THE QPHYL SYSTEM: A WEB-BASED INTERACTIVE SYSTEM FOR PHYLOGENETIC ANALYSIS

A thesis submitted in fulfillment of the requirements for the degree of Master of Science by research in the School of Information Technologies at The University of Sydney

Zhen Zhao

March 2008

Abstract

Phylogenetic tree reconstruction is a prominent problem in computational biology. Currently, all computational methods have their limitations and work well only for simple problems of small size. No existing method can guarantee that trees constructed for real-world problems are true phylogenetic trees for large and complex problems mainly because the existing computational models are not very biologically realistic. It has become a serious issue for many important real-life applications which often desire accurate results from phylogenetic analysis. Thus, it is very crucial to effectively incorporate multi-disciplinary analyses and synthesize results from various sources when answering real-life questions.

In this thesis, a novel web-based phylogeny reconstruction system with a real-time interactive environment, called Qphyl (short for quartet-based phylogenetic analysis) is introduced. The Qphyl system uses a new interactive approach to enable biologists to greatly improve the final results through effectively dynamic interaction with the computation, e.g., to move the computation back and forth to different stages so users can check the intermediate results, compare results from different methods and carry out certain manual refinements using their biological domain-specific knowledge in the decision making on how a tree should be reconstructed. Currently the alpha version of this web-based interactive system has been released and accessible through the URL: http://ww-test.it.usyd.edu.au/sogrid/qphyl/.

Acknowledgements

I'd like to thank my supervisor, Associate Professor Bing Bing Zhou, for his guidance and support throughout the year.

I would also like to thank Dr. Chen Wang, Penghao Wang and Monther Tarawneh, whose advice I sought during the year.

Thanks also go to my family and friends for their unwavering support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Overview

In biology, phylogenetics is a field with a growing impact on a variety of life science domains. The central aim is to discover the evolutionary relatedness among various groups of organisms (e.g., species and populations). The organisms are classified by taxonomy according to homology. Presently, the discovery and analysis of the patterns and processes that underpin biodiversity has a great number of practical applications throughout science and industry, including pharmaceutical drug discovery, medical diagnosis, conservation and forensics.

A lot of methods have been employed to reconstruct phylogenies. Frequently used methods include distance-based (e.g. neighbor-joining [34]), maximum parsimony [38], maximum likelihood [21] and Bayesian approaches [8]. All these phylogenetic methods depend on an implicit or explicit mathematical model describing the evolution of characters observed in the species included, and are usually employed for molecular phylogeny from a set of character-aligned DNA or amino acid sequences. These methods rely on a defined substitution model that encodes a hypothesis about the relative rates of mutation at various sites along the sequences. These substitution models aim to correct for differences in the rates of transitions and transversions in nucleotide sequences.

Modern phylogenetic software using these methods for phylogenies makes it easy to produce trees from aligned sequence data. However, as a general rule, most character datasets are so "noisy" that no truly "simple" solution is possible [48]. Therefore, no applications can perfectly handle real-life problems due to the limitations of the methods they use. Furthermore, these methods treat phylogenetic inference as a “black box” to users. Although some of the applications allow users to set up parameters of analysis before the computation starts (e.g. PHYML [51]), the computation procedure is still restricted within only one specific method.

Some of the packages are composed of a set of programs based on different methods (e.g. PHYLIP [52]). This kind of collection can satisfy most of the phylogenetic biologists with diverse preferences. Trees can be reconstructed by various methods; however, there are no interaction among different algorithms and methods. This is meaningless to compare generated trees for biologists because it’s difficult to identify computational mistakes.

Visualization is another pending issue of phylogenetic software. Most of applications need a third-party tree-visualization package (e.g. TreeView [49]) to graphically represent the reconstructed trees. But phylogeny data from a specific method often contain more information than be visualized by these tree-visualization packages. Sometimes this lost information is very important for biologists to analyse and evaluate the phylogenetic results, but filtrated by the visualization tools.

Last but not least, it is very inefficient for biologists to modify tree topologies via visualization tools after the tree reconstruction due to no interaction between the manual operation and automatic computation. i.e., even if a particular error has been noticed in a tree topology, a simple modification only aimed at this error still cannot guarantee to correct the tree to an accurate tree.

Thus, it is desirable to have an interactive platform which allows biologists to integrate computational and non-computational methods into the phylogeny reconstruction procedures. For this reason, a novel web-based interactive computing system for phylogenetic analysis, based on our quartet-based algorithm [14, 15], has been proposed. This system provides an interactive interface for biologists to effectively interact with the computation, (e.g., to move the computation back and forth to different stages so users can check the intermediate results, compare results from different methods and carry out certain manual refinements using their biological domain-specific knowledge in the decision making on how a tree should be reconstructed to greatly improve the final results), and a real-time visualization tool to represent every stage of the tree reconstruction graphically with computational information. The interactive interface enables the computation to effectively incorporate multi-disciplinary analyses and synthesize results from various sources. Currently the alpha version of this web-based interactive system has been released and accessible through the URL: http://ww-test.it.usyd.edu.au/sogrid/qphyl/.

1.2 Contribution

This thesis is produced with the aim to address the limitations of current systems and packages designed for phylogeny analysis and try to overcome their weaknesses by proposing an interactive approach based on our quartet-based algorithm. The main contributions include:

(i) The proposal of a new architecture design and implementation of web-based interactive system for phylogeny reconstruction.

(ii) An effective interactive approach which can be efficiently incorporated with our quartet-based algorithm and provides an interface for further integrations.

(iii) A sophisticated and user-friendly GUI with a powerful tree visualization tool.

1.3 Layout of the Thesis

The rest of thesis is organized into the following chapters. In Chapter 2, the algorithms and packages in phylogeny domain as described in popular research literatures are reviewed. In Chapter 3, we present an introduction of our quartet-based algorithm – a key issue relating to the development of the Qphyl System. In Chapter 4, we introduce the architecture of the Qphyl System by going through the design of both hardware and software components. In Chapter 5, we describe the back-end programs which provide the phylogenetic analysis services. In Chapter 6, we introduce the web-based GUI frontend and protocols of the system. The conclusion of research along with future studies is introduced in the last chapter.

Chapter 2 Background

Phylogenetic trees, also called evolutionary trees, like the one in Figure 2.1, are the visual representation of inferred phylogenies. A phylogenetic tree indicates the evolutionary relationships among various biological species or other entities that are believed to have a common ancestor. In a phylogenetic tree, each node with descendants represents the most recent common ancestor of the descendants, and the edge lengths in some trees correspond to time estimates. Generally, each leaf is called a taxonomic unit and internal nodes are called hypothetical taxonomic units because they cannot be directly observed.

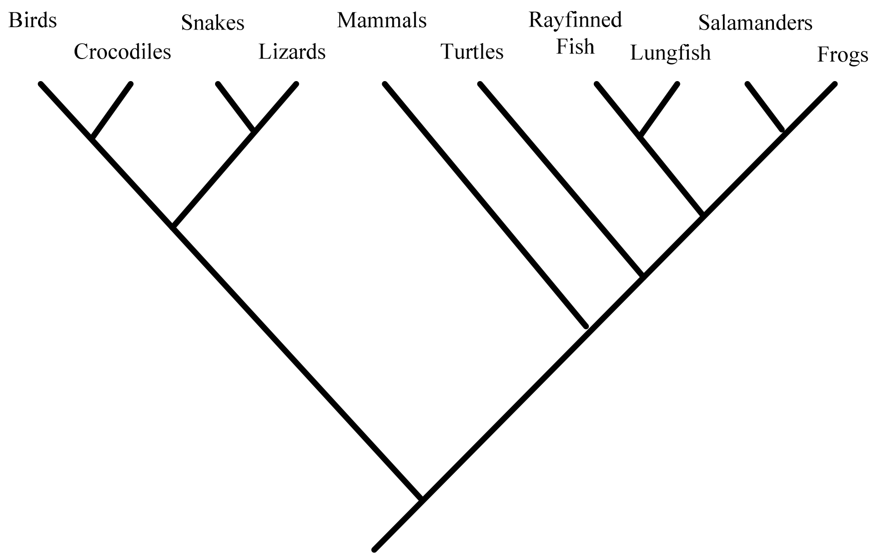


Figure 2.1: A phylogenetic tree (Maddison and Maddison, 1992)

The goal of phylogenetic analysis is to assemble a phylogenetic tree from a set of genes, species, or other taxa. Phylogenetic analysis has proven its usefulness in many important application fields, such as: tracing genes of known function and compare how they are related to unknown genes in order to predict unknown gene function, tracking of step-by-step transmission of disease and identifying disease reservoirs, tracking the diversity of a pathogen to select an appropriate vaccine for a particular region, identifying an organism, and determining protein folds [27, 39].

2.1 Computational Phylogenies

A great deal of attempts has been made to reconstruct the phylogenetic history of group of organisms. The most commonly used computational methods include distance-based methods, parsimony, maximum likelihood and MCMC-based Bayesian inference.

Perhaps Neighbor-Joining (NJ) is the most commonly used distance-based method in molecular genetics due to its quite good approximation and the computational speed advantage. Because it is a polynomial-time algorithm, its efficiency is very impressive. Furthermore, NJ is also useful to rapidly search for a good tree that can then be improved by other criteria. However, it seems that NJ is employed solely on the grounds of computational speed. The cost of this objective is the decreasing of the accuracy. In other words, although NJ can very quickly produce reasonable trees, but cannot necessarily produce the best tree for the given data. When evolutionary rates vary from site to site in sequences, distances should be corrected for this variation. When variation of rates is large, these corrections become important. But NJ forgets to use information from changes in one part of the tree to correct the others [28].

Maximum Parsimony (MP) probably is the most widely used tree-building method in systematics. The general idea of MP is that the evolution always picks up the easiest path to be executed. So under the MP, the preferred phylogenetic tree is the tree that requires the least number of evolutionary changes. It sounds quite reasonable and may be the easiest criteria to understand. MP is popular because it's a very simple and straightforward. However, the problems of the approach are obvious. The first problem is that MP often produces a number of equally most-parsimonious trees [48]. A large number of "best trees" is often seen as an analytical failure. Secondly, it is not statistically consistent. That is not guaranteed to produce the true tree with high probability, given sufficient data. The Joe Felsenstein's demonstration in 1978 indicates that MP can be inconsistent under certain condition. The third major problem is well-known as Long Branch Attraction, which is a phenomenon that when rapidly evolving lineages are inferred to be closely related, regardless of their true evolutionary relations.

The method of maximum likelihood (ML) [21] is a parametric statistical method, in that it employs an explicit model of character evolution. That's because, practically, the probability of a tree cannot be calculated only if we assume a specific set of probabilities of characters change. Mechanically, much like MP, in ML analysis functions trees are scored based on a character dataset and the tree with the best score is selected. Generally, such methods are potentially more powerful than non-parametric statistical methods like MP, but only if the data is produced by a reasonable approximation of the processes [28, 48]. The main strength of ML is that ML has a sound and longstanding basis in statistical methodology which makes its accuracy drivingly forceful. However, it is a very computer-intensive approach which is its primary practical weakness. The complexity of computation increases rapidly with larger numbers of species, because there could be a great many terms. It makes an analytical solution to ML estimation only possible for very simplest problems [28].

Bayesian methods are closely related to likelihood methods. It uses the same evolutionary models as does ML analysis and it also uses the likelihood function to quantify the trees. The difference is that, when assessing a particular tree in the ML analysis we are trying to find the combination of parameter values that produces the largest maximum-likelihood value, while in Bayesian analysis we are trying to find the average likelihood across all of the possible parameter values. Mathematically, the ML analysis uses the profile likelihood while Bayesian analysis uses the integrated likelihood. The Bayesian approach has become more popular due to advances in computational machinery, especially, Markov Chain Monte Carlo algorithms (MCMC) [8]. MCMC methods in Bayesian inference draw a random sample from the posterior distribution of hypotheses, in this case, trees. So it is possible to make probability statements about the true tree. This method will generate a good approximation to the desired result if the sampling is performed correctly and the sample is large enough.

On the other hand, the Bayesian approach demands specifying priors for the tree topology and branch length for all of the parts of the substation model. This means we have to be wary of how we choose to specify the priors. Therefore, the key to a successful Bayesian analysis is not only choosing a suitable evolutionary model, but also choosing priors and making sure that the MCMC is doing its job. Compared with ML analysis, there are three major practical advantages in Bayesian analysis. The first one is that the time that an analysis takes can be estimated accurately; the second one is that the time taken for analysis will not be increased dramatically if the evolutionary models are getting more complicated; the last one is that estimation of the degree to which each branch is supported by the data and model is given as an integral part of the analysis [28].

2.2 Phylogeny Packages

The implementation of computer programs for handling phylogenetic analysis is not as difficult as it seems. Logical data structure design and recursive function calls allow us to move rapidly through a tree, updating information or changing its structure. The most widely used phylogeny packages and programs for phylogeny inference include PHYLIP [52], PAUP [50], PHYML [51] and Tree-Puzzle [37, 53].

