SAMPLING DISCRETE COMBINATORIAL SPACES IN PHYLOGENETICS

by

Alexander Safatli

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2015

I dedicate this document to my loving family, of which also includes the colourful members of Blouin Lab, in appreciation of the lessons this journey has given me.

Table of Contents

List of Tables

List of Figures

Abstract

Phylogenetics is the study and identification of evolutionary structure, and phylogeneticists often present evolutionary inferences as leaf-labelled trees. We investigate the combinatorial nature of phylogenetic tree reconstruction as an applied problem of combinatorics.

This thesis strives to improve techniques for tree reconstruction and evolutionary inference by characterizing the fitness landscape of tree search. Introduced are some of the challenges faced when performing heuristic searches of the phylogenetic fitness landscape. Also introduced are strategies for restricting that search. I discuss techniques to sample the space, rather than merely find an optimum, including an applied metaheuristic of Ant Colony Optimization. This metaheuristic was found to sparsely sample a set of phylogenetic landscapes across their diameters. Finally, I present implementation details and discussion regarding a software framework that was developed to facilitate heuristic searches and exploration of the space. This framework was developed with a number of existing tools and creates new ones.

Acknowledgements

My supervisor, Dr. Christian Blouin, deserves the most praise for his invaluable guidance, friendship, and suggestions.

I would also like to thank Dr. Norbert Zeh, Dr. Robert Beiko, and the members of both their labs for many helpful suggestions, their feedback, and their support. For their diligence in teaching courses that I wrote reports relevant to my thesis, I also give thanks to Dr. Meng He and Dr. Vlado Keselj.

Lastly, I thank all of the friends and acquaintances I have made at Dalhousie University. They have made this journey something to remember. This journey has taught me humility, has granted me confidence, and has tempered me with strong ideals.

Chapter 1

Introduction

The field of phylogenetics possesses many problems that could benefit from the scrutinous eyes of mathematicians and computer scientists. New methods can be developed, or old ones can be analysed. The tree remains a staple data structure in computer science. What is more, we as computer scientists can help to find insight in many of the systematic problems posed to phylogeneticists. In particular, I use this thesis to provide insight to a problem of constructing trees from data.

1.1 Motivation

The idea of phylogeny, of the tree representation of evolutionary relationships between taxonomic units, was popularized by Charles Darwin. It is in his book entitled *On the Origin of Species* where the earliest depiction was made of a *phylogenetic tree* to represent these relationships. It is also there that Darwin notes that "the affinities of [those] of the same class [can be] represented by a great tree" [23].

Since these words were written, the development of sequencing technology has brought about an invaluable source of data for the investigation of evolutionary history. Matrices of biological characters, including DNA and protein sequence data, are now the primary source of signal for phylogenetic inference.

Modern day understanding of evolutionary theory proves to be far more complex than it was when that first tree was suggested. With this improved understanding comes a number of implications that challenge the model of a tree to explain evolutionary affinities between classes. When a tree model is used, an underlying assumption that is made is that of relation by *evolutionary descent*. That is to say, that genes are passed from an ancestor to its offspring. Other methods of genetic transfer have been identified, such as *lateral gene transfer*. Phylogenetic networks are now used where the assumption of strict vertical descent should be relaxed [34, 48, 60, 85]. This and other developments serve to provide more sources of conceptual representation that

biologists consider in their research. However, phylogenetic trees remain useful and fundamental components of evolutionary analyses.

A solution space for a set of taxonomic units is the state graph of all possible binary trees where each taxonomic unit is a leaf. I denote such a state graph as a phylogenetic tree fitness landscape. Combinatorial spaces can be modelled as state graphs in order to facilitate the design of heuristic searches for an optimal solution in that space. The optimal solution is the best-fitting tree with respect to a matrix of biological character data and an objective function. This tree is assumed to best represent the evolution of the set of taxonomic units [15, 19, 22].

The size of this space grows factorially with respect to the number of taxonomic units. In most instances, searching this space is computationally intractable. Formulating and testing approximate algorithms, or heuristics, to search this space is expensive and prone to error. Furthermore, a number of possibly good techniques to search the space may not be tested due to this cost and the poor understanding of properties of the search space.

Analyses based on sequence data require hypotheses formed from phylogenetic studies, and to find the optimal solution in the phylogenetic tree landscape. In absence of phylogenetic inference, an incorrect conclusion can be made that all biological sequences are equally informative.

Phylogenetic analyses are found in key areas of health, such as genomics, in the investigation of environments, and metagenomics, such as in studies of microbiomes. They are also useful in functional annotation, a field dedicated to identifying genes to perform new chemical reactions.

This work focuses on understanding, through visualization, the properties of this search space and on identifying ways to capture its shape by sampling using a metaheuristic.

1.2 Related Work

To characterize the structure of phylogenetic tree landscapes, I abstract the search space as a state graph. This investigation also includes work in visualizing the space. Next I discuss related work in these areas.

Graph theory possesses a number of concepts widely applied in many domains.

As a way to characterize problems, graphs are versatile and suitable. In phylogenetic landscapes, trees are represented as vertices in this graph. They are connected by edges to other trees if an appropriate rearrangement can be defined from one tree to transform it to another.

The first documented account of this application of graph theory to combinatorial optimization problems is by Charleston [19]. In this first account, formal definitions are made for hill climbing and local optima. These concepts are subsequently used to support his claim that "we can... determine characteristics of the original data set which may not be immediately apparent from the results of heuristic searching or bootstrapping techniques."

Other authors also studied combinatorial features of this space. Bryant focused on *splits* or bipartitions of trees in the space made by removing specific edges [16]. Furthermore, these splits are central to the geometry of tree space as described by [10] as they are shown to correspond to dimensions in orthants of a defined geometric space. Bipartitions and their role in the space will be discussed in further detail in Section 3.2 with reference to Bryant's work. Billera et al. also developed geometric models where the space is treated as a continuous space and introduced the concept of convex hulls to phylogenetics [10].

This thesis also makes the first documented application of a metaheursitic known as Ant Colony Optimization (ACO) to produce an explored graph and its trees as an instance of the search space. This is not the first time the ACO algorithm is applied to this problem, though. Two applications of ACO to produce possible tree topologies in a stepwise fashion have been proposed in [18, 27].

As for visualization, I first discuss the concept of multidimensional scaling (MDS). MDS is a way of retaining distance information of a multi-variate dataset in a reduced dimension by finding a set of vectors to present these distances. Previous work in tree space visualization includes [59, 2, 22, 49]. Each of these methods uses MDS to embed trees into a two-dimensional space. Pairwise tree distances are typically computed using the Robinson-Foulds distance metric [70]. Hillis et al. observed that this method is capable of identifying *islands* in the space, sets of topologically-similar trees. They also explore the use of bootstrapping and Bayesian sampling [49]. Finally, an investigation of the landscape by Bastert et al. has shown that spectra are capable

of characterizing a landscape and its respective basins and optima [7].

The work of Amenta and Klingner also identify a number of design decisions that are taken into consideration in this work, including the storage of tree bipartitions, the use of consensus trees, and the use of RF distance [59].

Visualization done in this thesis will *not* be making use of MDS itself because 2D projections are inadequate for analyses. This was concluded by Cullis [22]. Instead, more effort will be put into condensing the visual space and using channels such as node size and height on a canvas to represent the relative location of trees in the combinatorial space.

1.3 Objectives

This thesis intends to extend our understanding on combinatoric search spaces in phylogeny. This is in hope of improving tree reconstruction and evolutionary inference. By finding out more about the structure of the combinatorial problem and establishing a software framework, directions for investigation are identified.

This thesis introduces a software framework for the heuristic investigation of this space. With this framework, the search space is both visualized and sampled using a metaheuristic. All of the work done in this thesis sought to achieve the following:

1. Implement a framework to generate, visualize and analyze combinatorial search spaces and components of that space.

2. Develop a heuristic to capture the shape of tree space, and sample properties from that space, with a focus on the optimal region.

I will also discuss challenges of heuristic searches of the space, the value of existing techniques, and methods of restricting the space to make the search more tractable.

1.4 Outline of Thesis

In Chapter 2, I review concepts that are essential to the understanding of the work in this thesis, focusing on the areas of combinatorics, phylogeny, and visualization.

In Chapter 3, I describe the challenges I faced with applying heuristic methods to the phylogenetic search space. Furthermore, I discuss how the potential number of

solutions in the space can be reduced by applying restrictions to edge movement in the state graph.

In Chapter 4, I reconsider the traditional objective of heuristic search. In place of finding a near optimum tree, I search for characteristics of the space. I present a notion of sampling the space for its properties.

In Chapter 5, I discuss a software framework I developed to facilitate heuristic search and exploration of the phylogenetic search space by leveraging a number of existing tools and creating new ones.

In Chapter 6, I conclude this work, summarize my contributions, and discuss a number of directions for future work.

Chapter 2

Background

This chapter is divided into four sections. The first section introduces combinatorial search spaces and the role of heuristics in these spaces. The second section discusses the domain of phylogenetics. The third section discusses principles of visualization. Finally, the fourth section provides background on a dataset that was used for many of the evaluations in this thesis.

2.1 Combinatorics

The study of a finite or countable discrete structure is a branch of mathematics known as **combinatorics**. A branch of combinatorics is *combinatorial optimization*. Combinatorial optimization aims to select an optimum from a finite set of possible states where each state can be scored using an objective function [77]. Combinatorial optimization has a number of applications in many areas including auction or game theory, software engineering, and biology. Commonly known optimization problems are the travelling salesman problem and the minimum spanning tree problem.

2.1.1 Combinatorial Search Spaces

A **combinatorial solution search space** refers to the collection of all solution states and possible transformations between these states in a combinatorial optimization problem [47].

In a discrete combinatorial search space, the set of solutions X is a discrete and countable set. In contrast to continuous sets and functions, the concept of a local minimum can only be defined if a suitable notion of neighborhood is defined.

Metric search spaces possess a set X of states. A real-valued distance function, or metric

$$d : X \times X \to \mathbb{R} \tag{2.1}$$

assigns a real-valued distance to any pair of elements $x, y \in X$ such that the following properties hold

$$
\begin{aligned}
d(x, y) &\geq 0, \\
d(x, x) &= 0, \\
d(x, y) &= d(y, x), \\
d(x, z) &\leq d(x, y) + d(y, z),
\end{aligned}
$$

where $x, y, z \in X$ [72].

The definition of a neighborhood provides the means to determine what solutions are similar to each other and to define edges when modelled as a graph. Closed local regions in a graph, or more commonly referred to as neighborhoods, are a central concept for local search. Local search is one of the early techniques proposed to cope with combinatorial optimization problems [30]. Let X be the search space containing all solutions to the problem. A neighborhood is a mapping that assigns to each solution $x \in X$ in the search space a set of solutions y that are neighbors of x.

For these combinatorial search spaces with a defined metric, let a *fitness landscape* be (X, f, d), where an objective function f measures the quality of solutions. Let X be the set of solutions and d be the distance measure or metric [90]. Let $d_{\min} = \min_{x,y \in X}(d(x, y))$, where $x \neq y$, be the minimum distance between any two elements in the space. Therefore, two solutions x, y are neighbors if $d(x, y) = d_{\min}$. The value of $d_{\min}$ is typically equal to or normalized to one, as we will find it is in phylogenetic tree reconstruction.

Fitness landscapes are described using a graph G with a vertex set corresponding to X and an edge set $E = \{(x, y) \in X \times X | d(x, y) = d_{\min}\}$ [72] where the objective function assigns a fitness value to each solution in the set of vertices. A common assumption is that the resulting graph is connected. The distance between any two solutions can be represented as the number of edges on the path of minimal length between them in G. The maximum distance $d_{\max} = \max_{x,y \in X}(d(x, y))$ is the diameter of the landscape.

2.1.2 Heuristics

Heuristics are a class of algorithms or techniques in computer science. They are algorithms designed to find satisfactory solutions to difficult problems, often NP-hard problems, that avoid the complexities of exact methods. After finding optimal or near-optimal solutions, these solutions are often validated by sensitivity analyses to determine if changes to the model or parameters for the model make a change to the resulting optimal solutions [72].

The concept of heuristics as we know it came about in in the early 1950s with artificial intelligence. Newell et al. were the first to use heuristic as a noun, referring to a heuristic process, in 1963, where they contrast it to an algorithm providing optimal solutions [71, 36]. In this thesis, I define a heuristic as the finding of a solution without guarantee of it being the correct one.

This thesis will discuss a number of heuristics and the role they play in bioinformatics. Their appeal as a topic of research stems from the intractability of finding, with guarantee, the optimum state in a very large space. It is here where heuristics find a place.

Metaheuristics are non-deterministic iterative generation processes that guide subordinate heuristics to explore and exploit search spaces. Typically, these used to overcome the limitations of single heuristics for escaping local optima. Metaheuristics are not problem-specific and may make use of domain-specific knowledge at the lower heuristic level [12]. These are valuable for problems where finding the optimal solution has no predetermined, principled procedures [54].

2.1.3 Ant Colony Optimization

Decentralized, self-organizing systems called swarm populations are a concept used in artificial intelligence and other fields of computer science. They comprise single agents interacting locally with one another. **Swarm intelligence** refers to a general set of algorithms making use of these principles. Such systems are meant to emulate systems found in nature.

A majority of the content of this subsection (2.1.3) comprises the background of a published peer-reviewed conference paper to appear in the proceedings of the *BIOINFORMATICS 2015* conference [73].

Millonas articulated five basic principles of swarm intelligence [58].

- *Proximity*: the swarm population should be able to carry out space and time computations in response to the environment,
- *Quality*: the population should be able to respond to quality factors important for group and individul survival,
- *Diverse Response*: the population should not allocate all group resources along excessively narrow and ordered lines of movement,
- *Stability*: the population should not shift behaviour for every possible fluctuation of the environment that may not produce a worthwhile return of investment, and
- *Adaptability*: the population *should* change behaviour if it is worthwhile.

The collective behaviour of decentralized, self-organizing systems, as they are found in swarm intelligence, is a concept that is easy to reconcile with a problem that has multiple states with unpredictable outcomes. All five of the principles implicitly value the importance of adaptability to a changing environment and ensuring resources are not ever put towards any single solution. In a discrete space with an unknown structure, discrete *agents* with these principles can be made to behave in a way that allows them to explore varying regions of fitness of the landscape in an intelligent manner.

The **Ant Colony Optimization** (ACO) algorithm is a metaheuristic related to swarm intelligence. It models the foraging behaviour of ants with a collective behaviour to find paths between food sources and their nests. The biological mechanism for this involves the deposit of pheromones by ants as they traverse the space. Paths of higher pheromone concentrations are chosen with higher probability by any given ant agent. Therefore, cooperative interaction emerges to solve for shortest paths [12]. A parametrized probabilistic model is employed to emulate this deposition of pheromones.

The following will discuss the simple formulation of the metaheuristic (Simple ACO or S-ACO) as described in the original paper proposing the strategy in 1999 by Dorigo et al. [28].

Definition Let $\psi = (C, L)$ be a construction graph comprised of vertex set C, ACO solution components, and edge set L. ACO solution components can correspond to states or components of states found in a combinatorial optimization problem P. Set L comprises paths between solutions in ψ.

Artificial ants perform random walks on a graph ψ. Edge $l_{hk} \in L$ between vertices $c_h, c_k \in C$ has pheromone parameter value w_{hk}. The values of these parameters, along with any other relevant *a priori* heuristic information about the problem instance, allow ants to make probabilistic decisions on their movement in the construction graph. If a constrained problem is considered, these constraints are built in so that only feasible solution components can be added to a current partial solution. A solution to the optimization problem can be expressed in terms of a feasible path, or ant trail, on this graph comprised of selected ACO components [13, 28, 54].

Besides the activities of ants, other procedures found in an ACO algorithm include pheromone trail evaporation and daemon actions. Pheromone evaporation is when the intensity of a trail deteriorates automatically over time. Probabilistically, this avoids rapid convergence towards a sub-optimal region. Daemon actions are an optional element of the algorithm to implement centralized action that cannot be performed by single ants [28].

The probabilities involved in moving on the construction graph are transition probabilities that depend on the pheromone trail values. Ants progressively build possible solutions to the problem they are trying to solve. This is done by building edges and vertices, or reinforcing existing ones, on the construction graphs with ant agent movement. These construction graphs can be associated with partial or complete solutions to the problem. While moving, ants maintain a memory of what edges and nodes they have visited in ψ. When an ant has built a candidate solution to the problem, it retraces a path back to the source node and it dies [28]. Once an ant has found a feasible solution, they retrace their path to the colony. In this sense, ants have two modes of movement: *forwards* and *backwards* [29].

New ants are generated during one of the main phases of an ACO algorithm. The first phase of the algorithm involves ant generation and activity. The second phase is when pheromone evaporation takes place [28].

Each ant of a population or colony *is* complex enough to find a feasible solution by

itself. However, good quality solutions only emerge as a result of collective interaction [28].

