

**Definition 2.3.1.** Let $T$ be an $n$-leaf tree and $e$ an edge of $T$. The expansion of an edge, $e$, is the $(n+1)$-leaf tree, $T(e)$, generated when a new leaf is added to that edge. Let the expansion of an $n$-leaf tree, $T$, be the set of $(n+1)$-leaf trees that can be generated from expanding all edges of $T$ (see Figure 2.3).
If two trees differ by only a single NNI move, then the edges of the trees are identical, except for a single edge, that we call the “edge of difference.” Formally:

**Definition 2.3.2.** Consider two trees, $T_1$ and $T_2$, that differ by one NNI move. Let $e_d$, the edge of difference, be the single edge in the symmetric difference between the set of edges of $T_1$ and the set of edges of $T_2$.

We note that the size of the expansion of an $n$-leaf tree is independent of the given tree topology and depends only on the number of internal edges. Likewise, in a binary tree, the number of internal edges is a function of the number of leaves. For a given tree $T$ with $n$ leaves, there are $2n - 3$ trees with $n + 1$ leaves contained in the expansion of $T$. We first prove several useful lemmas about expansions of edges:

**Lemma 2.3.1.** Let $T$ be an unrooted binary tree, and let $e_1$ and $e_2$ be adjacent edges on $T$. $T(e_1)$ and $T(e_2)$ differ by one NNI move.

**Proof.** Let $e_1$ and $e_2$ be adjacent edges in an $n$-leaf tree. Let $S$ be the subtree whose root edge is incident with $e_1$ and $e_2$. The addition of a new leaf, $l_{n+1}$, creates two new edges: $e_3$, which connects the new leaf node to the tree and $e_4$, which separates $S$ and $l_{n+1}$. In $T(e_1)$, $l_{n+1}$ is between $e_1$ and $e_4$ and $S$ is between $e_4$ and $e_2$. The opposite occurs in the $T(e_2)$. In that case, $S$ is between $e_1$ and $e_4$ and $l_{n+1}$ is between $e_4$ and $e_2$. That is, $T(e_1)$ and $T(e_2)$ have the same tree topology save for the arrangement around $e_4$. Since the new taxon and the rooted subtree are on opposite sides of $e_4$, an internal branch, swapping them costs only one NNI move. Therefore, $T(e_1)$ and $T(e_2)$ differ by one NNI move. \qed
Figure 2.3: Expansion of an unrooted tree on \( n \) leaves to \((n + 1)\)-leaf trees. When \( n = 4 \), there are five possible edges to which to attach a new leaf, resulting in five 5-leaf trees in the expansion of the initial tree.

**Lemma 2.3.2.** Let \( T_1 \) and \( T_2 \) be two unrooted binary trees where \( T_1 \) and \( T_2 \) differ by one NNI move. Let \( e \) be an edge that is not the edge of difference, \( e_d \). \( T_1(e) \) and \( T_2(e) \) differ by one NNI move.

*Proof.* Let \( A, B, C, \) and \( D \) be the four subtrees whose root edges are incident to \( e_d \), between \( T_1 \) and \( T_2 \). By definition, the arrangement of the four subtrees in \( T_1 \) differ from their arrangement in \( T_2 \). Assume, without loss of generality, that \( e \) is an edge of \( A \), and denote by \( A' \) the subtree created by the addition of the new leaf to \( e \) in \( A \). Note that \( A' \) is identical in \( T_1(e) \) and \( T_2(e) \). The arrangement of \( A' \), \( B \), \( C \), and \( D \) around \( e_d \) is the only difference between the two trees. Therefore, \( T_1(e) \) and \( T_2(e) \) differ by one NNI move.

When focusing on a triple of consecutive trees on a Hamiltonian path of \( n \)-space, there are many subtrees which are identical across all three trees. The next lemma shows how the expansions of these subtrees can be traversed such that each node in the expansions is visited only once (see Figure 2.4):

**Lemma 2.3.3.** Let \( T_1 \), \( T_2 \), and \( T_3 \) be three unrooted binary \( n \)-leaf trees where \( T_1 \) and \( T_2 \) are NNI neighbors and where \( T_2 \) and \( T_3 \) are NNI neighbors. Let \( S_i \) be some rooted subtree on \( T_i \).
where \( i = 1, 2, \) or \( 3 \). If \( S_1 = S_2 = S_3 \), the union of the expansions of the edges in \( S_1, S_2, \) and \( S_3 \) has a Hamiltonian path such that the walk starts on \( T_i(p_i) \), where \( p_i \) is the root edge of \( S_i \), and ends on \( T_j(p_j) \), where \( p_j \) is the root edge of \( S_j \) and \( i \neq j \).

**Proof.** We proceed by induction on the size of the subtree.

**Base Case:** The subtree has two leaves and three edges: \( p_i \), which connects the root node to the internal node; \( l_i \), which connects the internal node to a leaf node; and \( r_i \), which connects the internal node to the other leaf node. All three edges are adjacent. By Lemma 2.3.1, the expansions of these edges are NNI neighbors. Since \( T_1 \) and \( T_2 \) are one NNI move apart, \( T_1(p_1) \) and \( T_2(p_2) \) are NNI neighbors by Lemma 2.3.2. The rest of the edges follow suit. That is, \( T_z(y_z) \) and \( T_{z+1}(y_{z+1}) \) are NNI neighbors where \( y \in \{ p, l, r \} \) and \( z \in \{ 1, 2 \} \).

We can enumerate the possible walks that start on \( T_i(p_i) \) and end at \( T_j(p_j) \) where \( i \neq j \) (see Figure 2.4). We identify the path through the \((n + 1)\)-leaf trees by the edge that is expanded:

- \( p_1 \rightarrow l_1 \rightarrow r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow l_3 \rightarrow l_2 \rightarrow p_2 \rightarrow p_3 \),
- \( p_1 \rightarrow l_1 \rightarrow r_1 \rightarrow r_2 \rightarrow r_3 \rightarrow p_3 \rightarrow l_3 \rightarrow l_2 \rightarrow p_2 \),
- \( p_2 \rightarrow r_2 \rightarrow r_1 \rightarrow p_1 \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow r_3 \rightarrow p_3 \).

We note that since the edges are not directed, each of the above three paths could be traversed in reverse. Thus, we have a Hamiltonian path of the expansions of the edges of the subtrees that begins on \( T_i(p_i) \) and ends on \( T_j(p_j) \) where \( i \neq j \).

**Inductive Step:** Assume that the subtree, \( S_i \), has three or more leaves and at least five edges: \( p_i \), which connects the root node to an internal node; and edges \( l_i \) and \( r_i \) which
are incident with $p_i$. By Lemma 2.3.1, the expansions of these edges are NNI neighbors. Further, since $T_1$ and $T_2$ are one NNI move apart, $T_1(p_1)$ and $T_2(p_2)$ are NNI neighbors by Lemma 2.3.2. The rest of the edges follow suit. That is, $T_z(y_z)$ and $T_{z+1}(y_{z+1})$ are NNI neighbors where $y \in \{p, l, r\}$ and $z \in \{1, 2\}$.

We show that a Hamiltonian path can start on $T_i(p_i)$ and end on $T_j(p_j)$ where $i \neq j$.

Without loss of generality, assume that $l_i$ is the root edge of the inner subtree, $C_i$. Let $T(C_1, C_2, C_3)$ be the union of the expansions of all the edges in $C_1$, $C_2$, and $C_3$ except for the expansions of two of the root edges, $l_i$ and $l_j$, whose visit we explicitly show. By the inductive hypothesis, a Hamiltonian path can start on $T_i(l_i)$, and end on $T_j(l_j)$ where $i \neq j$.

Case I: $S_i$ is a complex rooted subtree with a leaf attached to the root. $r_i$ connects the first internal node to a leaf (see Figure 2.4). The following are paths of the union starting at $T_i(p_i)$ and ending at $T_j(p_j)$, $i \neq j$:

- $p_1 \rightarrow r_1 \rightarrow l_1 \rightarrow T(C_1, C_2, C_3) \rightarrow l_3 \rightarrow r_3 \rightarrow r_2 \rightarrow p_2 \rightarrow p_3$,
- $p_1 \rightarrow r_1 \rightarrow l_1 \rightarrow T(C_1, C_2, C_3) \rightarrow l_2 \rightarrow r_2 \rightarrow r_3 \rightarrow p_3 \rightarrow p_2$,
- $p_2 \rightarrow p_1 \rightarrow r_1 \rightarrow r_2 \rightarrow l_2 \rightarrow T(C_1, C_2, C_3) \rightarrow l_3 \rightarrow r_3 \rightarrow p_3$.

Case II: $S_i$ is a complex rooted subtree with another complex rooted subtree attached to its root. $r_i$ connects the first internal node to another complex subtree, $D_i$. The following are paths of the union starting at $T_i(p_i)$ and ending at $T_j(p_j)$, $i \neq j$:

- $p_1 \rightarrow r_1 \rightarrow T(D_1, D_2, D_3) \rightarrow r_3 \rightarrow l_3 \rightarrow T(C_1, C_2, C_3) \rightarrow l_2 \rightarrow p_2 \rightarrow p_3$,
- $p_1 \rightarrow r_1 \rightarrow T(D_1, D_2, D_3) \rightarrow r_2 \rightarrow l_2 \rightarrow T(C_1, C_2, C_3) \rightarrow l_3 \rightarrow p_3 \rightarrow p_2$,
- $p_2 \rightarrow r_2 \rightarrow T(D_1, D_2, D_3) \rightarrow r_1 \rightarrow p_1 \rightarrow l_1 \rightarrow T(C_1, C_2, C_3) \rightarrow l_3 \rightarrow p_3$.

This completes the proof. □
Figure 2.4: Left: Base Case I for Lemma 2.3.3. The labels on the arrows and subtrees indicate the order of the traversal. \( S_x \) is a rooted cherry that is the same across \( T_1 \), \( T_2 \), and \( T_3 \). Note that the path through the expansions of the edges in \( S_x \) can be notated as a series of NNI moves, moving the leaf added in the expansions through the various edges in \( S_x \). Right: The three walks show that a Hamiltonian path can start at \( T_i(p_i) \) and end on \( T_j(p_j) \) where \( i \neq j \).

Lemma 2.3.3 focuses on the union of the expansions of edges in rooted subtrees that are identical across a triple of subsequent trees in a path of the \( n \)-leaf tree space, giving multiple paths that can traverse that union from different starting and stopping points. The following lemma shows how these paths can be “glued” together to form a path for the unions of the expansions of the complete trees. The difficulty in the proof is the “lining up” of the endpoints of paths to create a single longer path. Since Lemma 2.3.3 provides paths from any root edge of identical subtrees, it suffices to show how to traverse the edges of difference in the expansions.

**Lemma 2.3.4.** Let \( T_1 \), \( T_2 \), and \( T_3 \) be three unrooted binary trees where \( T_1 \) and \( T_2 \) are NNI neighbors and where \( T_2 \) and \( T_3 \) are NNI neighbors. For any edge \( e \) of \( T_1 \), there exists a Hamiltonian path of the union of the expansions of \( T_1 \), \( T_2 \), and \( T_3 \) starting at \( T_1(e) \).

