Overview

1.1. Introduction

Understanding the evolutionary relationships between organisms is one of the central objectives in biology. Originally the study of their evolution was based on physical similarities, diet, or their natural habitat. For example, organs that perform similar activities, digest the same food, or have the same structure are assumed to be from the same taxonomic group [22], in other words, they may stem from the same ancestor. Once the first genetic sequence was determined, scientists began to realize how much information is encoded in it. The main task since then has been to convert sequence information into physical facts or knowledge and thereby to understand the evolutionary relationships between species. This can be likened to translating an unknown human language but is possibly more complicated.

Biological research has begun to rely on computational methods due to the enormous developments in computer hardware and software technologies in recent
years. A huge amount of genetic data has been collected including biological information on genome sequences, genes, gene expressions, protein sequences and structures, taxonomy and other related information. The content of public databases like the GenBank continues to grow at an exponential rate as shown in Figure 1.1. The National Centre for Biotechnology Information (NCBI) has reported that the new GenBank Release 159.0 (April 2007) contains over 75 billion nucleotide base pairs [40] with an increase of 4 billion base pairs as compared to March 2007 release. With such a vast and increasing amount of data available, the challenge is how to analyse it all. A combination of techniques from computer science, mathematics, physics and biological science is required to accomplish such a task.

Figure 1.1 Growth of public database GenBank
The evolutionary relationships between species can be depicted using phylogenetic trees. A phylogenetic tree such as that in Figure 1.2 (also called an evolutionary tree or tree of life) [110] explains the evolutionary relationships among the various species believed to have a common ancestor. In a phylogenetic tree, each leaf node represents an existing species and the internal nodes represent a hypothetical ancestors, and where the edge lengths sometimes corresponding to time estimates. Each node in a phylogenetic tree is called a taxonomic unit. Internal nodes are generally referred to as hypothetical taxonomic units (HTUs) as they cannot be directly observed.

![Phylogenetic tree](image)

**Figure 1.2** Rooted phylogenetic tree, the three-domain system [110]
Chapter 1: Introduction

Phylogenetic study is not only a tool for mapping the history of life on earth; but also plays an essential role in the understanding of human health. It can contribute to the understanding of diseases and viruses, such as the rapidly mutating HIV [69], and in the discovery of new drugs. Phylogenetic trees can provide a lot of information about the mechanism of evolution as well as various evolutionary events and their causes, and are therefore, of great scientific interest.

Phylogenetic trees are inferred from molecular sequences using molecular phylogenetic methods. The majority of biologists use DNA and protein sequences as the identification for each organism. Since the molecular sequence of an organism is directly affected by its evolution, these sequences are a good source of information about the evolutionary processes. The methods for inferring phylogenetic trees are not exclusively biological, but also involve mathematics and algorithms. For example, while a mathematical approach is required to model the evolution, an efficient algorithm is needed to calculate all the required equations in that model. A variety of algorithms based on different criteria and assumptions have been developed to infer phylogenetic trees; most of these are available on Felsenstein’s web page at http://evolution.genetics.washington.edu/phylip/softwar.html.

The success of phylogenetic inference methods depends on two key aspects: the computational time required and, more importantly, the accuracy of the produced result. It is therefore important for computer scientists to design a robust and accurate phylogenetic inference method. This thesis takes a step in this direction.

This Chapter will introduce the genetic data used in phylogenetic analysis, explain the complexity of phylogenetic trees, outline the contributions made by this research and explains the structure of this thesis.
Chapter 1: Introduction

1.2. Genetic Sequence

The genetic sequence or DNA (Deoxyribonucleic Acid) molecule is the main source of information on life forms. All the necessary instructions for a particular organism to live are encoded in these sequences.

A DNA molecule consists of two strands of information twisted together in order to hold more information. The sequence is the order of nucleotide bases (individual letters of sequence) along a DNA strand. There are four types of nucleotide bases: adenine (A), cytosine (C), guanine (G), and thymine (T). During cell division two separate DNA sequences are produced: the entire DNA strand is separated, copied, complementary strands generated, and the duplicated DNA sequences are produced. The genetic information in the DNA is then copied into a messenger RNA (mRNA), which is a complementary DNA template strand where the DNA transcribes to the same bases except for Thymine which transcribes to Uracil. mRNA is used as a template for the translation process to synthesize proteins. Proteins are made up of amino acids; there are 20 amino acids (see Table 1.1). The different protein are expressed in different cells depends on the function of these cells.

It is well known that changes (or mutation) can occur in DNA sequences in the form of nucleotide base substitutions, deletions or insertions. Mutation is the change of one or more positions in the sequences. So, each position in the DNA sequences may change from one base (state) to four possible bases (A, C, G, and T). A mutation for proteins sequences is more complicated, because there are 20 amino Acids.
Table 1.1 Amino Acids

<table>
<thead>
<tr>
<th>Name</th>
<th>Three Letter code</th>
<th>One letter code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine</td>
<td>Ala</td>
<td>A</td>
</tr>
<tr>
<td>Cysteine</td>
<td>Cys</td>
<td>C</td>
</tr>
<tr>
<td>Aspartic Acid</td>
<td>Asp</td>
<td>D</td>
</tr>
<tr>
<td>Glutamic Acid</td>
<td>Glu</td>
<td>E</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>Phe</td>
<td>F</td>
</tr>
<tr>
<td>Glycine</td>
<td>Cly</td>
<td>G</td>
</tr>
<tr>
<td>Histidine</td>
<td>His</td>
<td>H</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>Ile</td>
<td>I</td>
</tr>
<tr>
<td>Lysine</td>
<td>Lys</td>
<td>K</td>
</tr>
<tr>
<td>Leucine</td>
<td>Leu</td>
<td>L</td>
</tr>
<tr>
<td>Methionine</td>
<td>Met</td>
<td>M</td>
</tr>
<tr>
<td>Asparagine</td>
<td>Asn</td>
<td>N</td>
</tr>
<tr>
<td>Proline</td>
<td>Pro</td>
<td>P</td>
</tr>
<tr>
<td>Glutamine</td>
<td>Gln</td>
<td>Q</td>
</tr>
<tr>
<td>Arginine</td>
<td>Arg</td>
<td>R</td>
</tr>
<tr>
<td>Serine</td>
<td>Ser</td>
<td>S</td>
</tr>
<tr>
<td>Threonine</td>
<td>Thr</td>
<td>T</td>
</tr>
<tr>
<td>Valine</td>
<td>Val</td>
<td>V</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>Trp</td>
<td>W</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>Tyr</td>
<td>Y</td>
</tr>
</tbody>
</table>

Models of evolution are based on the assumption that the substitution of nucleotides in sequences is a stochastic process [52], and many of the assumptions about the nucleotide substitution process have been made in order to calculate the probabilities of change from one state to another at a given site. The transition probability can be calculated with a parameter of time and described in a matrix $P$. The function $P_{ij}(t)$ represents the probability of change from state $(i)$ to state $(j)$ within $(t)$ units of time. The total probability for this occurring at site $(i)$ is equal to 1.

$$P(i \rightarrow j \mid t)$$ (1.1)
Chapter 1: Introduction

Many computational models have proposed to evaluate Equation 1.1. Some models are simple; others are fairly complex as will be further explained in the next chapter.

1.3. Problem Complexity

Phylogenetic inference is concerned with finding the tree that best describes the relationship between species according to their genetic sequences. The large amount of reasonable trees generated is problematic because the number of possible trees increases exponentially with the number of species considered. The number of rooted and unrooted trees for \( n \) sequences can be computed as in Equations 1.2 and 1.3 [29] and Table 1.2 shows the number of possible trees thus calculated.

\[
T_{\text{rooted}} = \prod_{i=2}^{n} (2i - 3) \quad (1.2)
\]

\[
T_{\text{unrooted}} = \prod_{i=3}^{n} (2i - 5) \quad (1.3)
\]

### Table 1.2 Number of possible trees for \( n \) sequences

<table>
<thead>
<tr>
<th>( n )</th>
<th>Number of rooted trees</th>
<th>Number of unrooted trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20</td>
<td>( 8.2007945326 \times 10^{21} )</td>
<td>( 221 \times 10^{20} )</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

The problem said to be an NP-hard problem. A number of methods have been developed to recover the correct tree from this large number of possible trees. Most of them are based on statistical theories [79] and optimization strategies. Some methods, such as Neighbour joining (NJ), are fast whereas others, such as the maximum likelihood (ML) method, are slow, and there is a clear trade-off between speed and accuracy.

1.4. Contribution

Reconstruction of phylogenetic trees from genetic sequences is still considered a hard problem to date. There is trade-off between accuracy and speed. Computation along good methods such as ML is extremely expensive, and the running time grows exponentially with increased data. Progress can be achieved by integrating the principles of expensive methods with faster methods to achieve the better results in reasonable time.

The main contribution of this work has been in the implementation and design of a novel quartet-based algorithm called QB that can leads significantly improved accuracy in phylogenetic trees. The parallel version of the QB algorithm has been implemented. The efficacy of the algorithm was tested using different datasets of different size. The experimental results using the benchmark generated by the LIRMM methods and the Algorithms in Bioinformatics Research Group have been included. The thesis also contains an interesting biological analysis of one of the eukaryote groups, the so called “excavate taxa” that is considered one of the keys to understanding early eukaryotic evolution.

The research of this thesis has been published in the following reviewed conference papers:


1.5. Thesis Structure

The remainder of this thesis is organized as follows:

**Chapter 2:** An introduction to phylogenetic tree inference is given along with an explanation on the process of phylogenetic inference. This is followed by some
more detailed information about sequence alignment. The available evolutionary models used in phylogenetic analyses are compared and the criteria defined for choosing the most appropriate model. Most of the computational methods for reconstructing evolutionary trees such as neighbour joining, maximum parsimony, maximum likelihood, and bayesian inference are presented. We further explain some of the commonly used heuristic search and tree rearrangement methods used for finding the best tree. Testing and comparing phylogenetic tree are discussed.

**Chapter 3:** This chapter will introduce a one-to-one mapping procedure between a given tree topology and its global quartet weight matrix. The global quartet weight matrix (QWM) that accumulates the quartet weights is explained. An \( O(n^2) \) algorithm is developed for generating a global quartet weight matrix from a given tree topology. Also, another \( O(n^2) \) algorithm is developed for reconstructing a tree from its quartet weight matrix.

**Chapter 4:** This chapter will introduce a new quartet based algorithm (QB). It will introduce several techniques to deal with inaccuracies in the sets of quartets. These techniques include the calculation of an average confidence value, quartet correction and critical points detection.

**Chapter 5:** In this chapter, we will present experimental results using simulated data sets with different size and compare the performance of the QB with other existing phylogenetic analysis methods.

**Chapter 6:** Experimental results using a real data set called excavate taxa is presented. Firstly, an introduction to the data sets and the current relationships among them are presented. Secondly, the data source and phylogenetic methods used in the experiment are explained. These results are presented and further discussed.
Chapter 7: This chapter introduces a parallel implementation of QB. Experimental results obtained from a 128 node IBM BlueGene/L using simulated data sets to compare the performance of the parallel versions of QB are presented.

Chapter 8: The content of the thesis is summarized followed by future works.
Chapter 2: Reconstructing Evolutionary Trees

2

Reconstructing Evolutionary Trees

This chapter reviews some of the important issues in phylogenetic inference. It presents the complexity of building evolutionary tree and explains the basic steps in phylogenetic analysis. Also, it covers the basic models and algorithms for phylogenetic tree inference. Furthermore, it describes some of the searching techniques used to find the best tree and some methods for assessing the confidence of the final result.

2.1. Introduction

Phylogenetic inference is the study of evolutionary history, sometimes called cladistics. It is a large research area in which the relationships between organisms are investigated using molecular sequence data. Being able to explain the connection
between species, populations, individuals and genes, the general term for which is taxa [66], has many important applications in biology and life science.

The idea is to compare specific characters of species based on the assumption that similar species are genetically close. The relationships between species are represented in the form of a tree as shown in Figure 2.1. A tree that shows common ancestor is a rooted tree, whereas if the common ancestor is not specified, it becomes unrooted. There are methods, such as using out group species, which can be used to re-root unrooted trees (out group species is chosen before constructing the tree, and the final tree then constructed by making this specie share a common ancestor with the rest of species) . The edge lengths measure the dissimilarities between two species or the length of time since their evolution as a separate species. However, genes vary widely in their rate of change, and the different rates in different genes leads to different results in phylogenetic inference. There is only one true tree and many inferred trees.

![Figure 2.1 Rooted and unrooted phylogenetic trees](image-url)
Since the first sequence was discovered and scientists realized how much information was stored in these sequences, many methods have been proposed for reconstructing evolutionary trees from molecular sequences. Some of these methods make direct use of the sequences. However, the general procedure is to compute a score for each possible tree or sub-tree, and then choose the tree that has the optimal score based on the used criteria. This is a straightforward way to find differences between species and analyse them statistically [34].

### 2.2. Basic Concepts

Phylogenetic trees are a logical way to represent the evolutionary relationships. The accuracy of molecular phylogenetic analysis is based on two things: the quality of the data and the performance of the analysis.

Figure 2.2 illustrates the main steps in phylogenetic analysis starting with the selection of data sequences and their alignment, choosing the analysis criteria and the evolutionary model, applying a phylogenetic inference method, and finally, evaluating the obtained tree in terms of its reliability.

There has been tremendous growth in the amount of data relating to biological sequences and structures in molecular biology over the past few years. Large-scale data collections (such as GenBank) provide enough data to begin to draw meaningful conclusions and explore the relationship between organisms.

However, the fold of molecular sequences of different lengths and types requires special treatment before any phylogenetic analysis can be carried out: the sequences need to be rearranged in an alignment that determines which positions along the sequence are derived from the original or common ancestor. In general, the main task of alignment is to maximize the similarities and minimize the number of
insertions and deletions. More details regarding this process are presented in section 2.3.

The next step is to determine which criterion to use. The most commonly used criteria are Maximum Parsimony, Distance, Maximum likelihood, and Bayesian. Each of these criteria has its own strengths and weaknesses, for example, distance-based methods are fast and simple, where maximum likelihood and Bayesian methods are more accurate but slow. Some methods may require an evolutionary
model. There are certain models used for nucleotide sequences and others for protein sequences. All of these models make assumptions on the data and the suitability of the model depends on the type of data. The choice of model will affect the resulting tree.

Over the years numerous phylogenetic inference methods have been proposed for reconstructing evolutionary trees (see [34] for a recent survey). Some methods incorporate the selection of optimal criteria with an evolutionary model. Since the number of candidate trees grows exponentially with the number of taxa, some methods combine tree search schemes with optimality criteria under a specific evolutionary model to improve the running time efficiency.

The obtained evolutionary tree still needs to be evaluated to see how well the tree is supported by the data and how statistically significant it is in comparison to other trees. The tree can be compared with other trees generated by other methods or under different evolutionary models. The evaluation process may depend on statistical tests or biological criteria.

In the next sections, steps in phylogenetic analysis will be discussed in more detail.

2.3. Sequence Alignment

The sequences are simple sets of characters with different lengths due to changes or mutations occurring during the time in which they have diverged from the common ancestor. These changes can be caused by errors during DNA replication or result from the damaging effects of mutagens such as chemicals and radiation. Mutations can be divided into three categories: substitutions, deletions, and insertions. The main challenge is how to define the similarities between present
sequences without having any information about their past. This is why alignment is considered as an important step in phylogenetic analysis [22]. Phylogenetic inference assumes that the sequences are correctly aligned and produces a phylogenetic tree(s) based on this information; thus the quality of the tree produced is affected by the accuracy of the alignment.

Sequence alignment is one of the most basic operations in bioinformatics. It involves inferring where deletions, insertions and substitutions may have taken place within sequences. Gaps may be inserted to make the sequences lengths equal. In most cases, it is hard to distinguish whether the gaps are caused by deletion from one sequence or insertion from the other. Therefore, insertions and deletions are usually treated as a single group of events called \textit{indel}. Usually, many possible alignments could be applied to the same set of sequences and an optimal alignment must therefore be determined according to criteria within a selected scoring system. For example consider the two sequences below:

\begin{align*}
S1: & \quad \text{GCTGAACG} \\
S2: & \quad \text{CTATAATC}
\end{align*}

The following is one of the optimal alignments for these two sequences:

\begin{align*}
S1: & \quad \text{GCTG – AA – CG} \\
S2: & \quad \text{– CTAT AA T C –}
\end{align*}

There are two fundamental types of alignment: Local alignment, which can be done on a segment of the sequence, and Global alignment, which is done on the entire sequences.
Many algorithms have been proposed for sequence alignment. There is pair-wise sequence alignment (such as the Dotplot program) and multiple sequence alignment (such as ClustalW program) which is built up progressively by a series of pair-wise alignments. Algorithms seek to minimize the gaps (dissimilarities) and maximize the number of matches between sequences, thus maximizing the score. Alignments are computed using dynamic programming. The basic algorithms are Needleman [70] for global alignment and Smith-waterman [94] for local alignment. Many computer programs such as ClustalW [106], Muscle [23], and T-Coffee [73] are based on these algorithms.

However, computed sequence alignment has its limitations. It is said to be an NP-complete problem [17] because time and memory both increase exponentially. The problems in inferring the correct alignment will not be further discussed here as they are not the focus of this thesis.

2.4. Models and Assumptions in Phylogenetics

Molecular phylogenetic analysis is based on the existence of an evolutionary history defined by changes in the ancestral genetic sequences [52]. The complete history of an organism can be traced along the mutations in the genomic line. The difficulty lies in the fact that phylogenetic trees can not be built precisely because neither the ancestor nor the exact mutations are always known. Thus a method is needed to analyse and compare huge amounts of molecular data. If the phylogenetic inference were based on the full knowledge of the evolutionary process, then inference methods would be free of errors. In reality, however, methods are based on explicit evolutionary assumptions. In order to correct for hidden mutation, a model of
evolution must be assumed. While some methods may not be based on evolutionary assumptions, they are not assumption free, assumptions maybe explicit or implicit.

Many models have been proposed to evaluate the rate of evolution, and the divergence time between taxa. The evolution rate is used as measurement of the change in an evolutionary lineage across many generations. Some of the models consider similar/different rates, while others assume different rates in different parts of the tree. The rate can be represented in a matrix $Q$, where $Q_{ij}$ represents the rate of change from base $(i)$ to base $(j)$ during small amounts of time ($dt$), and where $Q_{ij} = Q_{ji}$.

In the DNA models $Q$ will be a $4 \times 4$ matrix, since there are 4 states (A, C, G, and T). For amino acids there are 20 states.