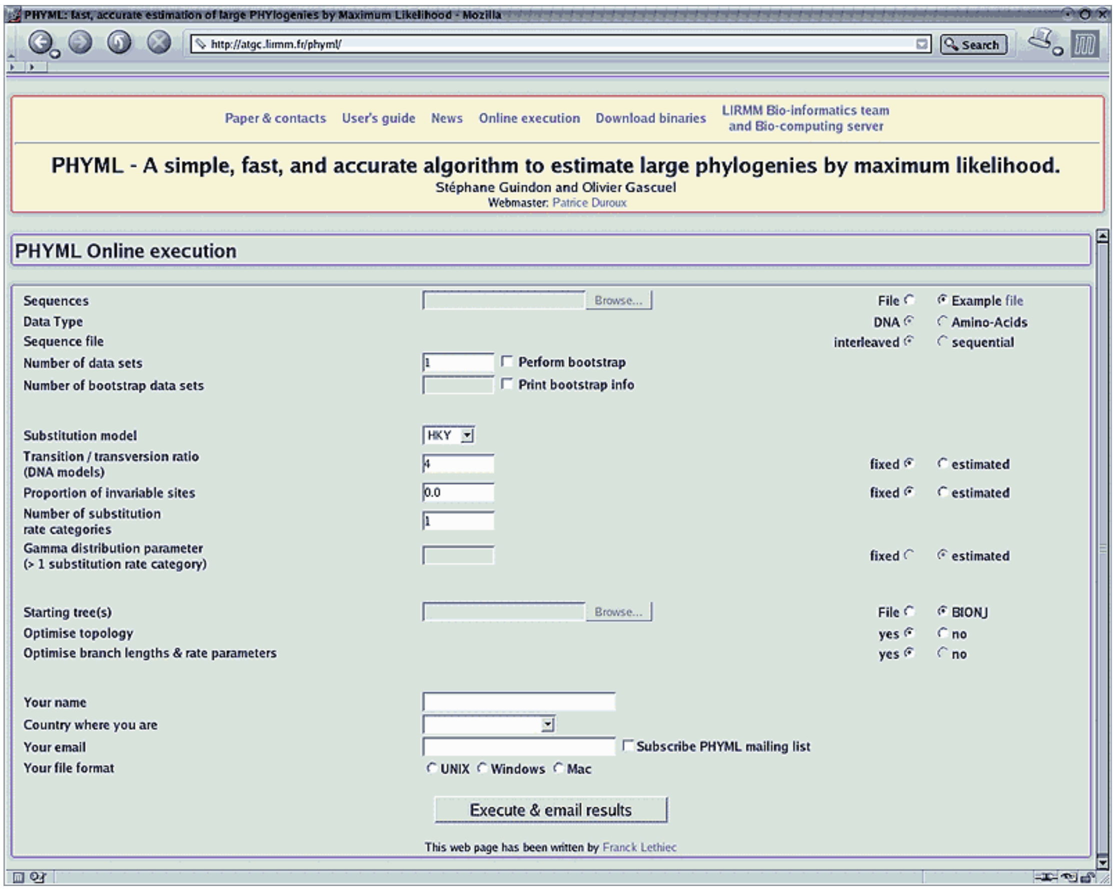
PHYLIP, which is short for Phylogeny Inference Package, is a free package of programs for inferring phylogenies. PHYLIP is an aggregate of programs based on the most popular methods including parsimony, distance matrix, compatibility, and likelihood methods, furthermore, bootstrapping and consensus trees. Duo to this specifically integrative speciality, PHYLIP can satisfy most of the phylogenetic biologists with diverse preferences, so it is the most widely-distributed phylogeny package, and to be the one responsible for the largest number of published trees. Unfortunately, PHYLIP doesn’t provide a graphical user interface for any of the executable phylogenetic program. Through the textual user interface, users can only control each program by a menu, which asks the users which options they want to set, and allows them to start the computation.

PAUP, which is a comprehensive package with parsimony, likelihood and distance matrix methods, has many more features and methods than PHYLIP. But it’s not free. Not like PHYLIP, some of the executables in PAUP are released with mouse-windows user interface. PAUP requires aligned sequences in a common data file format (NEXUS [5]) as inputs. After the tree generation the output tree file can be printed as a low-resolution graphically-represented tree (Figure 2.2) from the display buffer. However, if high resolution tree is required, a third party tree-printing software package, e.g., TreeView, is necessary to produce the tree representation for better quality.

```
/------------------------------------ Lemur catta
|
|                                                /---- Homo sapiens
|                                                |
|                                     /---------14 /------- Pan
|                                     |          \13
|                              /-----15            \--------- Gorilla
18                             |      |
|                /------------16      \-------------- Pongo
|                |             |
+---------------17             \-------------- Hylobates
|                |
|                \-------------------- Macaca fuscata
|
\----------------------------- Saimiri sciureus
```

Figure 2.2: A low-resolution tree created by PAUP

Compared to PHYLIP and PAUP, PHYML is very recent. PHYML is dependent on a method, which is declared as “A Simple, Fast and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood” [51]. This algorithm dramatically reduces computational time and makes the phylogenetic analysis from larger and more complex data sets much more practical. Besides an executable program with a textual user interface, PHYML also provides an online interface to the PHYML algorithm (Figure 2.3; URL: http://atgc.lirmm.fr/phyml/). This tool provides the user with a number of options, e.g. nonparametric bootstrap and estimation of various evolutionary parameters, in order to perform comprehensive phylogenetic analyses on large datasets in reasonable computing time. Users need to provide email address for the results to be sent to user by emails. The tree file shows the maximum likelihood phylogeny in Newick format [43], which can be graphically represented by most third party tree-printing software package. Also, trees can be viewed through an applet available on the PHYML Online server.



PHYML: fast, accurate estimation of large PHYlogenies by Maximum Likelihood - Mozilla

http://atgc.lirmm.fr/phyml/

Paper & contacts | User's guide | News | Online execution | Download binaries | LIRMM Bio-informatics team and Bio-computing server

PHYML - A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood.
Stéphane Guindon and Olivier Gascuel
Webmaster: Patrice Duroux

PHYML Online execution

Sequences	Browse...	File	[✓ Example file]
Data Type		[✓ DNA]	Amino-Acids
Sequence file		[✓ interleaved]	sequential
Number of data sets	1 ☐ Perform bootstrap		
Number of bootstrap data sets	☐ Print bootstrap info		
Substitution model	HKY		
Transition / transversion ratio (DNA models)	4	[✓ fixed]	estimated
Proportion of invariable sites	0.0	[✓ fixed]	estimated
Number of substitution rate categories	1		
Gamma distribution parameter (> 1 substitution rate category)		fixed	[✓ estimated]
Starting tree(s)	Browse...	File	[✓ BIONJ]
Optimise topology		[✓ yes]	no
Optimise branch lengths & rate parameters		[✓ yes]	no
Your name			
Country where you are			
Your email	☐ Subscribe PHYML mailing list		
Your file format	UNIX Windows Mac		

Execute & email results

This web page has been written by Franck Lethiec

Figure 2.3: The interface of PHYML Online

TREE-PUZZLE is a computer program to reconstruct phylogeny trees from molecular sequence data by ML. The name, TREE-PUZZLE, is from a fast tree search algorithm, quartet puzzling [13, 29, 33, 37], which allows analysis of large data sets and automatically assigns estimations of support to each internal branch. TREE-PUZZLE also computes pairwise ML distances [28] as well as branch lengths for user specified trees. It is fast, but its accuracy is not as good as the others.

2.3 Discussion

In the past, research efforts on computational phylogenetic analysis were dedicated to the design of heuristics which can quickly find near-optimal trees under a specific optimization criterion. Applications were designed and implemented based on these methods. However, all criteria are over-simplified and cannot realistically model the real evolutionary process. Thus all currently existing algorithms for phylogenetic analysis have their limitations. It has become a serious issue for many important real-life applications which often demand accurate results from phylogenetic analysis. Therefore, no existing packages based on these algorithms and methods can guarantee that the tree generated by them can correctly answer real-life questions.

It leads us to look for an integrative approach which can effectively incorporate multi-disciplinary analyses and synthesize results from various sources during the computation. Previously we introduced a quartet-based algorithm [14, 15], which is an agglomerative procedure and can be easily adapted for dynamic integration and interaction. In next chapter we will review basic concepts of our original quartet-based method for phylogenetic analysis.

Chapter 3 A Quartet-based Algorithm

There has been strong interest in reconstruction of large evolutionary trees from small sub-trees in the computational biology community in recent years [1, 6, 7, 20, 17, 18, 24, 26]. The quartet-base approach may be the simplest and the most cost-effective one for such reconstruction. However, real data usually contain noises or errors and the design of practical quartet-based algorithms often focuses on how to tolerate quartet errors. We recently derived a new quartet-based algorithm for phylogenetic inference [14, 15], which can significantly reduce the influence to phylogeny reconstruction from the inaccuracy of quartet weights. In this chapter an introduction of our algorithm is provided firstly, and then a discussion based on experimental results is given.

3.1 The Method

The quartet-based algorithm consists of three major stages (Figure 3.1): in stage one quartet weights for every possible quartet trees from a given number of aligned sequences are calculated; in stage two a global quartet weight matrix is generated to gather the quartet topological information from the quartet weights calculated in stage one; in stage three a full size tree is recursively reconstructed using this quartet weight matrix.

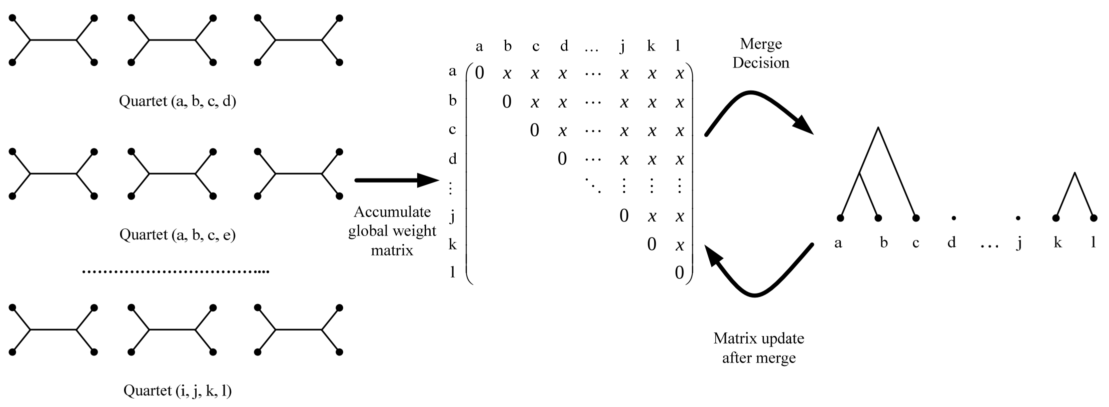


Figure 3.1: The three-stage procedure of the quartet-based algorithm

For a given *n* sequences and the associated set of quartets, we can generate a symmetric global quartet weight matrix of size $n \times n$, each row or column corresponding to one particular sequence. Let $(ij\,|\,kl, w)$ denote a quartet tree with a quartet weight *w*. Our global quartet weight matrix is generated by adding each *w* to entries *ij*, *ji*, *kl* and *lk*, using a complete set of quartets from the given sequences. A simple example of this global weight matrix and its associated quartets is shown in Figure 3.2. In this example there are five sequences and each quartet is associated with three quartet weights for three possible resolved trees.

$$\{1,2,3,4\} \begin{cases} (12|34,\ 1.0) \\ (13|24,\ 0.0) \\ (14|23,\ 0.0) \end{cases} \quad \{1,\ 2,3,5\} \begin{cases} (12|35,\ 0.3) \\ (13|25,\ 0.3) \\ (15|23,\ 0.4) \end{cases} \quad \{1,2,4,5\} \begin{cases} (12|45,\ 0.3) \\ (14|25,\ 0.3) \\ (15|24,\ 0.4) \end{cases}$$

$$\{1,3,4,5\} \begin{cases} (13|45,\ 0.3) \\ (14|35,\ 0.3) \\ (15|34,\ 0.4) \end{cases} \qquad \{2,3,4,5\} \begin{cases} (23|45,\ 0.0) \\ (24|35,\ 0.0) \\ (25|34,\ 1.0) \end{cases}$$

(a)

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0.0 & 1.6 & 0.6 & 0.6 & 1.2 \\ 2 & 1.6 & 0.0 & 0.4 & 0.4 & 1.6 \\ 3 & 0.6 & 0.4 & 0.0 & 2.4 & 1.6 \\ 4 & 0.6 & 0.4 & 2.4 & 0.0 & 0.6 \\ 5 & 1.2 & 1.6 & 0.6 & 0.6 & 0.0 \end{array}$$

(b)

Figure 3.2: A simple example of the global weight matrix: (a) the quartets and their weights and (b) the associated global quartet weight matrix.

Based on the global quartet weight matrix, a confidence value c_{ij}, which indicates how “confident” the quartet-based algorithm is to merge two sub-trees represented by node *i* and *j* respectively, can be calculated by using following equation:

$$c_{ij} = \frac{m_{ij}}{\binom{n-(n_i+n_j)}{\mathbf{2}} + m_i + m_j} \qquad (3.1)$$

Where:

m_{ij} : The entry value at row *i* column *j* in the matrix;

m_i : The number of the concerned quartet trees generated in the sub-tree *i* ;

m_j : The number of the concerned quartet trees generated in the sub-tree *j*;

n_i : The number of sequences in sub-tree *i*;

n_j : The number of sequences in sub-tree *j*;

n : The total number of sequences for phylogeny analysis;

Initially every leaf node is considered as a sub-tree. Therefore, each row in the matrix will be associated with the two variables and they are set to $m_i = 0$ and $n_i = 1$. Then tree reconstruction program recursively merges the two sub-trees which have the highest c_{ij}, and re-correct the corresponding quartet weight because the c_{ij} should equal to 1 if the associated quartets are fully and correctly resolved and directly connected in the original tree. If one of sub-trees or both of them have more than one leaf node, we calculate the confidence values for all leaf node pairs and use the averaged value of them as the confidence value $\bar{c}_{ij}$ for each pair of sub-trees. After one merge step, the quartet weight matrix is updated based on the quartet weight re-correction, and all the confidence values are re-calculated. The phylogeny is completely reconstructed if only a single tree remains. During the tree reconstruction, a sub-tree is identified by a representative leaf, which is the leaf with the smallest index in this group. It is an important notification for creating the communication protocols of the web-based interaction.

Since the matrix is not accurate, it may not always be the right decision to merge the two sub-trees that have the highest confidence value. After the highest confidence value $\bar{c}_{ij}$ is obtained, we then check whether there is another sub-tree *k* which has a reasonably high confidence value associated with one of the two sub-trees *i* and *j* by checking whether $\bar{c}_{ik}\ (\text{or } \bar{c}_{ki}) \geq \alpha\bar{c}_{ij}$, or $\bar{c}_{jk}\ (\text{or } \bar{c}_{kj}) \geq \alpha\bar{c}_{ij}$ where α is a threshold which is smaller than, but close to one. If the *k* is found, we will consider this as a critical point and *ij*, *jk*, *ik* will be recorded as three possible computing directions at this merge stage.

3.2 Discussion

The main advantages of our quartet-based algorithm [14, 15] are that:

(i) It is based on a one-to-one correspondence between a tree topology and an associated set of quartet trees, i.e., if the topologies of the quartet trees are correctly identified, the correct phylogeny can be reconstructed in polynomial time;

(ii) It can accurately identify critical points at which the tree merge ambiguity occurs. At a critical point, one of two or three merge options may lead to the correct tree reconstruction. This allows users to dynamically incorporate different methods (both automatic and/or manual) to come out with a better decision;

(iii) It is an agglomerative procedure check points can easily be set at the end of each merge stage, thus it can be easily adapted for dynamic integration and interaction.