**Proof.** Let \( T_1 \), \( T_2 \), and \( T_3 \) be three unrooted binary trees where \( T_1 \) and \( T_2 \) are NNI neighbors and where \( T_2 \) and \( T_3 \) are NNI neighbors. Let \( e_{d_{12}} \) be the edge of difference between \( T_1 \) and
Let $e_{d_{23}}$ be the edge of difference between $T_2$ and $T_3$. Let the expansion of a subtree, $S$, in an $n$-leaf tree, $T$, be the union of the expansions of the edges in $S$.

Let $e$ be an edge in $T_1$. We will construct a walk that traverses, exactly once, every tree in the union of the expansions of $T_1$, $T_2$, and $T_3$. The location of $e$, whose expansion, $T_1(e)$, is the starting point, determines our strategy for building the Hamiltonian path. Denote by $A$ and $B$ the subtrees that result from removing $e$ (but not its endpoints) from $T_1$. If one of the subtrees, say $A$, is identical across $T_1$, $T_2$, and $T_3$, then neither $e_{d_{12}}$ nor $e_{d_{23}}$ is in $A$, and by Lemma 2.3.3, there is a Hamiltonian path across the union of the expansions of $A_1$, $A_2$, and $A_3$, beginning at $e$ in $T_1$ and ending at $e$ in $T_i$ where $i = 2$ or $3$. So, assume that neither $A$ nor $B$ is identical across all trees. We proceed by cases on the relative locations of the edges of difference, $e_{d_{12}}$ or $e_{d_{23}}$, to $e$:

**Case I:** Assume that $e_{d_{12}}$ is on the path between $e$ and $e_{d_{23}}$ (that is, $e_{d_{12}}$ is the “closer” edge of difference to $e$). Let $e_1, \ldots, e_m$ be the path between $e$ and $e_{d_{12}}$ in $T_1$, and let $S_1, S_2, \ldots, S_{m-1}$ be the subtrees along the path. By hypothesis, the subtrees, $S_1, S_2, \ldots, S_{m-1}$, occur in all three trees. We apply Lemma 2.3.3 to each of the subtrees and link the resulting paths by visiting the the expansion of the subtree, $S_i$, followed by the expansion of the path edge, $e_{i+1}$, in $T_1$, $T_2$, and $T_3$, for each $i$, creating a path that extends to $T_2(e_{d_{12}})$ or $T_3(e_{d_{12}})$.

We may assume that our path thus far ends at $T_3(e_{d_{12}})$. Let $C$ denote the subtree, identical in $T_1$, $T_2$ and $T_3$, that we have traversed thus far. The root edge of $C$ is incident with $e_{d_{12}}$. Let $D$ be the subtree whose root edge is incident with $C$ in $T_2$ and $T_3$ (that is, $C$ and $D$ are on the “same side” of $e_{d_{12}}$ in $T_2$ and $T_3$) and let $F$ and $G$ be the remaining subtrees whose root edges are incident with $e_{d_{12}}$ in $T_2$ and $T_3$ (that is, $F$ and $G$ are on the “opposite side” of $e_{d_{12}}$ from $C$ and $D$ in $T_2$ and $T_3$). Since the root edges, $r_C$ and $r_D$, of $C$ and $D$ are adjacent in $T_3$, we can move from $T_3(r_C)$ to $T_3(r_D)$ without having to traverse
the expansion of \( e_{d_{12}} \) (that traversal is optional). If \( D \) does not contain \( e_{d_{23}} \), it is identical across all three trees and we can apply Lemma 2.3.3 to yield an extension of our path that traverses the union of the expansions of all three copies of \( D \), ending in \( T_1 \).

\( T_1 \) differs from \( T_3 \) around \( e_{d_{12}} \), so the root edge of \( C \) in \( T_1 \) is not incident with \( D \), but with one of the other two subtrees, say \( F \). To simplify the argument, we will assume that \( F \) does not contain the edge \( e_{d_{23}} \) (if \( F \) does contain \( e_{d_{23}} \), the proof follows by a slightly more complicated, but similar, argument). We can similarly extend our path to the union of the expansions of the subtree \( F \) in \( T_1 \), \( T_2 \), and \( T_3 \), ending in \( T_3 \). In addition, because \( D \) and \( F \) are on “opposite sides” of \( e_{d_{12}} \) in \( T_1 \), our path must first visit \( T_1(e_{d_{12}}) \).

We now have a path that traverses the expansion of three out of four of the subtrees whose root edges are incident with \( e_{d_{12}} \) and \( T_1(e_{d_{12}}) \). To reach the final of the four subtrees, \( G \), we cross \( T_3(e_{d_{12}}) \), then \( T_2(e_{d_{12}}) \). From there, we visit the expansion of the root edge of \( G \), which we traverse by an argument similar to the one above. We note that if \( D \), above, had contained \( e_{d_{23}} \), then we would use the argument for traversing \( T_1(e_{d_{12}}) \), \( T_2(e_{d_{12}}) \), and \( T_3(e_{d_{12}}) \) to traverse \( T_1(e_{d_{23}}) \), \( T_2(e_{d_{23}}) \), and \( T_3(e_{d_{23}}) \). Once we have traversed the expansion of the three copies of \( G \), we have a path that visits all the edges of \( T_1 \), \( T_2 \), and \( T_3 \), and thus all trees in the union of the expansions of those trees.

**Case II:** We must also consider the case where \( e \) lies on the path between the edges of differences, \( e_{d_{12}} \) and \( e_{d_{23}} \). While the argument for this case is similar to that above, there is the additional difficulty of “starting in the middle.” It is necessary to traverse the path from \( e \) to \( e_{d_{12}} \) and still have unvisited edges upon which to return so that the “other side” of \( e \) can also be visited. This can be accomplished by traversing only the path edges (and none of the attached subtrees) in the tree \( T_1 \) until the edge of difference is reached. The path is then built, as above, but on the return, the subtrees on the path are linked by visits to the
path edges in only $T_2$ and $T_3$. Once that section of the expansions of the trees has been visited, the remaining trees in the union of the expansions are visited (namely the union of the expansion of $B$ where $B$ is the subtree resulting from removing $e$ and that contains $e_{d_{23}}$).

The result is a Hamiltonian path of the union of the expansions of $T_1$, $T_2$, and $T_3$.

Case III: Note the special case where $e_{d_{12}} = e_{d_{23}}$. In this case, $T_1$ and $T_2$ are NNI neighbors, $T_2$ and $T_3$ are NNI neighbors, and $T_1$ and $T_3$ are NNI neighbors. Such a case is a simplified version of the previous cases, and thus is covered above.

\[ \Box \]

**Theorem 2.3.1.** For all $n$, there exists a Hamiltonian path through the $n$-leaf NNI tree space.

**Proof.** By induction on $n$, the number of leaves.

**Base Case:** When $n = 4$, $(2n - 5)!! = 3$. Let the four leaves be $a$, $b$, $c$, and $d$. Then, without loss of generality, $e_d$ in $T_1$ separates $a, b$ from $c, d$; in $T_2$ $e_d$ separates $a, c$ from $b, d$; and in $T_3$ $e_d$ separates $a, d$ from $b, c$. $T_1$ and $T_2$ are NNI neighbors, and $T_2$ and $T_3$ are NNI neighbors. By the previous lemma, there is a Hamiltonian path through the 4-leaf NNI tree space (see Figure 2.1).

**Inductive Step:** Assume there is a Hamiltonian path through the $n$-leaf NNI tree space. The walk visits the ordered set of trees, $T_1$, $T_2$, $T_3$, \ldots\ $T_{(2n-5)!!}$. By the definition of a Hamiltonian path, $T_x$ and $T_{x+1}$ are NNI neighbors where $1 \leq x < (2n - 5)!!$.

By the previous lemma, the union of the expansions of the triplet $T_y$, $T_{y+1}$, and $T_{y+2}$ has a Hamiltonian path where $y = 3z - 2$ and $1 \leq z \leq \frac{1}{3}(2n - 5)!!$.

Because $(2n - 5)!!$ is divisible by 3 when $n \geq 4$, there is an ordered set of successive triplets, $R_1, R_2, \ldots, R_{\frac{(2n-5)!!}{3}}$, where $R_1$ is the triplet of trees $T_1, T_2, T_3$, and $R_{\frac{(2n-5)!!}{3}}$ is the
triplet $T_{(2n-5)!-2}$, $T_{(2n-5)!-1}$, $T_{(2n-5)!}$. The unions of expansions on each of these triplets has a Hamiltonian path.

Consider $T_y$, the third tree in triplet $R_v$, and $T_{y+1}$, the first tree in $R_{v+1}$. $T_y$ and $T_{y+1}$ are NNI neighbors. Then, by Lemma 2.3.2, $T_y(e)$, where $e$ is not $e_d$, is an NNI neighbor of $T_{y+1}(e)$. The end of the Hamiltonian path through an expansion of a triplet can thus be connected to the beginning of the Hamiltonian path through the expansion of the succeeding triplet. As shown above, there is an ordered set of triplets which covers $n$-leaf tree space with a Hamiltonian path. The expansions of each of these triplets has a Hamiltonian path, and each of the walks can be linked by a single NNI move. Therefore, a Hamiltonian path exists through the $(n+1)$-leaf NNI tree space.

\[\square\]

### 2.4 Conclusion

We have shown that the shortest walk on the space of binary phylogenetic trees with $n$ leaves under the NNI metric is a Hamiltonian path. Since visiting each node exactly once is the minimal path length possible, this answers Bryant’s First Combinatorial Challenge on the length of the shortest walk of trees under the NNI metric. In addition, since every NNI move can be simulated by an SPR or TBR move, this also gives an alternative proof to the Hamiltonian paths of SPR and TBR tree space [16]. Our iterative approach to building a Hamiltonian path for the space of trees with $n+1$ leaves from a path of the smaller space of trees with $n$ leaves does not yield an algorithm for producing a Hamiltonian path directly nor do we see an obvious way to do this.
2.5 Acknowledgments

The authors would like to thank the anonymous reviewers for their thoughtful and helpful comments, which greatly improved this paper. For spirited discussions, the authors thank the members of the Tree Space Working Group at CUNY: Alan Joseph Caceres, Ann Marie Alcocer, Kadian Brown, Juan Castillo, Samantha Daley, John DeJesus, Michael Hintze, Daniele Ippolito, Jinnie Lee, Joan Marc, Oliver Mendez, Diquan Moore, and Rachel Spratt. This work was supported by grants from the US National Science Foundation programs in mathematical biology and computational mathematics (NSF #09-20920) and the New York City Louis Stokes Alliance for Minority Participation in Research (NSF #07-03449).
3 Hill-Climbing Search of Tree Space

Characterizing Local Optima for Maximum Parsimony

Abstract

We examine a well-known data set, the Rokas yeast data, and find that it contains multiple local optima both under NNI and SPR tree spaces. We then simulate data similar to the Rokas set and examine the landscape of the simulated data under SPR.