2.2 Phylogenetic Trees

Phylogeny strives to infer relationships amongst species, populations, individuals, or genes. In a more general sense, it aims to infer the relationship between *taxonomic units*. These inferences can be presented as phylogenetic trees. In an evolutionary sense, the notion of distance in these trees is analogous to degree of relation between taxa [64]. Time and evolutionary distance can be considered to be related if one were to assume a constant rate of evolution [91].

2.2.1 Phylogenetic Tree Reconstruction

Trees have been used to a great extent in biology to represent evolutionary relationships between species, subpopulations, and genes. These trees are generally semi-labelled, binary, and rooted. They can also be multifurcating or unrooted. They descend from a root node, bifurcate at other nodes and end at leaves labelled by the names of operational taxonomic units (OTUs). Internal nodes are thus inferred ancestors of these OTUs. Traditionally, these trees were inferred from morphological similarities, where two species that shared the most characteristics were considered siblings.

The topology of a phylogenetic tree is arguably its most important feature. A parent and subsequent children in these trees imply that, at some point in the past, an ancestral population gave rise to two or more lineages [64]. Extant taxa are those at the tips of the tree; any internal nodes above them are inferred ancestors [22].

Today, the inference of phylogenetic trees is predominantly done from sequence alignment data. A number of other data types can be used to build trees, but discussion of these are outside the scope of this thesis. Furthermore, the branch lengths of these trees represent evolutionary distance [26]. Sequence data enabled an improved understanding of the mechanisms for evolution through better models and more data to validate them. One insight it enabled is that not all species are related by evolutionary descent. Other methods of gene transfer have been identified that break this

assumption, an example being lateral gene transfer. Solutions such as phylogenetic networks have been proposed to relax this assumption [22].

A matrix of pairwise distances, paired with the use of a clustering technique, allows a tree to be inferred. A matrix of pairwise distances can be computed from the alignment of DNA and protein sequences. This drives the notion of phylogenetic inference. Given an MSA and a phylogenetic tree t , we can compute a measure of how well t fits a dataset. As a result, phylogenetic inference can be modelled as an optimization problem in order to optimize the topology of a phylogenetic tree to represent evolutionary reality.

A rooted tree assumes a root as a common ancestor. It is from this node that time flows to any other node in the tree. On the other hand, an unrooted tree specifies only interrelations with no direction in which evolution occurs. Note, though, that approximate assignment of a root can be made by designating an outgroup. Outgroups are OTUs that are distantly related to all other OTUs in the analysis [64].

The construction of these trees is a clustering problem where clusters represent the process of differentiation over a period of time [91]. Simple clustering techniques include Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [82] and Neighbour-Joining [74]. The former assumes a constant rate of evolution in the span of the dataset, but this assumption could be rejected in many cases; therefore, this technique is typically used only to guide more complex methods. The latter technique assumes that the distances used are an exact reflection of real evolutionary distances to create a rooted dendrogram . This is guaranteed to find the true tree if the distance matrix is an exact reflection of the tree. However, distance estimates break down at a certain point, making this method not always the most effective solution [91].

A number of methods exist for the construction of phylogenetic trees. What all of these methods share is an evaluation criterion, a fitness function that can score how well a particular tree solution fits the genealogical relationships in a sequence alignment. Examples of functions include maximum likelihood and parsimony [40]; optimization of this function would lead to a tree that optimally fits a matrix [7]. Tree reconstruction problems, or that is to say the constructions of optimal trees, are all NP-complete. Exhaustive searches to minimize f are restricted to values of $n \leq 12$ [26, 43].

Biologists have to therefore use approximate optimization algorithms. These algorithms are usually associated with certain starting points in the graph and certain random moves between trees that makes the approach non-deterministic. Optimization can therefore be approached by considering what are typically referred to as *discrete character approaches*. Such search methods include

- Parsimonious Criterion (Parsimony) Methods,
- Maximum Likelihood Criterion (ML) Methods, and
- Bayesian Statistical Methods.

Maximizing parsimony selects the simpler of any given and otherwise equally viable hypotheses and involves choosing a tree that supposes the smallest amount of evolutionary change to explain the data. This is a method first proposed by Fitch [40]. His algorithm, the Fitch algorithm, chooses the most straightforward relationships, but therefore comes with the problem of assuming all evolutionary changes have a similar probability. This does not effectively deal with saturation of data and fails to model evolution where selection is at play [91].

Maximizing likelihood criteria involves abstracting a set of items, which in this case would be sequences, into a model where we know all instances in the set are stochastically derived from a unique parent. This is grounded on a Markov process where a tree is sought that maximizes the likelihood of the data given in the Markov process. This is done by considering the tree topologies and distances [65].

Bayesian statistics makes use of the Bayesian statistical paradigm [9] in order to infer phylogeny by assessing statistical confidence with an expanded definition of probability from classical statistics [81] to determine a best score very much akin to likelihood.

We can also discuss exploring the topology space. Given a tree in the space, one can arrive at a different tree using a method of step-wise addition or tree-rearrangement.

2.2.2 Phylogenetic Tree Spaces

In biology, systematists and biologists often need to analyse large collections of phylogenetic trees that all are potential solutions to a phylogenetic problem, either because

of similar scores of fitness or because they all come out of, say, a Markov chain Monte Carlo (MCMC) Bayesian analysis. Many of these trees can be used for further processing, such as for consensus analysis used to summarize a collection of trees into a single tree using various techniques that each attempt to make trade-offs on what information is retained or lost after the consensus process [14].

A **phylogenetic landscape** (or tree space) refers to the entire discrete combinatorial space comprising all possible phylogenetic tree topologies for n taxa. Each of these landscapes is constructed from a matrix of biological character data. Landscapes possess a finite set $T = t_1, t_2, ..., t_i, ..., t_{x-1}, t_x$ of x configurations with a fitness function $f(t_i)$ forming a discrete solution search space and a finite graph $G = (T, E)$ where G can be viewed as a function of its vertices [19, 83].

The edge set E of G refers to the neighborhood relation on T joining two vertices by an edge if one can be transformed into the other using a single leaf addition or a local rearrangement operation. Local rearrangement uses a base tree and rearranges the topology of branches. There are three basic methods of rearrangement:

- Nearest Neighbour Interchange (NNI),
- Subtree Pruning and Regrafting (SPR), and
- Tree Bisection and Reconstruction (TBR).

Nearest Neighbour Interchange is an operator by which four subtrees within the main tree have their connectivities exchanged: there are three possible ways of connecting four subtrees with one being the original connectivity — therefore, an interchange creates two new trees [39]. This exchange can be modelled as a movement or regrafting and pruning of subtrees.

SPR is an extension of the NNI operation where subtrees can be regrafted at any arbitrary distance from where they are pruned. Any interior edge in the tree t can be used as a cut edge to separate the tree into new trees t' and t'' where either of these latter two trees can be grafted onto all possible edges of the other. The number of possible target edges for a regraft is equal to $2n - 8$, where n is the number of leaves of t. Notice that the number of distinct SPRs will be smaller; this process can yield the same neighbour in some cases [22].

Tree Bisection and Reconnection is an extension of SPR. It involves, for each cut interior branch in t the yielding of a set of trees t' and t'' and the consideration of all possible connections made between branches of each of these two with the other to generate rearrangements [22].

The phylogenetic tree space is central to an example of a combinatorial optimization problem [12]. The set of all trees T is known as the search space. Each element of the set is a candidate solution. An optimal solution in tree space is can be said to be a solution $t^* \in T$ with maximum fitness function value $f(t^*) \geq f(t) \forall t \in T$.

Let $\mathcal{T}(n)$ be a function of the number of taxa that produces the number of possible reconfigurations of tree topologies, or possible solutions, for that number of leaves on a phylogenetic tree. This determines the number of the nodes of the graph. Since phylogenetic trees are often rooted and binary, I define $\mathcal{T}(n)$ as the number of possible strictly binary *rooted* phylogenetic trees,

$$\mathcal{T}(n) = \frac{(2n-3)!}{2^{n-2}(n-2)!} \ . \tag{2.2}$$

The number of unrooted trees for n sequences or taxa is equal to the number of rooted trees for $(n-1)$ sequences [37].

Comprehensive investigation of the phylogenetic landscape would involve characterization of the landscape by acquisition of empirical properties including the number and relative scores and locations of local optima, the sizes of their basins of attraction, the number of components of G, and the respective maximal steepest climb lengths (MSCLs) [19].

Determining where one is in the search space relative to the optimal solution is very difficult. Given a representation of these landscapes in terms of $f(t_i)$, there is no trivial path to a global optimum. The fitness values of the topologies in this landscape define a large number of basins of attraction and local optima [22]. Computation of a measure of distance from a state to the global optimum is therefore difficult. The presence of multiple maxima is a confounding factor in heuristic searches which may lead to drawing incorrect conclusions on evolutionary histories.

2.2.3 Representation of Phylogenetic Trees

A Newick, or New Hampshire tree format, string can describe labelled trees with edge lengths. This format uses a grammar of parentheses and commas. All strings end with a semi-colon. Its construction is found by considering the tree's in-order traversal in its construction. This format is the standard way of storing a collection of phylogenetic trees in biological literature and was adopted in 1986 [38]. An example of a tree in this format is

```
(a,((b,(c,d)),e));
```

where its associated structure is found in Figure 2.1.

To minimize efforts needed to perform conversions between this standard format and other formats, the representation of trees in this project will be in the New Hampshire tree format.

If we wish to improve upon this format, we can go on to consider the information-theoretic lower bound of tree representations. Phylogenetic trees are often rooted and binary. This form of tree can actually be considered to be an *Otter tree* after the work of Richard Otter [66], formally defined as unordered rooted trees with n leaves where all nodes, including the root, have either zero or two children. The Wedderburn-Etherington number sequence [33] is therefore applicable here such that the nth number of this sequence represents the number of Otter trees with n leaves. Given this, the information-theoretic lower bound for the number of bits required to represent Otter trees with n nodes is equal to $(1.3122...)n + o(n)$ bits and is less than the number of bits required to represent n-leaf ordinal trees, which is $2n + o(n)$ [35].

With this in consideration, other forms of phylogenetic tree representation have been proposed in order to reduce space needed to represent phylogenetic trees, such as the TASPI format [14], as well as succinct tree representation. Succinct represents include the Level-Order Unary Degree Sequence (LOUDS) representation [50]. Those representations will be left for the reader to consider, and will be revisited in Section 6.2).

2.3 Visualization

The composition of phylogenetic trees as nodes and branches means there are few limitations on how their topology can be depicted. The most important feature to preserve is the connections between nodes. As a result, there is an abundance of techniques for visualizing trees.

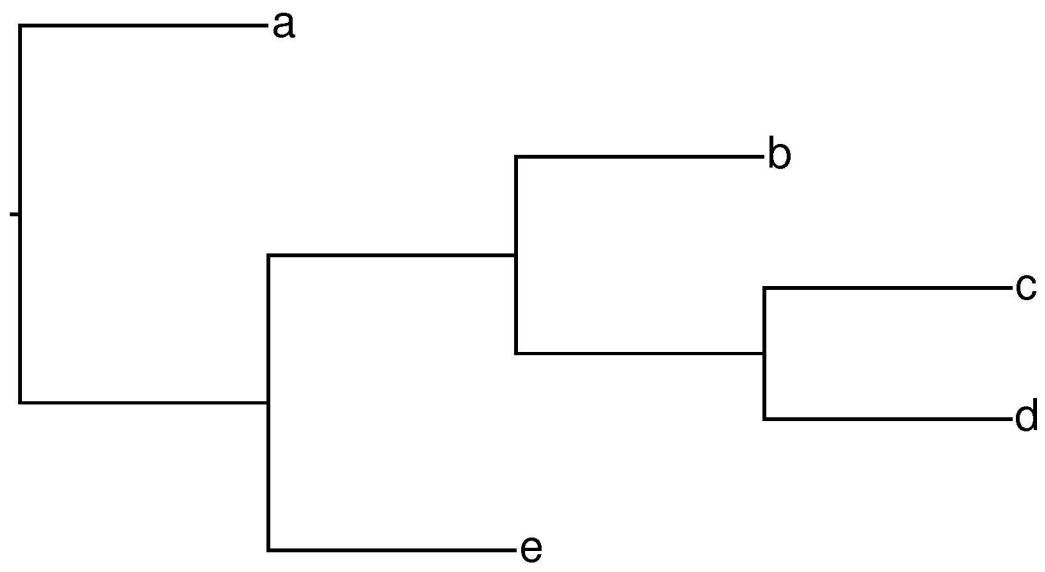


Figure 2.1: **A Phylogenetic Tree**. A phylogenetic tree presented in the manner typical for biological literature.

The most common technique for representing phylogenies is in a two-dimensional Euclidean plane (Figure 2.1). For very large trees, one can fold or collapse subtrees in order to save space, a process that can be manual or automated. Another method is to distort the space such that more can be shown in the same amount of area.

Other methods of representing trees include treemaps, a method best suited for classifications that lays out a tree into a series of nested rectangles, hyperbolic space in two or three dimensions, and three-dimensional fly-through or stacked tree layouts [67].

Visualizing a single tree is an important element of phylogenetic studies, but one can also investigate the larger, macroscopic, space that these trees are located in when in relation to other trees of the same number of leaves. In other words, one could visualize the combinatorial phylogenetic tree space.

However, techniques developed for graphs of up to 100 nodes typically fail to present coherent information; therefore, for greater number of leaves, traditional methods of visualization will become more difficult to use effectively. This tends to be a result of algorithmic or physical limitations [86]. For example, interactivity cannot be supported for such a large number of nodes, a high-complexity layout algorithm, or there is a lack of screen space to lay out the information in a clear manner.

Regardless of the approach, the problem of limited screen space is very difficult to overcome when considering a large number of nodes in a graph that one would wish to visualize. Simple solutions involve using the space more efficiently or by considering other geometric spaces, such as hyperbolic or three-dimensional spaces. Another solution involves presenting the user with only a limited subset of the data at any given time [86] or by creating a consensus to distil the information about multiple trees into one single summary tree [49].

2.4 Data Set

A number of sections in this thesis make use of a set or subset of biological character data associated with a family of bacteria. This data set was chosen because it is well studied and is comprised of a reasonable number of OTUs for in-depth analysis.

There are a number of bacterial families that are very abundant in the human microbiome [57]. A family in this grouping is the Lachnospiraceae family of bacteria (of the phylum Firmicutes and class Clostridia). The Lachnospiraceae family is significantly abundant in the digestive tracts of many mammals and found to be relatively uncommon anywhere else. This family is also used as an indicator for the production of butyric acid; this allows inferences associated with obesity and protection from colon cancer in humans. Recent claims suggest that not all genomes in this family produce it in either of two known pathways.

There are two known pathways responsible for the fermentation of the short chain fatty butyric acid. One goes through the enzyme butyrate kinase and one goes through butyryl-CoA:acetate CoA-transferase (BCoAT) [88]. It appears that this production is bound mainly to the organisms found within the class Clostridia [53] and has been found in many strains of the family Lachnospiraceae [4, 31, 20, 53]. Further

investigation has revealed that, of the two pathways, BCoAT activity is suggested to provide the dominant route for formation of the acid in the human colonic ecosystem containing a high concentration of acetate [52]. Only twelve of the thirty sequenced organisms investigated by Meehan et al. contained genes from either of these two pathways; it has been proposed that this is a genus-specific attribute in this family and a TBLASTN analysis did not seem to suggest otherwise [57].

The data acquired is that involved in the phylogenetic examination done by Meehan et al. which revealed the possibility of LGT events. Trimmed protein sequences of both enzymes, butyrate kinase and BCoAT, were collected for a series of genomes in the Lachnospiraceae family. To compare against, 16S trees were derived from the same genomes. There is a significance difference of the gene trees revealed by these two comparisons of alignments [57, 69]. The reason for choice of this data set for analysis comes about from this evidence of LGT events. A greater source for error regarding classification of the OTUs in this set suggests insight into the robustness of the techniques suggested in this thesis.

Two Lachnospiraceae-associated genes were chosen corresponding to the two enzymes associated with the pathways for the production of butyric acid. These were butyrate kinase and butyryl-CoA:acetate CoA-transferase (BCoAT).

Sequenced genomes from the family were acquired from the NCBI database on April 18, 2012 resulting in 30 genomes [57]. A total of 3500 protein sequences were acquired in FASTA format, aligned using MUSCLE 3.8.31 [32], and then trimmed using BMGE version 1.1 [21] with a BLOSUM30 matrix.

The result of this workflow was four files. Two are associated with the butyrate kinase protein and two are associated with the BCoAT enzyme. For each enzyme, an alignment of all trimmed protein sequences constitutes one file, and an alignment of all relevant 16S ribosomal RNA sequences for the respective genomes constitutes the other.

These files were obtained from the authors of [57] and are being used with their permission.

Chapter 3

Heuristic Searches in Phylogenetic Tree Space

A number of combinatorial optimization problems can involve the use of heuristics. Some problems may be suitable for black-box application of various heuristics or algorithms. However, more often than not, combinatorial optimization problems need to have heuristics designed or modified for them. The problem of finding an optimal tree in phylogenetics is no exception for the latter.