The general form of $Q$ for DNA sequences is given below:

$$Q = \begin{bmatrix}
-\mu(a\pi_c + b\pi_c + c\pi_T) & \mu a\pi_c & \mu b\pi_G & \mu c\pi_T \\
\mu a\pi_A & -\mu(a\pi_A + d\pi_G + e\pi_T) & \mu d\pi_G & \mu e\pi_T \\
\mu b\pi_A & \mu d\pi_A & -\mu(f\pi_A + j\pi_c + f\pi_T) & \mu j\pi_T \\
\mu c\pi_A & \mu e\pi_A & \mu j\pi_A & -\mu(\pi_A + k\pi_c + l\pi_T)
\end{bmatrix} \quad (2.1)$$

A number of parameters need to be estimated based on the used model, such as the base frequencies ($\pi_i$), mean Rates ($\mu$), and the relative parameters ($a$, $b$, $\ldots$, $l$) which correspond to the 12 possible types of substitution between distinct bases [52].

The substitution process is usually assumed to be reversible in most evolution models as in Equation 2.2 below:

$$\pi_i P(i \rightarrow j \mid t) = \pi_j P(j \rightarrow i \mid t) \quad (2.2)$$

where the total base frequencies are equal to 1 as in the following equation:
Substitutions of nucleotides can be divided into two types: transition (Ts) and transversion (Tv). Substitutions between pyrimidine (C ↔ T) and between purine (A ↔ G) are transitions, exchanges between purine and pyrimidine (A ↔ C, A ↔ T, C ↔ G, G ↔ T) are transversions.

Note that all models can be classified according to the number of parameters, the assumption of bases frequencies, and according to how many types of transition there are.

The JC69 [57] (or one parameter) model shown above is the simplest model. It assumes equal base frequencies ($\pi_A = \pi_C = \pi_G = \pi_T = 1/4$), and equal rates of substitution ($\alpha$). The JC69 model assumes no difference between transitions and transversions, and the rate matrix can be expressed in Equation 2.4 which can be obtained by setting the relative parameters in the general form to $a=b=c=d=e=f=1$. 

\[
\sum_{i=1}^{n} \pi_i = 1 
\]
where $\alpha = \mu/4$.

The probability is estimated using the following equation:

$$
P_{ij} = \begin{cases} 
\frac{1}{4} + \frac{3}{4} e^{-4\alpha t}, & i = j \\
\frac{1}{4} (1 - e^{-4\alpha t}), & i \neq j 
\end{cases} \quad (2.5)
$$

The F81 model [32] is an improvement on the JC69 model to allow different base frequencies, but it still uses single-type substitutions.

For F81 model:

$$
P_{ij}(t) = \begin{cases} 
\pi_i + (1 - \pi_j)e^{-\mu t}, & i = j \\
\pi_j (1 - e^{-\mu t}), & i \neq j 
\end{cases} \quad (2.6)
$$

Kimura’s two-parameter (K2P) model [60] proposed to allow different rates for transversions and transitions that can be expressed by $Ts/Tv$ ratio ($\kappa$). K2P contains two parameters: the rate of transitions ($\alpha$) and the rate of transversions ($\beta$). $Q$ can be obtained by setting the relative parameters in Equation 2.1 to $a=c=d=f=1$, and $b=e=k$:

$$
Q = \begin{bmatrix} 
-\alpha - 2\beta & \beta & \alpha & \beta \\
\beta & -\alpha - 2\beta & \beta & \alpha \\
\alpha & \beta & -\alpha - 2\beta & \beta \\
\beta & \alpha & \beta & -\alpha - 2\beta 
\end{bmatrix} \quad (2.7)
$$
The probability of change is calculated using the following equation:

\[
P_{ij} = \begin{cases} 
\frac{1}{4} + \frac{3}{4} e^{-\beta t} + \frac{1}{2} e^{-2(\alpha + \beta) t}, & i = j \\
\frac{1}{4} + \frac{1}{4} e^{-\beta t} - \frac{1}{2} e^{-2(\alpha + \beta) t}, & Ts \\
\frac{1}{4} (1 - e^{-\beta t}), & Tv 
\end{cases}
\]  

(2.8)

where the transition ratio \( Ts/Tv = \alpha/2\beta \).

F84 extends on the K2P model [61] to allow different base frequencies. The HKY85 model [49] is a synthesis of both the F84 and K2P models and allows unequal base frequencies and a substitution rate proportional to a rate parameter e.g. \( \alpha, \beta \). The Model assumes reversibility and a total base frequency equal to one so the substitution of the four nucleotide bases is constant over time. Below are the equations for calculating the probability matrices for each model:

For HKY85 and F84 models:

\[
P_{ij} = \begin{cases} 
\pi_j + \pi_j (\frac{1}{T_j} - 1) e^{\mu t} + \left( \frac{T_j - \pi_j}{T_j} \right) e^{\mu t X}, & i = j \\
\pi_j + \pi_j (\frac{1}{T_j} - 1) e^{\mu t} - \left( \frac{\pi_j}{T_j} \right) e^{\mu t X}, & i \neq j (Ts) \\
\pi_j (1 - e^{\mu t}), & i \neq j (Tv) 
\end{cases}
\]

(2.9)

where \( X = l + \pi_j (\kappa - l) \) for HKY85, and \( X = k + l \) for F84. Also, \( T_j = \pi_C + \pi_T \), if base \( j \) is a pyrimidine, and \( T_j = \pi_A + \pi_C \), if \( j \) is a purine base.

The General Time-Reversible model (GTR) [64] has six parameters for the rates of change between each pair of different nucleotides (\( \lambda_{ij} \)). If \( \lambda_{ij} = \lambda_{ji} \), then an additional parameter is added, and this make the computation more complex. The
Chapter 2: Reconstructing Evolutionary Trees

GTR model is derived by setting the relative parameters in Equation 2.1 to \( g=a, h=b, i=c, j=d, k=e, \) and \( l=f. \) Figure 2.4 represents the GTR model parameters.

Tamura-Nei (TN) model [104] is another complex model with two classes of transitions. \( \alpha_R \) is the probability of change from purine (A, C) to purine, and \( \alpha_Y \) is the probability of change from pyrimidine (G, T) to pyrimidine. There is a further probability of change from purine to pyrimidine (or vice versa), where \( \pi_R = \pi_A + \pi_G \) and \( \pi_Y = \pi_C + \pi_T. \) There are two substitutions types, a general one for all types of substitutions and a local one only for transitions [52].

![Figure 2.4 Representation of GTR parameters](image)

Models use the rate matrix to calculate the probability of change matrix (\( P \)):

\[
P(t) = e^{Qt}
\]  

(2.10)

In the case of a long branch length (\( t \)), the exponential can be evaluated by decomposing \( Q \) into its eigenvector and eigenvalue [100]. When the models become complicated, the maximum likelihood calculations need more time. Since ML
estimates the values of all parameters of the data, the calculation time increase with the number of parameters.

For protein or amino acid substitutions, the model must be expanded to 20 states or more in order to compute the probability of change from one amino acid to another. For example, The Poisson model [71] is an extended JC69 model for amino acids, and the proportional model by Hasegawa and Fujiwara [48] is an extension of the F81 model for amino acids. Dayhoff and his co-workers introduced a model of protein evolution [19] whereby because the large number of parameters, they are estimated prior to the analysis of the current data set. A number of empirical matrices have been proposed that used fixed relative rates such as Dayhoff’s PAM50, PAM100 and PAM250 matrices, the JTT matrix [56], the mtREV matrix [1], or the WAG matrix [108].

Models can be classified according to the number of parameters, assumption of base frequencies, and number of transition types they utilise.

**2.4.1. Choosing the Appropriate Model**

Models are generally based on a set of hypotheses about biology. These hypotheses translate into mathematical functions that represent the biological process. Any of these mathematical functions is based on a set of parameters in turn estimated from data through a number of equations. As explained previously, every model has a number of parameters; simple models can be obtained by restricting the parameters in the general matrix in Equation 2.1. Some of these models are complex, while others are simple. Each of them should be assessed, to find the best fit model to the data set being used for a particular tree-building analysis. An inappropriate choice of model affects the outcome of phylogenetic analysis [58].
Thus choosing an appropriate model from among a number of substitution models is one of the most important aspects to inferring accurate phylogeny.

There are three common measures used to select the best-fit substitution model: The hierarchical likelihood ratio test (hLRT), the Bayesian Information Criterion (BIC), and the Akaike Information Criterion (AIC). The hLRT is generally only applied to nucleotide model selection. The AIC and the BIC belong to a different class of model selection measures that compares all the models simultaneously according to a certain measure of fitness. There are also programs such as MODELGENERATOR [58] and Modeltest [77] which have been specifically developed to perform statistical analyses of the complete set of the available substitution models (for example, MODELGENERATOR version 0.82 supports 56 nucleotide and 80 amino acid substitution models). However, further examination of these does not fall within the scope of this research.

2.5. Tree-Building

Most computational methods for reconstructing phylogenetic trees fall into two general categories: distance-based (e.g. Neighbour-joining [87]) and character-based (e.g. maximum parsimony [28], maximum likelihood [31], and Bayesian approaches) [34]. Distance-based methods are computationally fast and depend on a pair-wise distance matrix estimated from the data, while character-based methods employ an evolutionary model to estimate the distance between species directly from sequences and are computationally expensive. A combination of distance and character based methods has also been proposed to improve the performance of phylogenetic inference methods by using the principles of distance-based methods to
get faster and employing the mechanism of character-based methods to get more accurate results, such as Phym [44].

2.5.1. Distance-based methods

Distance-based methods utilize a distance function to estimate how far apart a sequence is from the other sequences and then infer the tree from this data. These methods were introduced for reconstructing large phylogenies for large sets of sequences. It is a compromise of accuracy for computational speed. Typically, a distance matrix $D$ represents the differences between sequences (e.g. the amount of different nucleotide sites). A pair of sequences is closer if they have a minimum amount of differences. One of the important keys to the success of distance-based methods is the accuracy with which the chosen distance metric reflects the amount of divergence between two sequences.

In principle, distance-based methods can be divided into three classes: least square methods, minimum evolution approaches, and clustering methods.

2.5.1.1. Least Square Methods

Least Square methods are some of the best statistically justified approaches to distance-based methods [34]. The concept behind this approach is that there is an observed distance-matrix ($D$) and an expected set of distances ($d$) predicted on the basis of branch lengths. There is also a measure of discrepancy between the observed and the expected distances; the measure used in the least square methods is as in the following equation:
Chapter 2: Reconstructing Evolutionary Trees

\[ Q = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (D_{ij} - d)_{ij}^2 \]  

(2.11)

where \( w_{ij} \) is the weight of pair \( i \) and \( j \).

This weight is assigned differently in different Least Square methods: the first method assumes that all pairs have the same weight (unweighted least square \( w_{ij} = 1 \)) [15], another method assumes that the deviation in the matrices of two closely related species achieves more weight than the deviation in the matrices of two distantly related species (\( w_{ij} = 1/D_{ij} \)) [36]. Similarly, another method suggests that \( w_{ij} = 1/D_{ij}^2 \) [11].

In general, the Least Square method does not search for the topology only, but also for branch lengths that minimize the quantity in Equation 2.11. Finding Least Square trees is shown to be an NP-complete problem [20] and heuristic methods to reconstruct Least Square phylogenies have been implemented in packages such as PHYLIP [33] and PAUP* [103].

2.5.1.2. Minimum Evolution Methods

The Minimum Evolution method (ME) first used the sum of the absolute values of branch lengths as a criterion [59]. Even though the main function is to construct a tree and its branch lengths such that the length of the tree is the shortest, ME methods have become more closely related to Least Square methods in that they estimate edge length using Least Square principles [86, 21].

Since ME methods evaluate the branch length of the constructed tree using Least Square principles based on a distance matrix, they too have become an NP-
complete problem [34]. Therefore, heuristic searches have been proposed to reduce the computational cost.

The Neighbour-Joining algorithm (NJ) [87] is one of the most common distance-based methods that works on minimum evolution principles. NJ calculates a distance matrix $D$ (distances can be obtained from sequence data by a variety of approaches, e.g. Kimura’s 2-parameter estimate [39]), selects the pair of sequences $i$ and $j$ with the lowest distance, and joins them together via an internal node. The inner node replaces sequence $i$ and $j$ and the distances between this internal node and the other sequences, which do not need to be identical, are calculated. Thus the size of the matrix is reduced to $n-1$. The process is repeated until there are two nodes remaining which will be connected together to form a complete tree. Computer simulation studies have shown that NJ performs quite well.

The method BIONJ [38] modifies the NJ algorithm to allow the variance and co-variance of the distance in a simple model of evolution. It is shown that BIONJ can outperform minimum evolution [37]. Also, a weighted NJ algorithm called Weighbor [96] which separately calculates the likelihood for each of a series of overlapping quartets under the assumption that the distances are drawn from Gaussian distribution. Both algorithms BIONJ and Weighbor should do better than NJ, but the cost of this accuracy is that additional approximations are needed to keep calculations to $O(n^3)$. The speed of NJ is still being considered, for example the Fast Neighbour Joining algorithm (FNJ) recently developed an optimal reconstruction radius with running time complexity $O(n^2)$ [25].

Currently, NJ is useful for rapidly locating initial tree that can be improved by other criteria. For example, Phyml [44], one of the fastest and most accurate
algorithms, uses NJ to obtain an initial tree that is then improved under the maximum likelihood criteria using Nearest Neighbour Interchange Method (NNI).

### 2.5.1.3. Clustering Methods

Clustering methods are another class of distance-based methods without an explicit criterion. Instead they apply a particular algorithm to come up with a phylogenetic tree directly from a distance matrix. These methods are derived from a clustering algorithm [47]. Initially, each sequence is in its own cluster. At each step, the closest two clusters are combined into a higher-level cluster based on the distance matrix. The distance between any two clusters is taken to be the average of all distances between the pairs of sequences in the two clusters.

The Unweighted Pair-Group Method with Arithmetic Averages (UPGMA) [95] is a distance-based method that works by clustering. This is the simplest method to infer molecular-clock phylogenetic trees: the distance matrix is used to select the closest pair of sequences in the matrix and to join them together in a cluster whereby the common ancestor for this cluster is located in the middle of the branch connecting the pair. In this way, the matrix size is reduced to \( n-1 \), and the process is repeated until the size of the matrix has reached one.

The UPGMA algorithm makes two assumptions: firstly, the tree is additive, which means that the length of the path between any pair of sequences \( i \) and \( j \) in the tree must be equal to the sum of the branch lengths connecting them; and secondly, the tree is ultrametric, meaning that it is a special kind of rooted additive tree where the terminal nodes are all equidistant from the root. The UPGMA is still being used and implemented in most popular phylogenetic packages such as PAUP* [103] and PHYLIP [33].
2.5.1.4. Limitations of Distance-Based Methods

There has been a lot of development on distance-based methods. They are simple, easy to program and can be very fast, so it is suitable for large data set and that will guarantee the popularity of distance methods. However, there are certain limitations causing concern to some researchers.

Distance-based approaches may disregard certain information concerning the data which may affect the final phylogeny. Simple, they are based on distance matrix to reflect evolutionary relationship and compress sequence information into a single number. The negative side of reducing sequence data to distance matrix is that information is lost in the transformation. For example; by given the distance matrix it is impossible to go back to the original sequences. So distance matrices can not reflect the changes of character states. On the other hand, multiple substitutions at the same site may occur or different lineages may evolve at the same speed. This might cause them to look closer when applying sequence comparisons. Similarity and relationship is not the same thing, even though evolutionary relationship is inferred from certain types of similarity, two taxa can be most similar without being most closely-related. Also, one possible tree can be constructed.

Many corrections have been proposed to improve distance-based methods such as using different equation to calculate the distances between sequences or depend on an evolutionary model, but the problems remain.

2.5.2. Character- Based Methods

Distance-based tree building methods seem to be popular on the basis of running time. However, they should not be taken seriously on the ground of
accuracy. Character-based methods have the same general structure as distance based methods, but they search across a large space of phylogenies. So, instead of directly building a final tree, many possible trees are reconstructed and a heuristic search algorithm is needed to find the best tree. The best tree is chosen based on a score whereby the tree that has maximum (or minimum) score is selected. However, the main problem resides in examining all possible topologies (or a certain number deemed likely to be close to the true tree) to choose a final tree based on particular criteria. It is an extremely time-consuming process, again said to an NP-hard problem.

Three approaches are used and implemented in various computer programs: maximum parsimony, maximum likelihood, and Bayesian. These three approaches are based on the assumption that sites evolve independently.

### 2.5.2.1. Parsimony Criterion

Parsimony is the first method for inferring phylogenetic trees [34]. Like distance-based methods, maximum parsimony also assumes a model of minimum evolution, the only difference being that the minimum evolution is defined per site based on aligned sequences. The main goal for parsimony is to find a tree that requires a small amount of changes (e.g. number of nucleotide substitutions). Since there are usually many possible trees that have a minimal number of changes, a heuristic search algorithm is needed to find the most parsimonious one.

Since the main goal is to find the trees that entails the minimum number of events (i.e. state changes or evolutionary steps), the changes for each site are computed and an overall score is obtained by summing up the parsimony scores of all individual informative sites (similar sites are uninformative).
Figure 2.5 shows an example of computing the score for one site. The calculations have been done for a column of five aligned sequences, and the state (A, C, G or T) at each node is shown. The first step is to re-root the tree at an appropriate place, let us say with a node having a state T. Next, some information on the tree is updated from the top down; once the bottom is reached, the number of changes of state is available. Each inner node state contained in the parent state set has to be selected. Figure 2.5 shows two possible trees (b) and (c) with a parsimonious score equal to 4 changes (branches where changes occur are indicated by dotted lines). The score is computed for all remaining possible rooting of the tree and the minimum parsimony score obtained during this process corresponds to the parsimony score of the tree.

![Figure 2.5](image)

**Figure 2.5** Computation of a Parsimony score resulting in two possible assignments

Parsimony analysis includes a group of related methods, all of which minimize some evolutionary quantity but are different in their underlying evolutionary assumptions. The simplest methods are Fitch and Wagner parsimony: Wagner method [28] assumes that characters are measures on an interval scale, while Fitch method [35] generalizes Wagner parsimony to allow unordered multi-state characters
(e.g. nucleotide and protein sequences). Both methods allow free reversibility from one state to another. A consequence of reversibility is that the tree may be rooted at any point with no change in the tree length. Sankoff’s algorithm [88] is more complex. It builds a table of the cost of changes between states to then compute the cost of the tree by doing a post-order tree traversal.

The tree building algorithms for maximum parsimony searches face similar problems and deploy similar techniques as heuristic maximum likelihood searches. Those heuristics are outlined in section 2.6. Parsimony methods face the same problems as distance-based methods with long lineages, since they do not consider multiple substitutions at the same site. Felsenstein [30] showed how parsimony methods could fail for trees with strongly divergent rates of evolution among lineages. Also, Long Branch attraction is considered a general failure of parsimony criterion with equal rates of changes throughout the tree [51]. Parsimony is described as inaccurate and statistically inconsistent [34].

Parsimony criteria are still popularly used by biologists. An implementation of different parsimony methods can be found in the most widely used packages, such as the commercial PAUP* [103], and the “propars” and “dnapars” in Felsenstein’s PHYLIP [33] which is available as an open source code. A brief discussion of parsimony analyses can be found in Felsenstein’s work on phylogenies [34] while more detailed information is given in [52].