3.1 Introduction

Finding the optimal phylogeny, or evolutionary history, for a set of species is a central goal of biology. The maximum parsimony criterion for a phylogenetic tree is often used due to its simplicity in capturing evolutionary change and its ease of computation [31]. Given traits or characters for a set of taxa, the maximum parsimony criterion seeks the tree which has the minimal number of changes of character states across its edges. For any tree with $n$ taxa and sequences of length $k$, this score can be calculated in linear time in $n$ and $k$ [31]. However, to exactly find the optimal tree (that is, the tree with lowest maximum parsimony score) is computationally difficult [33]. The size of the search space of trees grows as the exponential

\footnote{Co-authors Ellen Urheim and Katherine St. John.}
of the number of taxa [64], and exact algorithms (e.g. [45, 46]) are thus only effective for small sets of taxa. Therefore, heuristic search methods are often used (e.g. [39, 61, 78]). The underlying structure of these methods follows the local search paradigm: at each step choose a “neighbor” of the current tree in the search space and repeat. In the simplest case, a “hill-climbing” approach is used, wherein the chosen tree at each step is the best-scoring neighbor, though more sophisticated techniques have been explored, such as simulated annealing [27] and Monte Carlo methods [61] (see [82] for a survey of popular approaches).

In this paper, we examine two popular ways to define neighbors of trees and their effectiveness for hill-climbing searching with well behaved data, NNI and SPR.

Empirical studies suggest that the search spaces resulting from these two operations differ greatly. Maddison [58] defined an island as a group of trees with scores less than some number $L$, all of which can be reached from each other via some operation without visiting a tree with score greater than $L$. Doing empirical studies on real data, he found that there were islands for all tree-manipulation operations he studied, including NNI and SPR.

Kirkup and Kim [53] investigated tree space under NNI and SPR. Using real data, they ran multiple heuristic hill-climbing searches on tree space using increasing numbers of taxa and found that, as the number of taxa increased, so did the number of local optima under both operations. However, the increase was markedly stronger for NNI, so that it ceased to be useful for searching at a fairly low number of taxa.

Whelan and Money [59] examined hill-climbing for another popular optimality criterion: the maximum likelihood criterion. They analyzed in-depth the well-studied yeast data set of Rokas et al. [67], which consists of 106 genes for 8 taxa of yeast. They scored all 10,395 possible trees under maximum likelihood under different models of evolution. While the search spaces for the genes under NNI yield multiple local optima, under SPR only 1 of the
106 genes had a search space with more than one local optimum under the GTR+Γ model of evolution, and only 5 had more than one local optimum under the simpler Jukes-Cantor model.

In this paper, we show that sequences simulated generated under the Jukes-Cantor model of evolution can yield a search space under SPR with a single local optimum and we calculate the bounds on parameters to guarantee this with high probability, echoing the results of Atteson [3] on the Neighbor Joining tree reconstruction algorithm, as well as work on balanced minimum evolution [14]. Namely, if character sequences are generated under the Jukes-Cantor model with suitable parameters from a model tree, then any search technique that chooses the best tree, with high probability, from the SPR neighbors, will find the model tree with high probability. We include experimental results on both biological and simulated data.

**Local Search and Hill-Climbing**

Given a search space, it is sometime fruitful to attempt the simplest, most naive search methods, as they are easy to implement and will often quickly resolve to an answer. The standard naive search algorithm is *hill-climbing*, wherein the neighborhood of a particular location is examined in order to determine the next move in the search. At each step, the location is updated to the most optimal location in the neighborhood, and the process is repeated. Hill-climbing is an example of a *local search* algorithm, as only the immediate neighborhood of a location is examined. Local search is powerful due to its simplicity, which makes it both efficient and easy to implement, but that same simplicity can be a disadvantage, as the search is subject to local minima. That is, because at any given step
For a fixed unrooted caterpillar tree, $T$, on 7 leaves, we let $\chi$ be a character sequence that perfectly defines $T$. Each node is a tree on $\chi$ and is colored by its parsimony score, with lighter colors being worse scores. The center node of each image is $T$. Nodes are arranged in concentric circles by their distance in space from $T$. Local optima and plateaus (which act as local optima for hill-climbing) are outlined with a thicker rule. Left: NNI space. Note the number of local minima, even among the worst-scoring trees, and those farthest from $T$. Right: SPR space. There is only one local minimum, $T$.

only the immediate neighborhood is examined, the current location may appear to the search algorithm to be a global optimum, but be, in fact, only a local optimum. (See [22] for a more in-depth discussion.)

### 3.2 Local and Global Optima for Perfect Data

We created perfect binary data (yielding a perfect phylogeny, $T$) for seven taxa and scored all 945 trees. This was a tractable number of trees, and we were thus able to graph both NNI- and SPR-spaces. We found that, while the SPR-space had a single local optimum, NNI-space had both multiple local optima and multiple plateaus. Some of these plateaus
occurred on least-optimal trees, and those at a maximal distance from $T$. It is thus the case that NNI is not amenable to hill-climbing searches, even on perfect data.

### 3.3 Empirical Results

Given that the landscape of NNI-space is hilly even for perfect data, whereas SPR is smooth, we next examined SPR-space with both real and simulated data.

**Empirical Results on Yeast Data Set**

We explored the maximum parsimony optimality criterion on the yeast data set [67] and found similar empirical results to those of Money and Whelan [59]: a majority of the genes—75 of 106—yield SPR-spaces with a single local optimum. An additional 15 genes have multiple globally optimal trees but no further local optima. Also, in every one of these cases the optimal trees are grouped on a single island. Thus hill-climbing search will work. There are a further six genes that have a single optimum, but these genes also contain at least one plateau, which will act as a local optimum for hill-climbing search.

**Empirical Results on Simulated Sequences**

We generated simulated data under the Jukes-Cantor model of evolution and evaluated the number of optima found for SPR-space. By generating aligned data on a small space, we were able to score the neighborhoods of every tree searching for local minima. Using simulated data allowed us to vary the amount of change in each tree, so we were able to examine how the ratio between sequence length and edge length, $\lambda$, affects the topography of the search space.
Figure 3.2: Graph of edge length ($\lambda$) versus average number of optima for 8-taxon SPR space. 20 simulated genes created on two topologies were tested for local optima. For graphing purposes, plateaus were counted as local optima and genes with multiple global optimal trees were given a score of 0. (All genes with $\lambda = .001$ had multiple optimal trees.) No trees with $\lambda < .55$ had multiple local optima.

We simulated genes from the Rokas et al. data set, which allowed us to plot $\lambda$ vs. the number of local optima. For $\lambda \in (.001, .05, .1, .15, \ldots, 8)$ we created two trees, a caterpillar and a balanced tree. We then evolved sequences down those trees using DAWG. The sequences were of length 1000, had no gaps and used the Jukes-Cantor model of evolution. The DAWG commands were:

Treescale = 1.0, Length = 1000,

Model="GTR", Freqs = \{0.25, 0.25, 0.25, 0.25\},

Params = \{1.0, 1.0, 1.0, 1.0, 1.0, 1.0\}, GapParams = \{0, 0, 0\}.

For each tree we ran 10 simulations, giving us a total of 340 simulated genes. As no genes had multiple global optima with differing multiple local optima, and in order to distinguish
between local and global optima in the results, we considered a gene with multiple global optima to have 0 local optima but a gene with a single global optimum to have one local optimum. We counted plateaus as local optima, as they behave like local optima for local search.

For trees with $\lambda = .001$, every tree had multiple global optima. For other values of $\lambda$ less than .55, there were either single or multiple global optima. The average number of local optima did not exceed 1 until $\lambda > .7$.

3.4 Discussion

The large number of local optima and plateaus, even on perfect data, some of which are on least-optimal trees, suggests that using multiple NNI moves as a substitute for a single SPR move is a misguided approach. One could argue that making multiple random NNI moves could work as a sort of simulated annealing, thus allowing local optima to be overcome. However, to walk an SPR neighborhood with NNI moves takes $\Theta(n^2)$ [40] operations. In other words, it varies with the number of taxa, and we have shown that, due to the number of plateaus, simply taking multiple NNI neighborhoods as a proxy SPR, search may get stuck in local optima even for well-behaved data.

3.5 Future Work

We have preliminary theoretical results that SPR-space has no local minima or plateaus for perfect data. Additional work must be done to prove that NNI-space will always have local minima or plateaus for perfect data.
We plan to use additional simulated runs in an attempt to parametrize when hill-climbing works, both under Jukes-Cantor, and for more sophisticated models.

Beyond the scope of this paper, but intriguing, nonetheless, is the question of whether similar results hold for the maximum likelihood optimality criterion. This is not clear, since it is possible to have multiple local optima in branch lengths for a single tree topology [75].

### 3.6 Acknowledgments

The authors would like to thank Mike Charleston, Kebaugh Gordon, Barbara Holland, and Ward Wheeler for insightful and helpful comments.
4 Landscapes of Tree Space

Towards Improving Searches for Optimal Phylogenies

Abstract

Finding the optimal evolutionary history for a set of taxa is a challenging computational problem, even when restricting possible solutions to be “tree-like” and focusing on the maximum parsimony optimality criterion. This has led to much work on using heuristic tree searches to find approximate solutions. We present an approach for finding exact optimal solutions that employs and complements the current heuristic methods for finding optimal trees. Given a set of taxa and a set of aligned sequences of characters, there are subsets of characters that are compatible, and for each such subset there is an associated (possibly partially resolved) phylogeny with edges corresponding to each character state change. These perfect phylogenies serve as anchor trees for

1Accepted (with major revisions requested) for publication to Systematic Biology; co-authors Katherine St. John and Ward C. Wheeler.
our constrained search space. We show that, for compatible character sequences, the parsimony score of any tree, $T$, is at least the parsimony score of the anchor tree plus the distance between $T$ and the anchor tree. As maximum parsimony is additive, the sum of the lower bounds on compatible character partitions provides a lower bound on the complete alignment of characters. This yields a region in the space of trees within which the best tree is guaranteed to be found; limiting the search for the optimal tree to this region could significantly reduce the number of trees that must be examined in a search of the space of trees. We analyze this method empirically using four different biological data sets as well as surveying 400 data sets from the TreeBASE repository, demonstrating the effectiveness of our technique in reducing the number of steps in exact heuristic searches for trees under maximum parsimony.