The experimental results show that the probability for the correct tree to be included in just a limited number of trees constructed by our method is very high [15]. When we selected just a few best ones (say three) from these trees under ML criterion, extensive tests showed that the algorithm outperforms many other algorithms and methods for phylogeny reconstruction in terms of tree topology. If we can effectively incorporate different methods and/or even other biological information to clear ambiguity at each critical point, we may obtain more accurate and reliable results for real-life problems. For this purpose, a new phylogenetic analysis system with an interactive environment, which provides a helpful interface for dynamic integration by allowing computation to progress back and forth to different merge stage to control the quality of analysis, is proposed.

Chapter 4 System Design

This chapter describes the prototype implementation of the Qphyl System. The Qphyl system, which is a web-based phylogenetic analysis system based on our quartet-based algorithm, uses a new interactive approach to enable biologists to effectively interact with the computation. The chapter will start with the workflow analysis, followed by the functional analysis. Then the architecture of the system will be introduced along with a detailed description of the distributed system which is based on the United Devices Grid MP Platform [4].

4.1 Workflow Analysis

The general requirement of the Qphyl System is molecular phylogeny reconstruction from a set of character-aligned DNA or amino acid sequences. As illustrated by Figure 4.1, the procedure to automatically generate a phylogenetic tree from a set of aligned of DNA sequences is as follows:

(i) A quartet weight generation program using ML to generate quartet trees from the input sequences, and each quartet is associated with three quartet weights for three possible resolved topologies.

(ii) A quartet-based tree reconstruction program reads the quartets generated previously to create a global quartet weight matrix, and recursively merges one pair of sub-trees at each step based on the global quartet weight matrix, until only a single tree remains. It should be noted that when every time two sub-trees are merged, the weights of quartets associated with the sequences belonging to the merged sub-trees will be corrected.

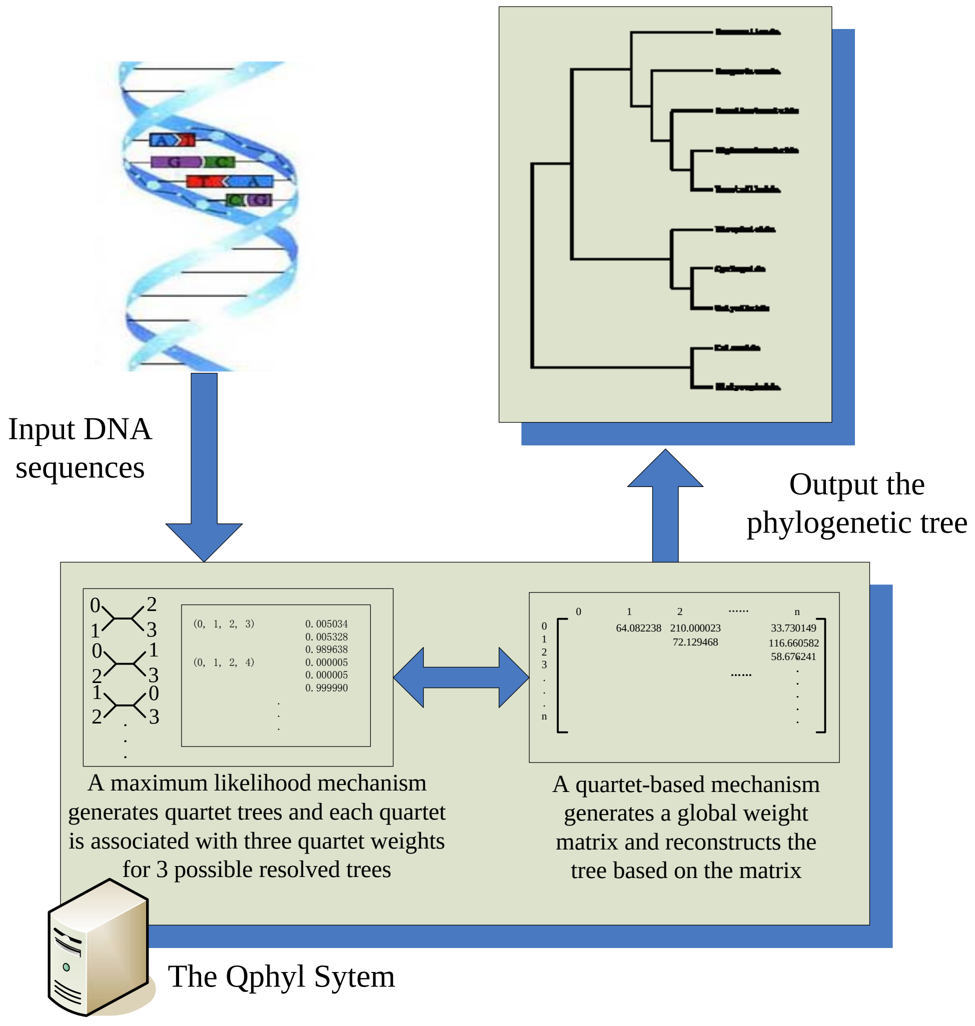

Input DNA sequences
Output the phylogenetic tree
(0, 1, 2, 3)
(0, 1, 2, 4)
0. 005034
0. 005328
0. 989638
0. 000005
0. 000005
0. 999990
64.082238
210.000023
72.129468
33.730149
116.660582
58.676241
A maximum likelihood mechanism generates quartet trees and each quartet is associated with three quartet weights for 3 possible resolved trees
A quartet-based mechanism generates a global weight matrix and reconstructs the tree based on the matrix
The Qphyl Sytem

Figure 4.1: Workflow about how to automatically generate a tree from a set of aligned DNA sequences through the Qphyl System

The desired interactive system is meant to allow users to interact with the computation for phylogeny reconstruction. The interaction consists of four essential functions which could be extended to more powerful and complicated ones. These functions involve step forward, step backward, manually joining and manually disjoining. With the embedded web-based interface, the procedure to generate a phylogenetic tree is much more flexible but the workflow is more complicated. Figure 4.2 shows the complete sequence of events that occurs during the phylogeny reconstruction.

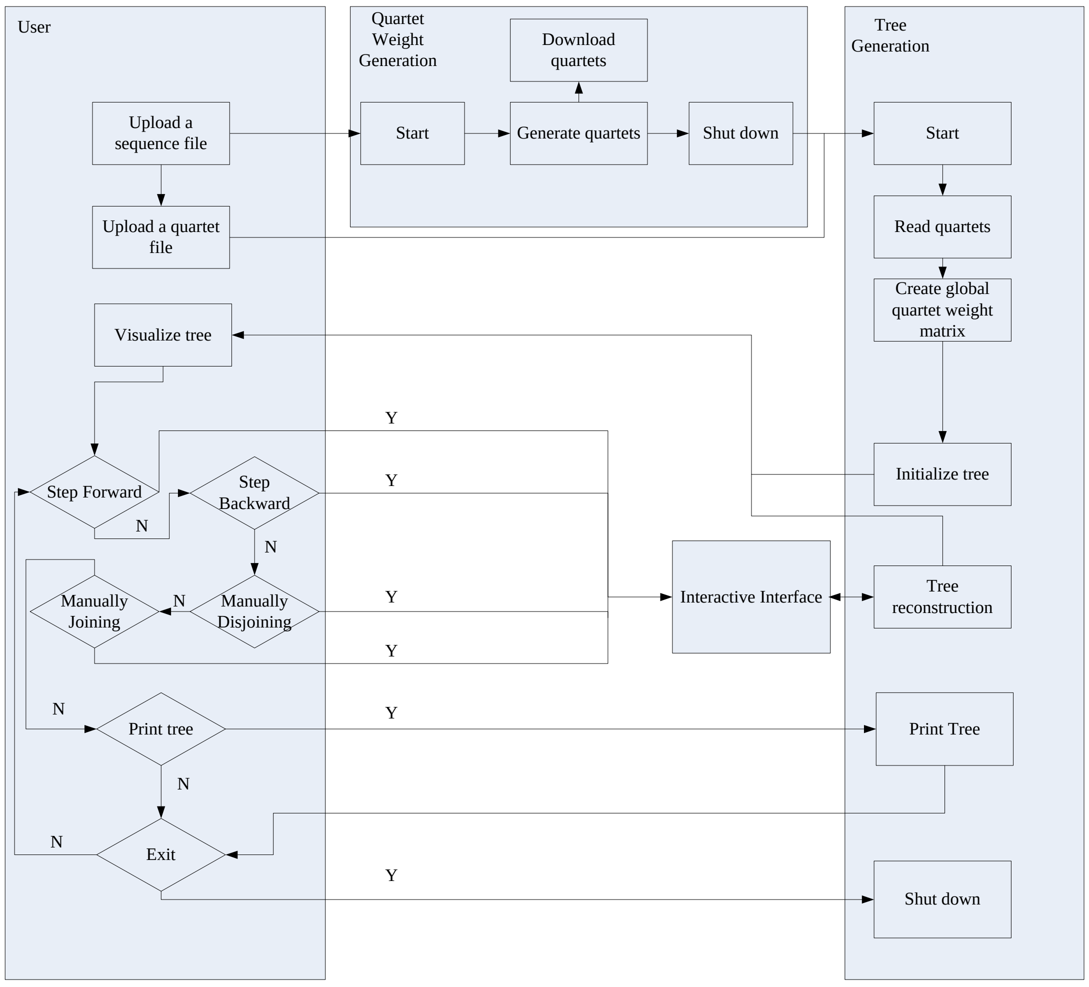


Figure 4.2: Complete workflow of the Qphyl System to generate a phylogenetic tree

Details are described as follows:

(i) Prior to the tree reconstruction, the system refers users to upload a sequence file to build a tree. As described previously, the first step of tree reconstruction is to generate complete set of quartets with quartet weights from uploaded sequences by the quartet weight generation program.

(ii) Once the quartets are completely generated, the tree reconstruction program is triggered to start up, and the quartet weight generation program transfers the computational results as input to the tree reconstruction program and shuts itself down.

(iii) Also, users can skip the quartet generation step by uploading a sequence file and the corresponding quartet file to the tree reconstruction program directly.

(iv) When the tree reconstruction program has been turned on, it reads the quartets and creates the corresponding global quartet weight matrix.

(v) The tree is initialized based on the initial global quartet weight matrix, and the status is graphically represented to users.

(vi) Then user can make decisions on computational direction and manually join or disjoin two nodes via the interactive interface. Each stage of the tree reconstruction after a specific operation is visualized to users.

(vii) After one specific operation, it is also available for users to download a text file of current tree topology in Newick format.

(viii) Since the previous computational status can be retrieved and an intermediate node can be manually disjoined even if the whole tree has been completely created, the triggers to shut down the tree reconstruction program only include: 1) user starts a new job; 2) the communication between the client-side and server-side is disconnected.

4.2 Functional and Non-functional Requirements

After examining the workflow of the Qphyl System, we can move our concentration to functional, interface design, and performance requirements. As discussed in the previous section, different interactive operations are involved into the tree reconstruction process. Therefore, it is desired a powerful tree visualization tool to make sure every stage of the tree reconstruction is clearly, accurately and distinctly represented.

Furthermore, our quartet-based algorithm is both compute and memory intensive [13]. However, as a web-based system, it is required to synchronously handle multiple tasks, efficiently respond and execute client-side requests, and complete in an acceptable time.

4.2.1 Functional Requirements

From the workflow analysis, a profile of the desired characteristics of the Qphyl System is created. These are listed as follows:

(i) User can create a job by either uploading a sequence file or uploading a sequence file and the corresponding quartet file.

(ii) Once the job is created, user can dynamically interact with the tree reconstruction. The basic operations include: step forward; step backward;

manually join and manually disjoin.

(iii) When executing stepping forward command, user can choose to stop computing or not if critical point occurs.

(iv) User can redirect the computational direction at critical points.

(v) For user’s convenience, previous stage of the tree reconstruction before the latest operation can be retrieved and the tree topology can be recovered.

(vi) User can download and print the tree topology in both text format (the Newick format) and graphical format.

(vii) User can download the quartet file after quartet generation.

4.2.2 User Interface Design Requirements

The key requirements for the user interface in terms of tree visualization and optimizing efficiency and usability include:

(i) The tree topology can be graphically represented at client-side.

(ii) For tree graphical representation, the computational information, including step number of each merge and confidence value of each merge, can be displayed at client-side.

(iii) For tree graphical representation, regular computational joining, critical point and manually joining can be distinguished by users at client-side.

(iv) For user’s convenience, after one operation, the modification on tree topology representation should be minimized.

(v) All the other information should be arranged on the screen in a way that helps users to focus on the key data, and hence make the correct judgment.

(vi) The tasks such as file uploading and downloading, manually joining and disjoining two nodes, stepping forward and backward should be performed in a straightforward manner.

4.2.3 Performance Requirements

We use the performance and capability requirements defined by the PHYLIP.

(i) The system should have the capability to efficiently analyze at least 11 600-site species. (PHYLIP declares that it may not be practical for more than 10-11 species to analyze by it based on ML criterion [52]).

(ii) The system should have the capability to handle multiple tasks parallelly. Based on users' sense, the response time should be acceptable.

4.3 Architecture Design

There are several design considerations in place when designing the system. First of all, such system should offer an efficient means of viewing, analyzing, and documenting phylogeny results. Secondly such system should be able to fleetly respond users' requests and make correct decisions. Thirdly such system can handle multiple executions parallelly. We will go through the design of both hardware and software components in this section.

4.3.1 System Architecture

The Qphyl System consists of a web-based user interface, a web server and a distributed system. The overall layout can be illustrated in Figure 4.3.