The first section of this chapter will investigate challenges of solution representation in the phylogenetic tree space that must be overcome by any techniques used to search it for solutions. The second section will identify properties of the problem that can be exploited for new search techniques.

3.1 Overcoming Challenges of Representation in Tree Space

The shape of a phylogenetic tree and its hypothesized pattern of branching are known as its *topology*. The more common ancestors, internal or named, two taxa share, to the exclusion of other species, the more closely related they are. Arrangements of the leaves of a phylogenetic tree is unimportant in the sense that every internal node of a tree can have its children notionally rotated without damaging the topology [44].

The redundancy of different tree representations for identical tree structures poses a major challenge for software implementation. A representation needs to be identified that is unambiguous for a tree structure in a space. Similarly, there needs to be a method to avoid adding trees to the space if they already exist in it.

3.1.1 Lowest-Order Taxon Rerooting Technique

In order to make Newick strings consistent amongst trees in a phylogenetic tree search space, an arbitrary but efficient rooting strategy is used to avoid redundancy. Before consideration of any tree within a search space, the tree is rerooted to the lexicographically lowest-order taxon name. If branch lengths and support values are subsequently

stripped from the Newick string, and only taxa labels are retained, only topological information is preserved for any given tree topology. This means that different rearrangements that lead to the same topology can still be recognized as not being a new addition to the space. This technique is not novel. It was described and used in [11].

3.1.2 Using PATRICIA Trees to Store Topologies

The probability of reevaluating a tree that was already constructed from a previous rearrangement operation rises with the amount of exploration done in a space. This duplication unnecessarily extends tree search. We can either:

1. Avoid reevaluating these trees, or

2. Ensure these trees are not added redundantly to the space upon reevaluation.

The first option would involve recognizing moves as leading to redundant trees before a rearrangement is performed, when a subtree is chosen to be rearranged. To facilitate this, space complexity could be unnecessarily large. We would have to store the unique component pieces of the tree topologies across the space and identify what movements of pieces lead to redundant constructions.

Since only the Newick strings for all trees are stored, the second option was preferred. A suitable data structure was sought for storing these unique string representations that would provide efficient lookup time and space complexity. This data structure came to be a variant of the traditional trie data structure.

The trie data structure is an ordered tree data structure used for associative arrays where keys are strings. Position in this tree is associated with a particular key and all descended of a node in this tree have a common prefix of a string associated with that node – the root is associated with the empty string and leaves correspond to the complete length of a stored string key.

A compressed trie structure was chosen and tested to allow quick access time and membership operations. Further effort was also applied to improve the compressed trie data structure to generalize it for tree traversal in order to provide additional functionality that may possibly provide tools for one day using the first, rather than the second, option.

A **PATRICIA tree**, or compressed trie, is a space-optimized representation of a trie data structure where nodes that only have a single child are merged with their parents [62]. Leaves of this structure would reference a value that is associated with some string key. Searching for some string in a PATRICIA tree is similar to search in a trie, except that when the search traverses an edge, the edge label is checked against a substring of the query and not just one character [61].

Constructing a fast, compact, and scalable trie structure for strings remains an open problem in Computer Science [3]. However, the PATRICIA tree is a compact structure that is seen as being an applicable solution for this lookup time. The PATRICIA tree is intended for sequences of strings and allows for $O(k)$ lookup time at worst case for query strings of length k. Furthermore, there is evidence of it having been used for a similar application before [11]. Other structures that were also considered include a hash table, suffix tree [6] or array, and wavelet trie [45]. I initially chose to forego the use of a hash table as they require an $O(k)$ *expected* cost for a lookup due to their need to hash the key. Also, hash tables do not allow for certain operations that a PATRICIA tree supports (see below). The suffix tree proved to be insufficient due to their use on large texts and not sequences of strings associated with independent objects. Lastly, the wavelet trie was deemed to be too complicated to implement within the scope of this thesis. Despite that, its complex implementation may have been worth its improvement with regards to space. The wavelet trie was designed to be a succinct structure making use of the idea of wavelet trees and providing storage complexity close to the information-theoretic lower bound.

While lookup time is an important concern, it is not the only one. As these landscapes can get very large, a data structure for lookup of trees should scale to large inputs. Let q be the number of trees in a landscape prior to an addition or deletion of a tree. The PATRICIA tree would occupy only $O(qn)$ space.

Let Σ be an alphabet of σ characters used to encode phylogenetic trees in the Newick format. When searching for a tree, we use a pattern string P of length p over Σ.

Consider the following formulation of a trie data structure. A trie can represent strings in $\{t_1, ..., t_q\}$ as a rooted, ordinal tree structure where child branches are labelled with characters from Σ and strings are terminated with a special character

\$ less than any other symbol. With a sorted order of σ children for each node in the trie, searching for a string P in this set takes $O(p)$ time. The space complexity of this structure is $O(Q\sigma)$ where Q is the total length of all q strings concatenated together and corresponds to the upper bound of the number of nodes in the trie [61, 80].

Operation	Time Complexity
`query(`P`)`	$O(p)$
`insert(`P`)`	$O(p+\sigma)$
`delete(`P`)`	$O(p)$
`prefixes(`P`)`	$O(q)$

Table 3.1: **PATRCIA Tree Structure Supported Operations**. Available operations of the PATRICIA tree structure and corresponding complexities.

A PATRICIA tree also represents the same set of strings as a rooted ordinal tree, but with labels of edges referring to substrings over the alphabet Σ rather than single characters. The query time for a string P in this structure remains $O(p)$ if a pointer/length representation for strings is used [61]. That is to say, that the representation of strings in memory presupposes a pointer to the beginning of a string as an array of characters and the string's length is known. Substrings can therefore be identified in constant time, comparisons can be made in constant time, and if a search compares all characters of P, the search is a success and reaches a leaf associated with that string. The space complexity for this compressed trie structure is comparatively better than the trie's. Because every node either (1) is a leaf or (2) has ≥ 2 children, the number of non-leaf or internal nodes is not greater than the number of leaves, and therefore the bounding terms of the complexity are merely the leaves of this structure where the complexity is $O(q\sigma + Q)$ [61].

An important consideration for the structure is its ability to support insertion and deletion of a string P in $O(p+\sigma)$ time. Inserting a string into the PATRICIA tree first involves a query operation. If the operation is found, nothing needs to be done. Otherwise, there are two cases for where the search stops with respect to the structure:

Case 1. The search stops in the middle of an edge during a substring comparison.

Case 2. The search stops at a node.

For Case 1, the edge in question is thereafter split into two new edges joined by a new node. For Case 2, the remainder of P which was not matched during the query operation is used as an edge label from the node the search stopped at to a new leaf associated with P. Both of these cases take $O(p+\sigma)$ time because they always involve a query for P, taking $O(p)$ time, and a creation of at most two new nodes follow, each with σ child branches.

Deleting a string also involves a query which should succeed, or otherwise nothing needs to be deleted. The leaf corresponding to P is found and removed. After this, if the parent of that leaf is left with a single child, that child is merged upwards with the parent node to form a modified node and edge containing the information from the single child. This takes $O(p)$ time because it involves a query that takes $O(p)$ time and a deletion of at most two nodes which can be considered to be constant time.

Prefix matches, too, can be made in an efficient time where a list of strings with a non-null-terminated string P prefix can be returned by searching for P until running out of characters to compare. From this point forward, every leaf in the subtree that follows from the node stopped at in the search corresponds to a string that is prefixed by P and this subtree is traversed in a time that is upper bounded by the number of leaves in that subtree, $O(q)$. This is useful for the application problem as this could allow for partial matches and searches by phylogenetic subtrees rather than individual trees, if such a match can be made using a common prefix [61].

3.1.3 Evaluation of PATRICIA Trees for Topology Searches

A number of other structures were considered for use in the space, and a former sequential Newick search was used to search for existing topologies across a linear list. The hypothesis here is that the PATRICIA tree will perform comparably with a trie and related structures, better than the previously implemented sequential search, and possess a reasonable space complexity for the stated problem. To evaluate this, benchmarking was performed against existing implementations.

The existing implementations that were tested against include the hash table, the traditional trie data structure, and the sequential Newick search. These were

chosen because they are readily available within the Python standard library, were constructed during the implementation of the PATRICIA tree structure, and already existed within the code framework respectively.

Benchmarking was performed on a desktop computer with an Intel© Core™2 Quad CPU at 2.66 GHz running Linux Mint 16 64-bit and possessing 8 GB of memory. A large number of queries and insertions are performed on each respective structure or implementation over this landscape of trees for different sized uniformly random subsets of the tree space of size $\{1000, 3000, 5000, 7000, 9000, 18000\}$. This landscape was created using data from Section 2.4. Its construction is tabulated in Section 4.2.2. In addition, queries are made for trees that are not present in the landscape. For each query operation, 25 replicate queries were performed in order to acquire a large enough sample size to perform meaningful statistics on the execution time of these operations. Scripts employed for testing these implementations considers both memory usage and computation time of these operations.

After performing benchmarking, results were obtained for both running times of a series of operations (Table 3.2) and the memory usage of the structure after insertion of a subset of trees from the phylogenetic landscape (Table 3.3).

We see that the PATRICIA tree performs insertions more slowly than the trie, but `query` operations are faster by at most an order of magnitude. Furthermore, the memory usage of the structure is considerably smaller than the trie's by orders of magnitude.

An interesting observation that can be made, however, is that the PATRICIA tree is actually slightly slower than the hash table in all respects for running time, and slightly larger than the hash table. While this is troubling, we can postulate this small difference may come about from implementation details as the scaling, from what data I have, appears to be equal between the two as the number of trees increased. When considering the source code of the hash table from the Python standard library, we see that they perform linear probing but ensure the dictionary is only ever at most $\frac{2}{3}$ full. Despite this, the PATRICIA tree implementation possesses the `prefixes` operation and a traditional hash table does not support this. The small difference in speed and memory usage could be accepted in light of this.

The insertion of trees were not tabulated for the sequential search implementation

due to the fact that this implementation presupposes trees have already been loaded into the structure. Furthermore, the only relevant statistic for this implementation comes from the speed of its operations. On this note, we are able to observe that the sequential Newick search implementation performs orders of magnitude slower for both `query` and `insert` operations compared to the other three implementations. This is in line with the fact that its complexity is $O(n)$ rather than the ones with respect to the search string's length for the other implementations.

A hash table implementation was used in the final software framework (Chapter 5), but it is recognized that the PATRICIA tree implementation could be favored if additional functionality is sought.

Table 3.2: **PATRICA Tree Implementation Time Benchmarking**. For the two operations, `insert` and `query`, the average running time for three actions are tabulated. Let Insert be as the `insert` operation, Search be a `query` on a tree that exists in the structure, and Decoy be a `query` on a tree that is not in the structure. These actions are performed on the four implementations that were benchmarked.

Number of Trees	Implementation Operation Avg. Running Time (μs)										
	PATRICIA Tree			Hash Table			Trie			Sequential Search	
	Insert	Search	Decoy	Insert	Search	Decoy	Insert	Search	Decoy	Search	Decoy
1000	180	41	2.0	2.0	1.0	1.0	7400	100	1.0	13000	26000
3000	86	43	2.0	1.0	1.0	1.0	7300	100	1.0	13000	26000
5000	87	44	2.0	1.0	1.0	1.0	6800	110	1.0	13000	27000
7000	92	45	2.0	1.0	1.0	1.0	7200	110	1.1	13000	26000
9000	190	45	2.0	1.0	1.0	1.0	7100	110	1.0	13000	26000
18000	190	46	2.0	1.0	1.0	1.0	6900	110	1.0	13000	26000

Table 3.3: **PATRICA Tree Implementation Memory Benchmarking**. Memory usage of respective structures after insertion of trees for a respective subset of the phylogenetic landscape.

Number of Trees	Implementation Memory Usage (MB)		
	PATRICIA Tree	Hash Table	Trie
1000	0.34	0.22	14.02
3000	1.01	0.70	40.39
5000	1.65	1.04	65.82
7000	2.30	1.96	91.12
9000	2.95	2.30	112.32
18000	5.86	3.81	213.78

3.2 Restricting the Space Using Branch-Locking

Using the term borrowed from nomenclature of a bipartite graph, a bipartition for a phylogenetic tree coincides with the definition of two disjoint sets U and V. A branch in a phylogenetic tree defines a single bipartition that divides the tree into two disjoint sets U and V. The set U comprises all of the children leaf of the subtree associated with that branch. The set V contains the rest of the leaves or taxa in the tree.

Decomposing phylogenetic trees into collections of splits can have important algorithmic consequences. In his work, Bryant has shown that constraining tree search to trees with splits contained within a given set can make our NP-hard optimization problem polynomial time solvable [16, 15, 17].

I present a different strategy to take advantage of the decomposition of trees into splits in order to reduce the number of solutions in the space.

3.2.1 Fixing Bipartitions as a Locking Technique

There may be preconceived knowledge of certain groupings of taxa that are certain to be evolutionary grouped in a clade or set of clades. Similarly, we could find empirical evidence from exploration of the space that provides a similar conclusion.

I claim that these particular bipartitions could therefore be *locked*. Locking a bipartition is a restriction placed on G such that no branch movement can be invoked across the branch associated with it. Rearrangements could therefore not be proposed for a tree topology that would violate this restriction. This serves to dramatically reduce the size of the search space and the construction graph.

In order to acquire empirical evidence from already explored spaces, we can consider the respective scores of resultant, new, topologies after rearranging bipartitions or branches in trees of the space. We define this as the "score of a bipartition". Using this information, I hypothesize that we could identify regions of the space that need not be explored due to poor scoring bipartitions. Avoiding to score such regions provides a significant economy in computation.

3.2.2 Finding Empirical Evidence from the Space

Considering the bacterial genome data of interest (Section 2.4), I investigate if I can use information associated with splits in phylogenetic trees. One method of visualizing properties in the space that was investigated involved a heterogeneous chart. This chart combines a sequence of box plots and lays them out across both axes. I tentatively name this heterogeneous chart a steepest-ascent box plot chart. Two examples of these charts are shown in Figures 3.1 and 3.2. using the BCoAT 16S rRNA landscape. This landscape was visualized in part and is both shown and described in Section 4.2.3.

The y-axis represents log-likelihood, an objective function that has been commonly used for evaluating trees in this thesis. The x-axis labels unique bipartitions present in trees across the space as a string of ASCII characters, to the exclusion of a colon character, each character indicating a taxon. These characters are on one side of a colon; the side it is on indicates what other taxon it is grouped with to comprise a bipartition. The box plots present are associated with one of these bipartitions and show the range of scores of trees that are found for trees that are constructed when a branch associated with that bipartition is rearranged to another part of the tree.

Tree ID	Log-Likelihood	Median Δ Log-Likelihood
0	-11947.60	-112.11
1773	-11250.73	-119.16
30728	-10776.84	-157.64
44410	-10174.84	-160.23
64524	-10061.48	-183.24
66509	-10048.18	-188.64
69152	-10037.54	-191.86
70617	-10030.27	-200.17
72670	-10023.54	-211.72
73541	-10019.41	-206.92
75482	-10017.24	-203.00
75676	-10017.24	-207.36

Table 3.4: **BCoAT 16S rRNA Maximum Improvement Scores from a Starting Tree**. Median Δ log-likelihood values across all bipartition rearrangement resultant tree scores along a path of best improvement from a FastTree starting tree (Tree ID: 0) in the landscape to an optimum (Tree ID: 75676).

These two example charts provide examples of the different insight these charts can provide depending on the trees sampled and their location in the space. Starting from a phylogenetic tree provided by the FastTree software [68], we are capable of acquiring meaningful data about the presence of particular bipartitions and their absence based on proximity to an optimum. We can observe in Figure 3.1 that as we find better scoring trees along the path from this initial position in the space, certain bipartitions will become absent in trees closer to the optimum and bipartitions that were not present in the initial tree will become present in better scoring trees. Furthermore, bipartitions that, when rearranged, provide better scoring trees become less prominent in trees that are closer to the optimum tree, indicating a sort of stability in score improvement upon rearrangement of a given tree.

If we look at the tree in a region of the landscape that consists of higher ranging trees (Figure 3.2), we see the score difference between trees along a path of improvement to be almost minimal. This is a stark contrast to the gradually decreasing, but initially large, difference in scores of trees as we get nearer to an optimum in the space found in the other chart.

If we look more closely at the median change in log-likelihood score that occurs across all of the bipartitions in these trees, and the possible rearrangements that can be done on those bipartitions (Table 3.4), we see a gradual decrease in this median. This tells us that over all bipartitions for those trees, the resultant scores of trees after rearrangement of a bipartition begins to worsen. Furthermore, rearranging splits in those trees provide worse trees more often.