Phylogenetic trees represent the evolutionary relationships among taxa, one of the primary endeavors of modern biology. Building on the principle of Occam’s Razor, several researchers have proposed [17, 29, 31] choosing the “most parsimonious” tree: the one with the least amount of evolutionary change across the edges. Foulds and Graham [33] showed that the problem of finding the most parsimonious tree for a given set of taxa—that is, determining the topology of the tree as well as the positions of the taxa on the leaves—is a computationally hard problem. When the given characters are compatible (that is, can be fitted onto the same tree with no homoplasy), the resulting tree is optimal (often called “perfect”), and can be quickly computed [41].
The naive approach to finding the exact optimal tree is to enumerate all possible tree topologies, score those trees, and choose the tree with the best score. Since the number of tree topologies grows exponentially in the number of taxa ($(2n-5)!!$ possible trees for $n$ taxa—[71, 64]), this is not possible for most data sets. Hendy and Penny [46, 62] developed a branch-and-bound algorithm [55] to find the most parsimonious tree. This method builds and scores a tree to determine an upper bound, and then backtracks on the build process, creating new trees by adding in taxa one by one. If the insertion of a new taxon yields a tree with a score greater than the bound, all trees that extend that tree will also have scores greater than the bound and therefore need not be explored. This pruning of a branch of the search can reduce the number of trees that need to be examined by a constant factor, but may still necessitate much of the search space being explored.

Due to the difficulty of finding the optimal tree(s) under the maximum-parsimony criterion, a great deal of work has been dedicated to tree search, and—because of the size of the search space—specifically to heuristic search (e.g., [78, 39]). Most of the heuristic search software follows a local search (or hill-climbing) paradigm: At each step, choose a neighboring tree (usually the best scoring) and repeat until a (local) optimum has been found or time has been exhausted. These search programs can be affected greatly by the starting point and choice of neighbors [18, 53]. Since the “ruggedness” of the space is not known in general [4], it is possible to have many local optima, stopping the search prematurely. Multiple starting points and sophisticated search techniques have been employed to make the most of the small sampling possible of this incredibly large space [78, 39].

Finding a global lower bound on the parsimony score, with respect to a given character sequence has been studied. A simple lower bound is the sum of the number of character states witnessed across all characters. Hendy, et al. [45] improved this by grouping characters
Figure 4.1: In a), the lower bound is graphed. The lower set of filled dots are the scores on the compatible subsets for each of three anchor trees, and the upper filled dots are the scores for each of those trees on the complete character set, $C$. The empty dots are other possible trees on the same set of labeled taxa. Initially, the trees that need to be considered fall below the solid black line. Once a tree with a lower score is found, the bound can be lowered, as with the dashed line. In b), the black lines are the summed distance from the anchor trees to the original best-scoring tree and the original space that must be exhaustively searched is in grey. Once a tree with a lower score is found, the summed distance is reduced. The grey lines are the summed distance to the new best-scoring tree, and the decreased search area is bounded by the dashed rule.

into small incompatible sets, each of whose lower bound can be calculated quickly. They show that the sum of these lower bounds gives a lower bound on the overall parsimony score. Holland, et al. [48] used the partition theorem of Hendy, et al. to develop a bound based on pairs of characters. The elegance of their approach is that it avoids the search for incompatible pairs and shows that the average over all pairs gives a lower bound.

Our approach follows the insightful work of Maddison [58] on “phylogenetic islands,” or connected regions of the space of trees that contain optimal or nearly optimal trees (see Figure 4.1). We quantify where these regions will occur by exploiting the underlying mathematical structure of the space of trees. In the spirit of the work of Hendy and Penny, we seek to bound the size of the space of trees that needs to be searched. Instead of a global lower bound, we develop a bound on all trees based on their distance to a set of “anchor trees,” or perfect phylogenies, that are compatible with subsets of characters. If the input
character sequences are compatible, then, as noted above, the exact optimal tree can be found quickly [41]. By carefully defining “neighbors,” we can show that, for non-additive characters (or for additive characters when the number of character states is two), trees that are first neighbors of the optimal tree have a score that is at least one more than the optimal, trees that are second neighbors of the optimal tree have a score that is at least two more than the optimal, and, similarly, trees that are $j^{th}$ neighbors of the optimal tree have a score that is at least $j$ more than the optimal. That is, as we move farther from the optimal tree, we prove mathematically that the score must increase. As characters from biological data are rarely completely compatible, the above will not work directly for most data, but serves as the first step in the approach. Since the maximum parsimony criterion is additive, we can calculate the parsimony score over each character separately and sum those scores to yield the score for all the characters. We partition the character sequences into subsets of compatible characters, determine the associated anchor tree for each, and apply the lower bound technique to each subset. The sum of the bounds of the compatible subsets yields an overall lower bound for every tree in the space. As in branch-and-bound algorithms, we exclude all trees that have a bound above our current best tree score. If we move too many steps away, the score must be worse than our current best tree score. Thus, we only need to examine trees that are within a fixed summed distance from the anchor trees (see Figure 4.1). We note that the search space, in general, is smaller if the sequence of characters is partitioned into fewer compatible subsets (and thus has a lower number of anchor trees). However, the proof of correctness for our approach does not depend on minimality, only on the fact that the partitioned sets contain compatible characters. With this in mind, we construct the compatible sets using a greedy algorithm. We also note that it is possible to find the exact minimal number of anchor trees via recent work of Gysel and
Gusfield [42], which will limit the search space further, but at the cost of additional time spent in choosing the anchor trees. While our approach might still yield a large space to examine, we show that it is in practice often much smaller than the original space, and, in addition, seems correlated to the consistency index of the best scoring anchor tree. This suggests that the consistency index can be used as a filter to limit the likely starting places for effective heuristic searches. Our approach is implemented using standard software packages, complements heuristic search, and can also be used as a filter to improve heuristic search.

4.1 Background

See the introduction for an introduction to the notation and terminology and see [72] and [22] for more thorough treatment of these topics.

On a set of characters and taxa for which no perfect phylogeny exists, there may be subsets of the characters that are compatible. Each of these subsets defines a perfect phylogeny that is an unresolved tree on the taxa. To avoid confusion between perfect phylogenies for the entire set and perfect phylogenies for the compatible subsets, we will call the trees thus derived anchor trees and their edges will be referred to as anchor edges.

Relaxed Robinson-Foulds Distance

Recall that the Robinson Foulds distance [65] is a metric that measures the dissimilarity between two trees by determining the number of edges by which the two trees differ. Formally,

Definition 4.1.1. Given two trees $T_1, T_2 \in \mathcal{T}$, the Robinson-Foulds distance, $d_{RF}(T_1, T_2)$, is the minimum number of contractions and resolutions necessary to convert $T_1$ to $T_2$. 
The RF distance can be computed efficiently [25].

To find the optimal maximum parsimony tree, we use a relaxed version of the Robinson-Foulds distance, whose neighborhoods have nice properties with respect to parsimony scores.

**Definition 4.1.2.** Given two trees $T_1, T_2 \in \mathcal{T}$, $d_{RF}(T_1, T_2)$, or the relaxed Robinson-Foulds distance, is the number of edges in $T_2$ that are incompatible with the tree $T_1$.

We use a modified version of the RF distance, $d_{rRF}$, to define neighborhoods around anchor trees. For fully resolved (binary) trees $d_{RF} = d_{rRF}$ and deleted edges are counted identically, but for unresolved trees, only the additions of non-compatible edges are counted toward the distance. As an example, given an initial derived tree, $T$, with edges $E$, any further resolutions that are done on $T$ to yield $T_1$ are not counted in $d(T, T_1)$. This is the case even if $T_1$ is fully resolved. However, if any edge $e_j$ is added to $T$, but is incompatible with $E$, so that an edge $e_i$ must be removed from $E$ before $e_j$ can be added, then $e_j$ is
counted to the distance (see Figure 4.2). We note that our relaxed distance is not a metric, as it is not symmetric, and there is no guarantee that the triangle inequality will hold.

4.2 Materials

In order to illustrate the merits of our new approach to exploring tree space, we analyzed four data sets: Metasiro americanus and three sets from TreeBASE, Rhoxocercosporidium spp., Armillaria, and Adansonia. In addition, we surveyed 400 treeBase data sets.

**Metasiro**

We ran our initial trials on a phylogeographic data set from *Metasiro americanus* (Opiliones: Neogoveidae)[19], a species of harvestman native to Florida. Harvestmen are found worldwide, and are useful in studies of phylogeography because they are highly localized, with limited dispersal abilities.

The data comprised 62 taxa, with 460 aligned base pairs. Of those 460 aligned characters, 418 were constant, while an additional six were autapomorphic, hence are compatible with all trees. That left 36 informative characters, so the final tree could not be fully resolved (as there are \(n - 3 = 59\) internal edges in the fully resolved tree, and each edge would need to be supported by at least one character). Of the 36 informative characters, 13 were completely compatible—that is, compatible with all other characters, and the remaining 23 characters broke down into two mutually incompatible sets of compatible characters, with respectively eight and fifteen characters, giving us just two perfect phylogenies, and therefore two anchor trees.
TreeBASE

We also conducted trials on data sets downloaded from TreeBASE [70]. After a search on TreeBASE for the keyword parsimony, 666 data sets were found and downloaded. We parsed the files, focusing on those where a tree could be automatically associated with a sequence of nucleotides. This was not possible for 266 of the data sets, with the most common issues being multiple or empty taxa blocks, and non-DNA character sequences. For the remaining 400 files, the constant and singleton characters were discarded. Using the trees provided with the data sets, the consistency indices of the sets were found. Finally, for the three data sets with the highest consistency indices, an exhaustive search was run. Those three data sets are:

**Armillaria, TreeBASE ID S899**

Coetzee et al. [20] ran a phylogenetic analysis on 14 isolates of the fungus genus *Armillaria* using the concatenation of ITS1, including 5.8S, ITS2, and the first intergenic spacer region (IGS-1). The concatenated characters consisted of 559 bases, of which 176 were parsimoniously informative.

**Rhexocercosporidium, TreeBASE ID S1786**

Reeleder [63] ran a phylogenetic analysis on nine isolates of the fungus genus *Rhexocercosporidium* and one outgroup, *Phialophora gregata*. The character data consisted of the concatenation of two sequence data sets, the first was an internal transcribed spacer region of ribosomal DNA, ITS1 and ITS2, including the 5.8S gene, the second was a portion of the β-tubulin gene. The concatenated characters consisted of 716 bases, of which 47 were parsimoniously informative.
Adansonia, TreeBASE ID S376

The third TreeBASE data set was from a phylogenetic analysis of baobabs, genus Adansonia. The taxa consisted of multiple specimens of the eight species in Adansonia, plus three out-group taxa from the clade Adansonieae, giving a total of 18 taxa. Baum et al. [5] ran several analyses on various character sets. Of those sets, the first in the downloaded Nexus file was the chloroplast rpl16 intron, consisting of 1350 bases, of which only 17 were parsimoniously informative.

4.3 Methods

We implemented our overall approach using the off-the-shelf software package POY [78] and Python scripts for processing data. Our approach consists of four steps:

1. partition the input sequence of characters into compatible subsets of characters;
2. construct and score the associated anchor tree for each compatible subset found;
3. calculate the starting configuration of the constrained search space; and
4. exhaustively search the constrained space, dynamically updating the remaining space to be searched.

These four steps are describe in detail below.