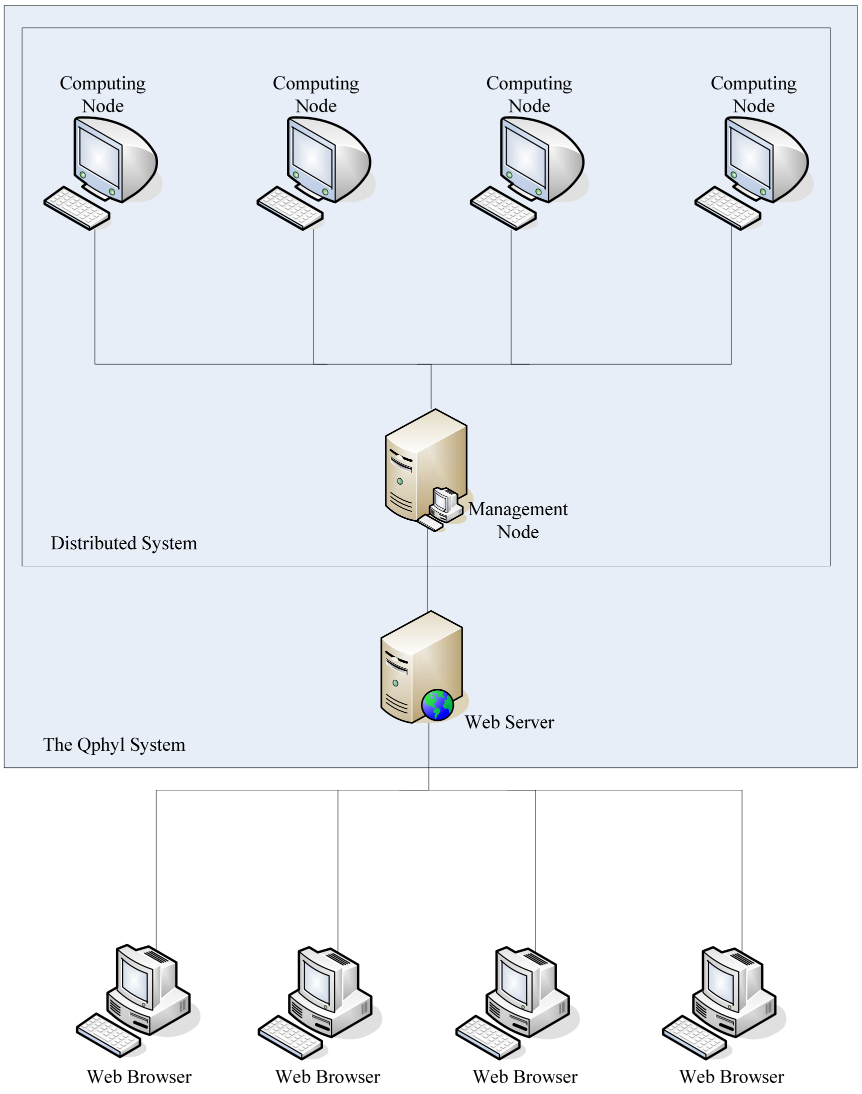


Figure 4.3: Overview of system architecture

The web server is responsible for accepting HTTP requests from users and serving them HTTP responses along with optional data contents. As the intermedium for users to communicate with the phylogeny computations, web server provides an interface to connect web-based user interfaces and the phylogeny reconstruction services.

The distributed system is for handling multiple tasks. The management node is a facility used to find available computing resources and allocate tasks to the backend computing nodes. During the computation it is responsible to send operational parameters to particular computing nodes, which are executing the corresponding tasks, and to return the computational results with identifications back to the web server in order to inform corresponding users. Once it captures the request to terminate a task, it locates the corresponding computing node and kills the process executing on it.

The backend computing nodes is a set of computing resources for phylogeny reconstruction. Their responsibility is to analyze phylogenies from sets of aligned sequences uploaded by users. The status of these computing nodes is detected by the management node and the output of phylogeny analysis is returned to users via the management node and web server.

This distributed system will be detailedly introduced in section 4.4.

4.3.2 System Overview

The system is split into several different components. Each component is responsible of handling certain tasks for different purposes. As we can see from the logical view shown in Figure 4.4, the tree visualization is executed by web browser at client-side in order to reduce the burden of the computation at server-side. It dynamically parses the phylogenetic analysis results, which are in XML format, and graphically represents them to users. It is also responsible to generate synchronous and asynchronous requests for interaction between users and backend computations. This interactive and dynamic properties exhibited by user interface are developed using AJAX and DHTML.

To start a new task, user uploads a sequence file or a sequence file with corresponding quartet file to the phylogeny service via uploading servlet. Once the uploading is completed, the phylogeny service node creates a unique session token of the task to identify the source of the task for further interaction, and allocates the task to an available backend computing node execution engine with this identification via the resource broker to start the phylogeny analysis. After the phylogenetic tree is completely initialized, the execution engine transfers the results, which is in XML format, to the service node. Then the service node returns this XML file including the session token back to the client who sent the request.

Once the client side has received the XML file of the initialized tree and the session token, the visualization tool parses the XML file and graphically represents the initial tree on the web browser. The following requests about interaction between the user and the backend computation are instantiated to parameters and transmitted in GET queries with the session token from client side to the web server, and the web server sends them to the phylogeny service node. Every time the service node receives a request about interactive operation, it identifies the backend computing node working on the corresponding task and triggers the execution of computation based on the request. After the execution has completed, the result is sent back to the client side in the same way as in the task initialization, and the visualization tool parses the XML file and then modifies the graphics of the tree topology.

User can start a new task or close the web browser to kill the current interaction process. Once the process is killed, the resources of the backend computing node are released and the session is closed.

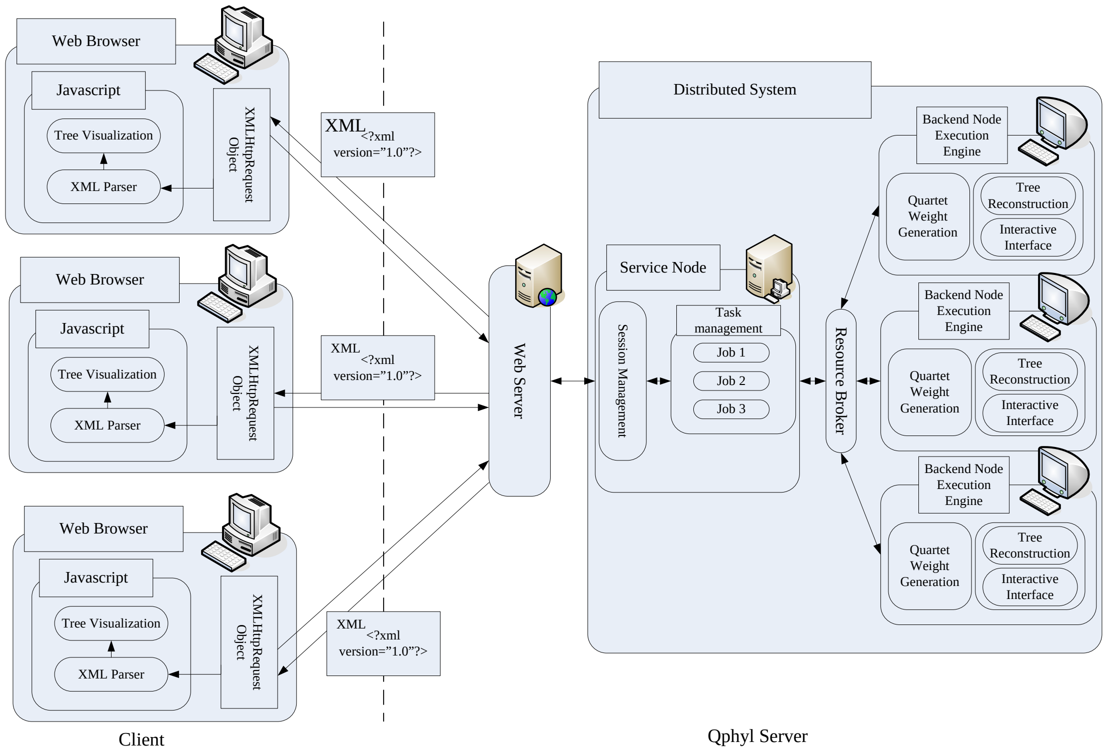


Figure 4.4: Overview of the system design

4.4 Distributed System

The distributed system, which uses a grid environment to handle multiple processes, consists of a management node, various numbers of computing nodes and a resource broker in between. The roles of the resource broker are to keep the information about available computing resources once registered there and to provide the information to the management node upon request. Once being contacted, an available computing node will receive the task from the management node, execute the task, and send the result back to the management node. The management node is responsible for contacting the resource broker for available resources, sending tasks to and receiving partial results from computing nodes.

To detect the actual performance of computing nodes, the management node maintains another list which records all the nodes the tasks were assigned before. When a computing node is first contacted, it will be recorded in the list and a small task is assigned to it and the start time (when the task is sent) is recorded. When the result is returned, the finish time (when the result received) is also recorded. The difference between the start and finish times will be considered in determining the size of the next task to be assigned to the node. If the computing node can finish the task quickly and immediately become available again, it will be assigned a larger task. When a computing node cannot finish the task fast enough, the size of the next task will be decreased. By detecting the actual performance of computing nodes, we are able to assign tasks of proper sizes to them and then make the computation more effective and efficient.

Once a task is created by reading a sequence file, a session between the client-side and server-side with a unique id is established. Every request from client-side is identified by the unique id and will be responded by the computing node which is handling the corresponding task. Thus, this distributed system can adequately satisfy multiple procedure requests and finally improve the robustness and flexibility of phylogeny inference.

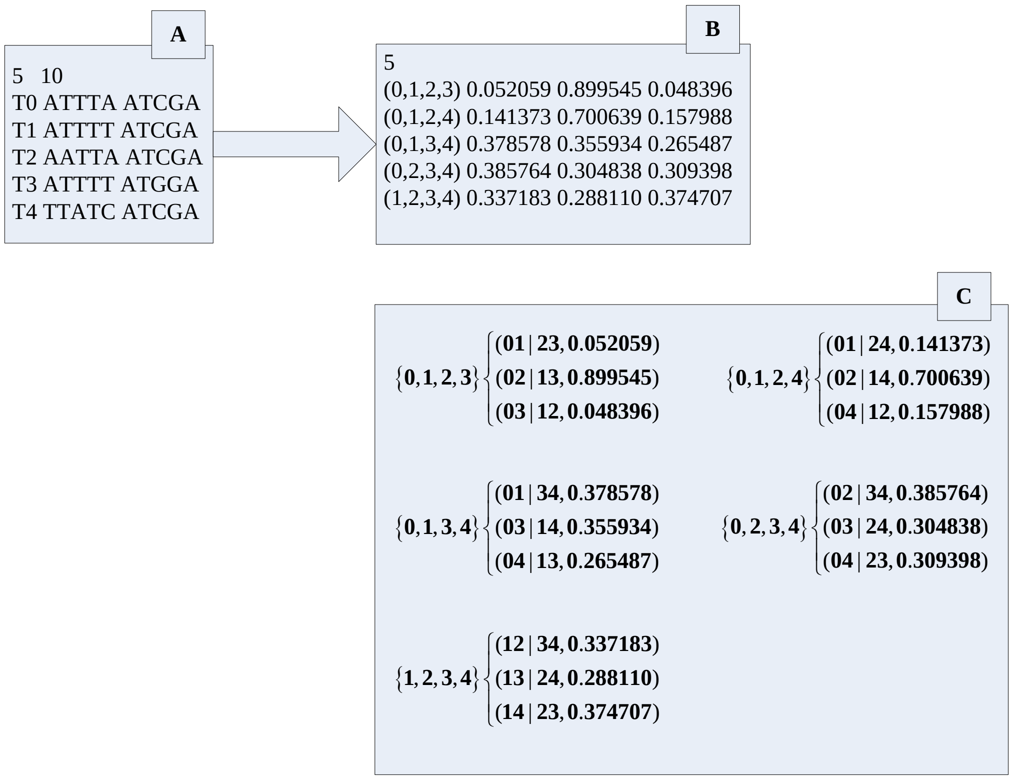
Chapter 5 Backend Programs

There are two back-end programs: a quartet weight generation program using ML and a tree reconstruction program using the quartet-based algorithm. The tree reconstruction program needs the output of the quartet weight generation as input for the tree reconstruction. An interactive interface is embedded into the tree reconstruction program to handle the interaction between users and phylogeny computations. In this chapter, we will introduce them respectively, and describe how computer memory is dynamically managed during interactions.

5.1 Quartet Weight Generation

The first stage of tree reconstruction by our quartet-base algorithm is quartet weight generation. Quartets are reconstructed from a set of aligned DNA, or Amino Acid sequences. We use ML as criterion because simulation studies show that ML methods generally outperform distance and parsimony methods over a wide range of conditions [28]. The quartet generation program first uses the functions from the likelihood-mapping procedure [29] to calculate the ML values for each of the three possible resolved trees and these likelihood values are then transformed into posterior probabilities, or Bayes weights.

A simple example of the quartet weight generation from a given set of aligned sequences is shown in Figure 5.1. Box A includes the content of 5 aligned sequences of 10 sites. Each sequence has a unique name. At beginning of the quartet generation procedure, these sequences are allocated with a unique id, from 0 to 4, to identify the sequences for both quartet generation and tree reconstruction. Box B represents the result of the quartet generation, and this output will be used as input to the tree reconstruction. Since there are three possible fully resolved trees for a quartet (*a*, *b*, *c*, *d*), we calculate w_i for i = 1, 2, and 3, by applying Bayes' theorem assuming a uniform prior for all three possible trees of a quartet. As illustrated in Box C, the expression of (*a*, *b*, *c*, *d*) w_1 w_2 w_3, is corresponding to $\{a,b,c,d\}\begin{cases}(ab\,|\,cd, w_1)\\(ac\,|\,bd, w_2)\\(ad\,|\,bc, w_3)\end{cases}$ which describes the three possible topologies with quartet weights.



```
5   10
T0 ATTTA ATCGA
T1 ATTTT ATCGA
T2 AATTA ATCGA
T3 ATTTT ATGGA
T4 TTATC ATCGA
```

```
5
(0,1,2,3) 0.052059 0.899545 0.048396
(0,1,2,4) 0.141373 0.700639 0.157988
(0,1,3,4) 0.378578 0.355934 0.265487
(0,2,3,4) 0.385764 0.304838 0.309398
(1,2,3,4) 0.337183 0.288110 0.374707
```

$$\{0,1,2,3\}\begin{cases}(01\,|\,23, 0.052059)\\(02\,|\,13, 0.899545)\\(03\,|\,12, 0.048396)\end{cases}\qquad \{0,1,2,4\}\begin{cases}(01\,|\,24, 0.141373)\\(02\,|\,14, 0.700639)\\(04\,|\,12, 0.157988)\end{cases}$$

$$\{0,1,3,4\}\begin{cases}(01\,|\,34, 0.378578)\\(03\,|\,14, 0.355934)\\(04\,|\,13, 0.265487)\end{cases}\qquad \{0,2,3,4\}\begin{cases}(02\,|\,34, 0.385764)\\(03\,|\,24, 0.304838)\\(04\,|\,23, 0.309398)\end{cases}$$

$$\{1,2,3,4\}\begin{cases}(12\,|\,34, 0.337183)\\(13\,|\,24, 0.288110)\\(14\,|\,23, 0.374707)\end{cases}$$

Figure 5.1: A simple example of quartet weight generation from a set of 5 aligned 10-site sequences

The quartet generation program can handle input sequence file in both interleaved and sequential formats [42]. A bunch of substitution models [45] are provided to satisfy most of the phylogenetic biologists with diverse preferences. For DNA sequence, the available substitution models involve HKY85, F84, TN93, GTR, custom, JC69, K80 and F81, while substitution models for Amino Acid sequences include JTT, MtREV, WAG, DCMut, RtREV, CpREV, VT, Blosum62, MtMam and Dayhoff.