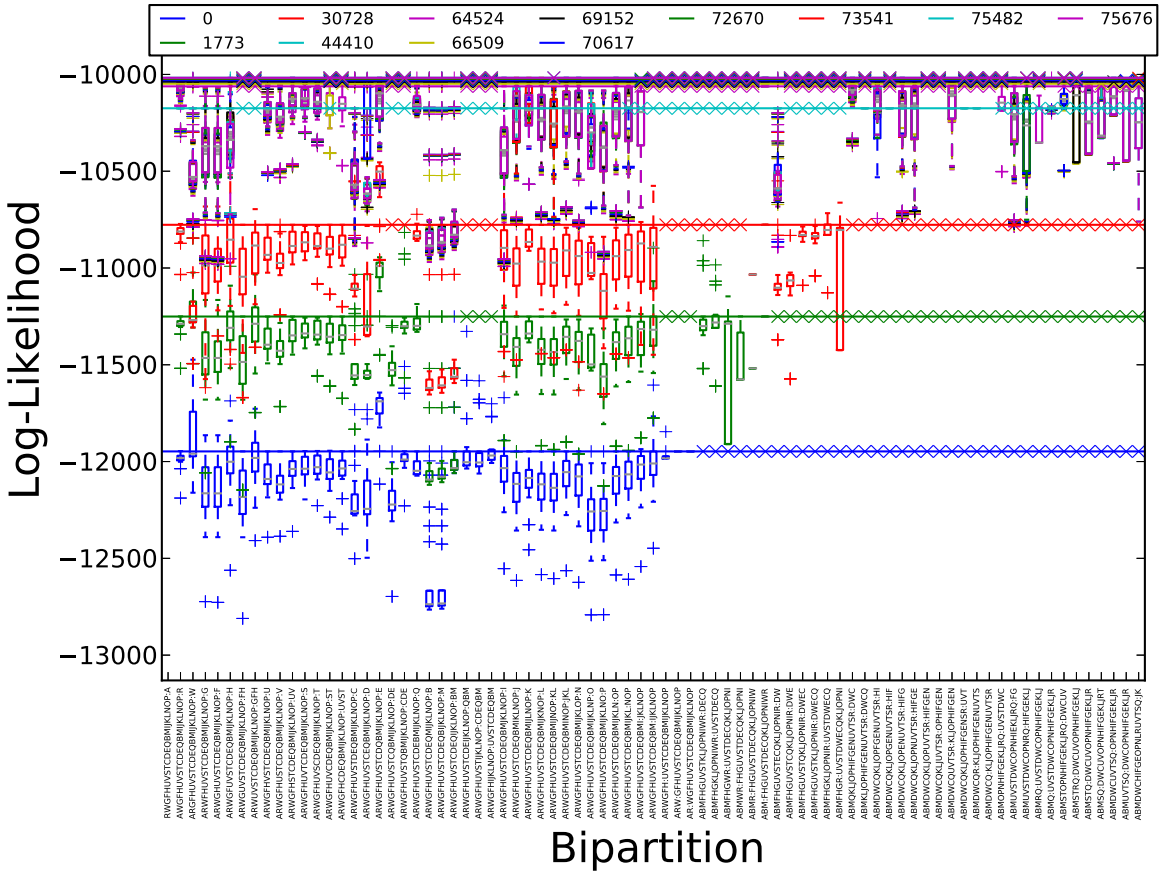


Figure 3.1: **BCoAT 16S rRNA Steepest-Ascent Box Plot Chart from a Starting Tree**. The visualized improvement landscape acquired for the butyryl-CoA:acetate CoA-transferase (BCoAT) 16S rRNA gene sequence alignment where the path of best improvement from a FastTree tree is shown using log-likelihood. These are shown as collections of box plots. The height of these box plots corresponds to log-likelihood score, associate to trees labelled by an integer, and the position on the x-axis corresponds to unique bipartitions.

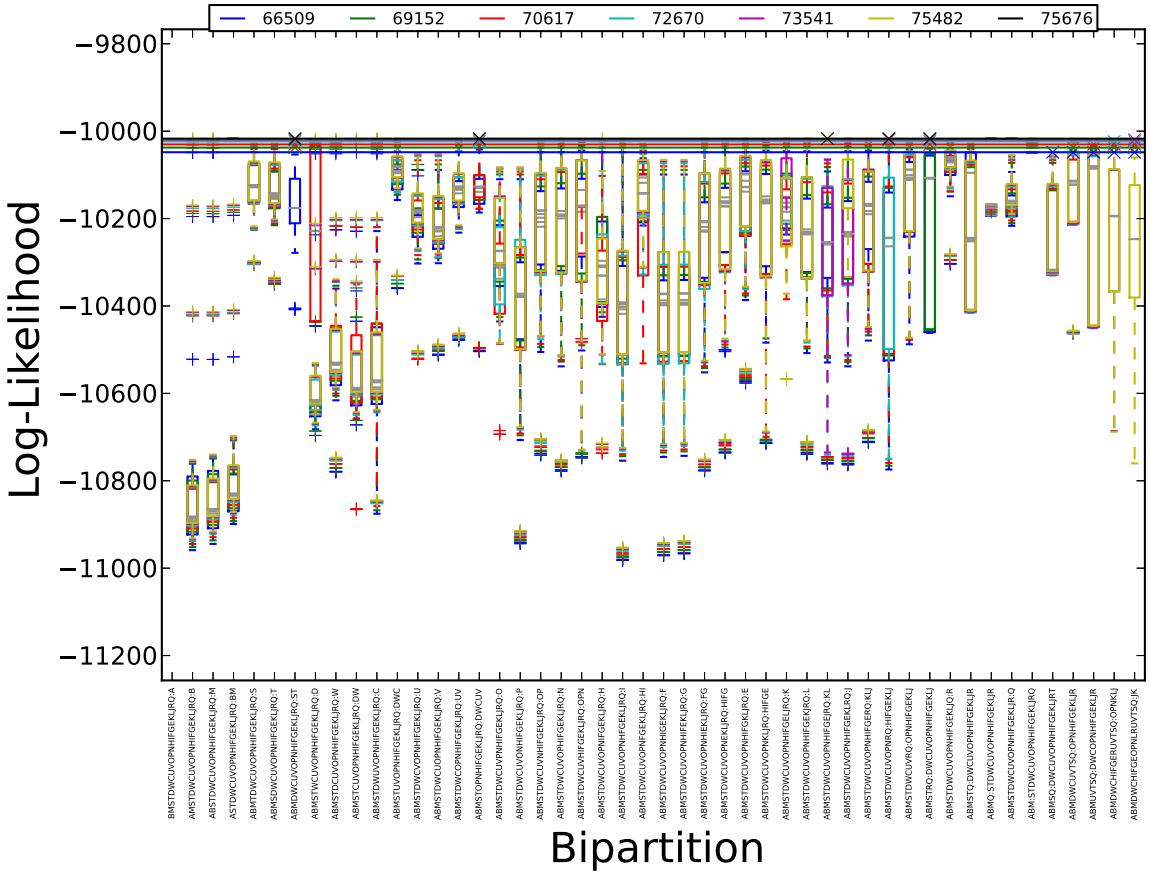


Figure 3.2: **BCoAT 16S rRNA Steepest-Ascent Box Plot Chart from an Arbitrary Tree**. The visualized improvement landscape acquired for the butyryl-CoA:acetate CoA-transferase (BCoAT) 16S rRNA sequence alignment where the path of best improvement from an arbitrary tree in the landscape is shown using log-likelihood. These are shown as collections of box plots. The height of these box plots corresponds to log-likelihood score, associate to trees labelled by an integer, and the position on the x-axis corresponds to unique bipartitions.

Chapter 4

Finding Structure in the Space

In this chapter, I abandon the traditional objective of heuristic searches in a fitness landscape. Whereas heuristics in the space are intended to find a near optimum solution, I reconsider this objective for one that serves to fulfil our objectives (Section 1.3). In place of finding an optimum solution, I formulate a heuristic and interactive strategy to define characteristics of the space.

In the first section, I will introduce the concept of sampling the space for its properties using a new metaheuristic method. In the second section of this chapter, I introduce an interactive software framework for searching and exploring the phylogenetic fitness landscape.

4.1 Metaheuristic Sampling of the Space

This section introduces a metaheuristic method of sampling the space. This is proposed for two reasons. Firstly, this will allow us to identify properties of the space that are not immediately garnered using traditional heuristic strategies. Secondly, this will also provide insight on the question of uncertainty of a true phylogenetic tree in a space which traditional heuristic methods attempt to find.

Little is known about the underlying structure of phylogenetic tree landscapes. The use of a heuristic provides insight into this structure if, instead of optimality by acquisition of a single structure, coverage of optimal features of the space is maximized in this discrete space. In other words, *sampling* from the space can also allow for identification of other local optima and features of the space. Such features include regions of poor score. Also present are terraces where there are multiple states of almost equal fitness function [75]. These are troublesome areas for a heuristic. Defining the boundaries of and deciding on an optimal move in such a region can

A majority of the content of this section (4.1) comprises the methods and results of a published peer-reviewed conference paper to appear in the proceedings of the *BIOINFORMATICS 2015* conference [73].

become an issue. If we identify properties of the structure of the problem, features can be uncovered in order to better design heuristics more capable of optimality. Sampling the search space can help us do this.

Furthermore, uncertainty of a true phylogenetic tree can arise from problems of statistical stability. Classical tree-building algorithms and heuristics look to find a single tree that is consistent with biological character data. We could still question whether or not this tree is correct. This tree can be seen to be an unknown parameter that is being estimated. There is the possibility that a small change in data, particularly arising from alignment or sequencing error, could result in a change of choice for this tree. Thankfully, assessment of sensitivity to sampling error can be done by methods such as bootstrapping. What remains, though, is a question of whether or not acquiring more of a sampling of trees from the landscape could provide a larger collection of equally informative trees. We will investigate this hypothesis by considering a confidence interval of collections of high scoring trees in sampled spaces [10].

4.1.1 Phylogenetic Landscape ACO (PLACO)

The purpose of our proposed application of the ACO algorithm (Section 2.1.3) is to sample the phylogenetic tree space. Because of the intractability of the problem, and the cost of computation, I will avoid poorly scoring regions unless such regions are necessary for ants to visit in order to move to a region of near globally optimal fitness. In order to do so in an effective and timely manner, it would also be beneficial to avoid consideration of all possible transformations for a topology and only sample a subset of them.

The ACO algorithm was adapted in a novel manner to sample phylogenetic tree landscapes in order to understand more about this structure. The proposed implementation provides a probabilistic model for exploring this combinatorial space. This probabilistic model allows the algorithm to circumvent the complexity that arises due to increasing the number of biological sequences.

In this algorithm, labelled as the Phylogenetic Ant Colony Optimization (PLACO) algorithm, a single population of ant agents explore a space G by visiting its trees. Pheromone trail deposition is expressed as an addition to edge weights proportional

to the change in f between one tree and the next. Each ant in the system performs a randomized walk to adjacent vertices in the graph until they run out of a quantity I define as energy. An ant that runs out of energy returns to the colony by retracing its path and subsequently dies.

Definition Let the rearrangement operator defining edges in E be a function $\lambda : T \to T$ such that $\exists e_{ij} \iff \lambda : t_i \longmapsto t_j \mid t_i, t_j \in T$ and $t_j \in E(t_i)$ and $e_{ij} \iff e_{ji}$. From some given topology t_i, the full enumeration of all possible topologies using λ forms a set of neighbors to t_i in G represented as $\mathcal{N}(t_i) = \{t_j \mid \lambda : t_i \longmapsto t_j\}$.

In both directions that an ant travels, away from and back to the colony, pheromone trails are deposited as an increase to edge weights. This corresponds with topology fitness along the forward path, but along the path in the return trip this is a function of the fittest topology visited. This can be computed to be equal to an exponentiation of the change in $f(t)$ from an origin topology to the next. I present the weight on the edge between t_i and t_j as $w_{ij} = 2^{(f(t_j)-f(t_i))/10}$. Negative changes in the fitness function will result in a smaller concentration of pheromones to be deposited by ants. The division by ten serves to reduce the effect of large differences on exponentiation without dramatically changing the effect of significant differences in cost function on ant movement.

Avoiding the computation of all neighboring reconfigurations to a topology can allow this heuristic to effectively scale to large n, as the degree of a vertex $t_i \in T$ in such a case increases proportionally to the square of n. When feasible solution components are discovered along the vertex set of G, ants located at t_i will tend to consider a set of topologies that does not encompass all topologies in its neighborhood. To do this, I propose that traversal in the space is reduced to a binary decision. This is a probabilistic decision influenced by the weights of existing edges and an initial weight for all of the unvisited edges that can be possibly formed.

Definition Let the set of existing topologies in $\mathcal{N}(t_i)$ for which a rearrangement has already been found be $\mathcal{N}^+$ such that $\mathcal{N}^+(t_i) \subseteq \mathcal{N}(t_i)$. In the absence of a fully enumerated set of topologies for t_i, $|\mathcal{N}^+(t_i)| < |\mathcal{N}(t_i)|$. Similarly, let the set of unvisited topologies, for which λ is *not* known, be $\mathcal{N}^-(t_i)$ such that $|\mathcal{N}^-(t_i)| = |\mathcal{N}(t_i)| - |\mathcal{N}^+(t_i)|$.

The movement of ants can be defined as a series of stochastic decisions to perform one of the following two tasks, each with a respective probability.

1. Explore an existing path where λ is known.

2. Explore an unvisited path and compute λ.

If I define the probability to explore an existing path as p^+ and the probability to explore a new path as p^-, they can be calculated as follows.

$$p^- = \frac{\dot{w}|\mathcal{N}^-(t_i)|}{\dot{w}|\mathcal{N}(t_i)| + \sum_{t_j \in \mathcal{N}^+(t_i)} w_{ij}} \tag{4.1}$$

$$p^+ = 1 - p^- \tag{4.2}$$

In Equation 4.1, $\dot{w}$ is a tunable parameter that corresponds to the default and arbitrary pheromone parameter value for unexplored edges. This need not be applied to known edges as the sum is already known. The use of this equation will increase the probability to explore new edges in the absence of a well-marked pheromone trail. With increasing pheromone concentration along explored trails from a topology, the probability for an ant to explore an unvisited path decreases. The path that is chosen is biased by pheromone concentration:

$$p(t_j) = \frac{w_{ij}}{\sum_{t_k \in \mathcal{N}(t_i)^+} w_{ik}} \mid t_j \in \mathcal{N}(t_i)^+ \tag{4.3}$$

Given a tree topology t_i, $|\mathcal{N}(t_i)|$ is difficult to calculate if the number wanted is the number of *distinct* tree topologies that can be formed after perturbation. For example, the number of possible SPR moves is equal to $4(n-3)(n-2)$ [22]. However, the actual number of distinct tree topologies formed by SPR moves is some amount less than this value. Therefore, an approximation can be made for $|\mathcal{N}(t_i)|$ to calculate the probability found in Equation 4.1.

The pheromone concentrations will be computed by an ant as $w_{ij} = 2^{f(t_j)-f(t_i)}$. Use of this equation ensures that negative changes in the fitness function will result in a smaller concentration of pheromones to be deposited by ants.

Pheromone evaporation will be a decay by some constant value left as another tunable parameter. If constant evaporation proves to be insufficient, a suggestion

from [54] will be considered by applying a learning rate to the pheromone evaporation. Therefore, the magnitude of this evaporation will be proportional to both a portion of the fitness of an edge and how many times it has been accessed by an ant.

Definition Let each ant in the system presently searching G be subjected to a restricted amount of movement. This restriction is defined by τ, the amount of energy that an ant initially possesses. The value of τ for an ant will be proportional to the diameter of G which is subsequently proportional to n. When an ant moves, that movement is followed by an expenditure of energy Λ.

Movement expenditure of energy Λ is calculated as found in Equation 4.4 if an ant travels along some path. More of an expenditure of energy is made if the path being explored has not yet been explored before. Otherwise, a fractional amount is expended depending on the edge weight.

$$\Lambda_{ij} = \begin{cases} 1/w_{ij}, & \text{if existing path} \\ 1, & \text{if unvisited path} \end{cases} \tag{4.4}$$

Once energy is depleted, the ant retraces its path back to the colony. The weight of pheromones deposited on a return trip by an ant will, instead of being with respect to adjacent vertices along the edge, be with respect to the fittest tree found along the return trip.

This sequence of items results in an algorithm (Algorithm 1) that should be capable of scaling to large inputs and performing an adaptive search of the graph analogous to a breadth-first search in unexplored regions of the graph, and depth first search where there is a strong pheromone concentration.

Because this variant can be implemented merely as a series of graph visitors and manipulation of edge weights, the computational load this would place onto the formation and exploration of a graph comprising tree space would be fairly small. Ants do not necessarily have to be anything more than positions and lists of edges representing history. Therefore, memory requirements here are little more than what is required to store the graph itself.

A complexity analysis will be given with respect to component functions in the top-level algorithm present in Algorithm 1. The function f used for scoring a tree

Algorithm 1 Phylogenetic Ant Colony Exploration

Require: Phylogenetic Landscape $G = (T, E)$ for n taxa
Require: Starting Colony Location in T
Require: $e \leftarrow$ Evaporation Constant, $0 < e \leq 1$
Require: `max` $\leftarrow$ Maximum Number of Ant Agents
Require: `init` $\leftarrow$ Starting Pheromone Concentration, `init` > 0

```
ants ← Population of Ant Agents
while still exploring do
    createAnt(ants, n)                          ▷ Generate new ant.
    for ant in ants do                          ▷ Perform all ant movements.
        if ant.getEnergy() > 0 then
            moveAntForward(G,ant)
        else
            moveAntBackward(ant)
            if ant.isDead() then
                Remove ant from ants
            end if
        end if
    end for
    for edge in E do                            ▷ Evaporate pheromones.
        edge.reduceWeight(e)
    end for
end while
```

topology will be considered as a single time unit of invariable complexity. The top level algorithm has an indefinite number of loop iterations that must be tested, practically equal to an amount of exploration being allowed.