Partitioning into Sets of Compatible Characters

For each data set, we partitioned the sequence of characters into compatible subsets. Determining whether a set of binary characters is compatible can be accomplished in linear time [41], while doing so on non-binary data is NP-Hard [8, 74], although a polynomial-time algorithm has been found for data with a fixed number of states [1]. As such, there are a
number of equally valid heuristic methods to solve this problem. We used two methods to
discover sets of compatible characters. On data sets with no initial tree, we used a simple
greedy algorithm (see the discussion of vertex cover in [22]) to discover sets of compatible
characters. For data sets that contained a tree for the character sequences, an iterative
process was used. At each step, an anchor tree was found. That anchor tree was then col-
lapsed by removing any edges not associated with a character in the subset. This (possibly
partially resolved) anchor tree was saved and all characters that were compatible with it
were removed from the set of characters to be processed. The initial tree was used as the
original anchor tree. Subsequent anchor trees were discovered by successive searches using
POY [78]. We used POY’s “search()” operation, which builds a set of Wagner trees, runs
TBR swaps, perturbs using ratchet, and fuses the resulting trees. At each step the search
time was doubled, until a maximum of four hours was reached. The searches continued until
either there were no characters left in the superset or two successive searches were unable to
find a tree with any compatible characters. Once the end case was reached the remaining
characters were checked for pairwise compatibility. Additional anchor trees were built using
both the pairwise-compatible characters and finally the remaining singletons.

**Scoring Anchor Trees**

For large sets of characters and small sets of taxa, it is possible that there exists a subset
of compatible characters large enough that every interior edge of a binary tree is associated
with some character, but in general this is not the case. Instead, each of these sets will
define a tree which is not completely resolved, hence a non-binary tree. Nonetheless, each
of these anchor trees will have a minimal parsimony score on that subset.
Figure 4.3: Upper bound on $d_{rRF}$: The score on $T$ is 1. After $e$ is removed, the difference in scores between $T$ and $T_1$ is 2, which is the lesser of $p$, the number of parent nodes of $v_1$ and $h$, the number of child nodes of $v_2$, (in this case, both $p$ and $h$ are 3), minus 1.

Recall that the diameter of our eventual search space is the difference between the theoretical minimum score on the data and the current best score. At each step, therefore, we scored the anchor trees on the complete character sequence. For fully resolved trees, it is to easy to determine the parsimony score [31]. However, the stored anchor trees are unresolved, and for unresolved trees, a tree resolution must be found. Finding this resolution is essentially the search problem on a slightly smaller space of trees, and thus computationally difficult [9, 83]. We therefore used POY to do constrained Wagner builds on the data, to get “good enough” scores for the anchor trees. POY first read in the character set and the anchor tree, which was used as a constraint, using the build(constraint: "treefile") command. Then a constrained swap was done and the best trees selected (swap(constraint: (depth: 0, file: "treefile")) select()). In the case that multiple trees were reported, the first reported score was used.
Determining the Constrained Search Space

To determine the constrained search space, we used the partition of the character sequence into compatible sets and the associated anchor trees as well as the static bound on parsimony to calculate the summed diameter of the constrained space. Any tree whose summed relaxed RF distance to the anchor trees is less than the summed diameter is part of the constrained search space. The static lower bound was calculated as the sum of the number of character state changes across all characters. These simple calculations were performed by a script written in Python. We outline the formulas used for calculating the summed diameter below.

For each compatible subset of characters, $S_c$, we can bound the parsimony score of trees in the space by their relaxed RF distance to the associated anchor tree $T_c$ for $S_c$, using the following theorem:

**Theorem 4.3.1.** Let $C$ be a compatible sequence of characters and let $T_C$ be the associated anchor tree. Then, for any tree $T$ leaf labeled by $C$,

$$l(C, T) \geq l(C, T_C) + d_{rRF}(T_C, T)$$

(Proof can be found in Section 4.6.) Since parsimony score is additive (e.g. $l(C, T) = \sum_{c=1}^{m} l(S_c, T)$), it follows immediately that:

**Corollary 4.3.1.** Let $C$ be a sequence of characters. Let $C_1, C_2, \ldots, C_m$ be a partitioning of $C$ into compatible subsets of characters. For each $i = 1, \ldots, m$, let $T_i$ be the anchor tree associated with $C_i$. Then for any tree $T$ leaf labeled by $S$,

$$l(C, T) \geq \sum_{i=1}^{m} l(C, T_i) + d_{rRF}(T_i, T)$$
To calculate $Diam$, the bound for the sums that determines which trees are part of the constrained search space, we first calculate the parsimony score on each anchor tree, $l(T_i, S)$ as well as the static lower bound for the space, $M$. The diameter is defined as:

$$Diam = \min_{i=1,...,m} l(T_i, S) - M$$

Searching the Constrained Search Space

As no software currently exists which allows for searching only in spaces constrained by $d_{rRF}$, we instead built trees using the anchor edges. We generated all possible combinations of anchor edges, where for any given tree $T_x$, the number of anchor edges is bounded below by the total number of anchor edges less the difference between the current best score and the theoretical minimum score. This set of trees could then be used as constraints (hereafter constraint trees), for a subsequent resolution. All resolutions of the constraint trees were discovered and scored.

4.4 Results

We present results for our search on the four data sets, as well as our survey of the treeBASE files.

Metasiro

These data are well-suited to our initial investigation because they reduce to binary data. That is, while they are DNA (and therefore four-state) data, at any given locus only two nucleotides are observed across the data, so they can be treated as binary.
Figure 4.4: Walking between the anchor trees on Metasiro. $T_1$, which is an optimal tree, has eleven internal edges, and $T_2$ has ten. These sets of edges intersect, giving a total of 15 anchor edges. $\min(l(T_i)) - M_t = 8$. Thus there are $\sum_{k=7}^{15} \binom{15}{k} = 22818$ unresolved trees to visit. However, the anchor edges are highly incompatible, so only 56 of those trees can be built. The trees were scored on the entire character set, and better scores are lighter.

The Metasiro data broke down into two sets of compatible characters, of sizes fifteen and eight characters. To each of these subsets was added the subset of completely compatible characters, of size thirteen. The expected number of anchor edges was therefore $15 + 8 + 13 = 36$, but due to characters in each set that were synonyms, there were a total of only 15 anchor edges. Each of the two subsets provided an anchor tree, with parsimony scores of 46 and 49, respectively. As there were 36 informative characters, each of which reduced to two states, the character set is binary. Thus the minimum possible score, $M_t$, for this data set was 36. Therefore, the only trees which needed to be enumerated in order to perform an exhaustive search are trees, $T_x$, that lay within $\sum_{a} d_{\text{RF}}(T_x, T_a) \leq 8 (\min(s_C(T_i))) - M_t$ where $T_a$ is an anchor tree. As there were a total of fifteen anchor edges, each tree to be examined needed to have at least seven anchor edges before resolutions, for a total of $\sum_{k=7}^{15} \binom{15}{k} = 22818$ unresolved trees. As the Metasiro data reduced to binary characters, the anchor edges were entirely incompatible, so the vast majority of those trees could not be built. In fact, there were a total of 57 unresolved trees, all of whose resolutions needed to
be explored in an exhaustive search. In the end, one of the two anchor trees—that defined by the larger compatible subset—was an optimal tree (see Figure 4.4).

**Rhexocercosporidium**

Information on this data set is represented in line two of Table 4.1. There were three anchor trees and a total of six anchor edges. The total number of unresolved constraint trees that could be built was three, with 405 total possible resolutions. These 405 trees were scored, and 45 optimal trees were found, including the tree given in the downloaded Nexus file, meaning that the authors reported an optimal tree for this dataset.

**Armillaria**

Information on this data set is represented in line one of Table 4.1. There were three anchor trees and a total of twelve anchor edges. The total number of unresolved constraint trees that could be built was 473. As many of the constraint trees were highly unresolved, there were a total of 1,905,291 trees to score. Of these, there were only nine were optimal trees, including the tree reported by the authors.

**Adansonia**

Information on this data set is represented in line three of Table 4.1. There were three anchor trees and a total of eleven anchor edges. The total number of unresolved constraint trees that could be built was only one. That tree had 2,835 possible resolutions, each of which was scored. 405 of the resolved trees were optimal, including the original tree from TreeBASE.
Table 4.1: The reduction in size of the search space is a function of both \( \min(l(T_i)) - M_t \) and the number of anchor edges discovered. Some data sets could not be reduced using our method, as \( |E| < \min(l(T_i)) - M_t \). In that situation, insufficient anchor edges are discovered to constrain the search space.

**treeBASE Survey**

As noted, we computed CI’s for 400 treeBASE data sets. In Table 4.1 we report the 20 data sets with highest CI scores. Of these, we chose the three sets with the smallest subsequent search spaces and did exhaustive walks of all necessary trees. We found that the reported trees were, in fact, among the sets of optimal trees for each of the three data sets.

We were able to significantly reduce the amount of the tree space necessary to search exhaustively (see Figure 4.1). In addition, for each of those high-CI data sets, we were able
to show that the trees given in TreeBASE were, in fact, optimal trees. This follows, as the CI's were computed using the trees given in TreeBASE, which were the best found for their respective data sets.

4.5 Discussion

We provide a new approach for finding the optimal phylogenetic tree under the maximum parsimony criteria. Extending the work of Maddison [58], we quantify where “phylogenetic islands” of optimal trees can occur by refining the static lower bounds determined by Hendy et al. [45] to new lower bounds that change across the tree space. As better potential solutions are found, the constrained search space dynamically shrinks, as in the classic branch-and-bound algorithms of Hendy and Penny [46, 62]. To determine the constrained search space, we first partition the input characters into compatible subsets, which induce “anchor trees” that guide the search.

As noted, we ran complete searches on the three TreeBASE data sets with the highest CI's. Using our method, we were able to run a complete search with a guaranteed optimal result on all three sets in just a few hours of time on a multi-core machine, including an exhaustive search of the areas in which we can guarantee the optimal trees will be found. Prior to this, only an exhaustive search or a branch-and-bound approach would guarantee that an optimal tree would be found. For the smallest of these sets, S1786, with only nine taxa, an exhaustive search of the entire space would certainly have been possible, but exhaustive searches would have been significantly more time-consuming for S899, and especially for S376, with 18 taxa. As one can see from examining Table 4.1, even for data sets with CI's around .85, the size of the space that one would have to search in order to be
sure that the true optimal tree had been found can be significantly reduced, moving some
sets with even large taxon sets into the realm of guaranteed results.

As each of the edges in the anchor trees is used to constrain the search space, our method
is most successful given datasets with high CI, which result in highly-structured anchor trees.
Likewise, a data set with a high CI implies that a large subset of compatible characters is
present. In general, we expect that the method will be more successful when fewer anchor
trees are found, as this will be the case when there are large subsets of compatible characters.
These large subsets will in turn give large sets of anchor edges whose members are compatible
with each other but largely incompatible with members of other sets. The size of the eventual
space to be searched is dependent on the number of resolutions of the constraint trees, and
as the anchor edges are used to generate constraint trees, large sets of anchor edges will give
more-resolved constraint trees, and incompatibility between sets will reduce the number of
constraint trees. Both of these outcomes will reduce the number of full-resolved trees to
search.