5.2 Tree Reconstruction

The tree reconstruction program uses our quartet-based algorithm to analyse the phylogeny of given sequences based on the quartet weights generated by the quartet weight generation program. As described in Chapter 3, a global quartet weight matrix is generated at the first stage, and based on this matrix, the tree is recursively reconstructed.

As displayed in Figure 5.2, the quartet weights in previous example in Figure 5.1 are integrated to a single symmetric global quartet weight matrix of size 5×5, each row or column corresponding to one particular sequence.

D

$$\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & \mathbf{0.000000} & \mathbf{0.572010} & \mathbf{1.905022} & \mathbf{0.709168} & \mathbf{0.732873} \\
1 & \mathbf{0.572010} & \mathbf{0.000000} & \mathbf{0.543567} & \mathbf{1.453142} & \mathbf{1.431280} \\
2 & \mathbf{1.905022} & \mathbf{0.543567} & \mathbf{0.000000} & \mathbf{0.736164} & \mathbf{0.734321} \\
3 & \mathbf{0.709168} & \mathbf{1.453142} & \mathbf{0.736164} & \mathbf{0.000000} & \mathbf{1.101525} \\
4 & \mathbf{0.732873} & \mathbf{1.431280} & \mathbf{0.734321} & \mathbf{1.101525} & \mathbf{0.000000}
\end{array}$$

Figure 5.2: A simple example of global quartet weight matrix corresponding to the example in Figure 5.1

Based on this global quartet weight matrix, the tree reconstruction program progressively identifies and merges two most closely related sub-trees, by using the equation 3.1 to calculate confidence value of each possible combination at every stage, until there left only a single tree.

To practically identify a critical point, the tree reconstruction program uses following method to dynamically assign a value to α :

If c_{ij} is the highest confidence value AND c_{ij} > 0.9

$$\alpha = \frac{\mathbf{3}c_{ij} \mathbf{-1}}{\mathbf{2}}$$

ELSE

$\alpha = \mathbf{0.85}$

Based on this α value, then tree reconstruction program checks whether $\bar{c}_{ik}$ **(or** $\bar{c}_{ki}$**)** $\geq \alpha\bar{c}_{ij}$ or $\bar{c}_{jk}$ **(or** $\bar{c}_{kj}$**)** $\geq \alpha\bar{c}_{ij}$, in order to find out at this step whether there exists another sub-tree *k* that has a reasonably high confidence value associated with one of the two sub-trees *i* and *j*. If such sub-tree *k* is found, this step will be considered as a critical point and three merge options will be saved for further identification and treatment.

Figure 5.3 demonstrates an unrooted phylogenetic tree which is generated by the tree reconstruction program from the sample set of sequences in the previous example in Figure 5.1. The tree reconstruction program merged T0 and T2 with confidence value 0.661983 at the first step, and then merged T1 and T4 with confidence value 0.576880 at the second. It should be noted that the second step is considered as a critical point by the program and highlighted in red color in this figure.

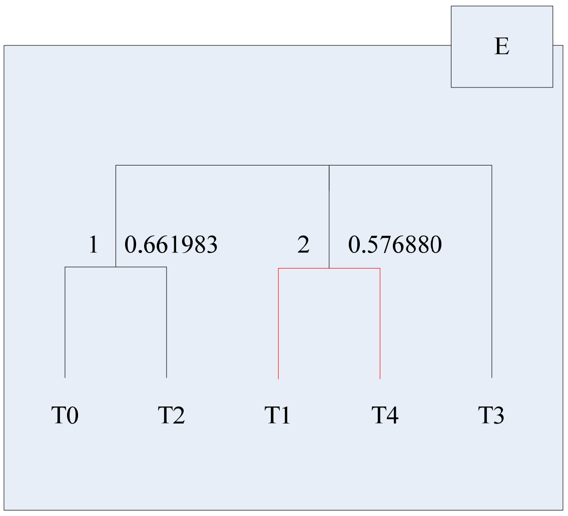


Figure 5.3: A 5-taxon unrooted tree from the sample sequences in Figure 5.1

5.3 Interactive Interface

As displayed in Figure 5.4, an interactive interface is embedded into the quartet-based algorithm for the purpose of dynamic interaction between user and the computation to control the agglomerative (or progressive) computing directions.

The interaction consists of four essential functions which could be extended to more powerful and complicated methods. These functions involve step forward, step backward, forcibly joining and forcibly disjoining.

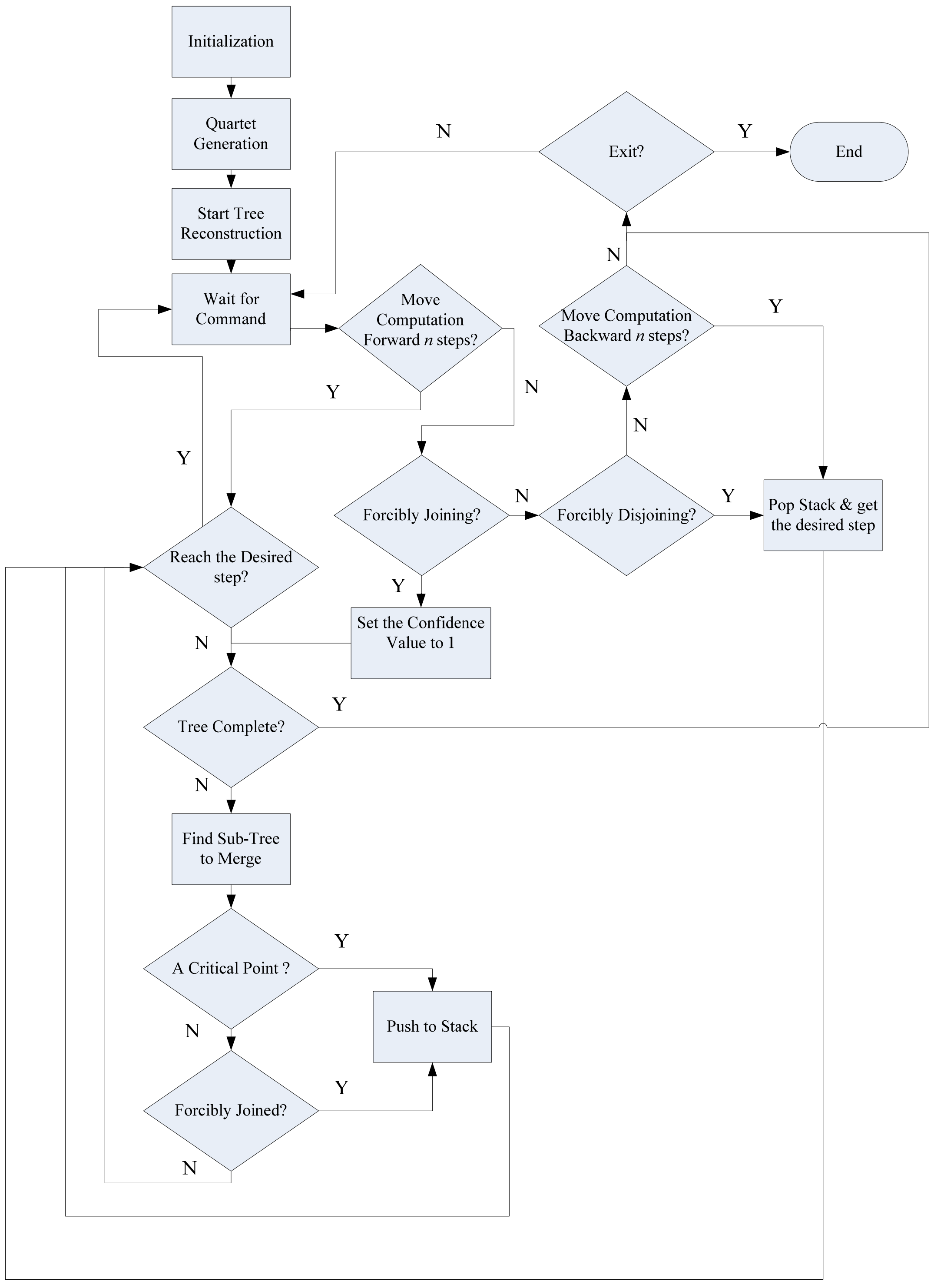


Figure 5.4: Workflow of the quartet-based algorithm with interactions

5.3.1 Computing Direction Control

Step forward is a basic operation required to execute the quartet-based algorithm to progressively reconstruct phylogenetic tree. A 1-step-forward operation, which is a normal step in the quartet-based algorithm, merges two sub-trees with the highest confidence value into one bigger sub-tree. An *n*-step-forward operation is composed of n 1-step-forward operations. If the whole tree has not been generated after or during *n*-step-forward operation, the total number of sub-trees becomes $n' - n$ where n' is the number of sub-trees before the operation. Since it takes $k - \mathbf{3}$ steps to generate a complete unrooted tree for k sequences, once n' is equal to 3, the rest 1-step-forward operations will be cancelled. Thus, in our interactive approach, if $n' \geq n$, the *n*-step-forward operation will be completely executed, else if $n' < n$, the computation will move forward n' steps.

Step backward, which is for retrieving previous computational stages, is an opposite operation to step forward. There are various ways to achieve the function of step backward. The first one might be to record every stage of computation including the quartet weight matrix, tree topology and a unique id for further retrieval. Because the quartet-based algorithm is agglomerative which means the following decisions are dependent on the previous ones, using stack to save these data is the most efficient way. When n-step-back is implemented, the stack will be traversed from the top to the bottom until the desired step has been found, and then, the data at this step will be restored.

However, in the quartet-based algorithm we have to calculate a number of $\mathbf{O}(n^4)$ quartet weights for all possible quartet trees first and all these weights need to be stored in the memory and used in the tree reconstruction stage. Each quartet is associated with three weights for three possible resolved trees. For a given set of *n* sequences, at each step the memory required will be $\binom{n}{\mathbf{4}} \times \mathbf{3} \times \mathbf{8}$ bytes in size, assuming a double variable is used for each quartet weight. Thus, to save every step of computation is memory intensive.

Another way might be to redo the computation from the starting point to the desired step. In this way instead of pushing all the stages into the stack, we need to save only one matrix. This can significantly save the computing resources but may take more time than the first method in some situations (e.g., 1-step-backward at the 100th step). One other significant problem is that the previous manually refined information will simply get lost after each recalculation.

For efficiently retrieving the previous step, in our interactive approach, we still save the internal computation steps into stack. Instead of recording all the steps' information, we execute push-into-stack after a certain number of steps since the last time we execute this. To retrieve a previous step, the computation will be re-executed from the step which is generated before the needed step and the closest to it. Then execute *n*-step-forward where *n* is the difference between the step-number of desired step and the step-number of jumping-off point of the new computation. If the desired backward step number is larger than the amount of executed steps, the interactive interface will retrieve the initial tree topology and foremost matrix.

5.3.2 Forcibly Joining and Disjoining

Forcibly joining is used to modify the tree topology and can be triggered by both automatic and manual operations. The forcibly joining in the interactive approach is quite straightforward: since the merge decision in our quartet-based algorithm is based on the confidence value (the value is between 0 and 1) and the possible combination which has a highest confidence value will be taken, we can set the confidence value of the desired combination to 1, and then this combination will be chosen to be merged by the quartet-based algorithm automatically.

It should be noted that forcibly joining has to been saved for the future retrieval. If we don't do that, when we try to go back to a step which is opportunely a forcibly joining, and this step is not saved in our interactive approach, the correct step won't be retrieved if the confidence value of this combination is not the highest one on this step before forcibly setting the confidence value of it to 1.

The confidence value saved in is still the "true" confidence value of this mergence before setting it to 1.

The essential of forcibly disjoining is to step backward. This is based on the assumption that the information which is generated after a specific combination will be useless if this combination is eliminated. This is in respect that the quartet-based algorithm is an agglomerative approach.

So when try to disjoin two nodes forcibly, we need to know the step index of this combination. Once the desired step is located, the step backward is executed to help us to disjoin two selected nodes.

5.3.3 Critical Point

Critical point is one of the most important concepts in the quartet-based algorithm because it defines ambiguities of the phylogeny reconstruction. When critical point occurs, the quartet weight matrix and three mergence options with confidence values should be stored for further redirection.

To redirect the computation at critical point, the computation should step backward to the step which is one step before the critical point, and pick up a new direction by forcibly joining the two nodes. Then following computations will be dependent on this new direction which means new computing redirection will significantly change tree topologies. Thus biologists can use the interactive platform to do experiments and compare the visible results to find the desired tree.

5.3.4 Interactive Operations in Qphyl

Based on the four fundamental operations, we can build a robust and flexible platform with plenty of sophisticated and powerful options. Table 5.1 shows the available interactive operations and corresponding parameters in the Qphyl System.

Param0	Param1	Param2	Operation Description
0	n(step number)	k(1 means stop when critical point occurs; 0 means straightforward but save the critical point)	Step forward
1	n(step number)	0	Step backward
2	Node1	Node2	Join node1 and node2
3	Node1	Node2	Disjoin node1 and node2
4	0	0	Refresh the tree
5	0	0	Undo
6	Node1	Node2	Get the confidence value of the possible combination between node1 and node2
7	Step id (the id of combination which is a critical point)	Direction(0, 1 or 2)	Redirect the computation at critical point
8	0	0	Exit
9	0	0	Save the current stage in TXT format

Table 5.1: Options and associated parameters for interaction

Basically, the first parameter, Param0 is the index of the operations. To make the computation forward, user assigns 0 to Param0 and the desired number of forward steps to Param1. It should be noted that critical point, which is very crucial in quartet-based phylogenies, may occurs during the step forward execution. User can ask the interactive interface to stop the computation automatically when critical point happens, by setting the value of Param2 to 1. If Param2's value is equal to the default value, 0, the computation keeps on executing to merge the sub-trees recursively until the desired number of forward steps has been achieved. However, every three possible combinations corresponding to critical points happened during the execution are still stored and available to be retrieved later.

To make the computation backward, user assigns 1 to Param0 and desired number of steps to Param1. Param2's value, which is equal to 0 by default, doesn't affect the execution.

To manually join and disjoin two sub-trees, Param0 should be set to 2 for joining and 3 for disjoining. Param1 and Param2 are two representative leaves of the sub-trees respectively. It should be noted that, the two sub-trees are allowable to be merged only if both of them have no ancestors in current computing status, and the two sub-trees can be disjoined only if they have a same ancestor.