2.5.2.2. Maximum Likelihood Criterion

Maximum likelihood estimation is a method of statistical inference which involves finding the evolutionary tree which yields the highest probability of having evolved from the observed data [31]. Felsenstein used it to infer phylogenetic trees
from nucleotide sequences [31] later Kishino, Miyata, and Hasegawa applied it to infer phylogeny from protein sequences [62]. The goal of maximum likelihood analysis is to find a tree topology with branch lengths that have the highest likelihood of the observed data. The following steps need to be addressed before the tree with the maximum likelihood can be determined: firstly, an evolutionary model must be selected, secondly, the likelihood of a given tree is computed, and then, the branch lengths must be optimised to obtain the maximum score for that tree.

The challenge is to search for the tree that has the highest likelihood value. This problem said to NP-complete due to the large number of potential trees, the number of possible trees growing exponentially with the number of sequences. Most developments thus far have concentrated on minimizing the computational cost of ML.

The likelihood value for a tree is computed based on a specific model of evolution, which provides the transition probabilities among branches of length \( t \) (time-segment). In order to reduce the computation cost of the likelihood, it is assumed that different sites evolve independently (though biologist may not agree with this). Under this assumption, the score of a tree can be computed site by site, and the product of the individual sites gives the score of the tree.

Furthermore, the evolution of different lineages is assumed to be independent. For base transition probabilities, a Markov process is assumed, such that the probability of transition from state \( i \) to state \( j \) is independent of the state of \( i \) regarding prior evolutionary events. The likelihood uses a time reversible model, which means that the evolutionary process can be followed backward or forward in time, such that:

\[
\pi P_{ij}(t) = P_{ij}(t) \pi_i
\]  

(2.12)
Unlike parsimony, to compute the likelihood for site *j*, all possibilities by which the tips sequences could have evolved need to be considered. The computation of the likelihood, given the evolutionary model, is explained by the example tree of Figure 2.6, where the tree in (a) is an unrooted tree of five sequences (1,...,5) and *x*, *y*, and *z* are hypothetical ancestors; and the tree in (b) is a rooted tree where *x*, *y*, and *z* could be \{A, C, G, T\} and the tips (1,...,5) are C, C, G, A and C respectively at site *j*. Initially, a rooted tree is considered, say at node *y* as in Figure 2.6-b. However, the choice of the root has no effect on the likelihood value since it assumes a model of reversibility [31].

![Figure 2.6](image)

**Figure 2.6** Computation of a likelihood value

The branch lengths of the tree are given by *t*. The states for the tips at site *j* are indicated on the rooted tree (Figure 2.6(b)). The total probability of site *j* can be expressed as in the following equation:

\[
L_j = \sum_{y=A}^{T} \sum_{z=A}^{T} \sum_{x=A}^{T} p( y, x, z, C, C, G, A, C )
\]  

(2.13)

and
\[ L = \prod_{j=1}^{n} L_j \]  

(2.14)

If the states at \( x, y, \) and \( z \) are known, the likelihood can be computed as the product of the probabilities of change along each branch and the prior probability at \( y \) (\( \pi_y \)) as in the following equation:

\[ L = \pi_y P_{yx}(t_1) P_{y5}(t_2) P_{yz}(t_3) P_{z2}(t_4) P_{xz}(t_5) P_{z4}(t_6) P_{z3}(t_7) \]  

(2.15)

\( x, y, \) and \( z \) states are unknown, so all possible states need to be taken into account (for nucleotide sequences there are four states A, C, G, T):

\[ L = \sum_{y=A}^{T} \sum_{x=A}^{T} \sum_{z=A}^{T} \pi_y P_{yx}(t_1) P_{y5}(t_2) P_{yz}(t_3) P_{z2}(t_4) P_{xz}(t_5) P_{z4}(t_6) P_{z3}(t_7) \]  

(2.16)

This expression contains many parameters that need to be estimated, i.e. for \( n \) species it has \( 4^{n-1} \) terms which rapidly becomes a large number. It can be reduced by shifting some summation to the right as in the following equation:

\[ L = \sum_{y=A}^{T} \pi_y \left( \sum_{x=A}^{T} P_{yx}(t_1) \left[ P_{x1}(t_4) P_{x2}(t_5) \right] \right) \times \]

\[ \left( \sum_{z=A}^{T} P_{yz}(t_3) \left[ P_{z4}(t_6) P_{z3}(t_7) \right] \right) \times \]

\[ P_{y5}(t_2) \]  

(2.17)

The pattern of the equation looks exactly the same as the structure of the tree. Thus, the likelihood can be computed by starting from the tips applying post-order traversal. A recursive procedure could be used to compute the overall likelihood value by using a conditional likelihood for a sub-tree at node \( k \) of the tree. Let \( L_{(k)}^{(s)} \)

36
be the likelihood of the observed data from node \( k \) on the tree further up, given that \( k \) has a fixed state \( s \). If \( k \) is a tip and consists of state (A), then the likelihood value of one scenario will equal one and the others zero \(( L_A^{(k)}(A) = 1, \text{ and } L_C^{(k)} = L_G^{(k)} = L_T^{(k)} = 0 )\).

Otherwise, if \( k \) has two descendants, \( i \) and \( j \), all four entries can be computed by applying the following equation:

\[
L_{Sk}^{(k)} = \left( \sum_{i=A}^{T} P_{SkSi}(b_i) L_{Si}^{(i)} \right) \left( \sum_{j=A}^{T} P_{SkSj}(b_j) L_{Sj}^{(j)} \right)
\]  
(2.18)

By executing this procedure recursively to the root node \( (y \text{ in the example}) \), the four conditional likelihoods become available at \( L_y^{(4)} \) and the overall likelihood of the tree for a specific site \( i \) is calculated by using the following equation:

\[
L^{(i)} = \sum_{j=A}^{T} \pi_j L_y^{(i)}
\]  
(2.19)

The base frequencies (\( \pi_i \)) have to be the prior probabilities, which are usually evaluated from the aligned data. It does not directly form part of the maximum likelihood process. The log likelihood values are computed due to numerical reasons.

\[
Ln(L) = \sum_{j=1}^{n} Ln(L_j)
\]  
(2.20)

ML uses the reversibility property which Felsenstein calls the “Pulley principles” [31]. In other words, regardless of whether they are rooted or unrooted trees, the likelihood values will remain unchanged. This reduces the computational complexity which, nevertheless, remains high compared to other methods. In general, ML technique outperforms all other methods, and has become the most-used
approach for building evolutionary trees. However, the main challenge in using ML criteria is the computational cost. Therefore, many search methods have been developed in an attempt to reduce the computational cost for ML calculations; these methods are described in section 2.6.

The branch lengths also need to be optimized in order to obtain the maximum likelihood for a specific tree. A numerical method can be used to progressively improve the likelihood of the tree. In FastDNAml [75] the faster converging Newton-Raphson method is implemented, whereas a simple method for the optimization of one-parameter functions (Brent method) is deployed in Phylm [44]. Although, finding the maximum likelihood is hard [16], theses methods have made a good improvement on the likelihood calculation. Experimental results in this research confirm that ML is not unique, and that an ML criterion can not guarantee obtaining the correct tree, instead it gives the probability of the topology for the input data.

2.5.2.3. Bayesian Inference

The Bayesian approach is closely related to maximum likelihood, the only difference is the use of prior distribution of the tree [34]. Also, Bayesian inference searches for a set of best trees rather than a single optimal tree. It is based on a quantity called the posterior probability of the tree:

\[
P(Tree \mid Data) = \frac{P(Data \mid Tree)P(Tree)}{P(Data)} \tag{2.21}
\]
Bayes’ Theory is simply used to combine the prior probability of a tree ($P(\text{Tree})$) with the likelihood ($P(\text{Data}|\text{Tree})$) of obtaining a posterior probability distribution of trees, and that is just a measure of how correct a tree is. Based on that measure, an evolutionary tree is inferred. Usually, equal prior probability is considered for all trees, and the likelihood is computed under an evolutionary model. The posterior probability involves a summation of all trees whereby each tree is considered over all possible combinations of branch lengths and model parameter values.

In practice, it is impossible to implement this procedure due to the large number of trees. Instead, a numerical method called the Markov Chain Monte Carlo (MCMC) [41] is used to approximate the posterior probability of a tree. MCMC is implemented in Bayesian inference applications such as the MrBayes program.[54]. Basically, it draws random samples from the posterior distribution of trees. There will, however, be uncertainty if the number of samples is not enough.

The concept involves getting a starting tree (random or user-specified tree) and a model of evolution. MrBayes’ manual suggests using a random starting tree [85]. More accurate and efficient results could be obtained with MrBayes by using user-specified trees computed by more accurate methods such as ML. MCMC involves two steps: (i) Minor changes on the topology and/or alteration of the model parameters are applied. (ii) The new tree is either rejected or accepted using the Metropolis probability algorithm [67]. In case that the new tree is accepted, the first step is repeated. The calculation is fast since only a local area of the tree has been changed and a conditional likelihood can be used to avoid re-computations. The process of state transition, which is called generations in the related terminology, has to be repeated enough times until the chain becomes stable.
Chapter 2: Reconstructing Evolutionary Trees

The main task is to determine at which point the chain becomes stable. Usually, this is assessed by looking at the best likelihood values over generations and noting when the likelihood becomes stable. The likelihood might continue to increase after a long interval of seeming stationary. More details regarding this issue can be found in [53].

Bayesian inference generates a collection of best trees and allows for the estimation of their posterior probability which in turn enables the assignment of consensus-based probability values. Bayesian inference has become more popular mainly through the release of MrBayes [54].

2.6. Heuristic Search Methods

Searching for the best phylogeny based on a given optimality criterion (e.g. ML, and MP) is computationally expensive. To overcome this, heuristic methods have been proposed to construct phylogenies in practical time [34]. Although, these heuristic methods have improved the performance of phylogenetic analysis, they cannot guarantee to find the best tree(s).

There are many ways to search for the best tree. One is to start from scratch and insert species progressively into the tree until the final tree is completed. Another way is to start with a complete initial tree containing all species generated by fast programs, such as NJ, and then progressively optimize this tree. Also, small sub-trees such as quartets could be built and then recombined into a complete trees.

Hill climbing search is the most used approach in all methods. It improves the current solution (tree or sub-tree) in step by step manner, by which the search keeps moving from the current tree to a neighbouring tree that has higher score until no further improvement can be found.
2.6.1. Stepwise Addition

Stepwise addition is one of the most widely used progressive algorithms. Proposed initially by Felsenstein in [31] and implemented in “dnapars” [33], it starts with three taxon trees and then progressively inserts the remaining $n-3$ taxa into the tree. After each insertion, the optimal branch lengths and likelihood values of the new topology are computed, and the tree with highest likelihood among all possible analysed topologies is selected for the next insertion. The process repeated until all species inserted in the tree. Basically, it is a hill climbing search such as the process outlined in Figure 2.7 where the dashed lines show the possible places for the fifth element. The tree could be further refined by applying rearrangement to the intermediate tree (local rearrangement) or the final tree (global rearrangement) using NNI, TBR, or SPR (to be discussed in the next section).

A quick-add has been implemented in FastDNAml [75] that optimizes the branch adjacent to the insertion point, and saves time by reusing the likelihood values. RAxML [98] uses a similar approach.

Although it is widely used, the tree obtained from this progressive algorithm depends on the order of the sequences. Different orders could produce different trees.
2.6.2. Tree Rearrangement

As mentioned before, algorithms may start with a complete initial tree containing all species (or sub-tree), then improve the quality of this tree by rearrange it. The most important part is the hill climbing search where a small rearrangement of branches in the tree is undertaken to find a neighbouring tree that is better suited in terms of the criterion used. This tree is considered and continued rearrangements carried out until no further improvement can be found. The most commonly used techniques are Sub-tree Pruning and Regrafting (SPR), Nearest Neighbour Interchange (NNI), and Tree Bisection and Reconnection (TBR).
Sub-tree Pruning and Regrafting (SPR) is carried out by removing a branch of a tree with a sub-tree attached to it. That sub-tree is then reinserted into the remaining tree in all possible places. SPR carries out a much wider search and is more likely to find a quality result from all the trees in the space. Figure 2.8 outlines the process.

**Figure 2.8** Sub-tree Pruning and Regrafting (SPR)
Tree Bisection and Reconnection (TBR) splits the tree into two sub-trees by erasing an inner branch. Thereafter all possible connections are made between a branch of one sub-tree and that of the other sub-tree. However, this is based on the tree topology. The process is outlined in Figure 2.9.

**Figure 2.9** A possible bisection and reconnection (TBR). A branch is removed and the two sub-trees are reconnected to create all the possible connections between the branches of one tree and those of the other.
Nearest Neighbour Interchange (NNI) exchanges four sub-trees located at every inner branch of the tree (interchange sub-trees positions). Basically, any tree could be considered as a quartet tree and for each quartet tree there is three topologies. This technique is implemented in Phym [44]. The process is outlined in Figure 2.10.

Figure 2.10 A possible Nearest Neighbour Interchange (NNI) exchanges. The four sub-trees located at every inner branch of the tree exchanged
The quality of these techniques depends on the quality of the starting tree and the criterion used. The process could be speeded up by combining progressive algorithms with tree rearrangement techniques. Also, in addition to the topology modification, branch lengths could be optimized to get better result.

2.7. Measure of Confidence

All methods infer a phylogenetic tree according to the criterion used. However, the main issue in phylogenetic inference is to obtain confidence in the final evolutionary tree. In other words, how close is a tree to the correct tree?

The traditional way to ascertain this is to generate a number of trees for the same data set using different models of evolution, alignment, or data types. Then, a final consensus tree is built using any package e.g. tree-puzzle [89], Paup* [103], or Phylip [33]. Consensus tree methods simply assign a confidence value to each sub-tree or clade (e.g. 95% which indicates high confidence). Most consensus programs can produce a strict majority rule, and an extended majority rule consensus tree. Strict majority consensus trees only include bifurcation trees that form part of all trees, but multifurcation could be used for non-resolvable nodes. Extended majority rule consensus methods accept a sub-tree if it appears in the majority of trees, it might also produce a multifurcation tree. The extended majority rule allows for the computation of strictly bifurcating consensus trees.

Bootstrapping is another way to measure the confidence in a tree. It works by building many replicas for the input sequences (with new alignment where columns could be deleted or others replicated). The only restriction is that the new alignment must have the same length as the original one. Bootstrapping is a good way to assess the robustness of the obtained tree by using replicated input data. However, the
computational cost is considerable which makes it difficult for large data sets. Also, how many replicates should be enough?

Recently, most programs include consensus methods, bootstrapping and other statistical approaches for assessing the confidence in the final tree.

2.8. Testing and Comparing Programs

Many approaches have been used to infer evolutionary history. Some approaches are simple and fast such as NJ, while others are complicated and searching through vast numbers of possibilities such as MP and ML. Recently, there has been a trade-off between speed and accuracy. However, the performance of every program needs to be assessed according to the basic issues of performance, namely speed, tree quality, the ability to handle different types of data and models of evolution, and optimization methods.

Accuracy has been tested using simulated data. However, an algorithm’s result must be produced with respect to biological prospective. Models of evolution may be considered as a biological assumption, but the effect of model choice on the results is an important issue requiring further attention. Simulated experiments discussed in chapter 4 show that ML may miss the true tree by choosing the tree with the highest likelihood. For simulated data, the accuracy of phylogenetic reconstruction methods is determined by comparing the produced trees with the true tree. When using real data, the true tree is unknown, therefore, the trees’ scores are compared to determine a potential best tree.

One of the most widely used programs for generating synthetic alignment is Seq-Gen [80]. In this program the true tree’s model of evolution is known prior, and the alignment contains less errors and gaps. For tree comparison, most studies use the
Robinson and Foulds rate (RF) [83]. In RF the distance between two trees is a relative measure for topological dissimilarities. (In chapter 5, we used RF program to measure the performance of our new algorithm with other program such as Phyml, RAxML, and FastDNAml.)

2.9. Summary

This Chapter provided an introduction to phylogenetic inference, which covers the complexity of the task and the basic models and algorithms for the reconstruction of evolutionary trees. Moreover, it discussed methods for evaluating generated trees and comparing the results from different programs. Finally, it described some techniques to find the best tree.

The large number of possible trees is a major problem in molecular phylogenetic analysis. The problem is often NP-hard. A number of methods have been applied to recover the correct tree from this large number of trees. The study of available methods has shown that the final results are normally trade-off between speed and accuracy. No existing method can guarantee to obtain correct tree.

The following chapters describe a novel algorithm quartet-based method, which can infer large trees and outperform existing methods in term of accuracy.
This chapter introduces a one-to-one mapping procedure between a given tree topology and its global quartet weight matrix. First, the concept behind quartet trees is explained, and then the quartet puzzling algorithm is described. This is followed by an introduction of the global quartet weight matrix (QWM) that accumulates the quartet weights from all possible quartets for a given tree topology. An $O(n^2)$ algorithm is presented for generating a global quartet weight matrix from a given tree topology. Furthermore, another algorithm is presented for reconstructing a tree topology from its quartet weight matrix.

### 3.1. Introduction

There has been interest in the reconstruction of large evolutionary trees from small sub-trees in computational biology [6, 7, 8, 9, 10, 26, 43, 55, 89, 100, 105]. Instead of solving the problem using the entire input data once, the large input could
be divided into small sets, the solution for each small set found independently. These solutions are combined to obtain the complete solution for the original problem. A divide and conquer approach is proposed to reduce the computational cost of complex methods.

Distance-based trees could be considered as constructions of evolutionary tree by finding a possible two-species tree, then start adding the other species to this tree. It is also possible to use three-species trees. A triple method of combining unrooted three-species tree into full trees has been developed based on two, three, and four species distancing filled by least squares [42]. However, the three-species tree has only one unrooted topology, and it is hard to distinguish any relationship from this topology. Thus, a four-species tree (quartet tree), which has different topologies seems to be more reasonable. Therefore, the quartet-based approach is the most cost-effective one for reconstruction evolutionary trees from small sub-trees.

3.2. Quartet Trees

Quartet trees are the smallest informative structures of binary unrooted trees representing a relationship among four different species. There are seven possible topologies for each of the four species (a, b, c, and d) [101] as depicted in Figure 3.1. Three of them are partially resolved (P1-3) and one is totally unresolved (U) while the other three topologies are totally resolved (T1-3) and can be used to distinguish a binary relationship between species.

Quartet-based approach consists of two stages: The first stage consists of inferring all possible \(^{n \choose 4}\) four sequences (or quartet) topologies over an input of \(n\) sequences. Any existing method, such as maximum parsimony, maximum likelihood, or neighbour joining, can be used to reconstruct the quartet set [52]. The
recombination stage takes the information provided by the quartet set and combines these quartets to form an estimate of the unknown phylogeny. The inferred set of small trees could be likened as clues to the final tree.