$|\mathcal{N}(t_i)|$ is the number of topologies that can be created after perturbation. For example, if λ is associated with the SPR operator, $|\mathcal{N}(t_i)| \approx 4(n-3)(n-2)$. The *moveAntForward* routine is capable of a best case single fitness assessment and move *or* of a worst case assessment of all previously explored edges which is $O(|\mathcal{N}(t_i)|)$. With SPR operators defining edges, therefore, the complexity of this function is $O(n^2)$. This would be the largest requirement for computation from this heuristic because of the need to choose from a set of at most $|\mathcal{N}(t_i)|$ edges each with respective probabilities that must be computed.

Retracing, on the other hand, has order $O(1)$ complexity, with respect to n, as the history of an ant is already known and there is only an access to the end of a collection data structure within the respective function (Algorithm 4). The number of times an ant expends energy affects the amount of retracing that an ant must do.

Practically, this algorithm should never fully enumerate all edges for every single tree. I argue that this algorithm can sample a large area of the search space for this combinatorial optimization problem with a reasonable number of iterations of the heuristic.

Component algorithms are outlined in Algorithms 2, 3, 4 and 5.

4.1.2 Evaluating Space Sampling

A number of tunable properties can be identified that serve as parameters to this heuristic.

The first of these parameters is the **starting state** of the heuristic and where the ant colony will be established. I hypothesize that this parameter will not affect the heuristic's ability to sample the tree space. Because of the probabilistic and stochastic nature of the algorithm, an ant will always have the choice of a collection of edges with associated non-zero pheromone concentrations. This does not differ for ant that is present at the ant colony. Therefore, sampling should still occur in a relatively consistent manner even if the starting colony is near the top, bottom, or in an arbitrary location in terms of fitness function value. What *would* change, of

Algorithm 2 *createAnt*: Creates a new ant agent.

Require: A collection of ants `ants`

Require: The number of taxa n

  if `ants.size()` $<$ `max` **then**

    `ant` $\leftarrow$ Create new ant agent at starting colony $\in C$

    $d \leftarrow n$

    `ant.setEnergy(`d`)`

    Add `ant` to `ants`

  end if

Algorithm 3 *moveAntForward*: Moves an ant along a trail.

Require: A graph $G = (T, E)$

Require: An ant agent `ant`

  $c \leftarrow$ `ant.getCurrentTopology()`

  $N \leftarrow$ Number of possible edges at c

  $u \leftarrow$ Number of unexplored edges at c

  $w \leftarrow$ Explored edge weights at c excluding any `ant` has already visited

  $p \leftarrow$ (`init` $\times u$)/(`init` $\times N +$ `sum(`w`)`)

  `explore` $\leftarrow$ **True** with probability p

  if `explore` **then** ▷ Explore an unvisited edge.

    `new` $\leftarrow$ Compute a new topology using λ on c

    Add `new` to C

    `edge` $\leftarrow$ Create new edge between c and `new` in E

    `ant.expendEnergy(1)`

    `edge.setWeight(`$a^{f(\texttt{new})-f(c)}$`)`

  else ▷ Explore an existing edge.

    `old` $\leftarrow$ *chooseEdge(*`ant`$, c$*)*

    `edge` $\leftarrow$ Edge between c and `old`

    `ant.expendEnergy(1/edge.getWeight())`

    `edge.addWeight(`$a^{f(\texttt{old})-f(c)}$`)`

  end if

  Add `edge` to `ant`'s memory of its path

Algorithm 4 *moveAntBackward*: Moves an ant back to the colony (retrace).

Require: An ant agent `ant`

`history` ← All edges `ant` has moved across in forward movement

`best` ← Best tree found along `ant`'s backwards movement

`edge` ← Last element of `history`

c ← Topology that `edge` leads to

Remove `edge` from `history`

Retrace movement along `edge` to c from `ant.getCurrentTopology()`

`edge.addWeight(`$a^{f(\texttt{best})-f(c)}$`)`

if `history` is empty **then**

`ant.die()`

end if

Algorithm 5 *chooseEdge*: Randomly choose an edge in a roulette fashion.

Require: An ant agent `ant`

Require: A topology $t \in T$

N ← Collection of topologies connected to t excluding any `ant` has already visited

w ← Collection of all pheromone concentrations for each topology in N

s ← `sum(`w`)`

for each topology i connected to t **do**

$p \leftarrow w[i]/s$

Associate p with edge connecting i and t

end for

e ← A topology in N selected randomly given associated probabilities

return e

course, is the quality of trees most often found and visited by an ant depending on the state of the colony's surrounding trees.

The **evaporation constant** e acts as the rate of decay of pheromone concentrations along edges. With $e > 0$, only paths that are regularly visited and have pheromone consistently deposited upon them will remain greater in pheromone concentration. This serves to reduce the probability ants go down paths of less activity. A greater value for e will mean the algorithm loses pheromone concentration information over a very small number of iterations. The extreme case is that all pheromone concentration is immediately lost ($e = 1$). Here, the meaning of pheromone concentration and would be similarly lost. Accordingly, a lower value for e moves the algorithm towards a state where concentrations never degrade unless over a large number of iterations. A value of $e = 0$ for the constant implies that all past trails possess an equal influence on ant state. An ideal value for this constant will result in edges retaining a sufficient amount of information of pheromone concentration while down-weighting those edges that generally are poor choices for movement.

The **maximum number of ant agents** is another parameter that affects the performance of the PLACO algorithm. Its increase allows a greater amount of exploration of the system for any given iteration. Furthermore, its increase provides opportunity for this algorithm to be performed in parallel with agent operations split amongst different computational devices or threads on a computer.

The final parameter is the **starting pheromone concentration**. If we consider Equation 4.1, the probability of an ant moving to a new edge in the algorithm depends on this parameter. General movement of ants, and therefore sampling of the space, is affected by this probability. Relative to the range of weights present on explored edges as values for pheromone concentration, the magnitude of this parameter will determine the probability of an ant to move towards a new edge instead of a visited one. If the value of $\dot{w}$ is larger than a typical value for pheromone concentration, unvisited edges will be visited more often than ones that have already been explored. In contrast, a smaller value would imply rarely exploring new edges.

In order to evaluate the effects of these parameters on the PLACO algorithm closely, various quantities of the produced graphs will be considered, each created utilizing different parameter values. These quantities include the range of fitness

function values and its diameter. The range of fitness values can be characterized by computing the mean and maximum fitness values found. Another evaluation metric can also involve considering the diversity of different splits among the produced trees, as a greater diversity of splits would imply a greater spread of trees found. Consideration of these properties should represent the breadth of sampling that was done across the space.

In order to test the effect of a number of free parameters in the proposed algorithm on its ability to sample the space, the proposed algorithm was run on existing sequence data. The algorithm was applied to both empirical biological sequence data ($n = 23$) as well as synthetic sequence data ($n = 9$), for which the full search space can be explored, in order to compare both the difference arising from different taxon set size and simulation. Stamatakis and Albright et al. both claim that simulated phylogenetic tree data tends to be less complex than real biological data. As a result, the landscapes that result from simulated data are not entirely representative in terms of the complexity that may result from actual data [1, 84]. This hypothesis will be investigated when considering the results of the experiments.

The empirical data used for this evaluation is described in Section 2.4.

We use e to denote the evaporation constant, m to denote the maximum number of possible ant agents, i to denote the exponent of the initial pheromone concentration of edges, and t_0 to denote the starting topology.

After running a series of experiments, 45 landscapes were created for both the empirical and synthetic data, each with unique sets of parameter values, testing the parameters e, m and i, to thoroughly span the domain of possible values for each parameter. These were all given a starting topology found by a heuristic known as FastTree [68]. For a selection of the landscapes with parameters that searched the widest portion of the space, replicate runs were performed with identical parameters and with a variation in starting location t_0. For all experiments, the number of iterations was kept fixed at 10000. This number of iterations will decide how well the algorithm converges depending on the size of the landscape and must be empirically determined.

Various quantities on resultant explored graphs were recorded. These quantities include the range of fitness function values, for which I use log-likelihood, range of

node degree, number of trees, and their graph diameters. Log-likelihood was calculated using the C phylogenetic likelihood library (libpll).

A final test that was carried out on the explored landscapes involved ranking all found trees by their log-likelihood. Then, I defined a confidence set of trees amongst the top 10% of these trees from results of the Approximately Unbiased (AU) test. The AU test is a procedure which provides a selection of trees which is most likely to to include the true tree amongst a selection of trees [78]. The AU test was applied to these selections of trees using the CONSEL application [79].

When investigating different sets of e, m and i parameters, it was found that the diameter for every landscape appears to remain constant regardless of how parameters were varied. The diameter of the phylogenetic space is $\theta(n)$, and the diameters found were slightly less than n. The diameters of the explored portions of the empirical data landscapes were between 15 and 16. For the synthetic data landscape, it was 5. I believe that these values are smaller than expectedbecause there are suboptimal regions in the landscape that are not reached by the PLACO algorithm.

What *does* appear to differ is the number and range of scores for trees visited for different parameters. The number of trees and bipartitions found differed mostly when e and m were varied. The number of bipartitions found appears to be proportional to the number of ants. A greater number of ants implies more work being done in every iteration of the algorithm. Furthermore, these ants also *interact* with each other through the deposited pheromone applied on edges.

The evaporation constant appears to significantly affect the quality of fitness of the topologies the ant agents visit. The best collection of trees is found around $e = 0.5$. Extreme e values leads to a drop in the ability of the algorithm to explore the most relevant regions of the space. This dramatically reduces the relative fitness of found trees. Pheromone concentrations across edges effectively encode a long-term memory of ants upon the surface. I believe that evaporation provides the algorithm the ability to forget poor regions and reinforce the exploration of higher likelihood regions.

I selected three triplets of parameters where search properties were satisfactory and kept them fixed to test for robustness of the search. Across replicate runs with these selections (Table 4.1), I found similar properties of broad exploration through the space. All of the replicate runs consistently generated landscapes with a large

number of trees. However, when I investigated results from the empirical data, a large deviation existed among the replicate runs for the number of bipartitions and the maximum degree found in the search space. The former deviation signifies variation in the algorithm's ability to find a great diversity of trees across the runs. The latter suggests inconsistent behavior when ants are causing edges to be created, possibly due to differently scored trees being visited. Despite this, the difference in log-likelihood is small and the nodes where the degree is largest are those that score higher. Therefore, while different breadths of tree diversity is being acquired, and while different trees are being visited between replicate runs, the ability of the algorithm to sample similarly scoring trees and regions does not appear to change. Notice, also, that this inference is less relevant if we discuss the synthetic dataset.

Being a smaller search space, the algorithm appears to sample the trees found in the smaller space thoroughly. Respectively, the algorithm explored a number of the possible trees in the empirical landscape on the order of $10^{-15}\%$ and of the synthetic dataset on the order of 1%. As the number of trees in the space is equal to $O(n!)$, this shows that although the algorithm is not searching a very large proportion of all possible trees, it is sampling a number of them that is sufficient to capture the shape of the landscape.

When choosing different starting topologies (Table 4.2), the deviations in number of bipartitions and average log-likelihood are magnified between both empirical and synthetic data. It appears that when a different, possibly worse starting topology, is chosen, more iterations need to be done in order to acquire more of a diversity in splits and to bring resultant landscapes consistently into regions of better scoring topologies. However, when a good path is found, the energy expenditure function should mitigate this effect.

When the AU test confidence sets of trees were computed for, the number of trees found to be present in these sets were similar for the sets of parameters tested and for both emperical and synthetic data. Even when starting topologies were varied, the number of trees found to be part of these sets did not seem to be reduced from those found by starting at a well-scoring topology. This suggests a tendency for the search to find well-scoring regions of trees.

Table 4.1: **Replicate Run Landscape Quantities**. The σ and $\bar{x}$ noted by each quantity indicates these are standard deviations and averages of each measure respectively. Let A represent the confidence set of trees computed by the AU Test. * Parameters are triplets where (a) $e = 0.25, i = -8, m = 5$, (b) $e = 0.50, i = 0, m = 10$, (c) $e = 0.75, i = 0, m = 10$, and (d) $e = 0.00, i = 8, m = 10$. † The number of bipartitions refers to the number of unique clades or topological splits in the trees.

Data Type	Set*	σ Num. Bipartitions†	σ Max Degree	σ Avg. Log-Likelihood	$\bar{x}$ Avg. Log-Likelihood	σ Max Log-Likelihood	$\bar{x}$ $\lvert A \rvert$
Empirical $n = 23$	(a)	4726	23.7	95.7	-12384.7	138	20
	(b)	10579	104	149	-12406.4	74.5	31
	(c)	13080	210	128	-12368.7	38.9	34
Synthetic $n = 9$	(d)	0.376	9.00	17.0	-11014.7	14.6	2
	(b)	0.855	8.90	13.3	-11016.0	9.68	3
	(c)	0.519	8.67	19.3	-11016.5	21.1	3

Table 4.2: **Varied Starting Topologies Run Landscape Quantities**. The σ and $\bar{x}$ noted by each quantity indicates these are standard deviations and averages of each measure respectively. Let A represent the confidence set of trees computed by the AU Test. * Parameters are triplets where (a) $e = 0.25, i = -8, m = 5$, (b) $e = 0.50, i = 0, m = 10$, (c) $e = 0.75, i = 0, m = 10$, and (d) $e = 0.00, i = 8, m = 10$. † The number of bipartitions refers to the number of unique clades or topological splits in the trees.

Data Type	Set*	σ Num. Bipartitions†	σ Max Degree	σ Avg. Log-Likelihood	$\bar{x}$ Avg. Log-Likelihood	σ Max Log-Likelihood	$\bar{x}$ $\lvert A \rvert$
Empirical $n = 23$	(a)	10883	63.2	616	-12955.5	739	16
	(b)	25297	182	257	-12512.3	182	37
	(c)	16210	171	602	-12743.7	538	27
Synthetic $n = 9$	(d)	0.870	13.3	14.4	-11005.9	22.2	2
	(b)	2.140	17.3	18.2	-11014.3	14.7	3
	(c)	1.630	15.9	21.4	-11010.8	36.6	3

It was discussed in [29] that as problems become more complex, the free parameters of an ACO algorithm become increasingly important to obtain convergence to optimal solutions. The deposition of pheromones was claimed to not be enough for convergence but worked in tandem with the effects of these free parameters. Furthermore, pheromone updates said to be based on solution quality were found to be paramount for faster convergence, in addition to larger population of ants and the effect of evaporation.

This appears to agree with the results we have found using this evaluation strategy. This metaheuristic was designed to sample a large number of regions of interest of the search space with a reasonable number of iterations and amount of time. We found that evaporation was effective in steering the search to well-scoring regions of the space, the number of ant agents extended the number of trees found, and that the highest scoring trees in the search were visited more often as indicated by their increased degree.

All results show that, when exploring both empirical and synthetic data, we can make three claims about the performance of the proposed algorithm. Firstly, the PLACO algorithm is capable of broadly exploring the combinatorial space in spite of the number of taxa. Secondly, across replicate runs we find consistent behaviour but variation in quality of trees. This implies a sparse but broad search where different topologies are being found. Thirdly, it does not matter where the algorithm starts in order to acquire a wide ranging set of trees and to sample properties of and the shape of the space.

4.2 Interactive Visualization of the Search Space

Scientists and others working with large amounts of data are always searching for new ways to express data that they collect. Scientific visualization is meant to emphasize features of greatest interest. Phylogenetic landscapes comprise a large amount of information and it is in this section I discuss the application of visualization methods to this space to start and map out an image for the landscape that can be interactively manipulated and explored.

A prototype system is developed that provides a projection of the search space onto a two-dimensional surface using existing tools for visualizing graphs in a web browser.

Interactivity is provided by the facility of clicking elements of this graph to explore local neighborhoods and to display information about trees and their topologies.

4.2.1 Interactive Features

A visualization of the tree search space can be carried out using browser-based software. It has been designed to support the following features with prospective biologist users in mind.

1. *Exploring* any given topology of any given tree in the space via any sort of explorative method,
2. *Searching* the space for a given topology,
3. *Identifying* the connectivity of a tree in the space with relation to all other trees,
4. *Locating* notable features of the search space, including but not limited to local and global optima and maximal steepest climbs [22] for a given scoring function,
5. *Annotating* regions of the space to add context and new features, and later retrieve or manipulate these annotations, and
6. *Comparing* any two different landscapes or spaces that possess different contexts or investigate different scoring functions, but share underlying data.