The command to refresh a tree is executed only if the Param0's value is equal to 4.

The purpose of this option is to retrieve the tree topology from the computing-node in order to recover the client-side information if errors happen during the data transmission.

Undo operation is available if latest operation was not “undo”. Using “undo” you can cancel the last operation and recover the tree to previous stage. To “undo” last operation, user just sets Param0 to 5, and leaves Param1 and Param2 with default values.

Before manually joining two sub-trees, user can get the confidence value of this combination by setting Param0 to 6, and Param1 and Param2 to the index of representative leaves of the two sub-trees.

By assigning 7 to Param0, the index of step where critical point occurs to Param1, and the id, from 0 to 2 (each of them denotes a specific computation direction), to Param2, user can redirect the computation at a particular critical point.

To terminate the interactive process, the three parameters should be equal to 8, 0, 0 respectively.

In addition, user can order the tree reconstruction program to create a text file recording the current tree topology in Newick format [43] by inputting three parameters with 9, 0 and 0 respectively. Since the data structure of a node can be simplified as follows:

```
typedef struct node {
node *leftdesc, *rightdesc, *ancestor; // left and right descendants and ancestor
int node_id;                           // the index of the node
}
```

The tree topology can be efficiently traversed by using recursive function calls. Suppose we want to move through the tree, carrying out for each record *p* a function f(*p*), a postorder tree traversal [46] from node *p* does the following:

(i) Traverses *p*'s left descendant (if any).

(ii) Traverses *p*'s right descendant (if any).

(iii) Carries out function f*(p)* at *p*.

For each node, it does not have f(*p*) done until this has been done for all its descendants.

By using this preorder tree traversal, the task to print the tree topology in Newick format can be very simple. The unrooted tree topology in Figure 5.7 can be expressed as follows:

((T0,T2),(T1,T4),T3);

5.3.5 Memory Management

Since our quartet-based algorithm is both compute and memory intensive [13], it is critical to efficiently manage computer memories in the Qphyl System by considering the usability.

As described in section 5.3.1, the interactive interface uses stack to save current step's tree topology and matrix, under the following conditions:

(i) 1-step-forward operation has been executed for *k* times, i.e., if *l* steps have been executed, and *l* is a multiple of *k*, the push-into-stack operation is executed.

(ii) The current step is a manual mergence.

(iii) The current step is a critical point.

For details, the data we have to save into the stack for further retrieval include:

(i) the global quartet based matrix at this step

(ii) the tree topology with the information about how every 2 sub-trees are merged (regular mergence, manual mergence or a critical point) in previous steps

(iii) a unique id equal to the step number on which the push-into-stack operation is executed

If we want to retrieve a particular step executed previously, we use an algorithm, described by the following pseudo code, to locate the target:

```
------------------------------------------------------------------------------------------------------
$n = n_c - n_d$                  // $n_c$ is the current step number before retrieval;
                                 // $n_d$ is the number of backward steps desired to be
                                 //executed
While $id > n$                   // $id$ is the number of step on which the push-into-stack
                                 //executed
    $Stack.top--$                // Move the pointer to the next
Pop Stack                        // restore the matrix and tree topology
Set $n' = Stack.id$
($n - n'$)-step-forward
------------------------------------------------------------------------------------------------------
```

In addition, the interactive interface also allocates an independent area to save the latest status, including the matrix and tree topology, before a particular operation. If user wants to undo the last operation, the data in this area will be reloaded to cover the current matrix and tree topology in order to restore the previous status.

Chapter 6 Web-based GUI Frontend

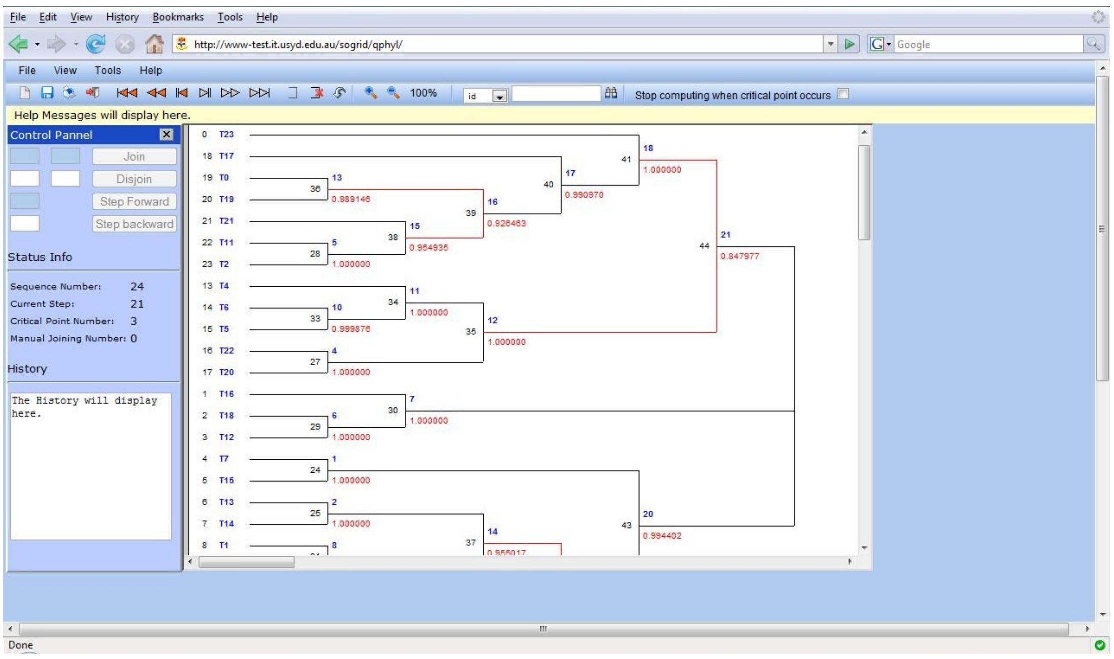
The responsibilities of the web-based GUI are to provide a graphical interface for users to interact with the backend computations and to parse and visualize the modification of tree topology after a specific execution. This chapter will start with the user interface design and tree visualization, followed by the protocols for communications. The result of the system will be presented by a given example at the end of the chapter.

6.1 User Interface Design

AJAX technology enables faster, more responsive web applications through a combination of asynchronous Javascript, DOM and XMLhttpRequest. Updates can be quickly and incrementally executed to a user interface without reloading the entire screen. In the AJAX model JavaScript calls to the server can update a single element in the UI with data retrieved from a server. Because the full screen does not need to be reloaded, the application is much more responsive to user actions.

As an interactive system, it is crucial that the status of backend computation can be correctly and distinctly exhibited to users. By considering the usability characteristics, all the operational options are arranged and constructed as a structured form input screen. As displayed in Figure 6.1, the environment of phylogenetic analysis is established on an integrated webpage, which can be divided into 3 parts, a toolbar including all the options, a control panel displaying the computation status information, and a canvas used to graphically represent tree topologies.

It can be viewed from the tool bar that the interactive operations include: re-initialization, *n*-step-back, 1-step-back, 1-step-forward, *n*-step-forward, auto-generation, manually joining, manually disjoining and undo. All these operations are established based on the four basic operations in the interactive approach: step forward, step backward, forcibly joining and forcibly disjoining.


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Google
File View Tools Help
100%
id
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
Status Info
History
Done

Help Messages will display here.

Sequence Number: 24
Current Step: 21
Critical Point Number: 3
Manual Joining Number: 0

The History will display here.

Figure 6.1: GUI of the Qphyl System

In addition, the interaction demands that all the requests sent to the computing nodes should be legal. For example, if the whole tree has been reconstructed, the step forward request is illegal and should be rejected. A mechanism to identify status of the computation is installed at client-side to avoid illegal operations. Basically, the mechanism enables and disables options based on the phases of tree reconstruction. There are three possible phases of tree reconstruction, which are tree initialization, execution of tree reconstruction, and completion of tree reconstruction. For instance, during the execution of tree reconstruction, all the options are valid, but to step forward and to merge two nodes options will be disabled once the whole tree has been reconstructed.

To get the phase of tree reconstruction, the GUI simply checks the current step number n_c: if $n_c = \mathbf{0}$, the phase is tree initialization; if $\mathbf{0} < n_c < k - \mathbf{3}$, where *k* is the number of sequences for phylogenetic analysis, the phase is executing tree reconstruction; and $n_c = k - \mathbf{3}$ indicates the current phase is completion of tree reconstruction.

Furthermore, for some particular options, further identifications about legality of operation are executed. These identifications include:

(i) The number of forward steps should be less or equal to the total number of rest steps to complete the tree reconstruction from current stage.

(ii) The number of backward steps should be less or equal to the number of executed steps.

(iii) Neither of the nodes can have an ancestor if user requests to manually join them.

(iv) The two nodes should have the same ancestor if user requests to manually disjoining them.

6.2 Tree Visualization

A bunch of algorithms for drawing unrooted trees have been designed and implemented. The most popular ones include the equal-angle algorithm [25], n-body algorithm [47] and the equal-daylight algorithm [23]. Basically these algorithms read a generated tree file (commonly in Newick tree format) and draw the tree in one time. However these algorithms are not appropriate for the interactive system mainly because the tree is reconstructed progressively and the tree topology is always transformed during computations. We need a more efficient way which can modify previous stage just based on the changes of topology after an execution. Thus, a specific tree visualization algorithm, which focuses on the topology modification, is designed and implemented in the system. We will use the set of five sequences in Figure 5.1 as example to describe how the algorithm works.

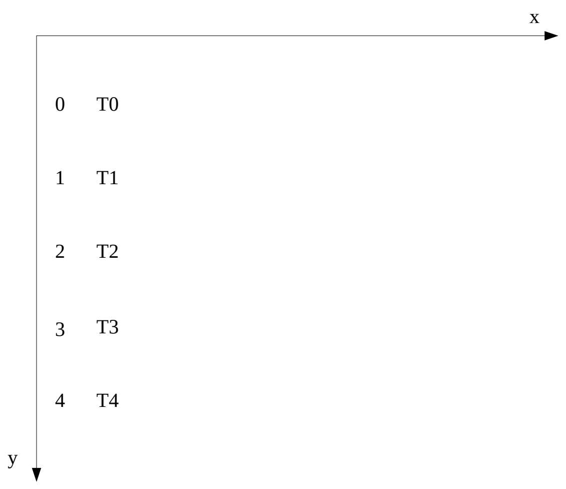


Figure 6.2: Initialized tree topology of 5 sequences

As displayed in Figure 6.2, the leaves are vertically arranged based on their indexes in a Cartesian coordinate system. The leaf with smaller index is located upon the one with larger index. Basically, once computation has moved forward by *n* steps, *n* corresponding intermediate nodes are created recursively and they are connected with their direct descendants by branches. To avoid the phenomenon named "crossing line", to merge two nodes, which are not neighborly closest to each other in the graphics but will share the same ancestor, causes node-swapping before the internal node is created. The node-swapping should follow two constraints:

1. If two sub-trees are supposed to be merged, the upstairs sub-tree's position is fixed. Meanwhile the sub-tree locating downstairs should get close to the sub-tree upstairs until they are the neighborly closest in the representation.
2. Premised with the first constraint, the modification of tree topology representation should be minimized.

The pseudo code shown below illustrates how the algorithm works to merge two sub-trees *i* and *j*, where $y_i < y_j$:

```
--------------------------------------------------------------------------------------------------------------
Set $N$ = the number of leaves in sub-tree $j$
Set $D = N \times d_y$                    // $d_y$ is a unit length of y-axis
Set $N'$ = the number of leaves between sub-trees $i$ and $j$
If $N' = 0$
    Create an intermediate node $l$
```

```
Else
    Loop through the nodes (leaves and internal nodes) between sub-trees i and j
        $y_m = y_m + D$        // $y_m$ is the ordinate of a particular node between
                               sub-trees i and j
    Loop through the nodes (leaves and internal nodes) in the sub-tree j
        $y_n = y_n - D$        // $y_n$ is the ordinate of a particular node in sub-tree j
    Create an intermediate node l
Set $x_l = x_{\max} + d_x$     // $x_{\max}$ is the largest abscissa in i and j;
                               // $d_x$ is a unit length of x-axis
Set $y_l = \frac{y_i + y_j}{2}$   // $y_i$ is the ordinate value of the leading node (root) in i;
                               // $y_j$ is the ordinate value of the leading node (root) in j
Draw new branches to connect the intermediate node with sub-trees i and j
-----------------------------------------------------------------------------------------------
```

After 1-step-forward, the tree’s topology is converted into one as demonstrated in Figure 6.3. It should be noted T1 and T2’s positions have been swapped.

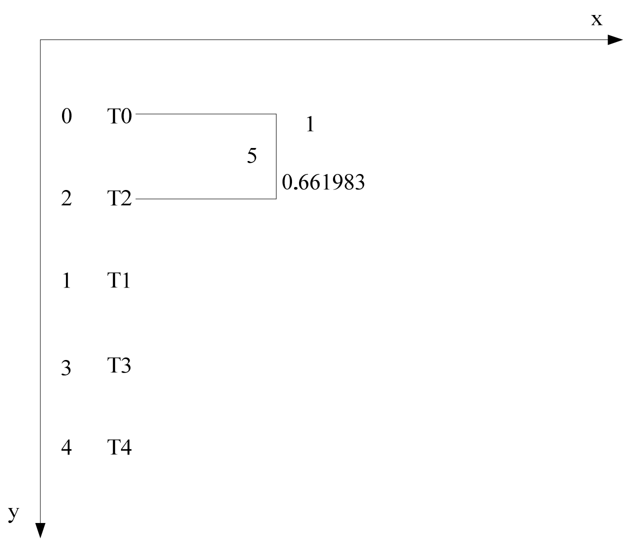


Figure 6.3: Tree topology of 5 sequences after execution of a forward step

Basically, there are three types of combination: non-critical-point automatic combination by algorithm, critical point, and manual combination. It is very important to identify these combinations because under an agglomerative criterion, the accuracy of following combinations is influenced by the previous ones. To identify a specific combination, a mechanism which uses colors to differentiate types of combination is employed.

For a non-critical-point automatic combination by algorithm we use black to color the branch; for a critical point, the branch is highlighted by red; for a manual combination, we use green as the branch color. Furthermore, during the computation, demonstration of the result of tree topology after manual joining and disjoining is displayed in yellow. Since the second automatic merge step in our example is a critical point, the branches created by this merge step are in red and the tree topology is represented as in Figure 6.4.