Many methods have been developed based on quartet trees. For example; Quartet Puzzling (QP) [89], The Short quartet method [27], IQPNN [68,107], and $Q^*$ [9]. In addition to these methods, several techniques have been proposed to compensate for the varying degrees of errors presented in the quartet set such as quartet cleaning (QC) [10].

The main question is how robust quartet-based methods are with respect to errors in the quartet set. A comparison between quartet methods [97] has shown that the $Q^*$ method is the least robust method, and that the QC method provides some

**Figure 3.1** The seven topologies for each quartet or four sequences (A, B, C, D).
error tolerance. However, both of these methods rate lower than quartet puzzling in terms of error tolerance. The accuracy of quartets is dependent on the used method for inferring these quartet trees.

On the other hand, assuming a fully resolved tree for each quartet is an unreasonable assumption, which may prevents the quartet-based approach from being widely adopted in practice. Therefore, the main concern in designing good and effective quartet-based algorithms is how to tolerate errors in the quartet trees. The running time might be an issue, but it is more important to obtain the most accurate results.

One of the most interesting aspects about quartet based methods is that they consist of separate individual stages, and these stages can be easily separately developed. For example, the quartet tree can be constructed via any algorithm. The tree construction from these quartets is a complete separate step, which can be optimized by using several approaches.

3.3. Quartet Puzzling Algorithm

Currently, quartet puzzling (QP) is one of the most widely used quartet-based method. This method reconstructs the maximum likelihood tree for each possible quartet and was originally developed to reduce the computational cost of maximum likelihood methods. QP is a three-step algorithm:

1. Maximum Likelihood step: The maximum likelihood tree for each of the \( \binom{n}{4} \) possible quartets is reconstructed.
2. Puzzling step: The quartet trees are combined into a complete tree containing all the sequences which are added sequentially in random order. The new sequence is added to a branch according to a voting procedure considering all the related quartets. The result of this step is an intermediate tree containing the full sequences. There will be a number of intermediate trees since the puzzling step is repeated many times to elucidate the landscape of possible optimal trees.

3. Consensus step: The final tree is obtained through a majority rule consensus of all intermediate trees. The resulting tree might be a binary or a multifurcation tree depending on the consensus step.

In the first step, for each quartet there are three fully resolved topologies exist with corresponding maximum likelihood values \(l_i\). All topologies with \(l_i = \max(l_1, l_2, l_3)\) are stored for the Puzzling step. If there are more than one topology with high likelihood value, one of them is chosen randomly. The QP algorithm is modified to use Bayesian weight or posterior probabilities of the likelihood [89]:

\[
\begin{align*}
\omega_i &= \frac{l_i}{l_1 + l_2 + l_3} \\
\end{align*}
\]  

(3.1)

and that improves the search for the maximum likelihood tree according to the criterion specified for the quartet puzzling.

Quartet puzzling takes both resolved and unresolved quartet trees into consideration. When adding a sequence to a sub-tree, it considers only those quartets associated with the new sequence and the sequences in the sub-tree, it is a local optimization process and very sensitive to quartet errors.
3.4. Global Quartet Weight Matrix

The three possible fully resolved trees for a quartet \( \{a, b, c, d\} \) can be depicted in Figure 3.2. In this Figure, \((xy \mid zt)\) indicates how the sequences, represented by leaf nodes, are divided into two pairs by cutting the middle edge (so-called bi-partitioning), and thus shows the neighbourhood relations of the quartet in terms of its topology.

All the quartets from a given \( n \) sequences can be plotted in a two-dimensional graph to see how many quartets are resolved (and how many unresolved) using the likelihood-mapping procedure [100]. In this procedure, the likelihood values for each of the three possible resolved trees are calculated and then transformed into posterior probabilities, or Bayes weights (or quartet weights) \( w_i \) for \( i = 1, 2, \) and \( 3 \), by applying Bayes’ theorem assuming a uniform prior for all three trees. Three of the weights of each quartet can be mapped onto a two-dimensional simplex. Using this mapping it can be seen how closely the maximum-likelihood (ML) method is able to resolve each quartet into one of the three possible trees. A generalized method, based on the same principle, named quartet mapping, was developed to measure other approaches to quartet tree construction and is used to compare different methods in their ability to indicate the correct topology [72].

![Figure 3.2](image-url) The three possible fully resolved trees for a quartet \( \{a, b, c, d\} \)
For any given $n$ sequences and its associated set of quartets, a symmetric global quartet weight matrix of size $n \times n$ can be generated. Let $(ij|kl, w)$ denote a quartet topology with weight $w$, the global quartet weight matrix (QWM) is generated by adding the quartet weight $w$ to entries $ij$, $ji$, $kl$, $lk$, using a complete set of quartets. A simple example is shown in Figure 3.3, in which there are five sequences and each quartet is associated with three weights for three possible resolved topologies.

\begin{table}[h]
\centering
\begin{tabular}{c|cccccc}
\hline
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0.0 & 1.2 & 0.6 & 0.6 & 1.2 \\
2 & 1.6 & 0.0 & 0.4 & 0.4 & 1.6 \\
3 & 0.6 & 0.4 & 0.0 & 2.4 & 1.6 \\
4 & 0.6 & 0.4 & 2.4 & 0.0 & 0.6 \\
5 & 1.2 & 1.6 & 0.6 & 0.6 & 0.0 \\
\hline
\end{tabular}
\end{table}

\textbf{Figure 3.3} A simple example of the Global Weight Matrix. (a) represents the quartets and their weights. (b) represents the associated matrix.
3.5. Reconstructing the QWM from a Given Tree Topology

For any given tree, a unique set of \( \binom{n}{4} \) quartet trees which are consistent with the original tree can be determined, i.e., each quartet tree separates the four leaves into two pairs in the same way as the original tree does through bi-partitioning. For each quartet there are three topologies, only one can be fully resolved tree in the set of these quartet trees with weight equal to one and it can be described as \((ab|cd, 1.0)\), while the other two have zero weights. From the given tree topology a unique global quartet weight matrix is also determined. This is because the matrix is generated using the quartet weights of the associated quartet trees. In the following section, an \( O(n^2) \) algorithm is presented to construct the quartet weight matrix (QWM) directly from a given tree topology.

To determine the entry value on row \( i \) and column \( j \) of the global quartet weight matrix from a given tree topology, the total number of quartet trees with the form \((ij|pq, 1.0)\) for \( p, q \neq i, or j \) must be calculated. This can be done using the bi-partitioning technique by first determining the path which is unique from leaf node \( i \) to node \( j \), as shown by the dashed line in Figure 3.4. For each internal node on that path, a cut is made on the third edge which is not on the path to separate the tree into two sub-trees. This ensures that leaves \( i \) and \( j \) will always be in the same sub-tree. Counting the number of leaves in the sub-tree which does not contain leaves \( i \) and \( j \), the number of quartet trees with the form \((ij|pq, 1.0)\) from this bi-partitioning is calculated through the following equation:

\[
Q_{ij} = \begin{cases} \frac{n_k}{2}, & n_k \geq 2 \\ 0, & n_k < 2 \end{cases}
\] (3.2)
Where $n_k$ is the number of leaves in that sub-tree which not includes $i$ and $j$ (Note that $\binom{n_k}{2} = 0$ when $n_k < 2$). Therefore, the total number of such quartet trees or the value for entry $ij$ in the quartet weight matrix will be:

$$m_v = \sum_{k=1}^{L} \binom{n_k}{2}$$

(3.3)

where $L$ is the number of the internal nodes on the path connecting $i$ and $j$.

**Figure 3.4** Calculating the number of quartet trees which have the form of $(ij|pq, 1.0)$ using the bi-partitioning technique, where $p$ and $q$ could be any other species except $i$ and $j$. 
3.5.1. Efficient Algorithm

The bi-partitioning procedure described in the previous section is not an efficient way to directly calculate each entry value of the quartet weight matrix. The process is far too lengthy: for each leaf node the paths to every other node must be found, then cuts must be set for every node on every path, and the number of nodes in the other sub-tree counted for each bi-partitioning. As such, the tree has to be traversed many times and there will be a large number of redundant calculations. Dynamic programming can be used to eliminate the redundant calculations and obtain a very efficient algorithm.

Since each internal node in an unrooted binary tree has three edges, a tree will be divided into three sub-trees when an internal node is removed. For example, in Figure 3.5 three sub-trees \(a, b\) and \(x\) are connected to each other by node \(l_o\) which has three edges connecting to nodes \(l_a, l_b, l_x\) in the sub-trees, respectively. Node \(l_a\) (or \(l_b\) or \(l_x\)) is called the leading (representative) node of a sub-tree since it is the only node linking the sub-tree to other sub-trees. The main concern here is to calculate the total number of quartet trees with the form \((ij|pq, 1.0)\) where \(i\) is a leaf node in sub-tree \(a\) and \(j\) a leaf node in sub-tree \(b\). It is obvious that nodes \(l_a, l_o\) and \(l_b\) must be on the path from leaf node \(i\) to leaf node \(j\). The calculation for the total number of quartet trees meeting the specified form can then be divided into three parts according to the three sub-trees: Firstly, into those quartet trees generated in sub-tree \(a\) from leaf node \(i\) along the path to the leading node \(l_a\). Secondly, the quartet trees generated in sub-tree \(b\) from node \(j\) to node \(l_b\), and thirdly those generated through a bi-partitioning on the third edge of node \(l_o\), not on the path from \(i\) to \(j\).
After removing the third edge of node $l_0$ that connected to sub-tree $x$, the tree becomes two sub-trees, one being the combination of sub-trees $a$ and $b$, the other being the entire sub-tree $x$. Assuming that the number of leaves in sub-trees $a$ and $b$ are $n_a$ and $n_b$, respectively, the number of leaves in sub-tree $x$ is $n - (n_a + n_b)$, where $n$ is the total number of leaves in the complete tree. The number of quartet trees that can be generated to have the form $(ij|pq, 1.0)$ through this bi-partitioning is simply calculated using the following equation:

![Diagram](image-url)
\[ Q_{ij} = \left( n - (n_a + n_b) \right) / 2 \]  \hspace{1cm} (3.4)

Assuming further that the total number of the quartet trees with the concerned form in sub-tree \( a \) from leaf node \( i \) along the path to the leading node \( l_a \) is \( m_i \) and that the number in sub-tree \( b \) from node \( j \) to node \( l_b \) is \( m_j \), the total number of quartet trees which have the form \( (ij|pq, 1.0) \) can then be obtained as follows:

\[ m_{ij} = Q_{ij} + m_i + m_j \]  \hspace{1cm} (3.5)

This is also the entry value on row \( i \) and column \( j \) of the global quartet weight matrix.

After every entry value \( (m_{ij}) \) associated with the leaves in sub-trees \( a \) and \( b \) has been calculated, the two sub-trees are merged into one. Node \( l_o \) will be the leading node and the number of leaves in it is equal to:

\[ n_o = n_a + n_b \]  \hspace{1cm} (3.6)

For each leaf node in sub-tree \( a \), \( \left( \begin{array}{c} n_b \\ 2 \end{array} \right) \) is added to \( m_i \), i.e.:

\[ m_i = m_i + \left( \begin{array}{c} n_b \\ 2 \end{array} \right) \]  \hspace{1cm} (3.7)

This is because when this new and larger sub-tree is merged with another sub-tree, it is necessary to know the number of concerned quartet trees from each leaf node \( i \) of the original sub-tree \( a \) to the new leading node \( l_o \). However, the number from node \( i \) to node \( l_o \) is already known, and it is simply a matter of making an additional bi-
partition by cutting the edge from node $l_o$ to node $l_b$. This will make the original sub-
tree $b$ be completely separated from the rest. Therefore, \( \binom{n_b}{2} \) more quartet trees of
the concerned form can be generated on the path from leaf node $i$, which is originally
in sub-tree $a$, to the new leading node $l_o$. For the same reason \( \binom{n_a}{2} \) is added to $m_j$
which is associated with each leaf node in sub-tree $b$, i.e.,

$$m_j = m_j + \binom{n_a}{2}$$ (3.8)

By applying Equations 3.7, an efficient algorithm can be formed to obtain the
entry values of the global quartet weight matrix from a given tree topology. In this
algorithm, each leaf node is associated with a variable $m_i$ to store the number of
quartet trees of the concerned form obtained from the current sub-tree. It seems that
for each node $i$, $n-1$ variables might be needed to store the numbers for different $ij$
pairs. However, a single variable will be enough since the path from node $i$ to the
leading node is the common sub-path for node $i$ to reach any other nodes outside the
sub-tree. Each sub-tree is associated with a variable $n_i$ to record the number of leaf
nodes in the sub-tree. Initially, every leaf node is considered as a separate sub-tree
containing a single node. Thus the value of $m_i$ is set to zero and the value of $n_i$ is
set to one. Sub-trees are recursively merged together according to the tree topology
using the above equations until there is only a single tree left.

The size of each sub-tree will be no greater than $n$. Each loop in the above
process takes only $O(n)$ operations. Thus it takes $O(n)$ operations to merge two sub-
trees. There are $n-1$ such steps to merge all leaf nodes into a single tree according to
the tree topology. Therefore, the total computational cost of this algorithm is $O(n^2)$. 

61
A pseudo code of the algorithm is given in the following:

For each leaf node \( i \):

Set \( n_i = 1 \)

set \( m_i = 0 \)

While (number of sub-trees > 1) do:

\{

Find two sub-trees \( a \) and \( b \) with common ancestor \( o \)

For each leaf node \( i \) in sub-tree \( a \), and leaf node \( j \) in sub-tree \( b \):

\[
m_{ij} = \left( n - \left( \frac{n_a + n_b}{2} \right) \right) + m_i + m_j
\]

For each leaf \( i \) in sub-tree \( a \):

\[
m_{i} = m_{i} + \left( \frac{n_b}{2} \right)
\]

For each leaf \( j \) in sub-tree \( b \):

\[
m_{j} = m_{j} + \left( \frac{n_a}{2} \right)
\]

node \( o \) becomes the leading node: set \( n_o = n_a + n_b \)

\}

3.6. Tree Reconstruction from a Given Quartet Weight Matrix

The previous section introduced an \( O(n^2) \) algorithm that generates the global quartet weight matrix from a given tree topology. With a simple modification, the previous algorithm can be used to reconstruct the original tree from its generated matrix. Thus, there is a one-to-one mapping between a given tree topology and its generated global quartet weight matrix.
In the algorithm for generating the quartet weight matrix, each row in the matrix corresponds to a particular leaf node and is associated with a variable \( m_i \) which records the number of the concerned quartet trees generated on the path from the leaf node to the leading node of the sub-tree. Also, each sub-tree is associated with a variable \( n_i \) to record the number of leaf nodes in the sub-tree. When there is more than one leaf node in a sub-tree, only one node is chosen as representative for that sub-tree.

Every leaf node is considered as a sub-tree. Therefore, each row in the matrix will be associated with two variables set to \( m_i=0 \) and \( n_i=1 \). Since there is only one generated matrix and no tree topology to follow when merging sub-trees, a special procedure is needed to find the correct edges in the original tree. With two representative nodes \( i \) and \( j \) of a two sub-trees, a value \( d_{ij} \) can be calculated using the following equation:

\[
d_{ij} = \left( n - (n_i + n_j) \right) + m_i + m_j
\]  

This equation looks exactly the same as Equation 3.4 that was used to calculate the entry values \( (m_{ij}) \) of the matrix. The two sub-trees are directly connected in the original tree only if \( d_{ij} \) is equal to the corresponding value \( m_{ij} \) in the matrix. Therefore, a confidence value can be defined as follows:

\[
c_{ij} = \frac{m_{ij}}{d_{ij}}
\]  

For any two sub-trees to be merged together the corresponding confidence value must be equal to one. When there is more than one leaf node in each sub-tree,
there is no need to calculate all the $c_{ij}$ values associated with every leaf node pair. When two sub-trees are directly connected in the original tree, the confidence value $c_{ij}$ for every associated node pair must be equal to one. Therefore, only their representative nodes need to be used. The variable called $g_i$ is an additional variable for the representative node $i$ to store index $j$ when $c_{ij}=1$. This can save considerable amount of computational cost.

The algorithm is better described using an example: Initially, every leaf node is considered as a sub-tree and contains only one leaf node ($n_i=1$). Thus, each row in the matrix is associated with three variables which are set to $m_i=0$, $n_i=1$, $g_i=null$. The first step is to calculate the desired value using Equation 3.9:

$$d_{ij} = \left(\frac{n-(n_i+n_j)}{2}\right) + m_i + m_j = \left(\frac{7-(1+1)}{2}\right) + 0 + 0 = 10$$

For all $i < j$, starting from the first row, each entry value is divided by this $d_{ij}$ to get its confidence value. As depicted in Figure 3.6(a), there are only three entries which have confidence values equal to 10.0 and their $g_i$ values set to the corresponding node indices. For example, where $g_2 = 3$, leaf nodes 2 and 3 should be merged together, and node 2 is chosen as representative of the new sub-tree and the values of the corresponding $m_2$ and $n_2$ are updated.

In Figure 3.6(b), after leaf node 2 and 3 are merged (as indicated by the thick lines), this sub-tree can be merged with leaf node 1 (as indicated by the thin lines). Having set $g_1 = 2$, means that node 1 is to be merged with the sub-tree with node 2 being its representative. After the merge, node 1 will become representative for the new sub-tree. Figure 3.6(b) also shows two merge operations for nodes 4 with 5, and for nodes 6 with 7.
Figure 3.6 A simple example of tree reconstruction from a global quartet weight matrix.

Figure 3.6(c) depicts a situation just before the last edge (the thin line) is added. The final tree must be the original tree from which the global weight matrix is generated. This is because when considering the merging of two sub-trees, the desired value \(d_{ij}\) is calculated using the same equation used to calculate each entry...
value $m_{ij}$ of the matrix according to the given topology. If the two sub-trees are indeed connected by a common ancestor in the original tree, then $d_{ij} = m_{ij}$, and the confidence value is equal to 1.0, i.e.; $c_{ij} = m_{ij} / d_{ij} = 1.0$.

A pseudocode of the new algorithm is given below:

Initialize n sub-trees: For row i:

set $n_i = 1$

set $m_i = 0$

While (number of sub-trees > 1) do: (main loop)

For each row $i$, (1 ≤ $i$ < Total number of sub-trees) do:

$g_i = 0$

for each column $j$, (1 < $j$ ≤ Total number of sub-trees & $j < i$) do:

Calculate $C_{ij}$

If ($C_{ij} = 1$), then $g_i = j$

Check $g_i$ for a nonzero

If (nonzero: $g_i = j$) then

Add an internal node to merge node $i$ and node $j$

Update $m_i$ and $m_j$:

$$m_j = m_j + \left( \frac{n_i}{2} \right)$$

$$m_i = m_i + \left( \frac{n_j}{2} \right)$$

Choose a representative node for the new sub-tree: (i)

Update $n_i$: $n_i = n_i + n_j$
In the previous example, each sub-tree has a representative and the values associated with the representatives are simply used and updated during the computation. Thus for each sub-tree, it requires at most $O(n)$ steps to find another sub-tree to merge with, providing such a tree exists. There is a total of $n$ sub-trees originally. Therefore, it is easy to verify that the total computational cost of this modified algorithm is also $O(n^2)$.