In addition to the method of enumerating the trees in the bounded space that we dis-
cussed above, the trees could be discovered—using existing software—by employing the SPR
distance (because \( d_{rRF} \) is not standard). As SPR is a superset of the RF distance—that is,
if the distance between two trees is \( k \) under \( d_{rRF} \) then the distance between the two trees
under the SPR distance is \( \leq k \). Thus,

\[
T \in SearchSpace \iff \sum_{i=1}^{m} l(T_i) + d_{rRF}(T_i, T) \leq D \\
\Rightarrow \sum_{i=1}^{m} l(T_i) + d_{rRF}(T_i, T) \leq D \\
\Rightarrow \text{for some} \ c, \ d_{SPR}(T, T_c) \leq D
\]

This method has some limitations. Specifically, we rely on the fact that, as one moves
away from an anchor tree, the scores of subsequent trees must increase. However, if the
sizes of the subsets of compatible characters are too small relative to the size of the set of all characters, then that behavior, while still occurring, can be overwhelmed by changes in the combined scores of all the other characters in the superset. That is, each compatible subset has a signal, whose strength relative to the signal of the superset must be large, lest it be lost in the noise of the signal of the superset. Also, it is possible that the number of edges $|E| < \min(l(T_i)) - M_t$. In that case, too few anchor edges are discovered, and it is not possible to constrain the search space. This can occur even given anchor edges with high CIs.

There are several directions for future work. The first are algorithmic challenges to improve the efficiency of the approach. For instance, due to limitations in the off-the-shelf software we used, we were not able to take advantage of the tightening bounds during our search of the bounded space to further reduce the size of the space. In addition, we might incorporate the approaches of Hendy and Penny [45] and Holland et al. [48] for calculating static lower bounds into the computational framework. Both these changes would shrink the number of trees currently explored without requiring improvements to the underlying theoretical results.

Finding a near-perfect phylogeny on binary characters is fixed-parameter tractable, where the parameter is the number of substitutions or transformations (i.e. the distance from the minimum tree). For trees with high CI’s, Blelloch et al. [7] give a polynomial-time algorithm to find the optimal tree. Using our approach to discover trees with high CI’s, one could then forego additional steps, and instead use the method of Blelloch et al. [7] to find the exact optimal tree.

Other possible extensions include using our framework to approach the problem of finding the maximum likelihood tree for a sequence of characters. The work of Money and Whelan...
[59] suggests that islands of trees exists for the SPR metric, but difficulties lie in determining
the optimal score for a tree topology given that multiple optima can exist across the choices
of branch lengths [75].

4.6 Proofs of Results

**Theorem 4.3.1.** Let $C$ be a compatible sequence of characters and let $T_C$ be the associated
anchor tree. Then, for any tree $T$ leaf labeled by $C$,

$$l(C, T) \geq l(C, T_C) + d_{rRF}(T_C, T)$$

**Proof:** Let $T_C = (V_C, E_C)$ be the minimal tree compatible with the compatible character
sequence $C$ and let $T = (V, E)$ be a tree with leaves labeled by $C$. The proof proceeds by
induction on $d = d_{rRF}(T_C, T)$.

If $d = 0$, then $T$ is a resolution of $T_C$ (i.e. every edge of $T_C$ also occurs in $T$). Since $T_C$ is
a perfect phylogeny for $C$, every edge corresponds to a character and further resolution of
polytomies will not lower the score. So, $l(C, T) = l(C, T_C)$.

Assume that $d > 0$. By definition of the relaxed RF distance, there exists, $d_+$ and $d_-$
such that:

- $d = d_+ + d_-$,
- $d_+$ is the number of edges in $T$ that are not compatible with $T_C$, and
- $d_-$ is the number of edges missing from $T$ that are in $T_C$.

We note that $T$ could possibly have other edges that are compatible with $T_C$. These edges
make no difference to the score, since, as in the base case, resolutions of perfect phylogenies
have the same score as the perfect phylogeny.
If $d_+ > 0$, then there is at least one edge $e'$ that is in $T$ but that is not compatible with $T_C$. Let $T' = (V, E \setminus \{e'\})$. By construction, $d_{rRF}(T_C, T') = d - 1$ and by inductive hypothesis, $l(C, T') \geq l(C, T_C) + d - 1$. By definition $e'$ is not compatible with $T_C$, implying there is at least one edge, $e_c$, from $T_C$ which is incompatible with $e'$. Since every edge in $T_C$ is associated with a character from $C$, there is at least one character $c \in C$ that is incompatible with $e'$ and must take at least one additional change across $T'$. Combining these two facts, we have:

$$l(C, T) \geq l(C, T') + 1$$

$$\geq (l(C, T_C) + d - 1) + 1 = l(C, T_C) + d$$

$$= l(C, T_C) + d_{rRF}(T_C, T)$$

If $d_+ = 0$, then $d_- > 0$, and there is at least one edge $e''$ that is in $T_C$ but has been contracted and not found in $T$. Let $T'' = (V, E \cup \{e''\})$. By construction, $d_{rRF}(T_C, T'') = d - 1$ and by inductive hypothesis, $l(C, T'') \geq l(C, T_C) + d - 1$. $e''$ is an edge of the original tree $T_C$, and, by definition of $T_C$, there is at least one character state change across that edge. Thus, contracting $e''$ raises the parsimony score by at least 1 (possibly more, if there are multiple character state changes across $e''$. Combining these two facts, we have:

$$l(C, T) \geq l(C, T'') + 1$$

$$\geq (l(C, T_C) + d - 1) + 1 = l(C, T_C) + d$$

$$= l(C, T_C) + d_{rRF}(T_C, T)$$

□

**Corollary 4.3.1.** Let $C$ be a sequence of characters. Let $C_1, C_2, \ldots, C_m$ be a partitioning of $C$ into compatible subsets of characters. For each $i = 1, \ldots, m$, let $T_i$ be the anchor tree associated with $C_i$. Then for any tree $T$ leaf labeled by $C$,

$$l(C, T) \geq \sum_{i=1}^{m} l(C, T_i) + d_{rRF}(T_i, T)$$

73
Proof: By Theorem 4.3.1, we have for each $i$, $l(C_i, T) \geq l(C_i, T_i) + d_{RF}(T_i, T)$. Since parsimony scores are additive, we have for any tree $T$, $l(C, T) = \sum_{i=1}^{m} l(C_i, T) \geq \sum_{i=1}^{m} l(C, T_i) + d_{RF}(T_i, T)$  

4.7 Acknowledgments

We would like to thank Ronald Clouse for the Metasiro data set and Mike Charleston, Sean Cleary, and Barbara Holland for insightful conversations. The paper was greatly improved by comments and suggestions of the editors, Andy Anderson and Lars Jermiin, and the referees: Mike Steel and two anonymous referees. Partial funding was provided by the US National Science Foundation (\#0920920 to KS). This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-05-1-0271 to WCW.
5 Efficient Search in Tree Space

Direct Optimization versus Alignment and Search\textsuperscript{1}

Abstract

Given a set of taxa and associated non-aligned sequence data, the generalized tree alignment problem (GTAP) is to find a phylogenetic tree—and associated vertex sequences—with the lowest cost for that data under maximum parsimony. We compare two approaches to GTAP—sequence alignment followed by tree search, and Direct Optimization—on several biological data sets.

5.1 Introduction

A central goal of biological systematics is mapping the relationships between organisms and groups of organisms—both extant and extinct—which involves the creation of phylogenetic trees using character sequence data. The generalized tree alignment problem (GTAP) is to find a phylogenetic tree—and associated vertex sequences—with the lowest cost for that data under maximum parsimony.

\textsuperscript{1}Co-author Ward C. Wheeler.
There has been an ongoing debate in the literature regarding sequence alignment [28, 81, 51], with several aligners available. In addition, much effort has been expended in improving search on aligned sequences [39]. At the same time, other paradigms for solving the GTAP are also available, chief among those being Direct Optimization (DO) [79, 80]. It has been our experience that DO gives significantly better results than the two-step process of alignment then search. In addition, the high degree of complexity in the settings of the software tools used for alignment and search only confuses the matter, as default settings are often used, and these defaults do not correspond between aligner and search engine. In this chapter we compare DO to two-step solutions directly. We also test whether the results of searches where alignment and search setting correspond are more optimal than those in which they do not. We find that DO results in the discovery of shorter trees, by a factor of 15%. In addition, using the two-step approach we found significantly (approximately 4%) shorter trees when using settings on alignment that match the settings of subsequent search.

5.2 Software tools

We ran comparisons on several pieces of alignment software. What follows is a brief description of each package.

MUSCLE v3.8.31 [28] performs progressive alignments using guide trees. To build the first tree, it uses the approximate $k$-mer pairwise distances. A $k$-mer is a subsequence sample of length $k$, and the $k$-mer distance used by MUSCLE is the fraction of $k$-mers in common in a compressed alphabet. This initial tree build is followed by progressive alignments using the Kimura distance [52].
MAFFT v7.029b [51] also uses a guide tree to direct alignment, but uses fast Fourier transform to speed up identification of homologous regions in sequences. Depending on the number of taxa, MAFFT uses either progressive or iterative alignment algorithms.

Clustal W 2.0.12 [56] predates both MUSCLE and MAFFT. It uses a guide tree to align sequences but does so using neighbor-joining [69] rather than UPGMA (Unweighted Pair Group Method with Arithmetic Mean, a hierarchical clustering method using a similarity matrix), as in MAFFT and MUSCLE. In addition, Clustal W weights the sequences, giving more-divergent sequences more weight, in an attempt to get more accurate results in pairwise comparisons.

Clustal Omega [73] is similar to Clustal W but speeds the process of creating the guide tree by re-encoding each sequence as an \( n \)-dimensional vector, which can be viewed as its similarity to \( n \) reference sequences. This allows for quicker clustering using hidden Markov models.

In contrast to the above packages, POY 5.0.0 beta2 [78] uses Direct Optimization. This approach starts by building a Wagner tree (or a set thereof). Search on the trees is done using relatively similar methods to TNT [39], but because the alignment and scoring are done as each tree is constructed (see discussion of DO, above), a POY search is computationally more complex than a TNT search.

After the alignment stage, we used TNT to search for the shortest trees. In an attempt to compare DO to two-step search more thoroughly, we also ran TNT on the results of POY searches. As POY’s output is a tree rather than an alignment, an implied alignment was created, and an additional search was conducted on that applied alignment using TNT.
5.3 Materials

We ran alignment and search on six biological data sets. The biological data sets were chosen for their sizes. Each of the software packages allows for the input of either genetic or protein data. In this case, we used genetic data, but all of the analyses could have been run using protein data, instead.