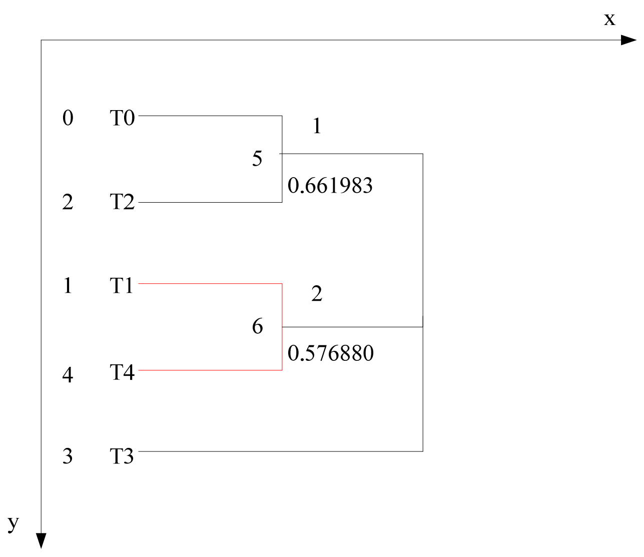


Figure 6.4: Tree topology of 5 sequences after execution of two forward steps.

The number of steps to complete an unrooted tree reconstruction is $k - 3$, where k is the number of sequences. Thus, in our example, the last three connections are automatically created after two forward steps. The whole tree of the 5 given sequences has been generated and visualized.

6.3 Protocols for Communication

As a web-based system, efficient protocols for communication are required to be established. To send requests to server to interact with the computation, the 3 parameters for interactions are embedded into the URL and sent by HTML GET method. The responses from server-side with information of updated topologies and computing status are in XML format and parsed at client-side for further modification.

The tree visualization tool modifies the tree topology graphical representation based on the following three XML files: tree initialization, internal node creation, and internal node deletion. The type of the XML file is recognized by checking the value of “name”, which is an attribute of the tag “Action”. For defining tree initialization, internal node creation, and internal node deletion, the values of “name” are “Initialize”, “Join” and “Disjoin” respectively.

In this section, the structure of these three XML files is introduced by using an example based on the phylogeny reconstruction from the given set of 5 sequences in Figure 5.1.

The structure of tree initialization XML file is defined as follows:

```
<?xml version = “1.0” encoding = “utf-8” ?>
<Action name = “Initialize”>
  <leaves n = “5”>
    <leaf id = “0”>T0</leaf>
    <leaf id = “1”>T1</leaf>
    <leaf id = “2”>T2</leaf>
    <leaf id = “3”>T3</leaf>
    <leaf id = “4”>T4</leaf>
    <leaf id = “5”>T5</leaf>
  </leaves>
</Action>
```

Detailedly, the number 5, which is the value of attribute “n” in the tag “leaves”, defines the number of sequences will be analyzed. The value of “id” in tag “leaf” is a unique index of this sequence and the value of “leaf” is the name of this sequence.

Internal node creation is an XML file describing the tree topology modification after step forward and manually joining operations. In the previous example, assume next operation is 2-step-forward, the response from phylogenetic inference service will be in the following format:

```
------------------------------------------------------------------------------------------------
<?xml version = “1.0” encoding = “utf-8” ?>
<Action name = “Join”>
  <steps n = “2”>
    <step id = “1” type = “0”>
      <leaf1> 0 </leaf1>
      <leaf2> 2 </leaf2>
      <penalty>0.661983</penalty>
    </step>
    <step id= “2” type = “2”>
      <leaf1>1</leaf1>
      <leaf2>4</leaf2>
      <leaf3>3</leaf3>
      <penalty1>0.576880</penalty1>
      <penalty2>0.517866</penalty3>
      <penalty3>0.571920</penalty2>
    </step>
  </steps>
</Action>
------------------------------------------------------------------------------------------------
```

In this XML file, the attribute “n” of “steps” is the number of computing steps in this operation. In one step of the computation, the attribute “type” indicates how these two sub-trees are merged. 0 means a non-critical point automatic mergence, 1 is associated with a manual mergence, and 2 is corresponding to a critical point. Basically, “leaf1” and “leaf2” are two representative leaves of the descendants of the created internal node and penalty is the confidence value of this mergence. If critical point occurs, there are three representative leaves with three penalties corresponding to three directions at the critical point (e.g. the step with id 2 in this XML).

Internal node deletion is an XML file presenting the tree topology modification after step backward and manually disjoining operations. The following sample is based on the assumption that, after the 5 sequences in our example has been completely analyzed, 1-step-backward operation is executed. The XML format of response from phylogenetic inference service will be as follows:

```
<?xml version = “1.0” encoding = “utf-8” ?>
<Action name = “Disjoin”>
<from>2</from>
<to>1</to>
</Action>
```

The values of tag “from” and “to”, are the jumping-off point and the end point of the backward operation respectively. Generally, after parsing the internal node deletion

XML file, the tree visualization tool deletes all the internal nodes created between the step defined by tag “from” and the step defined by tag “to.”

By synthetically and dynamically using these three XML files, we can simply define the topology modification after critical point redirection, undo, and re-initialization.

In addition, errors may occur during the interaction. An error is informed to the frontend based on XML format as well. The structure of the XML file is defined as follows:

```
----------------------------------------------------------------------------------------------------
<?xml version = “1.0” encoding = “utf-8” ?>
<Action name = “Error” type = n>
</Action>
----------------------------------------------------------------------------------------------------
```

The possible values of “*n*” and corresponding definitions can be found in Table 6.1.

Type	Definition	Possible Reason
0	cannot go forward	reach the end of the computation already
1	cannot go back	reach the beginning of the computation already
2	cannot join the 2 nodes	at least one of the 2 nodes is not a representative node at current step
3	cannot disjoin the 2 nodes	the 2 nodes are not couples
4	cannot read the stage of computation from the memory	no tree has been generated

5	cannot undo	the undo command cannot be executed if the last operation is undo
6	cannot calculate the confidence value	the 2 nodes cannot be joined at current step
7	cannot redirect the computation at critical point	the step of combination is not a critical point; or ,the id of direction is illegal
8	the other exceptions	

Table 6.1: Error definitions and possible reasons in protocols

From Table 6.1 we can see each type of error is corresponding to a specific interactive operation. Once an error occurs, the phylogeny service will inform users the exception and wait for the further commands: terminate the interaction or retrieve the tree topology and associated information from computing node's memory.

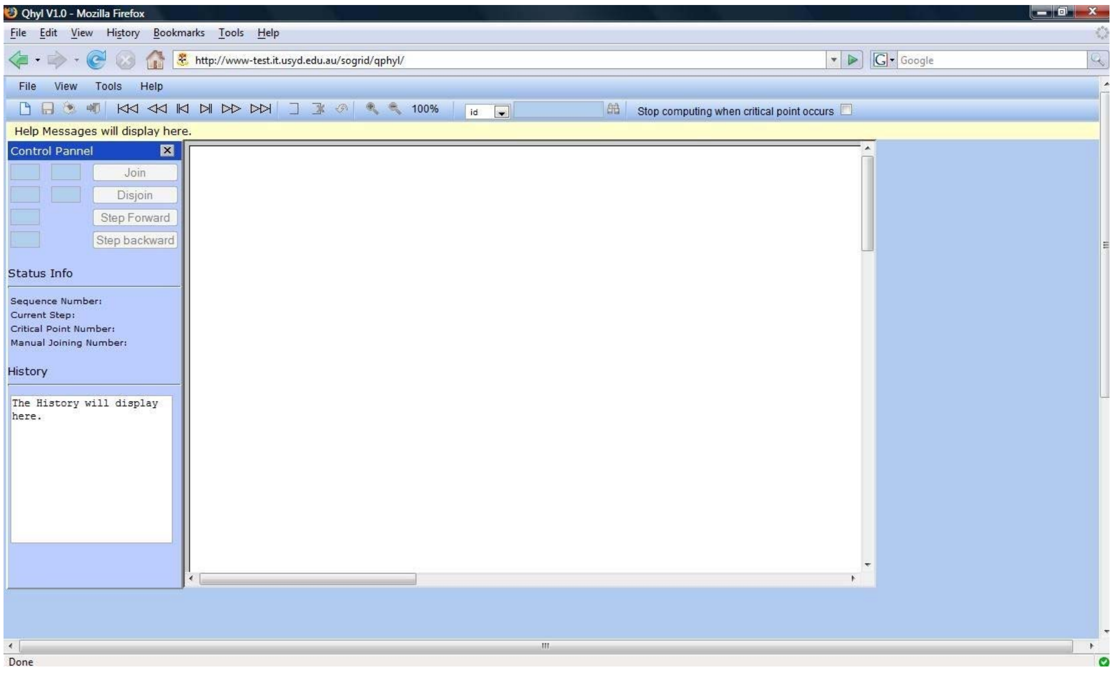
6.4 Results

This section introduces the results from the system design. An example is given during the introduction to describe how the interaction works for phylogeny reconstruction.

6.4.1 An Instance Using the Qphyl System

To use the Qphyl System, user should open the website of Qphyl via IE, Firefox or Maxthon (Firefox is recommended) by typing the following URL:

http://www-test.usyd.edu.au/sogrid/qphyl/, into the address bar. A graphical interface (Figure 6.5) will be displayed in user's web browser.


Qhyl V1.0 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Google
File View Tools Help
100%
id
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
Status Info
History
Done

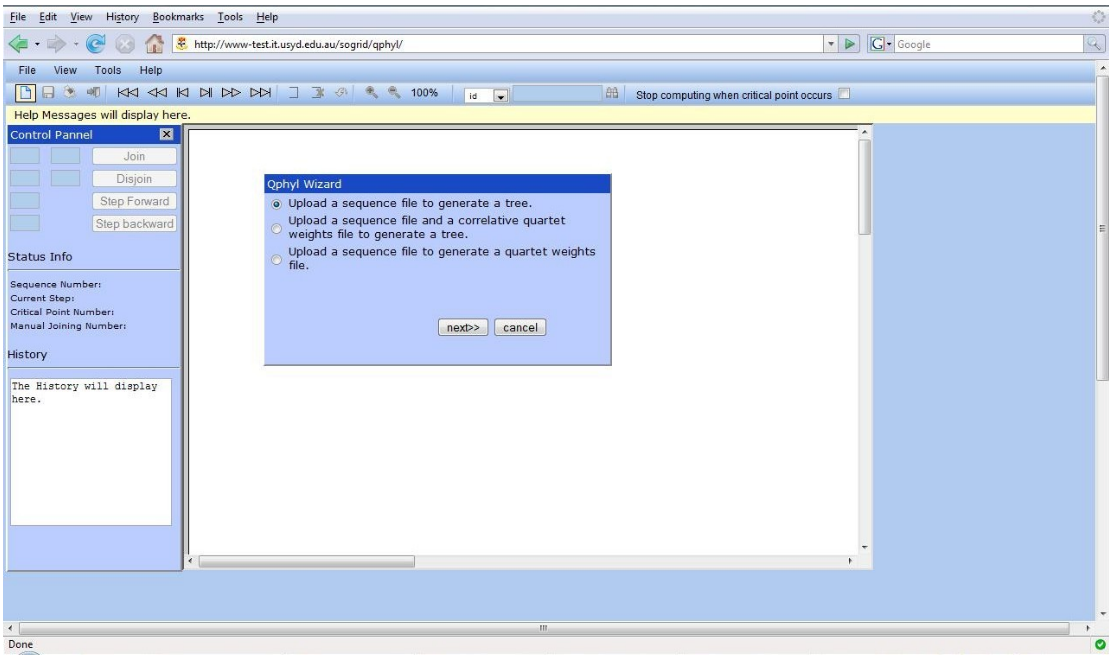
Help Messages will display here.

Sequence Number:
Current Step:
Critical Point Number:
Manual Joining Number:

The History will display here.

Figure 6.5: Screenshot of the Qphyl System interface

To start a new job, user clicks "File" from the main menu and then selects "New" or click menu Item "New" (🗋). Qphyl will launch the new-job wizard (Figure 6.6).


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Google
File View Tools Help
100%
id
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
Status Info
History
Qphyl Wizard
next>>
cancel
Done

Help Messages will display here.

Sequence Number:
Current Step:
Critical Point Number:
Manual Joining Number:

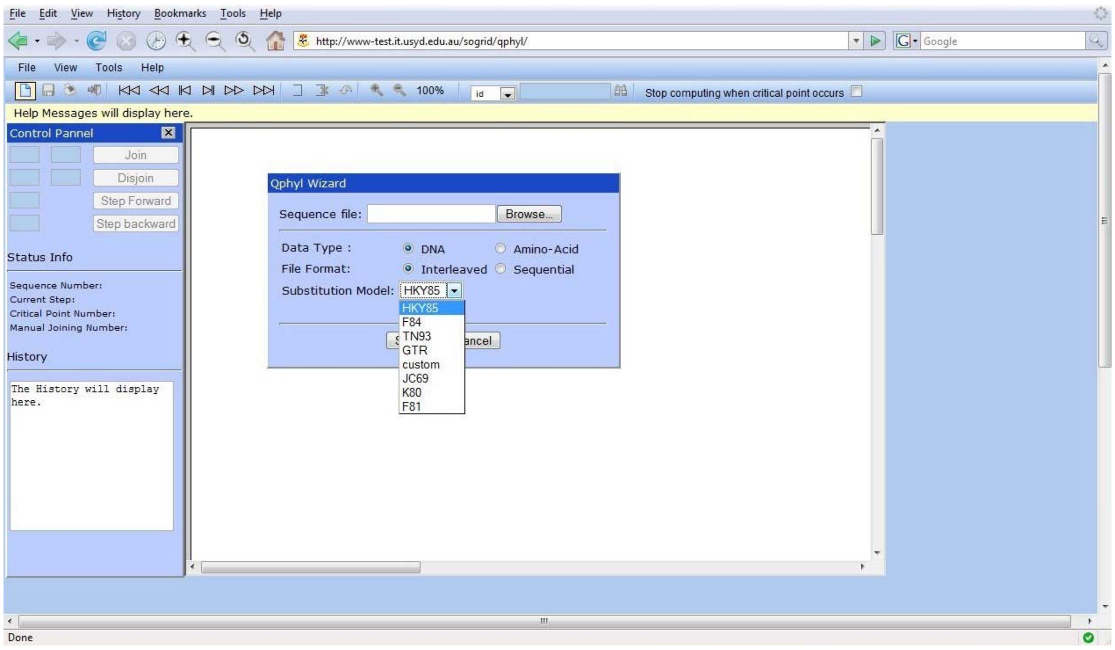
The History will display here.