### 3.7. Summary

In this chapter, a one-to-one mapping procedure between a given tree topology and its global quartet weight matrix is proved. The global quartet weight matrix accumulates the quartet weights from all possible quartets for a given set of sequences. Two algorithms have been discussed; the first one is to generate a quartet weight matrix from a given tree topology, and second one is to generate a tree topology from a given quartet weight matrix.

Based on this one-to-one mapping procedure, the next chapter will present a novel quartet-based algorithm called QB. Like the quartet puzzling, this algorithm rebuilds the evolutionary tree using quartet weights, however, it takes a global view by looking at all possible quartet trees when merging a pair of sub-trees. It is a progressively clustering algorithm that recursively merges one pair of sub-trees at each merge step based on global statistics provided by a symmetric quartet weight matrix. Also, additional techniques will be introduced to deal with quartet errors that may generate misleading results.
This chapter introduces a novel quartet-based algorithm (QB) which is based on the one-to-one mapping between a given tree topology and its global quartet weight matrix as described in the previous chapter. This algorithm incorporates a number of new procedures to deal with inaccuracies in the sets of quartets. These procedures include the calculation of an average confidence value, quartet correction, and the critical points.

4.1. Introduction

When all the quartet topologies are accurate, recombination is trivial. The task of assembling an overall phylogeny can be done in $O(n^5)$ time [111], where $n$ is the number of species. In practice, it is very difficult to obtain correctly resolved quartet
trees using existing methods [2, 78]. When some of the inferred quartet topologies are ambiguous or even erroneous, the problem of reconstructing phylogenies becomes simultaneously more interesting and complex, thereby also increasing the computational time. This is typically the case when reconstructing a phylogeny from short sequences.

Most of the quartet-based methods are focus on solving the maximum quartet consistency problem (MQC). They try to find an evolutionary tree from the input of sequences that maximizes the number of quartets consistent with the tree topology. The same problem could be treated as a minimum quartet error (MQE) one, i.e. finding an evolutionary tree that minimizes the number of quartets not included in the tree topology. However, while this problem cannot be solved by counting the number of quartets that agree with the tree topology, doing so might serve to test the consistency of the quartet set, just like likelihood mapping.

An algorithm for reconstructing the original tree from its generated global quartet weight matrix was discussed in the previous chapter. The same algorithm may be used to construct an evolutionary tree for a given set of \( n \) sequences if all the associated quartets are fully and correctly resolved. But, this is a theoretical ideal case. The quartets are rarely all fully and correctly resolved and the global quartet weight matrix generated from such erroneous quartet weights is inaccurate. Since the quartet can not be fully resolved, the quartet-based algorithm for tree topology reconstruction described in the previous chapter cannot be used directly without additional processes to deal with the inaccuracy of the quartet weight matrices. While these procedures might be expensive, they are vital if the main goal is to obtain more accurate results.
4.2. Average Confidence Value

The confidence value is a measure of closeness between two subtrees. Thus, to reconstruct the tree from its generated quartet weight matrix, one leaf node is selected as a representative for each sub-tree. Since the entry values of the global weight matrix are no longer ideal, different node pairs, one from each of the two subtrees, may produce different confidence values. It is impossible to determine which of these is the true value that will lead to the recovery of the correct topology.

A simple way to alleviate this problem is to consider every pair of leaf nodes. So, instead of calculating the confidence value for the representative node pair, the confidence values for every leaf node pair are calculated and the average of these values is used as a confidence value \( c_{ij} \) for each pair of sub-trees. For example, if one sub-tree has \( n_a \) leaf nodes in it and the other has \( n_b \) nodes, the total number of node pairs, one from each sub-tree, will be \( n_a \times n_b \). The average confidence value for the two sub-trees will be the sum of the confidence values for all the node pairs divided by \( n_a \times n_b \) as in the following equation:

\[
C_{ab} = \frac{\sum_{i} \sum_{j} c_{ij}}{n_a \times n_b}
\]  

(4.1)

where \( a \) and \( b \) are two sub-trees with \( n_a \) and \( n_b \) leaf nodes, respectively.

Since the entry values of the quartet weight matrix are inaccurate, the average confidence value \( \bar{c}_{ij} \) equal to one might not be obtained for a pair of sub-trees during the computation. In addition to the three variables \( g_i \), \( m_i \) and \( n_i \) associated with each sub-tree as depicted in Figure 4.1, a new variable \( \bar{c}_i \) is needed for recording the
highest average confidence value for sub-tree \( i \) in regard to another sub-tree \( j \) in the case of \( i < j \) and \( i \neq j \).

The average confidence value \( \overline{c}_{ij} \) is calculated at each step for every possible pair of sub-trees and the highest value for each sub-tree \( i \) is stored. These values are then compared and the two sub-trees which have the highest average confidence value are merged.

Calculating the average confidence value is a simple procedure to make sure that every leaf node in the two sub-trees contributes in the merging process.

### 4.3. Quartet Weight Correction

There remains another big challenge in the quartet set: Each quartet has three resolved unrooted topologies that are used to distinguish a binary relationship between the four species. In the ideal case, one of these topologies has a weight equal to one, and the other two have zero weights. Since it is hard to get all the quartets resolved in practice, an additional step is needed to restore the associated entries in the matrix to their “true” values after two sub-trees have been merged, i.e. the quartet weights are modified according to the currently reconstructed sub-trees and the quartet weight matrix adjusted accordingly.

After each merge the weights of all those quartets containing four nodes \{\( i, j, p, q \)\} need to be corrected to \((ij/pq, 1.0)\), where \( i \) is a leaf node in one merged sub-tree, \( j \) is a leaf node in the other, and \( p \) and \( q \) are leaf nodes not represented in either of the two sub-trees. The correction process carried out by adding the quartet weights of the two quartet topology \((iq/pj)\) and \((ip/qj)\) to the weight of \((ij/pq)\), and then set their weight to zero. In fact, the correction goes directly into the matrix. Our experimental results show that if the quartet weights are not corrected, the distributed errors may
significantly affect the correctness of the decision made during the following merge steps.

After these two modifications, the QB algorithm will look like the following:

For (each row i) do:
\[ n_i = 1 \]
\[ m_i = 0 \]

While (number of sub-trees > 1) do:
{
  For sub-tree i: (1 < i < total number of sub-trees) do:
    \[ g_i = 0 \]
    \[ c_i = 0 \]

  For sub-tree j: (i < j < total number of sub-trees) do:
    Calculate \( c_{ij} \)
    If \( (\bar{c}_{ij} > \bar{c}_i) \), then:
    \[ set ~ \bar{c}_i = \bar{c}_{ij} \]
    \[ set ~ g_i = j \]

  Find the largest \( \bar{c}_a \) and associates \( a = b \)
  Add an internal node to merge node a and node b
  For (each row associated with node i in sub-tree a) do:
    \[ m_i = m_i + \left( \frac{n_b}{2} \right) \]
  For (each row associated with node j in sub-tree b) do:
    \[ m_j = m_j + \left( \frac{n_a}{2} \right) \]

  Choose i as a representative node for the new sub-tree
  Update \( n_i: n_i = n_i + n \)
}

Quartet weight correction:

Update QWM by correcting the weight of those quartets containing \((i,j,p,q)\)
to \((ij|pq, 1.0)\) where i and j are leaves in sub-trees a and b
While quartet weight correction is an expensive process, it can significantly improve the accuracy of the results. In our implementation, it is not the quartet weight itself which is corrected; instead the corresponding entry value in the matrix is altered directly to reduce the computational overhead.

4.4. Example

Figure 4.1 depicts a simple example showing how our QB algorithm works for tree reconstruction from an inaccurate global quartet weight matrix. This example uses the tree previously depicted in Figure 3.4(a). The weights for all the associated quartets are obtained using the first part of TREE-PUZZLE (version 5.2) [89]. The global quartet weight matrix is then generated from these quartets, as shown in Figure 4.1(a). At the beginning, each leaf node is considered as a sub-tree, where \(m_i\) and \(n_i\) are set correspondingly. For subtree \(i\), all the average confidence values \(\bar{c}_{ij}\) for \(j>i\) are calculated, and the highest one is selected and stored in \(\bar{c}_i\) and the corresponding index stored in \(g_i\). Then the values in column \(\bar{c}_i\) are compared to find the highest confidence value at the current step and the two sub-trees are merged to form a larger sub-tree. In Figure 4.1(a), the largest value for \(\bar{c}_2\) is equal to 0.94 and the corresponding sub-tree index \(g_2 = 3\). According to this, sub-tree 2 will be merged with sub-tree 3 as indicated by the thin lines in the Figure. The smallest index from these two sub-trees becomes the index for the new sub-tree (in this case: 2). After that, \(m_i\) is updated for each leaf node as in equation (3.3) and \(n_i\) is updated to \(n_i = n_i + n_j\) for the new sub-tree.
Figure 4.1 An example of tree reconstruction from an inaccurate global quartet weight matrix
Before re-calcultating the average confidence values for the next merge, the quartet weights need to be corrected in order to restore the “true” values in the corresponding entries according to the current merge. As shown in Figure 4.1(b), the value in row 2, column 3 is restored to 10.0. Simple, we add the quartet weights of all related quartet (i.e. \(2x/3y\), where \(x\) and \(y\) could be any other sequences except 2 and 3) to the value in row 2, column 3 and set their weights to zero. Other values in certain other entries are also changed or “corrected” accordingly, e.g. the value in row 1, column 3 is restored to 5.99. After completing the weight correction procedure, the average confidence values are re-calculated for all the sub-tree pairs as a prerequisite for the next merge step.

This process continues until there are three sub-trees left, when one additional internal node is introduced to connect them together, as shown in Figure 4.1(d). It is easy to see that in this simple example the true tree has been reconstructed in terms of its topology.

### 4.5. Computational Cost

With the above two modifications the algorithm is able to deal with inaccurate quartet weights. At each merge step, the three major contributors to the total computational cost are:

1) The updating of \(m_i\) values.

2) The calculation of average confidence values for each sub-tree to find the highest one.

3) The quartet weight correction.

Since we have \(n \times n\) matrix, it takes \(O(n^2)\) to update \(m_i\) values. Additional loop is needed to calculate the Average confidence value and that increase the total
computation cost to $O(n^3)$. However, each quartet weight is corrected once and only once during the entire computation. Since the total number of quartets for a given set of $n$ sequences is $\binom{n}{4}$, the total cost for the algorithm will be $O(n^4)$. It should be noted that this cost is much lower than the cost for computing quartet weights using the maximum likelihood criteria, which requires $O(sn^4)$ operations, where $s$ is the length of the sequences, usually a few hundreds to thousands.

4.6. Critical Points and Multiple Trees

Many techniques have been proposed to tolerate quartet errors, and quartet correction is one of them. While these techniques improve the accuracy of the obtained results, it is impossible to recover the quartet errors completely. Therefore, they need to be dealt with in another way: by considering any strongly supported relationship suggested between the quartets.

In the previous algorithm, a simple search technique was used by progressively merging the two sub-trees with the highest confidence value to eventually arrive at one single tree. Since the matrix is not accurate, merging the two sub-trees that have the highest confidence value may not always be the right decision. Once the highest confidence value $\bar{c}_{ij}$ is obtained, a check is carried out to see whether there is another sub-tree $k$ associated with one of the two sub-trees $i$ and $j$ that also has a reasonably high confidence value. Figure 4.2 explains this by example. The closeness is defined according to the following condition:

$$\bar{c}_{jk} \text{ (or } \bar{c}_{ki}) \geq \alpha \bar{c}_{ij}, \text{ or } \bar{c}_{jk} \text{ (or } \bar{c}_{kj}) \geq \alpha \bar{c}_{ij}$$

where $\alpha$ is a threshold smaller than, but close to one.
(b) Take all three possibilities and continue with three different trees.

Figure 4.2 An example of a common critical point, where the confidence values for three possible merges are virtually identical.
To further alleviate the effect of quartet errors, the concept of critical points is introduced. When such sub-tree $k$ (that has a reasonably high confidence value) exists, it poses a difficulty for determining which two sub-trees should be merged since they all have reasonably high confidence values. A critical point is said to be encountered when such an ambiguous situation occurs during the computation. In this situation, it is difficult to define which pattern should be considered for the next merge. At each of these critical points three different super quartet trees with four sub-trees $i, j, k, l$ and the rest as its four super nodes at different places come into question. The problem is which one will lead to the correct tree topology.

It is hard to decide which pattern is the right one. A possible solution is to use popular objective function such as maximum likelihood criterion to select one of the patterns. Another possible solution would be to keep all the three different patterns at the critical point where the ambiguity occurs. However, to test the quartet-based concept, we decide to go with the second solution. Therefore, a limited number of trees could be generated for posterior analysis. Experimental results show that the probability is very high for the true tree to be among this limited number of generated trees. This opens a new research direction for further investigation of more efficient search algorithms for phylogenetic inference. In practice, a single tree may not be possible to correctly describe the relationships among species, and it is also often desirable to construct a number of trees for a posterior analysis. Figure 4.3 shows the workflow of the QB algorithm for multiple tree reconstruction.
Figure 4.3 QB workflow graph
In practice, not all of the three different super-quartet patterns can be kept at every critical point. Without a limit on the number of trees to be reconstructed, the worst case, where every merge step is a critical point, might end up generating about $3^n$ different trees, where $n$ is the number of merge steps. Thus, a stage parameter $s$ is used to limit the total number of trees. Each time a critical point is met; one or two extra trees are generated until $s$ stages are encountered for each tree. This will limit the maximum number of trees to be generated to $3^s$, so that if $s=4$, for example, the number of trees to be generated will be less than 81.

4.7. Invariable Threshold $\alpha$

In QB, multiple possible trees are constructed for a given set of sequences. The number of trees can be limited by prefixing the total number of stages. The question remains of whether the number of trees generated in the same number of stages can be further reduced while maintaining the high probability of the correct tree being among them, bearing in mind that extra trees are only generated when a critical point is encountered, which is identified by using a fixed threshold $\alpha$.

There is a problem associated with the fixed threshold in that it also identifies certain critical points that are unnecessary. When two sub-trees have a confidence value of 1.00, this value should be considered very high and in itself sufficient reason to merge the two sub-trees without ambiguity. However, if the threshold $\alpha$ were set to 0.85, a critical point might appear here and generate an extra tree or two, when either of these two sub-trees has a confidence value of above 0.84 with another sub-tree. When unnecessary critical points are created and the limit for the number of generated trees has been reached in the first few merge steps, this could result in skipping the generation of the true tree in later merge steps.
One way to alleviate this problem is to make the threshold $\alpha$ variable, which is adjusted in respect to the confidence values, e.g. $\alpha$ will be set higher when the confidence value is higher. A simple linear function was adopted to meet this requirement and is depicted graphically in Figure 4.4.

In the diagram, threshold $\alpha$ is depicted as a function of the confidence value $c$. When the confidence value is smaller than $c_0$, $\alpha$ remains a constant. When the confidence value is larger than $c_0$, $\alpha$ increases linearly with the rise in the confidence value.

![Figure 4.4 Threshold $\alpha$ as a function of confidence value $c$](image)

### 4.8. Summery

In this chapter, we introduced a new QB algorithm with several techniques which are able to deal with inaccurate quartet trees. These special techniques include: the average confidence value, quartet weight correction, and critical points. While
these techniques might be expensive, they are necessary to obtain more accurate results.

Calculating the average confidence value is a simple procedure to make sure that every leaf node in the two sub-trees contributes in the merging process. The quartet weight correction used to restore the associated entries in the matrix to their “true” values, according to the currently reconstructed sub-trees.

Furthermore; the concept of critical point is explained. When there is more than one possible merges during the computations, all have a reasonable confidence values, it is difficulty for determining which two sub-trees should be merged. A critical point is said to be encountered when such an ambiguous situation occurs. Therefore, limited number of tree could be generated, as it is often desirable to construct a number of trees for a posterior analysis. The number of generated tree could be quite large, and thus a parameter is suggested to limit the total number of trees.

In order to test the performance of the new algorithm introduced here, the next chapter presents an experimental results using simulated data.
In this chapter, we present experimental results using simulated data sets to compare the performance of the QB with other existing methods.

5.1. Introduction

The QB algorithm introduced in the previous chapters reconstructs a phylogenetic tree in three steps:

1) Quartet weights step: the weights for all of the \( \binom{n}{4} \) possible quartets are calculated using maximum likelihood criterion. Note that other criteria, such as parsimony and distance, could be used.
2) Matrix generation step: the Quartet weights are accumulated into the global quartet weight matrix of size $n \times n$, where $n$ is the number of sequences.

3) Tree reconstruction step: based on the one-to-one mapping, the tree inferred from the global quartet weight matrix.

Three techniques are included in the third step to deal with quartet errors. These techniques includes: Average confidence value, quartet weight correction, and critical points. Multiple trees could be generated based on the level of error in the quartet set.

In the following sections, the performance of QB was analysed and compared in term of accuracy with the current existing programs.

5.2. Data Set

In the experiments, the benchmarks developed by Ranwez and Gascuel for testing and comparing different phylogeny methods [81] were used. They are similar to the ones previously developed and used by Kumar [63] and Gascuel [38]. This benchmark contains six model trees, each consists of 12 leaf nodes, were used to generate test data sets for various situations. Three of the model trees, named AA, AB and BB, were molecular clock-like, while the other three, named CC, CD and DD, presented varying substitution rates among the lineages. Four evolutionary rates were represented, ranging from low, medium, fast to very fast, for which the maximum pair-wise divergence (MD) was from 0.1 to about 2.0 substitutions per site. From these model trees and varying evolutionary conditions, a total of 48,000
test data sets were generated using Seq-Gen. A more detailed description of the benchmarks and the test data sets is given in [81].

5.3. Methods and Parameters

The threshold $\alpha$ was set to 0.85. The number of stages was set to four values: Firstly, to zero (QB0) to allow only a single final tree; secondly, to 4 stages (QB4) to allow a maximum of 81 trees; thirdly, to 5 stages (QB5) to allow a maximum of 243 trees; and finally, the parameter was set to infinity (QBINF) to allow all possible trees. The average number of trees generated per data set and the percentage of correctly inferred trees was measured. Note that a correctly inferred tree here means the correct tree topology was found among a number of trees generated for a given data set.

In addition to showing the results for QB, the Tables also show the percentages of correctly inferred trees for the same data set achieved by other programs: DNAPARS (the parsimony program (version 3.66) of the PHYLIP package) [33], BIONJ (an improved neighbour-joining method [38]), and FASTDNAML (the maximum-likelihood program version 1.2) [75]. The data on their results was simply copied for the purpose of comparison [81], while PHYML [44], one of the currently most accurate and well-known programming packages, was run on the same data set.