The emphasis here is on being able to adjust focus in the visualization, thus a stringent requirement for flexible interactivity. Focus should be able to be put onto single nodes in the graph, each representing a unique phylogenetic tree, and then later changed in order to derive how that tree fits in a greater, more encompassing macroscopic view of the data.

Furthermore, using existing scoring functions, rather than devising new ones, places a lesser burden on a user to adjust to a new way of thinking about how the data is pieced together.

4.2.2 Constructing Landscapes for Evaluation

I will explore the effectiveness of our proposed method by considering the effectiveness of the following.

1. The construction of a landscape near the true global maximum of the space should construct an appropriate graph G of vertices and edges which can be quantitatively identified.

2. The interactive visualization of a constructed phylogenetic landscape G should allow the user to make sense of regions of the landscape and to make hypotheses on both the space nearest the global maximum and the behaviours of the heuristics involved.

This evaluation will be two-pronged. One component of evaluation will be an analytical identification of the search space that all of the relevant tree topologies comprise in order to acquire relevant information that characterizes that search space. The other component will be a visualization of the phylogenetic tree space or landscape of appropriate data. These will be discussed separately.

The graph G can be constructed to model a phylogenetic landscape. For the alignments which I will evaluate, constructed graphs would constitute a restricted subset of all possible trees in tree space. Given each respective sequence alignment, I followed through with an iterative process depicted in Figure 4.1.

FastTree was used in order to acquire a tree that should be somewhere near the top of the entire search space, in terms of appropriate f. This tree is used as the first vertex of respective graphs. Graph G is kept in a database while a heuristic H is employed to populate the landscape with trees. Between both the 16S rRNA and full protein sequence alignments, a FastTree tree was obtained using the 16S sequence alignment.

Two heuristics were used for the constructions. The first was a novel greedy hill-climbing heuristic, and the other is the heuristic found in the RAxML software [84]. The objective functions used were both parsimony and log-likelihood. Log-likelihood was scored using a WAG empirical model [89].

This process generated a restricted subset of the exhaustive phylogenetic landscape that could then be analysed for characteristic features. Improvements were made to the graph configurations by interactively investigating the unexplored neighbours of component trees. This was done on a case-by-case basis for all four landscapes until refined landscapes were formed that could no longer be improved by directly investigating the greediest moves.

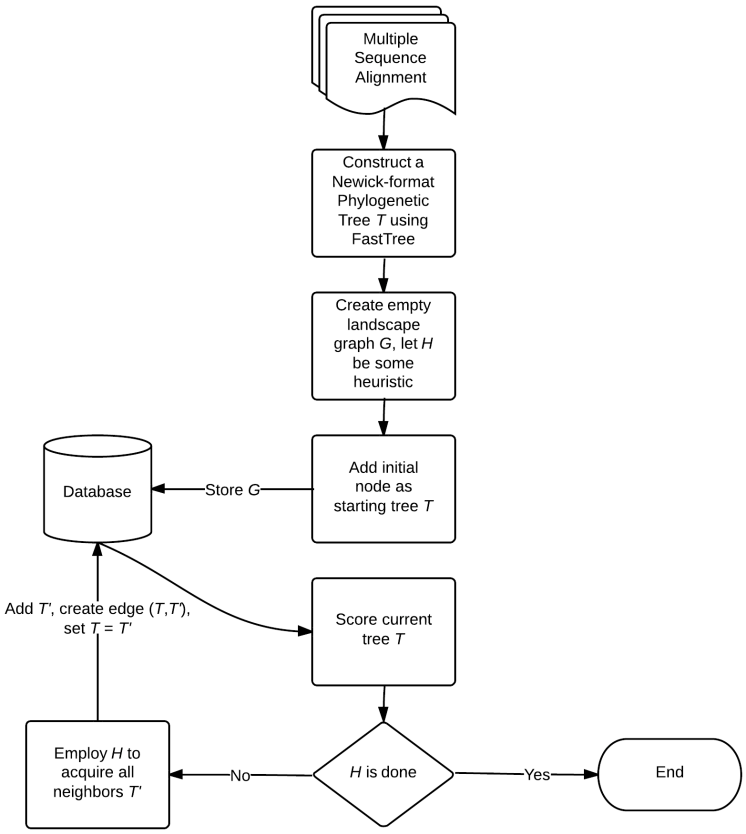


Figure 4.1: **Landscape Construction Workflow**. A flowchart presenting the process involved in the construction of phylogenetic landscape G.

Greedy Heuristic. The naive greedy heuristic that was used is dubbed here as a *smooth greedy search* by parsimony with refinement by likelihood. The heuristic performs a greedy climb of maximum parsimony via SPR rearrangements. Once a tree with maximum parsimony is found, all neighbours of this tree are acquired and their likelihoods are calculated. If no likelihoods exist that improve upon the maximum parsimony tree, the heuristic stops. Otherwise, search continues for a maximum parsimony tree again from the best scoring neighbour. This method is used to provide a faster method for building a landscape that a user can immediately explore since scoring parsimony for trees is considerably faster than scoring likelihood.

RAxML Heuristic. RAxML was run for each of the landscapes such that it output intermediate trees. The heuristic was performed such that exploration again started from a FastTree tree. Trees were added to the landscape if they were not

already present; otherwise, they were marked in the existing landscape as having been found using RAxML heuristic.

4.2.3 Visualization of Constructed Landscapes

An interactive visualization was built for each respective alignment and landscape to identify key qualitative properties. Visual channels, such as size and position, were utilized in this configuration such that the following was done.

1. *Nodes* in G that maximized improvement of f, for all surrounding nodes, were represented as circles.

2. *Edges* between these respective nodes in G were represented as undirected lines between the nodes.

3. The height of all nodes was scaled between the maximum and minimum of f; for this experiment, log-likelihood was used.

4. The horizontal position of a node was computed *via* a force-directed layout algorithm to remove visual cluttering.

5. The size of a node was drastically larger if it was an *optimum* or tree of interest (such as the starting tree of the landscape).

A resultant *improvement space* was formed that displayed the greediest view of the landscape. In order to better explain the notion of improvement space, it is useful to introduce the following algorithm that was performed on all respective nodes in G before the visualization was mapped to a landscape.

All those nodes that are the maximum improvement of another node were visualized, thus reducing the size of the landscape visualization without losing pertinent information.

This visualization was done in a browser and built from scratch using relevant libraries. It possessed the following technical components:

- A **database and scoring server** coded in Python that stored the phylogenetic landscape for an alignment and performed computationally expensive

Algorithm 6 Determine maximum improvement status of node N in G.

```
N_m = maximum improvement of N = null
acquire all neighbours of node N
N_f = cost function value of N
for neighbour K of N do
    K_f = cost function value of K
    if N_f < K_f and N_m is null or N_m < K_f then
        N_m = K_f
    end if
end for
```

operations such as scoring. Most expensive computation was done through the use of C bindings to libpll, the library underlying RAxML.

- A **client-side browser-based canvas** coded in JavaScript that was used in order to display the visualization, allowing for the viewing of individual trees, and enabling interactivity.

Interactivity involved the extension of the landscape to unexplored areas through the clicking of nodes, and for the viewing of individual trees or consensus trees and attributes.

Appropriate functions and methodology were applied to these graphs in order to acquire the following empirical observations for each graph G:

1. the number of components of G,
2. all local optima present in G,
3. the basins of attraction present for all optima,
4. the maximum steepest climb length (MSCL) for G, and
5. the global maximum of the restricted space described in G.

Applying a *breadth-first search* from the starting tree of the landscape, these properties were identified and compared between the different landscapes. The presence of LGT events may or may not present significant differences of features here.

In order to ensure that the visualization was providing accurate information, quantities that could be acquired from the graphs were compared to the visualizations to test for consistency.

I use the dataset described in Section 2.4 again here. Constructions of the four landscapes were done and analysed in order to acquire the following information found in Table 4.3. All respective landscape graphs contained only a single component.

Enzyme	Specimen	# Nodes	# Optima	MSCL	$T_{\max}$ Log-Likelihood	$T_{\max}$ Parsimony
BCoAT	*fp*	11155	1456	2	-7843	1889
BCoAT	16S	27550	4699	3	-10246	3702
kinase	*fp*	27128	147	2	-3943	1005
kinase	16S	21961	220	3	-3506	1851

Table 4.3: **Important Quantities From Constructed Landscapes**. For both enzymes butyryl-CoA:acetate CoA-transferase (BCoAT) and butyrate kinase (kinase), a **full protein** analysis was done of the enzyme (*fp*) along with the respective **16S rRNA** analysis, for all corresponding species which a protein sequence was found. Parsimony and log-likelihood of global maximum tree on basis of likelihood only ($T_{\max}$) is reported for each landscape. Included is the maximum steepest climb lengths (MSCLs) for each instance.

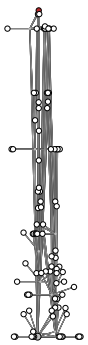


Figure 4.2: **BCoAT Full Protein Improvement Space Landscape**. The visualized improvement landscape acquired for the butyryl-CoA:acetate CoA-transferase (BCoAT) full protein sequence alignment where largest improvement trees in the landscape are circles and rearrangements of the trees to the next improvement are lines. Height is scaled by log-likelihood. Larger sized nodes are optima.

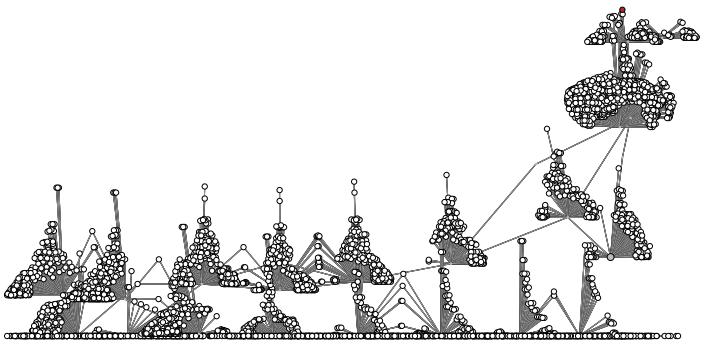


Figure 4.3: **BCoAT 16S rRNA Improvement Space Landscape**. The visualized improvement landscape acquired for the butyryl-CoA:acetate CoA-transferase (BCoAT) 16S rRNA sequence alignment where largest improvement trees in the landscape are circles and rearrangements of the trees to the next improvement are lines. Height is scaled by log-likelihood. Larger sized nodes are optima.

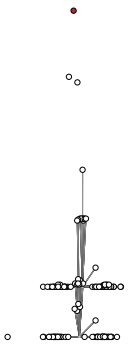


Figure 4.4: **Butyrate Kinase Full Protein Improvement Space Landscape**. The visualized improvement landscape acquired for the butyrate kinase full protein sequence alignment where largest improvement trees in the landscape are circles and rearrangements of the trees to the next improvement are lines. Height is scaled by log-likelihood. Larger sized nodes are optima.

Sequence Alphabet Can Affect Landscape Topology. Both viewing all four landscapes (Figures 4.2, 4.3, 4.4, and 4.5 respectively) and referring to the results found in Table 4.3, I am driven to believe that the heuristic-restricted phylogenetic landscapes constructed for the 16S rRNA sequence alignments (for both BCoAT and butyrate kinase) appear to be far more difficult to climb and more riddled with sub-optimal regions of local optima. More nodes comprise such graphs *vs.* their full protein sequence alignment counterparts which suggested the heuristic escaped from

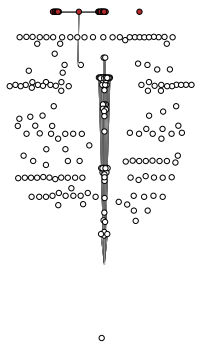


Figure 4.5: **Butyrate Kinase 16S rRNA Improvement Space Landscape**. The visualized improvement landscape acquired for the butyrate kinase 16S rRNA sequence alignment where largest improvement trees in the landscape are circles and rearrangements of the trees to the next improvement are lines. Height is scaled by log-likelihood. Larger sized nodes are optima.

local optima more often and that the heuristic encountered more regions of improvement in fitness when exploring likelihood space around local maximum parsimony trees. Respective maximum steepest climb lengths are also larger for these landscapes. Corresponding log-likelihoods and parsimonies cannot be compared because of differences in underlying models and sites.

The current hypothesis for this behaviour is that sequence alphabet can dramatically affect landscape topology. On the basis that 16S rRNA sequences are in a *nucleotide alphabet*, rather than an amino acid one (which the full protein sequence alignments *are* in), it is suggested that far more signal is present in the nucleotide sequence alignments. This results in the interaction observed with the (GTR) substitution model. Furthermore, there may be noise present in base pairing due to structural properties.

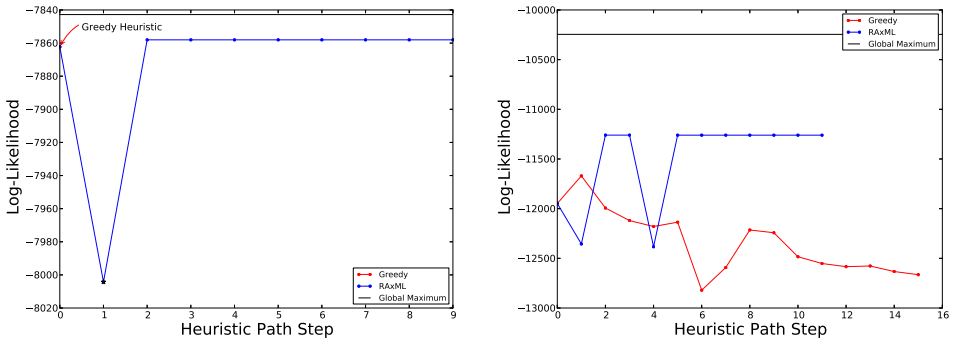
As such, a significant bias is most likely present in the 16S rRNA landscapes due to this variation between corresponding full protein sequence alignment landscapes. Therefore, difficulty will arise when making any claims as to significant differences arising from LGT rather than from this postulated phenomenon. This hypothesis could be tested by reintroducing the same methods with like-alphabet landscapes and making new observations, possibly by acquiring the nucleotide sequences for the full proteins.

Landscape Differences Between Enzymes Arise In Improvement Space. An interesting observation that has been made but does not have any obvious explanation is that the two different enzymes behaved differently in terms of the heuristic and how they are visualized in the improvement space. The BCoAT landscapes only possessed one component in both the actual landscape and in the improvement space constructed when considering only nodes of improvement. However, when considering the butyrate kinase landscapes, both the full protein sequence landscape and the 16S rRNA sequence landscapes (Figures 4.4 and 4.5) possessed multiple components in only the improvement space. This gives the impression that during the progression of the heuristic, moves were performed to maximize parsimony, but these moves were ultimately suboptimal in the consideration of maximum likelihood. Therefore, when visualized in an improvement space on the basis of log-likelihood, paths are no longer shown because those paths are paths that would lead to a suboptimal tree before moving to a better one.

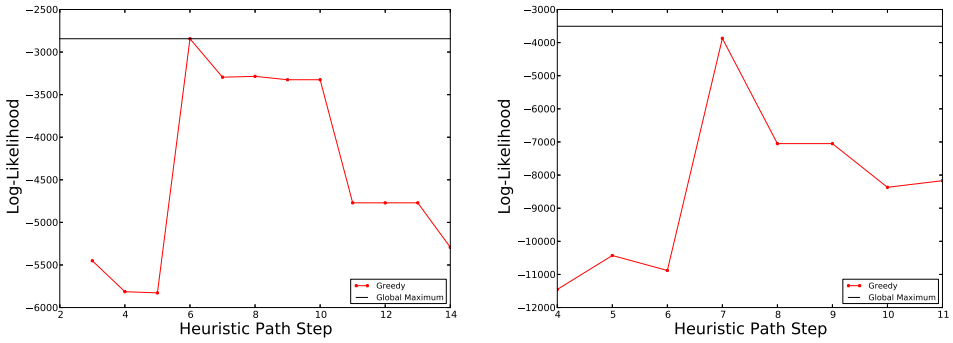
The only explanation that can be provided for this observation is that the kinase specimen possessed properties that drove the parsimony to believe it was improving but were not necessarily moves of better likelihood. This is possible evidence of effects of the long branch attraction problem [8] or a mismodelling of the substitution process.

Maximum Parsimony Convergence Did Not Guarantee Global Maximum. When observing the performance of the heuristics that were carried out (Figure 4.6), RAxML was not capable of outputting any intermediate trees when run on the butyrate kinase alignments.