- 62 taxa of Mantodea (mantids) consists of five genes (16S rDNA, 18S rDNA, 28S rDNA, cytochrome oxidase II and histone 3) [76]
- 208 taxa of Metazoa with sequences from 18s rDNA and 18s rRNA [81]
- 585 taxa of archaea comprised of small subunit sequence data from the European Ribosomal RNA Database [81]
- 1040-taxon set of mitochondrial small subunit data from the European Ribosomal RNA Database [81]
- 1553 taxa of fungi which we created from data in Genbank [6]
- 1762 taxa of Metazoa with sequences comprised of 33 18S rDNA sequences [37, 38]

5.4 Methods

For each of the six data sets, a two step process was used. First, the sequences were aligned by each of the aligners. The resulting alignments were then fed into TNT for subsequent search. In the case of Direct Optimization, we took an additional step. After the initial search with POY implied alignments were generated for each of the found trees, and a
second search was done using TNT on those implied alignments. Thus, for each data set eight total alignments were generated (two each for MAFFT, POY and Clustal Omega, one each for MUSCLE and Clustal W), and for each of the eight alignments, two TNT searches were run, giving a total of 192 runs through TNT, but 216 searches, including POY searches.

On several of the runs, POY found multiple trees. Rather than doing multiple TNT searches on each of the implied alignments, in these cases we simply used the first tree and implied alignment in the output file. As POY outputs these trees in no particular order, this seemed to be arbitrary, and therefore relatively “fair” to TNT.

Due to the complexity of the software packages used, researchers often run the applications using default settings. Interestingly, however, the default settings of TNT differ from the defaults of the alignment software. That is, TNT uses a gap opening cost of 0 and a gap extension cost of 1, whereas each of the aligners has a more sophisticated gap model. In fact, it is impossible to run TNT searches using the same settings as the alignment models, as all of the aligners studied here use affine gap models, and TNT does not. In order to measure the effect of this difference between the models, each of the data sets was aligned first with default settings in each aligner, and then with a gap cost of 0 and a gap extension cost of 1, which better models TNT’s gap-cost scheme. This allowed us to track the effect that this difference in settings might have on eventual tree lengths. In addition, when possible we set all substitutions to a cost of 1, as this is TNT’s default. We hereafter refer to this as 0:1:1 settings (0 gap cost, 1 gap extension cost, 1 substitution cost). We also note that TNT treats gaps as missing data, which is not the same as an indel, which is a hypothetical event, and not the same as a missing or mis-read character. In order to account for this, we set gaps as a fifth state for search.
Table 5.1: MAFFT settings for various data sets. For the 0:1:1 runs we needed to match MAFFT’s default settings as closely as possible. However, MAFFT’s default settings for any given size of data set were unclear from the documentation. These settings were thus chosen by running MAFFT on each data set under default settings, then examining MAFFT’s output to the interface to determine which settings were used for each data set. These determinations were then double-checked against the MAFFT manual.

For the alignment stage, we first ran MAFFT with default settings, then reran it with gap opening cost of 0 and extension cost 1 (--op 0 --ep 1 --lop 0 --lep 2). MAFFT uses different alignment settings depending upon the size of the input data. In order to recreate those settings while changing the defaults for gap costs, three different profiles were created: L-INS-i, with commands --localpair --maxiterate 1000, FFT-NS-i (--retree 2 --maxiterate 2) and FFT-NS-2 (--retree 2 --maxiterate 0). For each data set, MAFFT was first run with default settings. Its output to the command line was then read to determine which of the three profiles was used, and that profile was used when running MAFFT again with the custom gap costs. The individual data sets and profiles used are listed in Table 5.4.

As we were unable to run MUSCLE using a custom gap cost, it was run only with default settings. Likewise, Clustal Omega does not provide for the ability to run with custom gap costs, and it was therefore run only with default settings.

As with the other aligners, we ran Clustal W first with default settings on each of the data sets, then with 0:1:1 settings: -gapopen=1 -gapext=0 -ENDGAPS, and using a custom
base substitution cost matrix with costs of 0 for non-substitution and 1 for substitution.

We in turn ran POY using both less and more aggressive search strategies. The first run used build(), which builds Wagner trees using DO. The more aggressive run (build() swap(transform(static.approx)) swap()), builds Wagner trees, then creates an implied alignment and does a hill-climbing search looking at SPR neighborhoods, followed by an additional hill-climbing search on SPR neighborhoods using DO.

In addition, we exported the applied alignment from the resulting POY search, essentially using POY only to do alignment, with search done by TNT.

After every alignment was generated, we used a Python script to convert the resulting Fasta files to Hennig format for input to TNT. In addition, at this point, all gaps were changed to a fifth state, as mentioned above.

In order to measure the amount that both search and alignment affect the eventual tree length, we ran TNT with two settings for each alignment, first, a relatively modest run with mult, then with much more aggressive settings: mult; xmult=replications 10 ratchet 50 drift 20 fuse 5;.

5.5 Results

For tabulated results see Tables 5.2 and 5.3.

For every data set, the 0:1:1 settings gave shorter trees in the subsequent search than the default alignment settings. On average, there was a 3.5% improvement in tree length for those setting.

In general, running TNT with more aggressive settings gave very little difference in tree length, .04% on average. In contrast, there was a more marked improvement, of 1.59% on
average, between standard and aggressive runs of POY.

TNT was able, in every case, to find a shorter tree than POY, if given POY’s implied alignments as input. The difference was .03% on average. We assume this effect is due to TNT’s more aggressive search. Whereas POY was doing only SPR swaps, TNT’s search included two versions of simulated annealing as well as portions of a genetic algorithm.

The most striking difference is the difference in tree lengths for DO versus two-step alignment. Comparing the aggressive TNT search to the aggressive POY search, POY gave 14.78% shorter trees than the two step alignment. But because the difference in results was so minimal between TNT’s more- and less-aggressive runs, even the less aggressive DO runs were 13.53% better than the more aggressive TNT results.

We did not find that one aligner gave significantly shorter trees after search. For four of the data sets, MAFFT gave the best results, and for two sets Clustal W gave the best results. Likewise for the worst performance: Both Clustal Omega and MUSCLE gave the longest tree three times. Clustal Omega is specifically designed for use on protein sequences, and gives warnings when presented with gene data, which might explain its poor showing.
Table 5.2: Alignment results

<table>
<thead>
<tr>
<th>Data set</th>
<th>No. of Taxa</th>
<th>Aligner</th>
<th>Settings</th>
<th>Aligned Length</th>
<th>Tree Length After Standard Search</th>
<th>Tree Length After Aggressive Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mantids</td>
<td>62</td>
<td>Clustal Omega</td>
<td>default</td>
<td>1829</td>
<td>1159</td>
<td>1159</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>Clustal W</td>
<td>default</td>
<td>1828</td>
<td>1055</td>
<td>1055</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>Clustal W</td>
<td>one-one</td>
<td>1893</td>
<td>1042</td>
<td>1042</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>MAFFT</td>
<td>default</td>
<td>1832</td>
<td>1033</td>
<td>1033</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>MAFFT</td>
<td>one-one-l-ins-i</td>
<td>1872</td>
<td>1006</td>
<td>1006</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>MUSCLE</td>
<td>default</td>
<td>1853</td>
<td>1044</td>
<td>1044</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>POY/POY</td>
<td></td>
<td>1909</td>
<td>945</td>
<td>939</td>
</tr>
<tr>
<td>Mantids</td>
<td>62</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>1909</td>
<td>942</td>
<td>942</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>Clustal Omega</td>
<td>default</td>
<td>2863</td>
<td>38544</td>
<td>38536</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>Clustal W</td>
<td>default</td>
<td>2866</td>
<td>31105</td>
<td>31103</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>Clustal W</td>
<td>one-one</td>
<td>4777</td>
<td>29317</td>
<td>29317</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>MAFFT</td>
<td>default</td>
<td>3377</td>
<td>32842</td>
<td>32837</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>MAFFT</td>
<td>one-one-fft-ns-i</td>
<td>3792</td>
<td>29024</td>
<td>29014</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>MUSCLE</td>
<td>default</td>
<td>3307</td>
<td>35237</td>
<td>35228</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>POY/POY</td>
<td></td>
<td>6625</td>
<td>26876</td>
<td>26817</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>6625</td>
<td>26861</td>
<td>26861</td>
</tr>
<tr>
<td>Metazoa short</td>
<td>208</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>6651</td>
<td>26812</td>
<td>26812</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>Clustal Omega</td>
<td>default</td>
<td>1810</td>
<td>39840</td>
<td>39814</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>Clustal W</td>
<td>default</td>
<td>1757</td>
<td>39362</td>
<td>39340</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>Clustal W</td>
<td>one-one</td>
<td>3100</td>
<td>40560</td>
<td>40548</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>MAFFT</td>
<td>default</td>
<td>1837</td>
<td>39359</td>
<td>39337</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>MAFFT</td>
<td>one-one-fft-ns-2</td>
<td>2518</td>
<td>41304</td>
<td>41292</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>MUSCLE</td>
<td>default</td>
<td>1705</td>
<td>40760</td>
<td>40734</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>POY/POY</td>
<td></td>
<td>6770</td>
<td>37914</td>
<td>37118</td>
</tr>
<tr>
<td>Archaea</td>
<td>585</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>6770</td>
<td>37777</td>
<td>37775</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>Clustal Omega</td>
<td>default</td>
<td>3908</td>
<td>96656</td>
<td>96590</td>
</tr>
</tbody>
</table>

Note that POY standard and aggressive settings are in one row, and the settings are therefore left blank.
Table 5.2: Alignment results

<table>
<thead>
<tr>
<th>Data set</th>
<th>No. of Taxa</th>
<th>Aligner</th>
<th>Settings</th>
<th>Aligned Length</th>
<th>Tree Length After Standard Search</th>
<th>Tree Length After Aggressive Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>Clustal W</td>
<td>default</td>
<td>3111</td>
<td>91032</td>
<td>91014</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>Clustal W</td>
<td>one-one</td>
<td>7198</td>
<td>86669</td>
<td>86652</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>MAFFT</td>
<td>default</td>
<td>5064</td>
<td>94205</td>
<td>94166</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>MAFFT</td>
<td>one-one-fft-ns-2</td>
<td>5501</td>
<td>87997</td>
<td>87938</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>MUSCLE</td>
<td>default</td>
<td>2435</td>
<td>128178</td>
<td>128041</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>POY/POY</td>
<td></td>
<td>14474</td>
<td>79191</td>
<td>77832</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>14474</td>
<td>79065</td>
<td>79065</td>
</tr>
<tr>
<td>Mitochondrial</td>
<td>1040</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>14060</td>
<td>77819</td>
<td>77819</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>Clustal Omega</td>
<td>default</td>
<td>2622</td>
<td>83726</td>
<td>83654</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>Clustal W</td>
<td>default</td>
<td>2539</td>
<td>68973</td>
<td>68910</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>Clustal W</td>
<td>one-one</td>
<td>4543</td>
<td>67135</td>
<td>67060</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>MAFFT</td>
<td>default</td>
<td>2844</td>
<td>69185</td>
<td>69125</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>MAFFT</td>
<td>one-one-fft-ns-2</td>
<td>3695</td>
<td>68501</td>
<td>68437</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>MUSCLE</td>
<td>default</td>
<td>2350</td>
<td>76132</td>
<td>76014</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>POY/POY</td>
<td></td>
<td>11610</td>
<td>63473</td>
<td>62000</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>11610</td>
<td>63232</td>
<td>63202</td>
</tr>
<tr>
<td>Fungi</td>
<td>1553</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>11652</td>
<td>61944</td>
<td>61943</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>Clustal Omega</td>
<td>default</td>
<td>4891</td>
<td>243149</td>
<td>242941</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>Clustal W</td>
<td>default</td>
<td>4282</td>
<td>198870</td>
<td>198780</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>Clustal W</td>
<td>one-one</td>
<td>12535</td>
<td>187940</td>
<td>187882</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>MAFFT</td>
<td>default</td>
<td>7448</td>
<td>205416</td>
<td>205263</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>MAFFT</td>
<td>one-one-fft-ns-2</td>
<td>9066</td>
<td>187180</td>
<td>187093</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>MUSCLE</td>
<td>default</td>
<td>3554</td>
<td>249222</td>
<td>249076</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>POY/POY</td>
<td></td>
<td>37005</td>
<td>169595</td>
<td>165563</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>37005</td>
<td>169447</td>
<td>169443</td>
</tr>
<tr>
<td>Metazoa long</td>
<td>1766</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>36317</td>
<td>165524</td>
<td>165522</td>
</tr>
</tbody>
</table>