5.4. Results

The results are depicted in two Tables: Table 5.1 for DNA sequences with a length of 300 and Table 5.2 for DNA sequences with a length of 600. The quotient $x/y$ in the Tables denotes the percentage of correctly inferred trees over the average
number of trees generated per data set. The figures are all rounded to their nearest integers.

Table 5.1 Experimental results for sequence length 300

<table>
<thead>
<tr>
<th></th>
<th>AA</th>
<th>BB</th>
<th>AB</th>
<th>CC</th>
<th>DD</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MD=0.1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNA/PARS</td>
<td>19</td>
<td>16</td>
<td>15</td>
<td>13</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>BIONJ</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>23</td>
<td>20</td>
<td>21</td>
<td>16</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>PHYML</td>
<td>23</td>
<td>22</td>
<td>22</td>
<td>15</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>QB0</td>
<td>22</td>
<td>11</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>QB4</td>
<td>63/57</td>
<td>49/53</td>
<td>52/54</td>
<td>56/45</td>
<td>61/47</td>
<td>59/44</td>
</tr>
<tr>
<td>QB5</td>
<td>68/94</td>
<td>57/89</td>
<td>58/82</td>
<td>58/68</td>
<td>61/69</td>
<td>61/65</td>
</tr>
<tr>
<td>QBINF</td>
<td>70/124</td>
<td>58/117</td>
<td>59/120</td>
<td>58/86</td>
<td>64/88</td>
<td>62/80</td>
</tr>
<tr>
<td><strong>MD=0.3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNA/PARS</td>
<td>28</td>
<td>36</td>
<td>26</td>
<td>46</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td>BIONJ</td>
<td>33</td>
<td>34</td>
<td>34</td>
<td>56</td>
<td>57</td>
<td>56</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>58</td>
<td>53</td>
<td>54</td>
<td>70</td>
<td>68</td>
<td>69</td>
</tr>
<tr>
<td>PHYML</td>
<td>58</td>
<td>56</td>
<td>57</td>
<td>67</td>
<td>67</td>
<td>69</td>
</tr>
<tr>
<td>QB0</td>
<td>44</td>
<td>30</td>
<td>21</td>
<td>37</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>QB4</td>
<td>87/44</td>
<td>78/40</td>
<td>79/46</td>
<td>88/24</td>
<td>91/26</td>
<td>89/26</td>
</tr>
<tr>
<td>QB5</td>
<td>91/61</td>
<td>81/55</td>
<td>83/60</td>
<td>89/28</td>
<td>92/28</td>
<td>90/31</td>
</tr>
<tr>
<td>QBINF</td>
<td>92/66</td>
<td>82/61</td>
<td>83/68</td>
<td>89/29</td>
<td>92/28</td>
<td>90/32</td>
</tr>
<tr>
<td><strong>MD=1.0</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNA/PARS</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>BIONJ</td>
<td>22</td>
<td>20</td>
<td>21</td>
<td>62</td>
<td>63</td>
<td>65</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>44</td>
<td>36</td>
<td>37</td>
<td>82</td>
<td>83</td>
<td>81</td>
</tr>
<tr>
<td>PHYML</td>
<td>39</td>
<td>31</td>
<td>34</td>
<td>73</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>QB0</td>
<td>43</td>
<td>19</td>
<td>13</td>
<td>41</td>
<td>39</td>
<td>32</td>
</tr>
<tr>
<td>QB4</td>
<td>86/54</td>
<td>67/51</td>
<td>65/56</td>
<td>88/31</td>
<td>90/31</td>
<td>90/32</td>
</tr>
<tr>
<td>QB5</td>
<td>91/79</td>
<td>72/79</td>
<td>72/87</td>
<td>89/38</td>
<td>94/37</td>
<td>92/39</td>
</tr>
<tr>
<td>QBINF</td>
<td>91/89</td>
<td>74/90</td>
<td>72/101</td>
<td>90/41</td>
<td>95/37</td>
<td>92/42</td>
</tr>
<tr>
<td><strong>MD=2.0</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNA/PARS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BIONJ</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>29</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>59</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>PHYML</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>QB0</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>QB4</td>
<td>45/71</td>
<td>19/73</td>
<td>20/72</td>
<td>45/59</td>
<td>65/57</td>
<td>58/59</td>
</tr>
<tr>
<td>QB5</td>
<td>56/144</td>
<td>26/158</td>
<td>29/167</td>
<td>53/100</td>
<td>70/94</td>
<td>65/100</td>
</tr>
<tr>
<td>QBINF</td>
<td>60/207</td>
<td>31/344</td>
<td>34/294</td>
<td>55/129</td>
<td>71/124</td>
<td>67/130</td>
</tr>
</tbody>
</table>

In Table 5.1, the basic version of QB algorithm, i.e. QB0 for which the number of stages was set to 0 (to produce a single tree), did not perform as well as the other three methods except for model tree AA. For this tree some of the results obtained by
Chapter 5: Performance Evaluation

QB0 were better than those achieved by BIONJ and DNAPARS, and some were comparable to those achieved by PHYML and FASTDNAML.

**Table 5.2** Experimental results for sequence length 600

<table>
<thead>
<tr>
<th></th>
<th>AA</th>
<th>BB</th>
<th>AB</th>
<th>CC</th>
<th>DD</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MD=0.1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNAPARS</td>
<td>57</td>
<td>53</td>
<td>52</td>
<td>54</td>
<td>57</td>
<td>56</td>
</tr>
<tr>
<td>BIONJ</td>
<td>52</td>
<td>50</td>
<td>54</td>
<td>60</td>
<td>57</td>
<td>56</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>69</td>
<td>63</td>
<td>65</td>
<td>66</td>
<td>61</td>
<td>62</td>
</tr>
<tr>
<td>PHYML</td>
<td>71</td>
<td>65</td>
<td>66</td>
<td>65</td>
<td>61</td>
<td>62</td>
</tr>
<tr>
<td>QB0</td>
<td>56</td>
<td>43</td>
<td>36</td>
<td>46</td>
<td>39</td>
<td>37</td>
</tr>
<tr>
<td>QB4</td>
<td>94/25</td>
<td>88/25</td>
<td>91/24</td>
<td>92/16</td>
<td>93/17</td>
<td>93/19</td>
</tr>
<tr>
<td>QB5</td>
<td>95/29</td>
<td>89/28</td>
<td>92/27</td>
<td>92/17</td>
<td>93/18</td>
<td>93/19</td>
</tr>
<tr>
<td>QBINF</td>
<td>95/29</td>
<td>89/29</td>
<td>92/29</td>
<td>92/18</td>
<td>93/18</td>
<td>93/19</td>
</tr>
<tr>
<td><strong>MD=0.3</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNAPARS</td>
<td>65</td>
<td>77</td>
<td>69</td>
<td>83</td>
<td>86</td>
<td>88</td>
</tr>
<tr>
<td>BIONJ</td>
<td>76</td>
<td>77</td>
<td>81</td>
<td>90</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>93</td>
<td>87</td>
<td>91</td>
<td>94</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>PHYML</td>
<td>93</td>
<td>87</td>
<td>92</td>
<td>96</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>QB0</td>
<td>83</td>
<td>74</td>
<td>64</td>
<td>83</td>
<td>84</td>
<td>78</td>
</tr>
<tr>
<td>QB4</td>
<td>99/14</td>
<td>97/16</td>
<td>99/16</td>
<td>99/5</td>
<td>99/7</td>
<td>99/6</td>
</tr>
<tr>
<td>QB5</td>
<td>99/14</td>
<td>97/16</td>
<td>99/16</td>
<td>99/5</td>
<td>99/8</td>
<td>99/6</td>
</tr>
<tr>
<td>QBINF</td>
<td>99/14</td>
<td>97/16</td>
<td>99/16</td>
<td>99/5</td>
<td>99/8</td>
<td>99/6</td>
</tr>
<tr>
<td><strong>MD=1.0</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNAPARS</td>
<td>22</td>
<td>33</td>
<td>25</td>
<td>17</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>BIONJ</td>
<td>62</td>
<td>62</td>
<td>65</td>
<td>92</td>
<td>92</td>
<td>93</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>85</td>
<td>77</td>
<td>82</td>
<td>99</td>
<td>99</td>
<td>98</td>
</tr>
<tr>
<td>PHYML</td>
<td>82</td>
<td>76</td>
<td>78</td>
<td>96</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>QB0</td>
<td>77</td>
<td>63</td>
<td>49</td>
<td>87</td>
<td>82</td>
<td>78</td>
</tr>
<tr>
<td>QB4</td>
<td>99/24</td>
<td>95/26</td>
<td>96/26</td>
<td>99/8</td>
<td>100/13</td>
<td>99/11</td>
</tr>
<tr>
<td>QB5</td>
<td>100/25</td>
<td>96/28</td>
<td>97/30</td>
<td>99/8</td>
<td>100/13</td>
<td>99/11</td>
</tr>
<tr>
<td>QBINF</td>
<td>100/26</td>
<td>96/28</td>
<td>97/30</td>
<td>99/17</td>
<td>100/13</td>
<td>99/11</td>
</tr>
<tr>
<td><strong>MD=2.0</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNAPARS</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BIONJ</td>
<td>22</td>
<td>18</td>
<td>17</td>
<td>76</td>
<td>71</td>
<td>73</td>
</tr>
<tr>
<td>FASTDNAML</td>
<td>35</td>
<td>26</td>
<td>28</td>
<td>93</td>
<td>91</td>
<td>94</td>
</tr>
<tr>
<td>PHYML</td>
<td>29</td>
<td>18</td>
<td>20</td>
<td>66</td>
<td>64</td>
<td>68</td>
</tr>
<tr>
<td>QB0</td>
<td>42</td>
<td>20</td>
<td>12</td>
<td>48</td>
<td>42</td>
<td>40</td>
</tr>
<tr>
<td>QB4</td>
<td>86/50</td>
<td>63/52</td>
<td>64/56</td>
<td>89/40</td>
<td>93/33</td>
<td>93/36</td>
</tr>
<tr>
<td>QB5</td>
<td>91/74</td>
<td>67/84</td>
<td>72/87</td>
<td>92/57</td>
<td>94/40</td>
<td>94/45</td>
</tr>
<tr>
<td>QBINF</td>
<td>92/79</td>
<td>69/106</td>
<td>73/103</td>
<td>92/62</td>
<td>94/40</td>
<td>94/48</td>
</tr>
</tbody>
</table>

In Table 5.2, the accuracy of QB0 increased, but was still not as good as expected. It seems that sequence length had a good impact on the results, perhaps because long sequences give more information than short sequences. On the other
hand, the accuracy of quartet-based algorithms is dependent on the quality of quartet weights (or quartet trees generated). Even though the globally integrated information from all the quartets is used at each step to determine the merge operation, the basic version of QB algorithm is still sensitive to the errors of quartet trees. This is, however, the case for all quartet-based methods.

The two Tables 5.1 and 5.2 also show the results from using QB over 4, 5, and unlimited stages. It was noted that when the number of trees was allowed to increase, the percentage of correctly inferred trees among the generated trees was higher with QB than all other methods. This was the case in all the categories and in certain cases was much higher. For sequence lengths being 300 and $\text{MD} \approx 0.1$, for example, the average number of trees constructed using QB4 was only around 50. This number, or even the maximum number of trees allowed to be constructed, is tiny when compared with the total number of possible trees for a set of 12 sequences. However, the average percentage of correct trees from among the constructed trees from all six model trees was about 3.5 times higher than in using DNAPARS, or BIONJ, and about 3 times the percentage obtained using FASTDNAML. For QB5 and QBINF the figures were even higher.

This comparison seems unfair since the QB method builds a number of trees for each data set whereas the other methods construct only one tree. However, all methods for reconstructing evolutionary trees are subjected to errors and no existing phylogenetic method can guarantee better results by constructing only a very limited number of trees. While the whole tree might not be correct, some part of the tree may look reasonable.
The significant results obtained from these experiments verify the effectiveness of the quartet-based method (QB) and support the issue of multiple tree generation for posterior analysis.

5.5. Invariable Threshold $\alpha$

The previous experiments were repeated to compare QB with variable thresholds to QB with a fixed threshold. The number of stages was set to 4 (QB4) to allow a maximum of 81 trees to be generated for each test data set. The threshold $\alpha$ was set to 0.85 in the fixed threshold algorithm called QB4 (a const). In the variable threshold algorithm, which is called QB4, the initial threshold $\alpha_0$ was set to 0.85 and the initial confidence value $c_0$ was set to 0.90 (0.85, 0.9). Some of the experimental results for DNA sequences with a length of 300 are presented in Table 5.3.

<table>
<thead>
<tr>
<th>MD</th>
<th></th>
<th>AA</th>
<th>BB</th>
<th>AB</th>
<th>CC</th>
<th>DD</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>QB4($\alpha$ const)</td>
<td>63/57</td>
<td>49/53</td>
<td>52/54</td>
<td>56/45</td>
<td>61/47</td>
<td>59/44</td>
</tr>
<tr>
<td></td>
<td>QB4(0.85, 0.90)</td>
<td>63/53</td>
<td>50/50</td>
<td>53/50</td>
<td>56/42</td>
<td>61/46</td>
<td>59/42</td>
</tr>
<tr>
<td>0.3</td>
<td>QB4($\alpha$ const)</td>
<td>87/44</td>
<td>78/40</td>
<td>79/46</td>
<td>8/24</td>
<td>91/26</td>
<td>89/26</td>
</tr>
<tr>
<td></td>
<td>QB4(0.85, 0.90)</td>
<td>88/39</td>
<td>79/37</td>
<td>80/40</td>
<td>8/22</td>
<td>91/24</td>
<td>89/24</td>
</tr>
<tr>
<td>1.0</td>
<td>QB4($\alpha$ const)</td>
<td>86/54</td>
<td>67/51</td>
<td>65/56</td>
<td>88/31</td>
<td>90/31</td>
<td>90/32</td>
</tr>
<tr>
<td></td>
<td>QB4(0.85, 0.90)</td>
<td>87/51</td>
<td>67/50</td>
<td>65/54</td>
<td>88/27</td>
<td>90/29</td>
<td>90/29</td>
</tr>
<tr>
<td>2.0</td>
<td>QB4($\alpha$ const)</td>
<td>45/71</td>
<td>19/73</td>
<td>20/72</td>
<td>45/59</td>
<td>65/57</td>
<td>58/59</td>
</tr>
<tr>
<td></td>
<td>QB4(0.85, 0.90)</td>
<td>45/70</td>
<td>19/72</td>
<td>20/72</td>
<td>47/59</td>
<td>65/56</td>
<td>58/57</td>
</tr>
</tbody>
</table>
In Table 5.3, it can be seen that using the variable threshold ($\alpha$) has a significant impact on reducing the number of obtained trees and that the reduction is about 5.9 percent on average. Further more, by using the variable threshold, the percentage of correct trees included in the generated trees was not decreased but, in fact, increased somewhat in certain cases. These results were consistent with the expectations.

5.6. QB with Maximum Likelihood (QB+ML)

The number of generated trees was reduced with high confidence of retaining the true tree among them. This raises another important issue, namely, how to identify the correct tree from among them via posterior analysis?

The maximum-likelihood (ML) method allows to the estimation and comparison of the likelihood of different evolutionary hypotheses (each comprising a model of the tree and substitution process), given the observed data [31]. It involves the definition of a labelled binary tree followed by optimization of the estimates of the lengths of the edges on that tree. Experimental results often show that the ML method outperforms other methods, such as maximum parsimony and minimal evolution. As a result, the ML method is widely used in a variety of applications.

It is well known that high computational complexity is a problem associated with ML methods and most of them implement heuristic local search optimization techniques to reduce this. Since the search space had been successfully limited to a small number of trees, the decision was made to implement QBML (which is QB + Maximum Likelihood). In a straightforward process, QB would be used to reconstruct a limited number of trees (the previous experimental results show that the probability of the correct tree being among the generated trees is very high). Then,
the likelihood values of these generated trees would be calculated and the ones with the highest likelihood values are chosen.

5.6.1. Experiment on 12 Sequences

Using the results from the previous experiments on the 12 sequences of length 300, the likelihood values of the generated trees was calculated and just a few best trees (1, 2, 3, 4, and 5) with the highest likelihood values were chosen. If there was only a single maximum likelihood point for a given phylogenetic tree, using ML criterion should allow the identification of the true phylogenetic tree among these trees.

In Table 5.4, QB4+ML-j denotes the use of QB with a fixed threshold \(\alpha\) set to 0.85 and 4 stages to construct multiple trees from which \(j\) best trees were chosen under the ML criterion. These best trees were then compared to the model tree using the Robinson and Foulds topological distance (RF) method [83]. In choosing \(j\) best trees, the corresponding columns in the Table shows the percentages of correct trees among these \(j\) best trees for six different model trees. The figures have all been rounded to their nearest integers.

When a single tree with the largest likelihood value was selected (QB4+ML-1), the results were much better than in BIONJ and DNAPARS, and similar to those obtained using FASTDNAML and PHYML, sometimes even better. It can be seen that QBML performs well in comparison to the other methods. Also, when allowing the selection of more than one tree, the performance of QBML was much better than both FASTDNAML and PHYML in almost all categories.
Table 5.4 Results using QB4 with ML for sequence data set lengths of 300

<table>
<thead>
<tr>
<th>MD</th>
<th>BIONJ</th>
<th>FASTDNAML</th>
<th>PHYML</th>
<th>QB4+ML-1</th>
<th>QB4+ML-2</th>
<th>QB4+ML-3</th>
<th>QB4+ML-4</th>
<th>QB4+ML-5</th>
<th>QB4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>18</td>
<td>23</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>20</td>
<td>21</td>
<td>16</td>
<td>18</td>
<td>18</td>
<td>32</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>22</td>
<td>23</td>
<td>16</td>
<td>18</td>
<td>18</td>
<td>35</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>32</td>
<td>33</td>
<td>32</td>
<td>32</td>
<td>33</td>
<td>40</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>34</td>
<td>35</td>
<td>34</td>
<td>33</td>
<td>34</td>
<td>41</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>63/57</td>
<td>49/53</td>
<td>52/54</td>
<td>56/45</td>
<td>61/47</td>
<td>59/44</td>
<td>23</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>53</td>
<td>54</td>
<td>70</td>
<td>68</td>
<td>69</td>
<td>69</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>56</td>
<td>57</td>
<td>67</td>
<td>67</td>
<td>70</td>
<td>69</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>32</td>
<td>42</td>
<td>76</td>
<td>81</td>
<td>77</td>
<td>58</td>
<td>46</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>42</td>
<td>44</td>
<td>82</td>
<td>88</td>
<td>83</td>
<td>61</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>46</td>
<td>48</td>
<td>83</td>
<td>89</td>
<td>84</td>
<td>61</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>49</td>
<td>49</td>
<td>83</td>
<td>89</td>
<td>85</td>
<td>62</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>86/54</td>
<td>67/51</td>
<td>65/56</td>
<td>88/31</td>
<td>90/31</td>
<td>90/32</td>
<td>45/71</td>
<td>19/73</td>
<td>20/72</td>
</tr>
<tr>
<td>0.3</td>
<td>33</td>
<td>34</td>
<td>34</td>
<td>56</td>
<td>57</td>
<td>56</td>
<td>33</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>53</td>
<td>54</td>
<td>70</td>
<td>68</td>
<td>69</td>
<td>69</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>56</td>
<td>57</td>
<td>67</td>
<td>67</td>
<td>70</td>
<td>69</td>
<td>70</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>32</td>
<td>42</td>
<td>76</td>
<td>81</td>
<td>77</td>
<td>58</td>
<td>46</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>42</td>
<td>44</td>
<td>82</td>
<td>88</td>
<td>83</td>
<td>61</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>46</td>
<td>48</td>
<td>83</td>
<td>89</td>
<td>84</td>
<td>61</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>49</td>
<td>49</td>
<td>83</td>
<td>89</td>
<td>85</td>
<td>62</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>86/54</td>
<td>67/51</td>
<td>65/56</td>
<td>88/31</td>
<td>90/31</td>
<td>90/32</td>
<td>45/71</td>
<td>19/73</td>
<td>20/72</td>
</tr>
</tbody>
</table>

The results of the average RF values calculated for each data set are depicted in figure 5.1, 5.2, 5.3 and 5.4. It can be seen that this limited number of trees is not very far from the true tree, the average RF values for QB trees are less than or similar.
to the average RF values for PHYML trees. According to this, even if the true tree is not included among the selected best trees, they are still close to the true tree.