With the progression of the heuristics alone, it is interesting to observe that the greedy heuristic came far closer to a global maximum in the butyrate kinase dataset than it did in the BCoAT dataset. The global maximum here refers to one on the basis of likelihood and only in a far localized region of the search space surrounding the restricted landscape formulated in this experiment. This may be a result of RAxML not being present to drive the landscape into another region of the space, however. Furthermore, the butyrate kinase initial trees found in the first steps of the heuristic were not capable of being scored by the libpll implementation for log-likelihood scoring; since libpll underlies RAxML, this suggests a possible reason why



(a) BCoAT



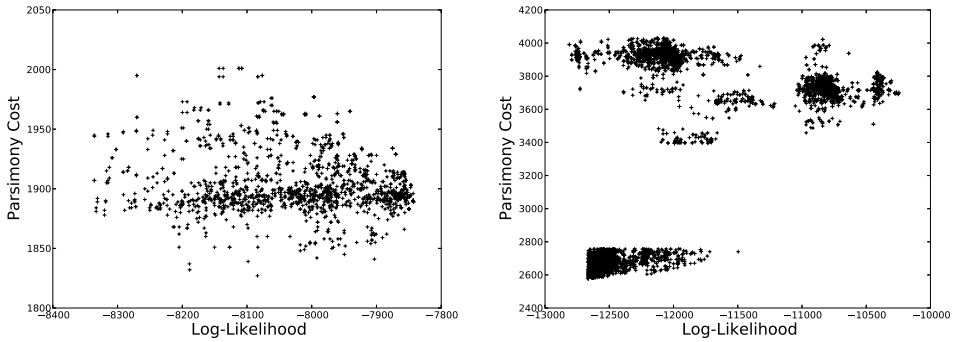
(b) butyrate kinase

Figure 4.6: **Performance of Heuristics**. Progression of the smooth greedy and RAxML heuristics when performed on the respective full protein and 16S sequence alignments. Full protein sequence alignment progressions are shown on the left and the 16S rRNA sequence alignments are shown on the right.

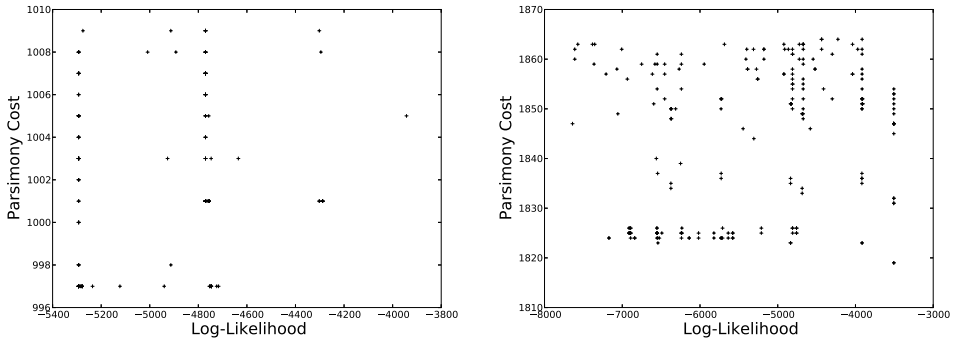
the RAxML heuristic failed.

It is also important to note that *manual intervention* was required in order to refine the four landscapes and to reach a better global maximum in all four cases. This was done with the utilization of the interactive visualization software and choosing to explore all possible greedy moves in the improvement space in order to ensure no new neighbours can be explored by SPR rearrangements. The global maxima found by this intervention is shown by a solid line in the charts present in Figure 4.6.

These results suggest that parsimony may not be as appropriate a measure of agreement with data was suggested by some sources. This could be due to the nature of the landscape at this region near the absolute global maximum of the exhaustive space, or it could be due to properties found within the alignments themselves including heterogeneity. Further evidence is given in Figure 4.7, the charting of both



(a) BCoAT



(b) butyrate kinase

Figure 4.7: **Relationship of Parsimony and Likelihood**. Relationship of parsimony cost and log-likelihood for the respective full protein and 16S sequence alignments. Full protein sequence alignment progressions are shown on the left and the 16S rRNA sequence alignments are shown on the right.

properties showing no recognizable correlation, especially for butyrate kinase, but showing interesting clustering for BCoAT.

Chapter 5

Hypothesis Testing in Search Spaces

A number of hypotheses were posed in this thesis, either explicitly or implicitly. Because there was a lack of tools that allow analysis and manipulation of the fitness landscape associated with phylogenetics, a framework was needed in order to allow testing of these hypotheses.

Chapter 2 introduced some of the challenges faced when performing heuristic searches of the phylogenetic fitness landscape as well introduced strategies for restricting that search. Chapter 3 introduced techniques to sample the space rather than merely find an optimum. This chapter discusses the software framework I developed to facilitate this heuristic search and exploration of the phylogenetic search space by leveraging upon a number of existing tools and creating new ones.

5.1 Implementation Details of the Software Framework

Introduced here is a cross-platform library called Pylogeny intended for heuristic search and analysis of the phylogenetic tree search space, as well as the design of novel algorithms to search this space. This framework is written in the Python programming language, yet it uses efficient auxiliary libraries to perform computationally expensive steps such as scoring. As a programming interface, Pylogeny addresses the needs of both researchers and programmers who are exploring the combinatorial problem associated with phylogenetic trees.

The source code and library requires only a small number of dependencies. Python dependencies include NumPy [87], a ubiquitous numerical library, NetworkX [46], a graph and network library, Pandas [56], a high-performance library for numerical data, and P4 [42], a phylogenetic library. An additional dependency that is required is a C phylogenetic library called libpll that underlies the RAxML application and

This chapter serves as a majority of the content of a software paper currently in submission for the open access peer-reviewed *PeerJ Computer Science* journal under the title "Pylogeny: an open-source Python framework for phylogenetic tree reconstruction and search space heuristics".

is used to score likelihood of trees [84, 41]. Optionally, the BEAGLE [5] package could be used for scoring as well. Most dependencies are automatically resolved by a single command by installing the package from the PyPi Package Index. Primary documentation is available on the library's website and alongside the library. All major classes and methods also possess documentation that can be accessed using a native command.

5.1.1 Summary of Features

The functionality to maintain a phylogenetic landscape is implemented in the `landscape` class defined in the `landscape` module of this library. This object interacts with a large number of other classes and supports tree scoring using standard phylogenetic methods. Many of the more relevant objects are tabulated and explained in Table 5.1.

The Pylogeny library can read sequence alignments and character data in formats including FASTA, NEXUS, and PHYLIP. Tree data can only currently be read in a single format with future implementations to allow for a greater breadth of formats. Persistence and management of character data is performed by an alignment module, while trees are stored by their representative string in a tree module. They can be instantiated into a richer topology object in order to manipulate and rearrange them.

5.1.2 Phylogenetic Tree Manipulation and Scoring

For the purposes of this framework, if instantiated into a `topology` object, phylogenetic trees are modelled in memory as being rooted. Therefore, manipulation and access of the tree components, such as nodes and edges, presupposes a rooted structure. Unrooted trees, either multifurcating or bifurcating, can nevertheless still be output and read. Support is also present for splits or bipartitions (as in the `bipartition` object) of these trees, required by many phylogenetic applications such as consensus tree generation [55].

Iterators can be created for visiting different elements in a tree. Unrooting, rerooting, and other simple manipulation can also be performed on a tree. For more complex manipulation, rearrangement operators (using the `rearrangement` module) can be performed on a tree to convert it to another topology. To save memory and

Table 5.1: **Pylogeny Framework Functionality**. Overview of the basic objects in the Pylogeny library.

Class Name	Module Name	Description
`alignment`	`alignment`	Represents a biological sequence alignment of character data.
`treeBranch`	`base`	Represents a branch in a tree structure.
`treeNode`	`base`	Represents a node in a tree structure, such as a phylogenetic tree, and its associated information.
`treeStructure`	`base`	A collection of `treeNode` and `treeBranch` objects to comprise a tree structure.
`executable`	`executable`	An interface for the running of some binary application (in a Unix shell).
`heuristic`	`heuristic`	An interface for a heuristic that explores a state graph.
`graph`	`landscape`	Represents a state graph.
`landscape`	`landscape`	Represents a phylogenetic tree search space, modelled as a `graph`.
`vertex`	`landscape`	Represents a single node in the phylogenetic `landscape`, associated with a tree, and adds convenient functionality to alias parent landscape functionality.
`landscapeWriter`	`landscapeWriter`	Allows one to write a `landscape` to a file.
`landscapeParser`	`landscapeWriter`	Allows one to parse a `landscape` that was written to a file.
`newickParser`	`newick`	Allows one to parse a Newick format string of characters representing a (phylogenetic) tree.
`rearrangement`	`rearrangement`	Represents a movement of a branch or node on one tree to another part of that same tree.
`topology`	`rearrangement`	An immutable representation of a phylogenetic tree on which movements can be performed.
`bipartition`	`tree`	Represents a bipartition of a phylogenetic tree.
`tree`	`tree`	Represents a phylogenetic tree; does not contain structural information and defines features such as its Newick string, fitness score, and origin.
`treeSet`	`tree`	Represents an ordered, disorganized collection of trees that do not necessarily comprise a combinatorial space.

computation, rearrangements are not performed unless the resultant structure is requested, storing movement information in a transient intermediate structure. This avoids large-scale instantiations of new topology objects when exploring the search space.

Scoring topologies using parsimony or likelihood is done by calling functions present in the library that wrap libpll or the high-performance BEAGLE library. These software packages are written in C or C++, the latter of which allows for increased performance by using the Graphics Processing Unit (GPU) found in a computer for processing.

5.1.3 Tree Search Space Graph Construction and Search

The tree search space is abstracted as a graph where a number of graph algorithms and analyses can be performed on it. I do this by utilizing routines found in the NetworkX library which has an efficient implementation of the graph in C. Accessing elements of the graph can be done by iteration or by node name, and properties of the space can be identified by function.

Exploring the space is done by performing rearrangements on trees as `topology` objects where different methods of exploration include a range of enumeration and stochastic-based sampling approaches. In order to make Newick strings consistent amongst trees in a phylogenetic tree search space, an arbitrary but efficient rooting strategy is used to avoid redundancy. Rearranging trees in the search space reroots resultant trees to the lexicographically lowest-order taxon name. This means that different rearrangements that lead to the same topology, with a possibly different ordering of leaves, can still be recognized as not being a new addition to the space. Restriction on this exploration is supported by allowing limitations on movement by disallowing breaking certain bipartitions.

A minimal example to demonstrate constructing a landscape from an alignment file, and finding trees in the space, is found below. The landscape is initialized with a single tree corresponding to an approximate maximum likelihood tree as determined using the FastTree executable [68].

```
from pylogeny.alignment import *
from pylogeny.landscape import landscape

# Open an alignment compatible with the strict
# PHYLIP format.
ali = phylipFriendlyAlignment('a.fasta')
startTree = ali.getApproxMLTree()

# Create the landscape with a root tree.
lscape = landscape(ali,starting_tree=startTree,
                   root=True)

# Explore around that tree.
lscape.exploreTree(lscape.getRoot())
```

5.1.4 Existing Phylogenetic and Heuristic Programs

The library supports executing other software on its objects. Implementations are present in the framework to call on the FastTree [68] and RAxML heuristics for finding an approximate maximum likelihood tree. There is also an implementation for the use of TreePuzzle [76] and CONSEL [79] in order to acquire confidence intervals of trees as defined by the Approximately Unbiased (AU) test [78]. Further implementations can be created by extending a base interface found in the library.

An example of code to demonstrate the use of CONSEL, to generate a confidence interval of trees with default settings, is as follows:

```
from pylogeny.alignment import alignment
from pylogeny.executable import consel

# ali   = Open an alignment file.
# trees = A set of trees (e.g., a landscape).
# ...

AUTest   = consel(trees,ali,'AUTestName')
interval = AUTest.getInterval()
```

5.2 Design of Search Space Heuristics

Interfacing to the rest of the framework in order to design a heuristic that can search a landscape has been designed to be merely an act of implementing a single interface class (Table 5.1). A number of heuristics have already been designed and implemented in this manner for the framework. These include a number of greedy hill-climbing heuristics, some with minor variations, using either parsimony or likelihood scoring techniques. Also included is the PLACO algorithm described in Section 4.1.1.

5.2.1 Applying Search Space Heuristics

Performing a heuristic search of the combinatorial space comprised by a phylogenetic landscape can be done with relative ease using this library. Not only can the heuristic's steps be later analysed, the resulting space that is explored can also be later viewed and investigated for its properties. The `heuristic` module has a number of already defined approximate methods to discover the global maximum of the space, and with understanding of the object hierarchy, one can create their own.

As an example, one could perform a greedy hill-climbing heuristic on the search space by comparing the trees' parsimony scores. To do this, they would instantiate a `parsimonyGreedy` object from the `heuristic` module and provide an existing landscape and tree in that landscape to start the climb at. The minimal code to achieve a search from the first tree of a landscape would be:

```
from pylogeny.alignment import alignment
from pylogeny.landscape import landscape
from pylogeny.heuristic import parsimonyGreedy

# ali    = Open an alignment file.
# lscape = Construct a landscape.
# ...

h = parsimonyGreedy(lscape,lscape.getNode(0))
h.explore()
```

5.2.2 Acquiring Usable Quantities From Landscapes

In order to get a variety of the quantity measures used to evaluate different hypotheses in this thesis, a secondary package was designed that is not currently packaged in the Pylogeny framework. This *analysis* package provides a number of hooks into the core framework to allow for computing quantities from already constructed or explored landscapes.

The package has three components.

1. A primary analysis middleware module defining most computational methods,
2. A module that focuses on analysing paths of best improvement from a tree, and
3. A module that focuses on finding SPR or RF-distances between trees.

The primary module is intended to find the following quantities from a search space alongside a number of plotting routines for drawing figures.

- The number of edges in the state graph,
- The number of nodes in the state graph,
- The only known global optimum in the landscape,
- All local optima in the landscape,
- Degree counts for all edges,
- All basins of attraction,
- The maximum steepest climb length (MSCL),
- A calculation of how much of the full search space has been explored,
- An approximation of the space diameter,
- Extractions of subgraphs using certain restrictions and parameters, and
- Unique splits across all trees.

These are computed using standard library functions in addition to some statistical methods from the NumPy library.

Chapter 6

Conclusion

Finding a solution for an NP-hard problem typically involves using techniques which are capable of finding a reasonable solution in practical time. One such problem is phylogenetic tree reconstruction.

In tree reconstruction, we have a combinatorial optimization problem comprising a discrete search space with an unknown underlying structure. There is a need for further understanding of what heuristics are capable of exploiting, exploring, and sampling this space. Acquiring this information will lead to an understanding of how heuristics can function in these spaces and what structure these spaces can have.

6.1 Contributions

This thesis sought to extend our understanding on combinatoric search spaces in phylogeny. This was in hope to improve techniques for tree reconstruction and evolutionary inference. By finding more about the structure of the combinatorial problem and establishing a framework to manipulate them, directions for investigation were identified, and hopefully room is put aside for future researchers for further work.

This thesis first introduced some of the challenges faced when performing heuristic searches of the phylogenetic fitness landscape. It also introduced strategies for restricting that search. Later discussions brought into light techniques to sample the space rather than merely find an optimum. Finally, implementation details and discussion was put forward regarding a software framework that was developed during the course of this thesis' research to facilitate this heuristic search and exploration of the phylogenetic search space. This framework was developed by leveraging upon a number of existing tools and creating new ones.

Through these avenues of discussion, we acquired a better understanding of interactions of properties of the data with resultant tree space configurations. Ultimately,

we also came to devise new techniques to accomplish this task including a metaheuristic based upon swarm intelligence.

6.2 Future Work

The research I performed brought forward many interesting questions. This section will provide a number of avenues for future research.

6.2.1 Challenges of Representation

To find an optimal solution to the problem of acquiring trees from the space efficiently and with a low memory footprint, more work can still be done. When discussing the problem (Section 3.1.2), I mentioned a number of other relevant data structures. One of them was the wavelet trie which is a succinct data structure performing the same operations as the PATRICIA tree with a much smaller amount of memory. Another structure is the adaptive radix tree that was explored in preliminary research, meant to be more conscious of main memory indexing [51]. I believe these both may be more ideal than the PATRICIA tree but will require further work to implement. Because the space of trees does not get very large in heuristic searches, this may be an avenue that only needs to be explored if the number of trees sampled from the space begins to become unwieldy.

I had also performed no further work on implementing the generalizing improvement to the trie structure. The implementation created in Python for the PATRICIA tree is capable of supporting any sequence that is indexable, can be queried for length, and can be queried for subsequences. Because this provided satisfactory performance, I dismissed the construction of a working system using the PATRICIA tree with component objects of a tree, rather than a Newick string. As for its correctness and an analysis of its time and space complexity, this would still be identical to the traditional PATRICIA tree as no other changes are made to its functionality. The only difference that may arise are in respective complexities of accessing and related operations on the sequence in question. To be more precise, if a tree structure is stored in the PATRICIA tree, we would have a multi-level tree structure where node traversal is better controlled and other information regarding these nodes in the phylogenetic

trees of the space can be accessed in real-time as traversal occurs through relevant trees in the space to a query.

6.2.2 Restricting the Space

I suggested that *a priori* knowledge on OTU relationships could provide a strategy to place restrictions on the space via branch-locking. A method to acquire this knowledge has been proposed and explored, at a shallow level, for an ongoing class project. It involves text mining and information extraction from biological literature to retrieve coarse phylogenetic relationships of significance. This could be extended for future research.