Note that POY standard and aggressive settings are in one row, and the settings are therefore left blank.
Table 5.3: Comparative alignment results

<table>
<thead>
<tr>
<th>Taxa</th>
<th>Aligner</th>
<th>Settings</th>
<th>% Shorter, Aggressive</th>
<th>% Difference, Standard to Aggressive</th>
<th>% Shorter, Implied Alignment Search</th>
<th>% DO Improves on Two-Step, Both Aggressive</th>
<th>% Standard DO Improves on Aggressive Two-Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>Clustal Ω</td>
<td>default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td>18.98%</td>
<td>18.46%</td>
</tr>
<tr>
<td>62</td>
<td>Clustal W</td>
<td>default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td>11.00%</td>
<td>10.43%</td>
</tr>
<tr>
<td>62</td>
<td>Clustal W</td>
<td>one-one</td>
<td>0.00%</td>
<td>1.23%</td>
<td></td>
<td>9.88%</td>
<td>9.31%</td>
</tr>
<tr>
<td>62</td>
<td>MAFFT</td>
<td>default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td>9.10%</td>
<td>8.52%</td>
</tr>
<tr>
<td>62</td>
<td>MAFFT</td>
<td>one-one-l-ins-i</td>
<td>0.00%</td>
<td>2.61%</td>
<td></td>
<td>6.66%</td>
<td>6.06%</td>
</tr>
<tr>
<td>62</td>
<td>MUSCLE</td>
<td>default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td>10.06%</td>
<td>9.48%</td>
</tr>
<tr>
<td>62</td>
<td>POY/POY</td>
<td></td>
<td></td>
<td>0.64%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>0.00%</td>
<td></td>
<td>0.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>Clustal Ω</td>
<td>default</td>
<td>0.02%</td>
<td></td>
<td></td>
<td>30.41%</td>
<td>30.26%</td>
</tr>
<tr>
<td>208</td>
<td>Clustal W</td>
<td>default</td>
<td>0.01%</td>
<td></td>
<td></td>
<td>13.78%</td>
<td>13.59%</td>
</tr>
<tr>
<td>208</td>
<td>Clustal W</td>
<td>one-one</td>
<td>0.00%</td>
<td>5.74%</td>
<td></td>
<td>8.53%</td>
<td>8.33%</td>
</tr>
<tr>
<td>208</td>
<td>MAFFT</td>
<td>default</td>
<td>0.02%</td>
<td></td>
<td></td>
<td>18.33%</td>
<td>18.15%</td>
</tr>
<tr>
<td>208</td>
<td>MAFFT</td>
<td>one-one-fft-ns-i</td>
<td>0.03%</td>
<td>11.64%</td>
<td></td>
<td>7.57%</td>
<td>7.37%</td>
</tr>
<tr>
<td>208</td>
<td>MUSCLE</td>
<td>default</td>
<td>0.03%</td>
<td></td>
<td></td>
<td>23.88%</td>
<td>23.71%</td>
</tr>
<tr>
<td>208</td>
<td>POY/POY</td>
<td></td>
<td></td>
<td>0.22%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>0.00%</td>
<td></td>
<td>0.02%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>585</td>
<td>Clustal Ω</td>
<td>default</td>
<td>0.07%</td>
<td></td>
<td></td>
<td>6.77%</td>
<td>4.77%</td>
</tr>
<tr>
<td>585</td>
<td>Clustal W</td>
<td>default</td>
<td>0.06%</td>
<td></td>
<td></td>
<td>5.65%</td>
<td>3.62%</td>
</tr>
<tr>
<td>585</td>
<td>Clustal W</td>
<td>one-one</td>
<td>0.03%</td>
<td>-3.07%</td>
<td></td>
<td>8.46%</td>
<td>6.50%</td>
</tr>
<tr>
<td>585</td>
<td>MAFFT</td>
<td>default</td>
<td>0.06%</td>
<td></td>
<td></td>
<td>5.64%</td>
<td>3.62%</td>
</tr>
<tr>
<td>585</td>
<td>MAFFT</td>
<td>one-one-fft-ns-2</td>
<td>0.03%</td>
<td>-4.97%</td>
<td></td>
<td>10.11%</td>
<td>8.18%</td>
</tr>
<tr>
<td>585</td>
<td>MUSCLE</td>
<td>default</td>
<td>0.06%</td>
<td></td>
<td></td>
<td>8.88%</td>
<td>6.92%</td>
</tr>
<tr>
<td>585</td>
<td>POY/POY</td>
<td></td>
<td></td>
<td>2.14%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that POY standard and aggressive settings are in one row, and the settings are therefore left blank.

Averages of above lines: 0.04% 1.59% 3.55% 0.03% 14.78% 13.43%
Table 5.3: Comparative alignment results

<table>
<thead>
<tr>
<th>Taxa</th>
<th>Aligner</th>
<th>Settings</th>
<th>% Shorter, Aggressive Search</th>
<th>% Difference, Standard to Aggressive DO</th>
<th>% Shorter, Implied Alignment 0:1:1 Settings</th>
<th>% DO Improves on Two-Step, Both Aggressive</th>
<th>% Standard DO Improves on Aggressive Two-Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>585</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>0.01%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>585</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>Clustal Ω</td>
<td>default</td>
<td>0.07%</td>
<td>4.79%</td>
<td>15.48%</td>
<td>17.99%</td>
<td>15.90%</td>
</tr>
<tr>
<td>1040</td>
<td>Clustal W</td>
<td>default</td>
<td>0.02%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>Clustal W</td>
<td>one-one</td>
<td>0.02%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>MAFFT</td>
<td>default</td>
<td>0.04%</td>
<td>4.79%</td>
<td>15.48%</td>
<td>17.99%</td>
<td>15.90%</td>
</tr>
<tr>
<td>1040</td>
<td>MAFFT</td>
<td>one-one fft-ns-2</td>
<td>0.07%</td>
<td>6.61%</td>
<td>11.49%</td>
<td>9.95%</td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>MUSCLE</td>
<td>default</td>
<td>0.11%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>POY/POY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>Clustal Ω</td>
<td>default</td>
<td>0.09%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>Clustal W</td>
<td>default</td>
<td>0.09%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>Clustal W</td>
<td>one-one</td>
<td>0.11%</td>
<td>2.68%</td>
<td>7.55%</td>
<td>5.35%</td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>MAFFT</td>
<td>default</td>
<td>0.09%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>MAFFT</td>
<td>one-one fft-ns-2</td>
<td>0.09%</td>
<td>1.00%</td>
<td>9.41%</td>
<td>7.25%</td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>MUSCLE</td>
<td>default</td>
<td>0.16%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>POY/POY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>POY/TNT</td>
<td>POY default</td>
<td>0.05%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>POY/TNT</td>
<td>POY aggressive</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>Clustal Ω</td>
<td>default</td>
<td>0.09%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>Clustal W</td>
<td>default</td>
<td>0.05%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>Clustal W</td>
<td>one-one</td>
<td>0.03%</td>
<td>5.48%</td>
<td>11.88%</td>
<td>9.73%</td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>MAFFT</td>
<td>default</td>
<td>0.07%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>MAFFT</td>
<td>one-one fft-ns-2</td>
<td>0.05%</td>
<td>8.85%</td>
<td>11.51%</td>
<td>9.35%</td>
<td></td>
</tr>
</tbody>
</table>

Note that POY standard and aggressive settings are in one row, and the settings are therefore left blank.

Averages of above lines: 0.04% 1.59% 3.55% 0.03% 14.78% 13.43%
Table 5.3: Comparative alignment results

<table>
<thead>
<tr>
<th>Taxa</th>
<th>Aligner</th>
<th>Settings</th>
<th>% Shorter, Aggressive</th>
<th>% Difference, Standard to Aggressive</th>
<th>% Shorter, Implied DO</th>
<th>Search 0:1:1 Settings</th>
<th>% DO Improves on Two-Step, Both Aggressive</th>
<th>% Standard DO Improves on Aggressive Two-Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1766</td>
<td>MUSCLE</td>
<td>default</td>
<td>0.06%</td>
<td></td>
<td></td>
<td></td>
<td>33.53%</td>
<td>31.91%</td>
</tr>
<tr>
<td>1766</td>
<td>POY/POY</td>
<td></td>
<td></td>
<td>2.44%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>POY/TNT</td>
<td>POY</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1766</td>
<td>POY/TNT</td>
<td>aggressive</td>
<td>0.00%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that POY standard and aggressive settings are in one row, and the settings are therefore left blank.

Averages of above lines: 0.04% 1.59% 3.55% 0.03% 14.78% 13.43%
5.6 Conclusion

As the implied alignments generated by POY are specific to the phylogenies found, it may seem slightly odd to compare its results to a separate aligner. Nonetheless, in every case, DO found a shorter tree than the two-step process of alignment and search. In addition, as TNT’s rather more aggressive search strategy was insufficient to find significant improvement in the tree lengths (approximately the same amount of improvement it was able to make over POY’s search), it seems that the majority of the result of a two-step search is a relic of the alignment stage, which “ties the hands,” as it were, of the subsequent search step. This hypothesis is bolstered by the more noticeable improvement between standard and more aggressive DO searches.

As noted above, TNT treats gaps not as indels but as missing data. Attempting to analyze the effects of this difference is beyond the scope of this paper, but is certainly cause for consideration.

5.7 Future Work

These same analyses will be run on simulated data, using DAWG to create gapped sequences on sets of 50, 200 and 600 taxa. The sequences will be generated using two different distributions of edge lengths, uniform and exponential.

Acknowledgements

A special thanks to Ward Wheeler, who gave valuable insight at all stages of this research.
Bibliography