Figure 5.1 The average RF values for the trees generated by QB4-ML and Phyml, where MD=0.1
Figure 5.2 The average RF values for the trees generated by QB4-ML and Phyml, where MD=0.3

Figure 5.3 The average RF values for the trees generated by QB4-ML and Phyml, where MD=1.0
Figure 5.4 The average RF values for the trees generated using QB4-ML and Phyml, where MD=2.0

5.6.2. Experiment on 24 Sequences

In the second experiment, a data set was used containing 24 sequences with a length of 600 based on 5000 random trees from 24 taxa. The data set was downloaded from the Montpellier laboratory of computer science [12]. These 5000 trees are different in shape and evolutionary rate. The internal branch lengths are also not equal. In this experiment a variable threshold (α) was used by setting $\alpha_0$ to 0.85 and $c_0$ to 0.90. Some of the experimental results are presented in Tables 5.6 and 5.7.

Table 5.5 Comparative results for a 24-sequences data set

<table>
<thead>
<tr>
<th></th>
<th>PHYML</th>
<th>QB3</th>
<th>QB4</th>
<th>QB5</th>
<th>QB7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27.7</td>
<td>35.2/25</td>
<td>47.5/61</td>
<td>56.5/159</td>
<td>63.1/371</td>
</tr>
</tbody>
</table>
In Table 5.6, the results produced by the QB algorithm using different numbers of stages were compared with the result of PHYML. Using PHYML on these synthetic 24 sequence DNA data sets yielded 1385 (27.7%) correct trees in terms of topology. With only 3 stages allowed, QB obtained 1760 (35.2%) correct trees by constructing an average of 25 trees for each data set. Much better results were obtained when more trees are allowed to be generated.

**Table 5.6** Comparative results using QB and ML criterion for the 24 sequences

<table>
<thead>
<tr>
<th></th>
<th>PHYML</th>
<th>QB4</th>
<th>QB4 + ML-1</th>
<th>QB4 + ML-2</th>
<th>QB4 + ML-3</th>
<th>QB4 + ML-4</th>
<th>QB4 + ML-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1385</td>
<td>2374</td>
<td>1017</td>
<td>1432</td>
<td>1695</td>
<td>1846</td>
<td>1957</td>
<td></td>
</tr>
<tr>
<td>27.7%</td>
<td>47.5/61</td>
<td>20.3%</td>
<td>28.6%</td>
<td>33.9%</td>
<td>36.9%</td>
<td>39.1%</td>
<td></td>
</tr>
<tr>
<td>QB5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2824</td>
<td>1095</td>
<td>1567</td>
<td>1877</td>
<td>2053</td>
<td>2192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56.5/169</td>
<td>22%</td>
<td>31.3%</td>
<td>37.7%</td>
<td>41.1%</td>
<td>43.8%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After multiple trees were constructed for each test set, similarly to what was done in the first experiment for the 12 sequences, a few best trees were chosen under the ML criterion. Table 5.7 shows the number of correct trees identified when choosing at most 5 best trees from those generated by QB over 4 and 5 stages. The results show that both QB4-ML-j and QB5-ML-j perform better than PHYML when a few trees are allowed to be selected.

The distance between the best selected trees (1, 2, 3, 4, 5) and the true tree using the Robinson and Foulds method was calculated. The average RF value for each test data set was calculated, and the results are shown in Table 5.8. Again, the average RF values for the QB+ML-1 trees are similar to and sometimes even less
than the average RF values for PHYML trees (where the average RF value for the trees generated by PHYML is equal to 1.66).

Table 5.7 Average RF values for the 24 sequences trees

<table>
<thead>
<tr>
<th></th>
<th>ML-1</th>
<th>ML-2</th>
<th>ML-3</th>
<th>ML-4</th>
<th>ML-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>QB4</td>
<td>1.65</td>
<td>1.78</td>
<td>1.87</td>
<td>1.95</td>
<td>2</td>
</tr>
<tr>
<td>QB5</td>
<td>1.53</td>
<td>1.66</td>
<td>1.73</td>
<td>1.82</td>
<td>1.87</td>
</tr>
</tbody>
</table>

5.7. Multiple Maximum Likelihood Points

An interesting and very important, but largely ignored question associated with the ML method, is whether there exists only a single maximum likelihood point for a given phylogenetic tree. Though Mike Steel has presented simple analytical results to argue that the ML point is not unique [99], his view so far has attracted only little attention. Till now the design of ML-based algorithms has focused mainly on reducing the computational complexity, and the accuracy of the algorithm has been judged in terms of how large an ML value it can produce for a given problem. Molecular biologists often only use values obtained from an ML-based algorithm to test phylogenetic hypotheses from genetic sequences.

The previous results show that QB+ML outperform other methods. But still unable to identify all the correct trees from a limited number of trees generated by QBN over 4 and 5 stages. Figure 5.1 show that a number of trees may have the same likelihood values. Therefore, using ML as a criterion to select one tree was impossible in this situation. This is clear evidence that certain incorrect trees can have likelihood values at least as large as that of the correct one! This shows that multiple maximum likelihood points do exist for a phylogenetic tree. Multiple trees
were built using QB+ML and the results were as expected and discussed in chapter four.

Figure 5.5  Likelihood values for 1037 trees. Values were calculated using Phyml and the highest value was achieved by more than one tree.

This important result suggests that ML criterion alone may not be sufficient to determine the true phylogeny for certain problems even if a truly globally optimal tree can be obtained under ML criterion. Therefore, it is very important in practice to construct multiple trees using different criteria for posterior analyses to ensure more accurate results.

5.8. Summary

This chapter presented experimental results using simulated data. The results show that the QB method can produce significantly more accurate results. The results also show that it is very important and thus desirable to construct a number of trees for a posterior analysis since for many hard problems in practice no existing
methods can guarantee the construction of a correct tree. The high probability of the correct tree being among the very limited number of trees constructed by the QB method is substantiated by the results, and important information on where these trees differ (i.e., critical points) is also provided.

Chapter 6 presents the results of experiments carried out on a real data set call excavate taxa.
Testing the Relationships Among Excavate Taxa

This chapter presents the results of the phylogenetic analysis of a real data set called Excavate taxa using the quartet-based method QB. To begin with, a detailed description is presented of the relationships among Excavate taxa and their relatives. This is followed by a section on the importance of collaboration between biology and statistical methods, and how molecular phylogenetic methods and morphological evidences can be effectively incorporated into the tree reconstruction.

6.1. Introduction

Since Darwin, biologists have thought that living species diverge from a small number of ancestral species, and should be traceable back to the very first organism in a universal evolutionary tree. Since that time, simulating the evolutionary history
of species has become a major task in computational biology, while building the tree of life that connects all organisms has become the major goal in biology.

Determining the phylogeny of uni-cells is one of the keys that lead to understand the evolutionary history of complex cells. Excavate taxa is one of the interesting groups within Eukaryotes, a group thought to be among the most ancient living Eukaryotes. Therefore, determining the relationships among excavates and seeing whether they form a monophyletic group is a major interest in Eukaryotic evolutionary study.

The morphologically based hypothesis postulates a relationship between Diplomonads, Retortamonas, Carpediemonas, Heterolobozea, Trimastix, Jakobids, and Malawimonas – the so called Excavate taxa [90]. Ultra-structural data suggests that excavates share a common ancestor and that they form a clade or grade [90] and are united primarily by similarities of cell structures [91]. However, there are indications that other taxa may be relatives to Excavate taxa such as Parabasalids (which lack of most excavates characteristics) share a common ancestor with Diplomonads [14]. Also, a computed molecular phylogenetic tree based on 18S RNA sequences has revealed a strong relationship between Trimastix and Oxymonads [18]. Based on combined morphological and molecular evidence for Excavate taxa and their relatives, an infra-kingdom excavate composed of the Excavate taxa, Parabasalids, Euglenozoa, and Oxymonads has been created [13].

Although the monophyletics of the entire excavate is uncertain. Some of the clusters are robust and have shown in β-, α-tubulin, and 18s RNA phylogenies. While many phylogenetic analyses have been done on Excavate taxa and their relatives [5, 18, 24, 65, 93], many relationships among excavates remain uncertain. To determine the relationships among different groups is an extremely difficult problem. Analyses
using different methods or even the same method on different datasets often produce contradictory results. The results from different studies often show conflicts between morphological and molecular phylogenetic analyses [5, 46, 92]. For example, phylogenetic analysis does not support Jakobids being monophyletic, but on the other hand biologists are uncomfortable with seeing Jakoba incarcerata separated from other Jakobids (Jakoba libera and Reclinomonas americana). Also, the position of Malawimonas is still unresolved. Structural data suggests that Malawimonas is similar to Jakobids [91], but it does not branch with Jakobids directly in many molecular phylogenetic analyses. Malawimonas also shares a lot of structural similarities with Trimastix and Oxymonads [90], and Oxymonads are classified close to Diplomonads [91]. However, recent molecular and morphological data suggests that Malawimonas and Oxymonads have a close relationship with Trimastix [18, 92]. Further more, the relationships between Diplomonads, Carpediemonas and Parabasalids are undefined precisely due to the large conflicts [46].

6.2. Materials and Methods

In this experiment, an attempt was made to include sequences from as many excavate groups as possible. These sequences were collected from previous studies. Additionally, different methods for phylogenetic analysis were adopted in order to show how the results of these methods can be effectively incorporated in QB during the merging process.

6.2.1. Data Source

Aligned amino acids data set of β-tubulin composed of 431 sites containing 21 species [102] was downloaded from Treebase [82] under the study accession number
S910 and the matrix accession numbers M1508. Treebase stores phylogenetic trees and the data matrices are used to generate them from published research papers. In order to do a comprehensive study of excavates, additional sequences were downloaded from Roger lab website [84], where extensive research has been conducted into phylogenetic analysis. Finally, a sequence from GenBank for *Streblomastix strix* was downloaded to represent Oxymonads (accession is ABC97355).

A total of 24 sequences were collected and realigned using the ClustalW multiple alignment program in Bioedit 7.0.5.3 [45]. Table 6.1 contains the species names and their groups. Figure 6.1 shows some of the excavate taxa and their relatives.

**Table 6.1** The 24 species included in the experiment

<table>
<thead>
<tr>
<th>Taxon name</th>
<th>Taxon Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trimastix pyriformis pyriformis</td>
<td>Trimastix</td>
</tr>
<tr>
<td>Trimastix pyriformis sulphatus</td>
<td></td>
</tr>
<tr>
<td>Trimastix marine</td>
<td></td>
</tr>
<tr>
<td>Malawimonas jakobiformis</td>
<td>Malawimonas</td>
</tr>
<tr>
<td>Reclinomonas americana</td>
<td></td>
</tr>
<tr>
<td>Jakoba libera</td>
<td>Jakobids</td>
</tr>
<tr>
<td>Jakoba incarcerare</td>
<td></td>
</tr>
<tr>
<td>Chlorarchnoion</td>
<td></td>
</tr>
<tr>
<td>Cercomonas</td>
<td>Cercozoa</td>
</tr>
<tr>
<td>Trypanosoma brucei</td>
<td></td>
</tr>
<tr>
<td>Euglena gracilis</td>
<td>Euglenozoa</td>
</tr>
<tr>
<td>Trypanosoma cruzei</td>
<td></td>
</tr>
<tr>
<td>Naegleria gruberi</td>
<td>Heteroloboeza</td>
</tr>
<tr>
<td>Plasiebystra</td>
<td></td>
</tr>
<tr>
<td>Taxoplasma gondii</td>
<td>Aveolata</td>
</tr>
<tr>
<td>Stylonychia lemae</td>
<td></td>
</tr>
<tr>
<td>Plasmodium faciparum</td>
<td></td>
</tr>
<tr>
<td>Pythium ultimum</td>
<td>Stramenopiles</td>
</tr>
<tr>
<td>Volvox carteri</td>
<td>Viridiplantae</td>
</tr>
<tr>
<td>Pisum sativum</td>
<td></td>
</tr>
<tr>
<td>Spironucleus barkhanus</td>
<td>Diplomonads</td>
</tr>
<tr>
<td>Carpediemonas</td>
<td>Carpediemonas</td>
</tr>
<tr>
<td>Trichomonas vaginals</td>
<td>Parabasalids</td>
</tr>
<tr>
<td>Streblomastix strix</td>
<td>Oxymonads</td>
</tr>
</tbody>
</table>
Figure 6.1 Some of the excavates taxa and their relatives [84]
6.2.2. Phylogenetic Analysis

Several trees were constructed for the same data set using different methods, i.e., the maximum-likelihood (ML), the maximum parsimony (MP) and the minimum-evolution distance (NJ) analyses. Since the new QB method is an agglomerative procedure and the topology of a quartet tree is easily readjusted (quartet weight correction procedure) in contrast to pair-wise distancing among taxa, the results obtained from other methods could be taken into consideration at each critical point during the merging process. Such an integrated approach, might achieve more accurate and reliable results. However, when the decision still cannot be made after serious consideration of all available information, both computational and experimental, multiple trees will need to be built for posterior analysis.

The maximum likelihood (ML) tree was inferred by PHYML 4.2 [44], one of the fastest and most accurate programs for ML tree reconstruction. The Jones-Taylor-Thornton (JTT) amino acid substitution model was used with Γ distribution approximated by four equally probable discrete categories; an α parameter and discrete Γ distribution were estimated from the data using Tree Puzzle 5.2 [89]. This model was suggested by the ModelGenerator program which supports 56 nucleotide and 80 amino acid substitution models. Both Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) agree on that model. Maximum Parsimony and Neighbour Joining analyses were done using DNAPARS and NJ in the PHYLIP 3.66 package [33].

The final results were obtained using QB with an integration of results from other methods and morphological evidence. In the experiment, the initial value of the threshold $\alpha$ in QB was set to 0.85 for detecting critical points. To assess the
constructed trees a 500 replicate ML bootstrap analysis was performed using PHYML under the same model.

6.3. Experimental Results

Due to the large conflict of relationships among Excavate taxa; the results regarding the relationships for each group are discussed separately, and an attempt will be made to resolve the conflicts by combining molecular phylogenetics (using different methods) and morphological analyses.

6.3.1. Trimastix

The Trimastix group contains three species: Trimastix pyriformis pyriformis, Trimastix pyriformis sulcatus and Trimastix marine. They form a group with strong support in all methods and this is consistent with previous studies [4, 5, 18, 24, 46, 65, 92, 93]. Ultra-structural data strongly links Trimastix with core-Jakobids (Jakoba libera and Reclinomonas americana) [74]. This link has already been shown previously [102] with the exclusion of some groups such as Oxymonads, Parabasalids, Carpediemonas, and Diplomonads. Recently, molecular and morphological studies have shown that Trimastix are most closely related to Oxymonads and not a direct sister to Jakobids [92]. QB analysis showed that the nearest group to Trimastix are Oxymonads with strong support for this result (86% confidence value and 80% bootstrap support) and consistent with previous studies.
6.3.2. **Parabasalids, Diplomonads, and Carpediemonas**

According to the morphologically based “excavate hypothesis” [90], Diplomonads, Carpediemonas, Trimastix, core-Jakobids, and Malawimonas share a lot of similarities and are called Excavate taxa. Parabasalids may not be part of the Excavate taxa, as this group lacks most, if not all, excavates characteristics [46]. However, a previous molecular phylogeny study shows that Parabasalids share a common ancestor with Diplomonads [14]. In this experiment, the result from the maximum parsimony (MP) analysis shows that Diplomonads are a sister group to Malawimonas, where Parabasalids and Carpediemonas form a separate clade. In this scenario, MP disagrees with neighbour-joining (NJ) and maximum likelihood (ML) analyses, which show Parabasalids and Diplomonads sharing a common ancestor (with 96% bootstrap support) and places Carpediemonas as their sister group (with 95% bootstrap support). QB analysis agreed with NJ and ML with a strong support in terms of a confidence value of 99%.

6.3.3. **Malawimonas**

Malawimonas may not be considered part of the Jakobids [4, 13, 91]. While they share a lot of structural similarities with Trimastix and Carpediemonas [90], several studies cast doubt as to whether Malawimonas are particularly closely related to Jakobids [4, 90]. Also, a previous study on a limited group of excavates strongly supports Trimastix and Jakobids sharing a common ancestor, where Malawimonas are closer to Trimastix than Jakobids [102]. Therefore Malawimonas are considered as a separate group [4, 13] and their relation to other Excavate taxa is still completely unresolved.
ML analysis places Malawimonas as a sister group to the Carpediemonas-Diplomonads-Parabasalids clade with 75% bootstrap support, whereas NJ analysis places Malawimonas as a sister group to the Oxymonads-Trimastix clade. A different scenario is drawn from MP analysis, where Malawimonas and Diplomonads form a separate group. QB analysis showed three scenarios or patterns, all with good support in terms of confidence values as shown in Figure 6.2. Thus, it was not possible to determine which two groups should be joined together based on the results of these molecular phylogenetic analyses.

Since the confidence values for these possible merge patterns are about the same and reasonably high, all three patterns for the latter merge steps were kept. When continuing the merge process using different patterns, three different subtrees
were obtained. These subtrees together with their confidence values are depicted in Figure 6.3. At the very next step each of these subtrees would be merged with another sub-tree containing core-Jakobids (Jakoba libera and Reclinomonas americana). While it is known that Malawimonas look structurally more alike to Jakobids [76, 90] they share a lot of structural similarities with Carpediemonas [90]. It is clear that the sub-tree in Figure 6.3(a) has a much higher confidence value than the other two and also that it places Malawimonas and Jakobids closer together in the complete final tree (see Figure 6.4). Thus, this sub-tree may be considered more reasonable, since it is consistent with the morphological evidence.