6.2.3 Metaheuristic Sampling

Two alternative means to perform sampling of the space using metaheuristics are proposed here, one which involves the use of memetic algorithms (MAs), and another that introduces branch-locking to the proposed PLACO algorithm.

MAs are subsets of evolutionary algorithms. Like other evolutionary algorithms, memetic algorithms have the property of being easily adapted to new problem domains. MAs try to mimic cultural evolution as put forward by the idea of memes in Richard Dawkins' *The Selfish Gene* [24]. Cultural evolution is said to be fraught with *good improvements* that are created because of problem specific knowledge. This so-dubbed *fast-feedback* flow of information from phenotype knowledge to genotype level differs from what is regarded as the processes of biological evolution [63].

Memetic algorithms form a hybrid algorithm combining a population-based global search, such as those found in genetic algoriths, and heuristic local searches. MAs are even sometimes referred to as hybrid evolutionary algorithms. Therefore, an MA that would be suitable for the phylogenetic tree space would involve a number of populations K that can each involve a population-based metaheuristic. Similarly, these K populations could involve a number of individuals performing a local search heuristic intended for a single individual, such as hill-climbing.

With regards to the PLACO algorithm, the concept of branch or bipartition locking could be introduced in order to establish *a priori* problem domain knowledge for

the proposed algorithm. Further variations to this application may involve performing bipartition locking based on acquired probabilities. Daemon actions could also be associated with the metaphorical increasing or decreasing of temperature of the system in order to have faster or slower decay of pheromone parameters based on proximity to an optimum.

Future work investigate the maintenance of multiple populations in the space. For example, we could build into the algorithm an ability for it to iteratively create colonies. This can accomplish to more densely move across the space and focus on regions of particular interest.

6.2.4 Landscape Construction and Visualization

For the evaluation performed of the construction and visualization of landscapes done in Section 4.2.3, further comparisons can be done of the data set landscapes. A change in data that could have been helpful would have been to start with the trees proposed by Meehan in [57] rather than a FastTree generated tree.

If time had permitted, another means of visualization to explore would be to decompose the graphs of the landscapes formed into an amplitude spectrum and into barrier trees. An investigation of the landscape as a barrier tree by Bastert et al. has shown that amplitude spectra are capable of characterizing a landscape and its respective basins and optima [7]. An additional step proceeding landscape generation would have involved transforming this graph into an algebraic framework. A point set is said to be able to be generated comprising rearrangement moves to new configurations *via* NNI or SPR with the use of maximum parsimony and maximum likelihood as f. The resulting fitness landscape could then lead to an amplitude spectrum and barrier tree that can characterize the landscape in a concise and visually meaningful manner. By comparing respective spectra, a more certain conclusion could have been drawn regarding the hypothesis. Statistical verification and significance of difference between landscapes could have similarly been verified by considering the probability distributions of all comprising likelihood scores.

Another experiment can also involve combining respective landscapes together to identify the location of the starting basins of heuristic-obtained trees when compared between two different alignments. The data set used in this thesis has evidence

for possible lateral gene transfer. Rescoring on the basis of one alignment or the other could have provided very interesting results regarding how the phenomenon of lateral gene transfer is actually moving the model of likelihood and therefore the combinatorial space itself.

Finally, in order to have condensed similar regions of the landscape to get a better view of more informative regions, consensus trees could have been built using a linear-time algorithm proposed by Day [25]. This would have formed so-called terraces where resultant tree scores are identical or close enough to be considered equally viable solutions [75].

Bibliography

[1] Evan Albright, Jack Hessel, Nao Hiranuma, Cody Wang, and Sherri Goings. A comparative analysis of popular phylogenetic reconstruction algorithms. *Midwest Instruction and Computing Symposium (MICS) 2014 Proceedings*, 2014.

[2] Nina Amenta and Jeff Klingner. Case study: Visualizing sets of evolutionary trees. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 71–74. IEEE, 2002.

[3] Nikolas Askitis and Ranjan Sinha. Engineering scalable, cache and space efficient tries for strings. *The VLDB Journal*, 19(5):633–660, 2010.

[4] GT Attwood, K Reilly, and BKC Patel. Clostridium proteoclasticum sp. nov., a novel proteolytic bacterium from the bovine rumen. *International journal of systematic bacteriology*, 46(3):753–758, 1996.

[5] Daniel L. Ayres, Aaron Darling, Derrick J. Zwickl, Peter Beerli, Mark T. Holder, Paul O. Lewis, John P. Huelsenbeck, Fredrik Ronquist, David L. Swofford, Michael P. Cummings, Andrew Rambaut, and Marc A. Suchard. Beagle: An application programming interface and high-performance computing library for statistical phylogenetics. *Systematic Biology*, 61(1):170–173, 2012.

[6] Ricardo A. Baeza-Yates and Gaston H. Gonnet. Fast text searching for regular expressions or automaton searching on tries. *J. ACM*, 43(6):915–936, November 1996.

[7] Oliver Bastert, Dan Rockmore, Peter F Stadler, and Gottfried Tinhofer. Landscapes on spaces of trees. *Applied mathematics and computation*, 131(2):439–459, 2002.

[8] Johannes Bergsten. A review of long-branch attraction. *Cladistics*, 21(2):163–193, 2005.

[9] David Berry. Statistics: A bayesian perspective. 1996.

[10] Louis J Billera, Susan P Holmes, and Karen Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

[11] Christian Blouin, Davin Butt, Glenn Hickey, and Andrew Rau-Chaplin. Fast parallel maximum likelihood-based protein phylogeny. In *ISCA PDCS*, pages 281–287, 2005.

[12] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003.

[13] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[14] Robert S Boyer, Warren A Hunt Jr, and Serita M Nelesen. *A compressed format for collections of phylogenetic trees and improved consensus performance*. Springer, 2005.

[15] David Bryant. Hunting for trees, building trees and comparing trees: theory and method in phylogenetic analysis. *Department of Mathematics, University of Canterbury, New Zealand*, 88, 1997.

[16] David Bryant. The splits in the neighborhood of a tree. *Annals of Combinatorics*, 8(1):1–11, 2004.

[17] David Bryant and Mike Steel. Constructing optimal trees from quartets. *Journal of Algorithms*, 38(1):237–259, 2001.

[18] Daniele Catanzaro, Rafflaele Pesenti, and Michel Milinkovitch. An ant colony optimization algorithm for phylogenetic estimation under the minimum evolution principle. *BMC Evolutionary Biology*, 7(1):228, 2007.

[19] Michael A. Charleston. Toward a characterization of landscapes of combinatorial optimization problems, with special attention to the phylogeny problem. *Journal of Computational Biology*, 2(3):439–450, 1995.

[20] Cédric Charrier, Gary J Duncan, Martin D Reid, Garry J Rucklidge, Donna Henderson, Pauline Young, Valerie J Russell, Rustam I Aminov, Harry J Flint, and Petra Louis. A novel class of coa-transferase involved in short-chain fatty acid metabolism in butyrate-producing human colonic bacteria. *Microbiology*, 152(1):179–185, 2006.

[21] Alexis Criscuolo and Simonetta Gribaldo. Bmge (block mapping and gathering with entropy): a new software for selection of phylogenetic informative regions from multiple sequence alignments. *BMC evolutionary biology*, 10(1):210, 2010.

[22] Jeffrey HF Cullis. *A Framework for the Construction, Visualization, and Characterization of Phylogeny Search Space*. Dalhousie University (Canada), 2008.

[23] Charles Darwin. *On the origins of species by means of natural selection*. 1859.

[24] Richard Dawkins. *The selfish gene*. Oxford university press, 2006.

[25] William HE Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2(1):7–28, 1985.

[26] William HE Day, David S Johnson, and David Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical biosciences*, 81(1):33–42, 1986.

[27] Huy Q Dinh, Bui Quang Minh, Hoang Xuan Huan, and Arndt Von Haeseler. Acophy: a simple and general ant colony optimization approach for phylogenetic tree reconstruction. In *Swarm Intelligence*, pages 360–367. Springer, 2010.

[28] Marco Dorigo, Gianni Di Caro, and Luca M Gambardella. Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172, 1999.

[29] Marco Dorigo and Thomas Sttzle. *Ant Colony Optimization*. A Bradford Book, 2004.

[30] Ding-Zhu Du and Panos M Pardalos. *Handbook of combinatorial optimization: supplement*, volume 1. Springer Science & Business Media, 1999.

[31] Sylvia H Duncan, Adela Barcenilla, Colin S Stewart, Susan E Pryde, and Harry J Flint. Acetate utilization and butyryl coenzyme a (coa): acetate-coa transferase in butyrate-producing bacteria from the human large intestine. *Applied and environmental microbiology*, 68(10):5186–5190, 2002.

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

[33] I. M. H. Etherington. Non-associate powers and a functional equation. *The Mathematical Gazette*, 21(242):pp. 36–39, 1937.

[34] Laurent Excoffier and Andre Langaney. Origin and differentiation of human mitochondrial dna. *American Journal of Human Genetics*, 44(1):73, 1989.

[35] Arash Farzan and J.Ian Munro. A uniform approach towards succinct representation of trees. In Joachim Gudmundsson, editor, *Algorithm Theory SWAT 2008*, volume 5124 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin Heidelberg, 2008.

[36] Julian Feldman and Edward A Feigenbaum. *Computers and Thought: A Collection of Articles by Armer...[et Al.]*. McGraw-Hill, 1963.

[37] Joseph Felsenstein. The number of evolutionary trees. *Systematic Biology*, 27(1):27–33, 1978.

[38] Joseph Felsenstein, J Archie, W Day, W Maddison, C Meacham, F Rohlf, and D Swofford. The newick tree format, 1986.

[39] Joseph Felsenstein and Joseph Felenstein. *Inferring phylogenies*, volume 2. Sinauer Associates Sunderland, 2004.

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

[41] T. Flouri, F. Izquierdo-Carrasco, D. Darriba, A.J. Aberer, L.-T. Nguyen, B.Q. Minh, A. von Haeseler, and A. Stamatakis. The phylogenetic likelihood library. *Systematic Biology*, 2014.

[42] Peter G. Foster. Modeling compositional heterogeneity. *Systematic Biology*, 53(3):485–495, 2004.

[43] Les R Foulds and Ronald L Graham. The steiner problem in phylogeny is np-complete. *Advances in Applied Mathematics*, 3(1):43–49, 1982.

[44] T Ryan Gregory. Understanding evolutionary trees. *Evolution: Education and Outreach*, 1(2):121–137, 2008.

[45] Roberto Grossi and Giuseppe Ottaviano. The wavelet trie: Maintaining an indexed sequence of strings in compressed space. *CoRR*, abs/1204.3581, 2012.

[46] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[47] Youssef Hamadi. *Combinatorial Search: From Algorithms to Systems*. Springer, 2013.

[48] Lenwood S Heath and Naren Ramakrishnan. *Problem Solving Handbook in Computational Biology and Bioinformatics*. Springer Science & Business Media, 2010.

[49] David M Hillis, Tracy A Heath, and Katherine St John. Analysis and visualization of tree space. *Systematic Biology*, 54(3):471–482, 2005.

[50] Guy Jacobson. Space-efficient static trees and graphs. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 549–554. IEEE, 1989.

[51] Viktor Leis, Alfons Kemper, and Thomas Neumann. The adaptive radix tree: Artful indexing for main-memory databases. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 38–49. IEEE, 2013.

[52] Petra Louis, Sylvia H Duncan, Sheila I McCrae, Jacqueline Millar, Michelle S Jackson, and Harry J Flint. Restricted distribution of the butyrate kinase pathway among butyrate-producing bacteria from the human colon. *Journal of bacteriology*, 186(7):2099–2106, 2004.

[53] Petra Louis, Pauline Young, Grietje Holtrop, and Harry J Flint. Diversity of human colonic butyrate-producing bacteria revealed by analysis of the butyryl-coa: acetate coa-transferase gene. *Environmental microbiology*, 12(2):304–314, 2010.

[54] Sean Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013.

[55] T Margush and Fred R McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 43(2):239–244, 1981.

[56] Wes McKinney. pandas: a foundational python library for data analysis and statistics. 2011.

[57] Conor J Meehan and Robert G Beiko. A phylogenomic view of ecological specialization in the lachnospiraceae, a family of digestive tract-associated bacteria. Technical report, PeerJ PrePrints, 2013.

[58] Mark M Millonas. Swarms, phase transitions, and collective intelligence. *arXiv preprint adap-org/9306002*, 1993.

[59] Ingrid Montealegre and K St John. Visualizing restricted landscapes of phylogenetic trees. In *Proc. of the European Conference for Computational Biology (ECCB 03)*. Citeseer, 2002.

[60] Bernard ME Moret, Luay Nakhleh, Tandy Warnow, C Randal Linder, Anna Tholse, Anneke Padolina, Jerry Sun, and Ruth Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):13–23, 2004.

[61] Pat Morin. Data structures for strings. 2014.

[62] Donald R. Morrison. Patricia: Practical algorithm to retrieve information coded in alphanumeric. *J. ACM*, 15(4):514–534, October 1968.

[63] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[64] David W. Mount. Sequence and genome analysis. *Bioinformatics: Cold Spring Harbour Laboratory Press: Cold Spring Harbour*, 2, 2004.

[65] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90–100, 2003.

[66] Richard Otter. The number of trees. *Annals of Mathematics*, 49(3):pp. 583–599, 1948.

[67] Roderic D. M. Page. Space, time, form: viewing the tree of life. *Trends in Ecology & Evolution*, 27(2):113–120, 2012. ¡ce:title¿Ecological and evolutionary informatics¡/ce:title¿.

[68] Morgan N Price, Paramvir S Dehal, and Adam P Arkin. Fasttree: computing large minimum evolution trees with profiles instead of a distance matrix. *Molecular biology and evolution*, 26(7):1641–1650, 2009.

[69] Victor Satler Pylro, Luciano de Souza Vespoli, Gabriela Frois Duarte, and Karla Suemy Clemente Yotoko. Detection of horizontal gene transfers from phylogenetic comparisons. *International journal of evolutionary biology*, 2012, 2012.

[70] D. F. Robinson and Leslie R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981.

[71] Marc HJ Romanycia and Francis Jeffry Pelletier. What is a heuristic? *Computational Intelligence*, 1(1):47–58, 1985.

[72] Franz Rothlauf. *Design of modern heuristics: principles and application*. Springer Science & Business Media, 2011.

[73] Alexander Safatli and Christian Blouin. Application of ant colony optimization for mapping the combinatorial phylogenetic search space. In *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms 2015 (BIOINFORMATICS 2015)*, Lisbon, Portugal, January 2015.

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

[75] Michael J Sanderson, Michelle M McMahon, and Mike Steel. Terraces in phylogenetic tree space. *Science*, 333(6041):448–450, 2011.

[76] Heiko A. Schmidt, Korbinian Strimmer, Martin Vingron, and Arndt von Haeseler. Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3):502–504, 2002.

[77] Alexander Schrijver. *A course in combinatorial optimization*. TU Delft, 2000.

[78] Hidetoshi Shimodaira. An approximately unbiased test of phylogenetic tree selection. *Systematic Biology*, 51(3):492–508, 2002.

[79] Hidetoshi Shimodaira and Masami Hasegawa. Consel: for assessing the confidence of phylogenetic tree selection. *Bioinformatics*, 17(12):1246–1247, 2001.

[80] M. Shishibori, K. Ando, M. Okada, and J.-I. Aoe. A key search algorithm using the compact patricia trie. In *Intelligent Processing Systems, 1997. ICIPS '97. 1997 IEEE International Conference on*, volume 2, pages 1581–1584 vol.2, Oct 1997.

[81] Jennifer S. Shoemaker, Ian S. Painter, and Bruce S. Weir. Bayesian statistics in genetics: a guide for the uninitiated. *Trends in Genetics*, 15(9):354–358, 1999.

[82] P. H. A. Sneath and R. R. Sokal. Unweighted pair group method with arithmetic mean. *Numerical Taxonomy*, pages 230–234, 1973.

[83] Peter F Stadler. Landscapes and their correlation functions. *Journal of Mathematical chemistry*, 20(1):1–45, 1996.

[84] Alexandros Stamatakis. Raxml version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 2014.

[85] Juan Diego Trujillo and Carlos Cotta. An evolutionary approach to the inference of phylogenetic networks. In *Parallel Problem Solving from Nature-PPSN IX*, pages 332–341. Springer, 2006.

[86] J. van Ham. *Interactive Visualization of Large Graphs*. Technische Universiteit Eindhoven, 2005.

[87] Stfan van der Walt, S. Chris Colbert, and Gal Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[88] Karl A Walter, Ramesh V Nair, Jeffrey W Cary, George N Bennett, and Eleftherios T Papoutsakis. Sequence and arrangement of two genes of the butyrate-synthesis pathway of clostridium acetobutylicum atcc 824. *Gene*, 134(1):107–111, 1993.

[89] Simon Whelan and Nick Goldman. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular Biology and Evolution*, 18(5):691–699, 2001.

[90] Sewall Wright. *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*, volume 1. na, 1932.

[91] Marketa J. Zvelebil and Jeremy O. Baum. *Understanding bioinformatics*. Garland Science, 2008.