Figure 6.3 Three possible combinations for Malawimonas. The numbers on the trees show the confidence values generated by QB.
6.3.4. Jakobids

The group Jakobids contains Reclinomonas americana, Jakoba libera, and Jakoba incarcerator which are closely related [91]. However, all molecular phylogenetic analyses [24, 50, 65, 74, 76, 91, 93, 102] placed Jakobids in two separate groups, the first containing Reclinomonas americana and Jakoba libera, and the second containing Jakoba incarcerator, which branches away from the other Jakobids. QB analysis strongly agrees with the classification that Reclinomonas americana and Jakoba libera form their own clade with strong support (100% bootstrap and a 97% confidence value), and Jakoba incarcerata remain separate from other Jakobids, as shown in Figure 6.4.

With an unexpected placement for Jakoba incarcerata it can be seen that molecular phylogenetic analyses do not support Jakobids being monophyletic. If all Jakobids are forced into one clade, the tree gets far less support (less than 50% confidence value and a low likelihood value). Therefore, Jakoba incarcerata might be a separate Jakobids group. Recently, a new species Andalucía godoyi has been isolated and analysis shows that this group forms a strongly supported clade with Jakoba incarcerata, while other Jakobids form a different separate clade (with 100% bootstrap support) [65].

The remoteness of Jakoba incarcerata from other Jakobids might be caused by a lack of data. It is expected that the current conflict may be resolved as more data becomes available in the future.

6.3.5. Euglenozoa and Heterolobozea

Euglenozoa are usually considered to be related to Heterolobozea based on some structural features [76] and some molecular phylogenies [24].
phylogenetic studies show a specific relationship between Heterolobozea-Euglenozoa and Jakobids, especially Heterolobozea [90].

Figure 6.4 Tree with multi-scenarios for Excavate taxa. The tree produced by the QB algorithm shows all possible positions for Malawimonas which is biologically close to Jakobids, Trimastix, and Carpediemonas. Also, Euglena gracilis could either join Heterolobozea or stay with the other Euglenozoa. Jakoba incarcerata is still removed from the core Jakobids. The numbers along the tree show the confidence values for each possible scenario.

MP analysis shows a specific relationship between Heterolobozea and Jakobids, that is, Naegleria gruberi form a clade with Euglenozoa, and the other Heterolobozea, namely Plaesiobystra, have a close relationship with Jakobids. NJ
and ML analyses confirm a good relationship between Euglenozoa-Heterolobozea and Jakobids. However, the relationship between Euglenozoa-Heterolobozea and Jakobids has a weak bootstrap support (60%). QB analysis showed a very weak relationship between Heterolobosea-Euglenozoa and Jakobids (less than 50% support in terms of its confidence value). However, all methods suggest a very close relationship between Euglenozoa and Heterolobozea.

QB analysis revealed a critical point where Euglena gracilis has two possible merge patterns, as shown in Figure 6.4, both with strong support in terms of its confidence value. The first merge pattern suggests that Euglena gracilis is a sister group to Heterolobosea with a 0.85 confidence value. The second pattern supports Euglena gracilis grouping with the other Euglenozoa with a 0.84 confidence value. ML and NJ analyses agree on the first scenario, while MP supports the second. At this point, there is no way to decide which scenario is or should be the correct one. More data may be needed to resolve this issue.

6.4. Summary

In this chapter, we tested the performance of QB algorithm on real data set called excavate taxa, one of the most interesting group in the eukaryotic evolution. The result shows that QB algorithm has the ability to show any conflict in the relationship between species, and multiple trees could be produced for posterior analysis. The critical point techniques can provide information about the relationships between species in more details, how close they are, and detect other reasonable possible trees. This makes QB algorithm flexible in such a way that can combines morphological and molecular phylogenetic analyses to yield more accurate tree.
The results also show that some excavate clusters, such as Trimastix-Oxymonads, core-Jakobids (Reclinomonas americana and Jakoba libera), and Parabasalids-Diplomonads-Carpediemonas, are robust and independent of the method used. Moreover, the remoteness of Jakoba incarcerata might be caused by lack of information. However, in all methods Jakobids form two separate groups: Jakoba incarcerata as a single group and the other core Jakobids together. Malawimonas have three possible positions (Figure 6.2) all of which have strong support in term of a confidence value and none of which can be considered more apt and a basis on which to continue. Therefore, the only solution was to keep all possible patterns for the next merge. By continuing the merge with the three different patterns, the next result yielded three sub-trees. A conflicting result was achieved by other molecular phylogenetic methods (NJ, Phyml, and MP). However, based on molecular evidence, the pattern in Figure 6.3-A appeared more reasonable. Also, since Heterolobozea and Euglenozoa are closely related to each other, this might explain why Euglena gracilis has an equally close relationship with Heterolobosea and the other Euglenozoa. Moreover, a specific relationship between Euglena-Heterolobozea and Jakobids was revealed with weak support (in terms of a confidence value and bootstrap).

It is hard to distinguish an exact relationship between real species. Therefore, at each critical point, QB tried to make a decision by considering result from other molecular phylogenetic methods and morphological analysis. This combination between phylogenetic analyses and morphological analyses is a good approach to obtain more accurate and reasonable tree using all the available data.
This chapter introduces a parallel implementation of QB algorithm. Experimental results obtained from a 128 node IBM BlueGene/L using simulated data sets to test the performance of the parallel versions of QB are presented.

7.1. Introduction

One major disadvantage associated with most quartet-based algorithms is that they require $O(n^4)$ computational steps to complete, where $n$ is a given number of molecular sequences in the analysis. This is simply because they need to generate $\binom{n}{4}$ quartet trees in order to obtain a reasonably good result. When the problem size $n$ is large, the memory size also becomes an issue, this is because a number of $\binom{n}{4}$ quartet weights for all possible quartet trees must be calculated and stored in the first
stage so that they can be used in the second stage. Further more, each quartet is associated with three weights for three possible resolved trees. For a given set of \( n \) sequences, the memory will be \( \left( \frac{n}{4} \right) \times 3 \times 8 \) bytes in size, assuming a double variable is used for each quartet weight. For example, 1.55GB is needed to store all the quartet weights when \( n = 200 \). Therefore, the algorithm is both computationally and memory intensive.

A parallel program package of Tree-Puzzle [89], a well-known method based on quartet weights, can only handle problems of size up to 250 regardless of how many processors are used in the computation. This is because it requires the whole set of quartet weights to be stored on every processor for the construction of intermediate trees in the puzzling stage.

The current trend is to design fast algorithms for phylogenetic analysis, e.g. those described in [44, 68][1, 2]. Given its excellent theoretical properties, the quartet-based method should never be overlooked. High-performance computing machines can be adopted to handle the higher computational and memory requirements of quartet-based algorithms.

The following section describes a parallel implementation of the QB algorithm which is able to tackle problems of a much larger size.

### 7.2. The parallel Algorithm

In this implementation, the master-worker paradigm is adopted. Workflow graphs for the master and workers are depicted in Figures 7.1 and 7.2 respectively.

In stage one, the \( n \) sequences are broadcast to all the worker processors from the master for quartet weight generation. The generation of quartet weights in
Parallel is a good application of a well-known method for combination enumeration [3]. The computation can be done very efficiently in parallel. First, each quartet is given a number from 1 to $\binom{n}{4}$. This set of natural numbers is then divided into $p$ subsets of equal size (or very close if not exactly equal), where $p$ is the number of processors involved in the computation. Since there is a simple one-to-one mapping between a complete set of $n$ quartets and a set of natural numbers, with just two integer numbers, one indicating the first quartet and the other indicating the last quartet, each processor knows exactly which subset of quartets it needs to generate. After the quartet generation, a subset of quartet weights is kept on each processor and used for computation in the second stage.

In stage two, a global quartet weight matrix is constructed by accumulating the weights of all possible quartet trees to the corresponding entries in the matrix. To construct this matrix each worker processor first builds a local weight matrix (since the quartet weights are evenly distributed among processors), which is collected by the master processor.

Once the quartet weight matrix has been built, the construction of phylogenetic trees can commence using the same procedure described previously. It should be noted that this tree reconstruction procedure takes far less time to complete than the quartet weight generation in stage one. As discussed, it only takes $O(n^3)$ time to complete when quartet weight correction is not an issue. It was decided to let the master processor do the tree reconstruction with worker processors helping with quartet weight correction (or quartet weight matrix correction) after each merger of two sub-trees completed by the master.
Chapter 7: Parallel Implementation

Figure 7.1 Master workflow graph
Figure 7.2 Workers workflow graph
Since multiple trees may be constructed during the computation, the program requires the modification of the quartet weight matrix, but not the original quartet weights. The quartet weights are stored across the processors, and the master processor needs to broadcast the merge information to worker processors each time two sub-trees have been merged. Each worker processor first updates the partially merged tree in accordance with the structure on the master processor and next constructs a local weight correction matrix. It is necessary to keep a local copy of the state of sub-trees on each worker processor so that it knows which quartet weights need correcting and the communication cost can be minimized. The master processor then collects all the local weight correction matrices to update its global quartet weight matrix accordingly.

It takes \( n - 2 \) steps to merge all \( n \) sequences. Therefore the total communication cost for tree reconstruction is the costs for \( n - 2 \) broadcasts of merge information from the master to all worker processors, plus the costs for \( n - 2 \) collective messages to receive the weight correction matrices from the workers.

For multiple tree construction a stack on the master processor is used to keep track of the possible tree merges (critical points) awaiting evaluation. To minimize communication costs the worker processors maintain the same stack. Therefore, additional information must be broadcast to the worker processors when a critical point is encountered. This message can be combined with the message containing the merge information to further reduce the communication overheads.

### 7.3. Experimental Results

The parallel implementation of QB was tested on a 128 node IBM BlueGene/L cluster. The IBM BlueGene/L is a new generation massively-parallel computing
system designed for research and development in computational science [12]. It is an extremely high compute-density system with relatively modest power and cooling requirements. Each BlueGene/L node consists of two 700MHz CPUs and 512 MB memory (256MB per CPU). The Blue Gene/L has multiple network architecture implemented to allow efficient communication between processors. A 3D torus network is used for node communication with neighbours. During a program run some communication calls are more global than others (e.g. all to one, one to all, and all to all). For these, Blue Gene/L provides another network: the collective network. This collective network connects all the computed nodes in the shape of a tree whereby any node can be the tree root (originating point). The Blue Gene/L uses this network to implement MPI collective communication calls. Parallel QB algorithms make effective use of this collective communication network to achieve a good performance. The barrier (global interrupt) network is the third dedicated hardware network the Blue Gene/L provides for efficient MPI communication.

The BlueGene/L operates in two primary modes, co-processor and virtual-node. The co-processor mode dedicates one CPU per node for computation and the other CPU for communication. This mode is particularly suited for communication bound computations. The virtual-node mode allows the two CPUs in each node to act as an independent node, effectively doubling the number of available processors for computation at the cost of communication speed. Due to the collective nature of communication and length of computation time in the parallel algorithm, the virtual-node mode not only doubled the number of available processors. Since the virtual-node mode also provided a linear speedup in comparison to the co-processor mode, this was the mode selected for the experiments.
In this experiment, computation time, memory usage and communication costs of
the algorithm were measured for a number of different DNA sequences and across
a number of different CPUs. Synthetic DNA sequence data with lengths of 4000 was
used and the size of input was varied between 50 and 400 sequences. Some of the
experimental results are presented in Figures 7.3 and 7.4.

![Figure 7.3 Speedup vs 32 CPUs for different numbers of sequences (or taxa)](image)

Figure 7.3 shows the performance in terms of speedup (versus 32 CPUs) for
two different sized problems. When the problem was small, i.e., \( n = 50 \), the total
computational cost was not high in comparison with the communication. Thus the
performance was sub-optimal when a large number of processors were involved in
the computation. This indicates that there is no need to use a large number of
processors to tackle small problems. When the number of input sequences was large, e.g., \( n = 150 \), the overall speedup approached linear as the computational time for quartet generation began to dominate.

![Figure 7.4 Percentage of Computation Time](image)

Figure 7.4 shows the percentages of the total computation time used for quartet generation in stage one and tree construction in stage two. It can be seen that the total computation and communication costs for the second stage was only a fraction of the total computational cost for a large number of long sequences.

As the number of sequences increases, a large memory is needed to store the quartet weights. On a 128 node IBM BlueGene/L cluster each node has a local memory of size 512MB so the size of the memory collectively is thus over 65.5GB.
For a problem size of 400, the memory requirement is about 25.2 GB. Since the quartet weights are evenly distributed across the processors, it is easy to tackle problem sizes of 400 or even larger on the 128 node cluster. Figure 7.5 shows the computation time (in seconds) for problems from sizes 50 to 400 using all 256 CPUs in the cluster.

![Figure 7.5 Speedup of program with 256 CPUs](image)

The QB algorithm is able to make effective use of the fast collective communication network provided by the IBM BlueGene/L cluster: For the given problems the communication time remained nearly constant as the number of processors increased. For example, the total running time for 250 sequences run on 256 CPUs for a single tree construction was about 1.3 hours, but the total
Communication time was only 12 seconds, which is about the same as that when using a smaller number of CPUs.

**7.4. Summary**

This chapter introduced a parallel implementation of the QB algorithm. Since the QB algorithm is both computationally and memory intensive, it needs parallelization for tackling large size problems. In distributing the quartet weights evenly across the processing nodes and making effective use of a fast collective network on the IBM BlueGene/L cluster, problems of a much larger size could be tackled and the results of the experiments show that close to linear speedup is achievable even when the number of processors involved in the computation is large. This demonstrates that QB as a parallel algorithm is very efficient and also confirms that the IBM BlueGene/L cluster is an excellent and powerful machine for scientific computing. Also, the results show that the parallel implementation has no effect on the accuracy of QB. In fact, it could be an advantage to obtain more accuracy, since it allows the use of high number of stages and more possible trees can be generated.
Conclusion and Future Work

This final chapter provides the conclusion of this research followed by a brief discussion of aspects to be considered by future work in the field of phylogenetic analysis.

8.1. Conclusion

The computation of the tree of life that connects all living organisms on earth remains one of the main challenges in computational biology. While it is already possible to construct small phylogenetic trees for a group of taxa based on molecular sequences, the huge amount of molecular sequences on public database such as GenBank offer the opportunity to study the evolutionary history among large number of species. However, building large phylogenetic trees is still a real challenge.

The most accurate methods for reconstructing evolutionary trees use aligned data sequences and are based on statistical models, where the evolutionary models
might contain a biological contribution. Currently, the most common approaches are maximum likelihood, maximum parsimony, distance-based and Bayesian inference. Each of them has its own weaknesses and strengths, some being complex and expensive, others simple and fast.

The quartet-based method is another method for reconstructing large evolutionary trees from a set of quartet trees. A quartet tree is the smallest informative structure of binary unrooted tree representing a relationship between four different species. The main advantage of quartet-based method is that there is a one-to-one correspondence between a tree topology and a set of quartet trees, and so if the topology of every individual quartet can be accurately identified, the entire evolutionary tree will be correctly constructed in polynomial time[6]. The work presented in this thesis has focused on the design and development of a quartet-based algorithm called QB.

The basic concept of the new quartet-based algorithm QB is first explained. Every given binary tree topology is associated with a unique set of quartet trees. These quartet trees are in turn can generate a unique global quartet weight matrix. An efficient method is introduced to reconstruct the original tree topology from its generated global quartet weight matrix. As a result, a one-to-one mapping between a given tree topology and its associated global quartet weight matrix is proved. This one-to-one mapping forms a solid base of our new quartet-based algorithm.

The new QB algorithm has a significant advantage over other quartet-based methods. It takes a global view by considering all the possible quartet trees when merging a pair of sub-trees. Also, several techniques are included to effectively deal with inaccurate quartet trees and thus increase the accuracy of the final tree. These techniques include:
1- The average confidence value: It is a simple procedure to make sure that every leaf node in the two sub-trees contributes in the merging process.

2- Quartet weight correction: Since the quartet trees can not be fully resolved, their weights need to be corrected to their true value based on the current tree topology. It is a complex process, and so in order to minimize the cost, only the related quartets are corrected directly into the quartet weight matrix.

3- Critical Points: A critical point is where it is hard to decide between more than one possible merge pattern during the computations, all have a high confidence values, and it is difficult to determine which pattern should be merged. Therefore, limited number of tree could be generated, as it is often desirable to construct a number of trees for a posterior analysis. The number of generated tree could be quite large, and thus a threshold $\alpha$ is used to determine the critical points.

The performance of the new QB algorithm is tested using synthetic data sets, which contains test data sets for various situations. The result demonstrated that our QB algorithm outperform most existing methods. When the number of generated trees was allowed to increase, the percentage of the correct tree to be among the limited number of generated trees is very high. This demonstrates the importance of the critical point. By using the maximum likelihood criteria to select the one single best tree from this limited space, The QB algorithm performs similar and some time better than PHYML and FASTDNAML. This result reveals that maximum
likelihood has some limitation and may not find the correct tree. Even though an optimal solution under ML can be obtained, incorrect trees may have a likelihood value greater than the correct tree.

The performance of the QB algorithm is tested on real data sets as well. The data sets are considered as one of the most important group in Eukaryotic evolution. The results demonstrated that QB is able to yield significantly more accurate tree than other methods. One great advantage of QB on is that it allows us to effectively incorporate the results obtained from other methods (molecular and morphological) when critical points encountered during the tree reconstruction. This integrated approach can achieve more accurate and reliable results for real data set.

The reconstruction of the quartet weights is the most expensive procedure in quartet-based methods. Nowadays, it should be a less significant problem than accuracy, since high performance computing machines can handle high computation and memory requirements. A parallel version of the QB algorithm is designed and implemented. It shows a good speedup can be obtained by distributing the quartet weights across the processing nodes. Experimental results show that a close to linear speedup can be obtained, even when the number of processors involved in the computation is large.

**8.2. Future Work**

The performance of the new QB algorithm is still can be improved in many ways, some of the future work is mentioned below:

- **Finding the best tree**: The QB algorithm is able to generate a limited number of trees. Our results show that the percentage of the correct tree to be
included in that small number of trees is very high. By using ML (which is widely used in a variety of applications) to select an optimal single tree, QB performs similar to other ML methods. Since the percentage of the correct tree to be included in that small space of trees produced by QB is very high, the performance of QB should be better than other existing methods. Thus, it is good to further explore good search techniques in order to be able identify more correct trees in this limited space of trees.

- **Integration**: During the computation, the QB algorithm is able to find critical points, where the merge decision is difficult to make as there are several choices having similar confidence values. We demonstrated using real data set that incorporate different phylogenetic analysis methods and consider other morphological evidences if available at each critical points allows us to achieve better result. A future direction is to automatically integrate all available information from different sources to increase the accuracy.
References