[✓ Upload a sequence file to generate a tree.]
Upload a sequence file and a correlative quartet weights file to generate a tree.
Upload a sequence file to generate a quartet weights file.

Figure 6.6 Screenshot of new job wizard

Then three options are available and each option corresponds to one aim of phylogeny analysis. These options are: (i) Upload a sequence file to generate a tree; (ii) Upload a sequence file and a correlative quartet weights file to generate a tree; (iii) Upload a sequence file to generate a quartet weights file. The first two options will launch a mission to generate a phylogenetic tree from uploaded file(s).

To select the first option will lead the user to the next step for quartet weight generation (Figure 6.7).



Stop computing when critical point occurs

Help Messages will display here.

Status Info

Sequence Number:
Current Step:
Critical Point Number:
Manual Joining Number:

History

The History will display here.

Qphyl Wizard

Sequence file:

Data Type : [✓ DNA] Amino-Acid
File Format: [✓ Interleaved] Sequential
Substitution Model: HKY85

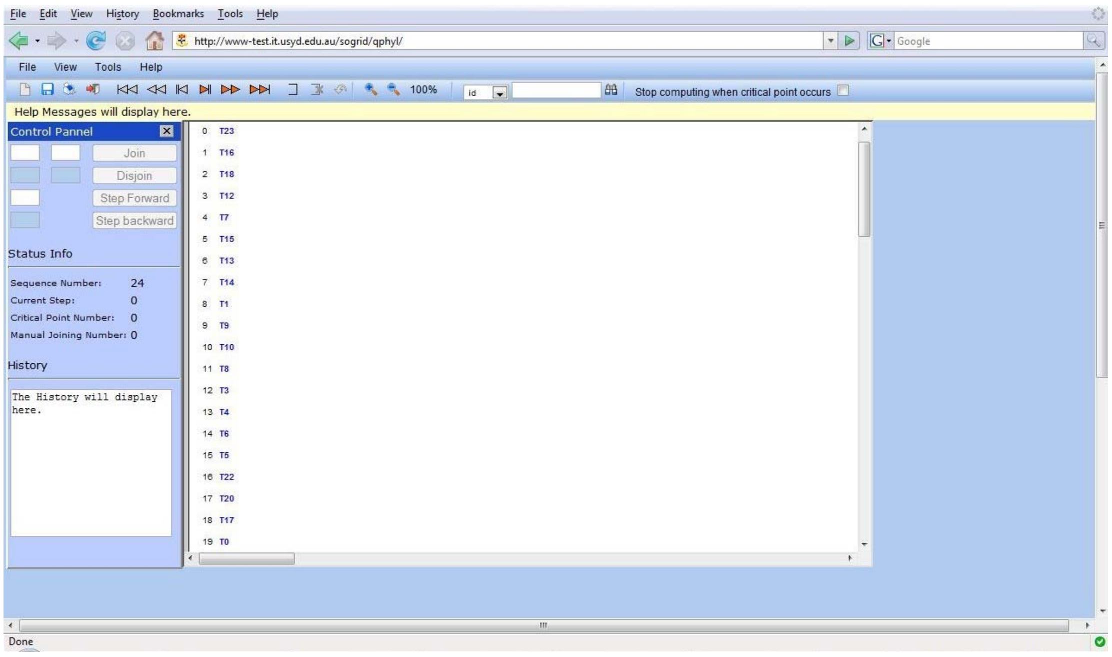
Figure 6.7 Screenshot of a panel used to set parameters for quartet weight generation

As demonstrated in Figure 6.7, after the uploading sequence file selection, user need to set up three parameters, data type (DNA or Amino Acid), file format (interleaved or sequential) and substitution model, for quartet generation. The available substitution models for quartet generation from DNA sequences include HKY85, F84, TN93, GTR, custom, JC69, K80 and F81,while substitution models for Amino Acid sequences include JTT, MtREV, WAG, DCMut, RtREV, CpREV, VT, Blosum62, MtMam and Dayhoff. Once the sequence file is uploaded, the quartet weight generation task will start automatically on server-side. It should be noted that it always takes much time to generate quartet weight. Once the quartet weight file is generated, user will be informed and the tree reconstruction stage will start up.

The stage of quartet weight generation can be skipped if user already has a quartet weight file corresponding to the sequence file in waiting for analysis. By choosing the second option, user can upload the sequence file and quartet weight file, and then start the tree reconstruction.

The aligned-sequence file for uploading has to be a text file with an extension “.seq”, which user can prepare by using any word processor or text editor. The sequences in the file have to be in PHYLIP format. The quartet file for uploading, which has to be a text file with an extension “.qrt”, can be prepared by the Qphyl system. By choosing the third option in the wizard, after uploading a sequence file, a corresponding quartet file will be generated by server and available to be downloaded automatically after the generation.

After the quartet generation, the tree will be initialized as displayed in Figure 6.8. Now user can interact with the phylogeny computation.


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Google
File View Tools Help
100%
id
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
History
Done

Help Messages will display here.

Status Info

Sequence Number: 24
Current Step: 0
Critical Point Number: 0
Manual Joining Number: 0

The History will display here.

0 T23
1 T16
2 T18
3 T12
4 T7
5 T15
6 T13
7 T14
8 T1
9 T9
10 T10
11 T8
12 T3
13 T4
14 T6
15 T5
16 T22
17 T20
18 T17
19 T0

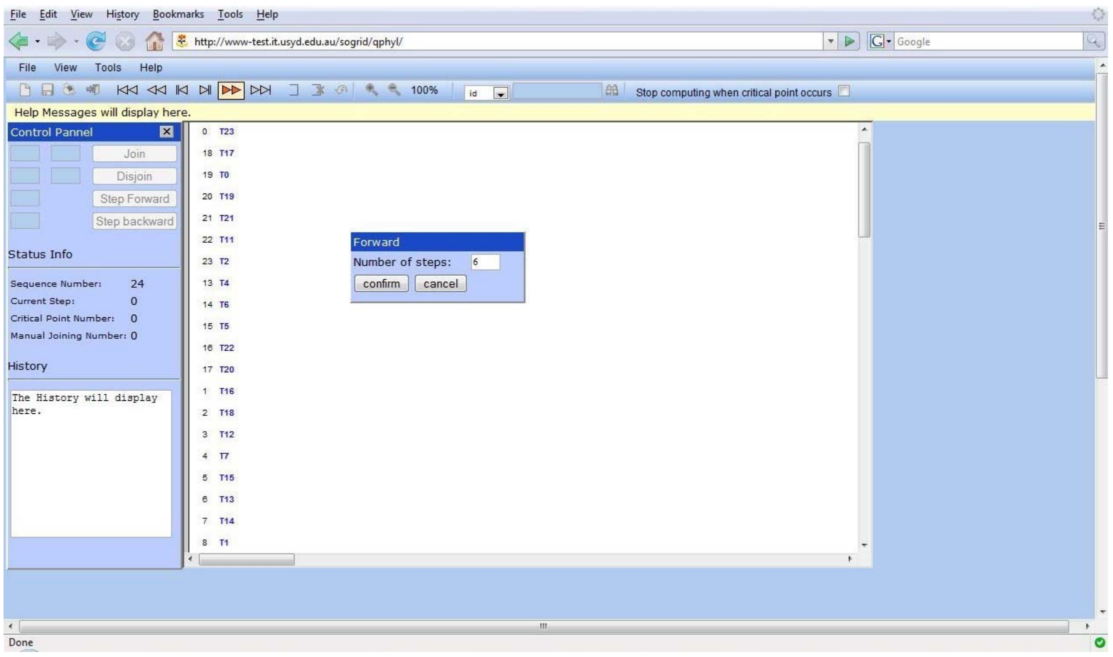
Figure 6.8: Screenshot of an initialized tree

All the leaves with unique index are represented graphically in a specific order. The leaf with smaller numeric index is located upon the one with larger numeric index.

On the control panel locating on the left, status information of computation can be detected. From Figure 6.8 It can be seen that there are 24 sequences, current step id is 0, no critical point occurs and no manually joining has been executed.

Currently, available operations for interaction include "one step forward" (▶|), "forward" (▶▶), "forward to the end" (▶▶|), and "manually join" (❐). Invalid buttons are grayed out. It should be noted the checkbox of "Stop computing when critical point occurs". If this checkbox is checked, once a critical point takes place, the computation will be stopped automatically and waiting for the further command.

Figure 6.9 indicates an operation of “forward”. After clicking the “forward” button, a dialog box is splashed. Here user can input the number of steps he/she wants the computation take and then confirm the operation. User doesn’t have to be worried about an illegal number, because the confirm button is capable only if the input number is reasonable. In this example, 6 steps forward will be taken. Figure 6.10 shows the result after this operation.



Help Messages will display here.

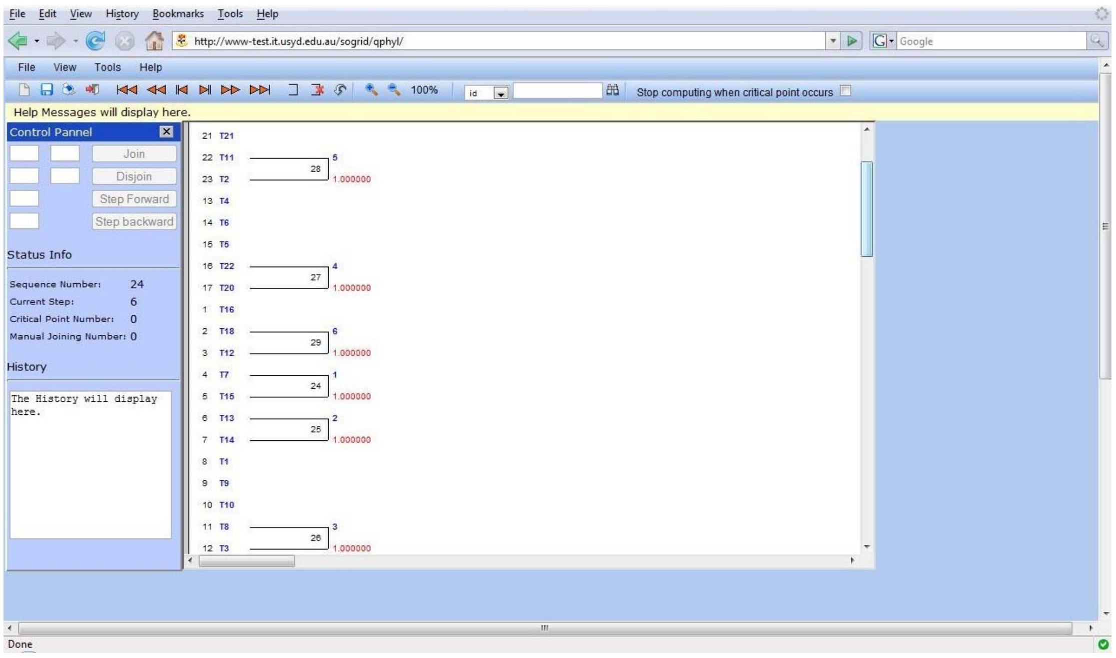
Sequence Number: 24
Current Step: 0
Critical Point Number: 0
Manual Joining Number: 0

The History will display here.

Number of steps: 6

```
0   T23
18  T17
19  T0
20  T19
21  T21
22  T11
23  T2
13  T4
14  T6
15  T5
16  T22
17  T20
1   T16
2   T18
3   T12
4   T7
5   T15
6   T13
7   T14
8   T1
```

Figure 6.9: Screenshot of step forward (1)


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
File View Tools Help
100%
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
Status Info
History
Done

Help Messages will display here.

Sequence Number: 24
Current Step: 6
Critical Point Number: 0
Manual Joining Number: 0

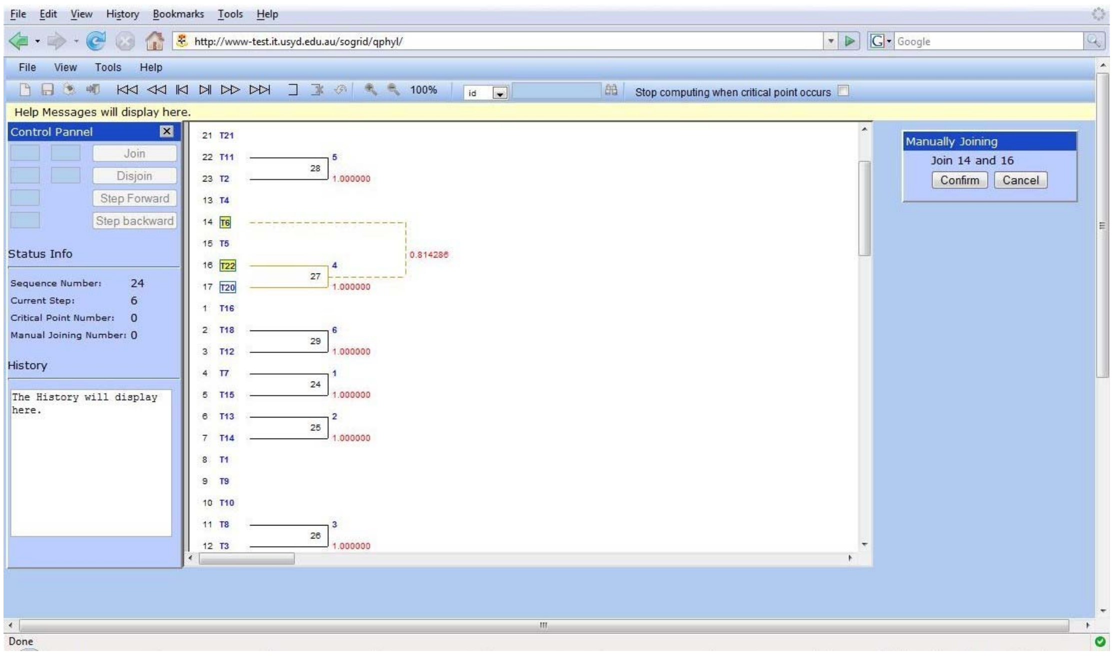
The History will display here.

Figure 6.10: Screenshot of step forward (2)

It could be noticed that now more buttons are available to use.

To execute “step backward” (◀◀) is quite similar with “forward” operation, once the number of steps inputted into the text-input area is confirmed, the computation will return to the desired step by retrieving the corresponding status.

Figure 6.11 shows how to use the manually-joining operation.



Help Messages will display here.

Status Info

Sequence Number: 24
Current Step: 6
Critical Point Number: 0
Manual Joining Number: 0

History

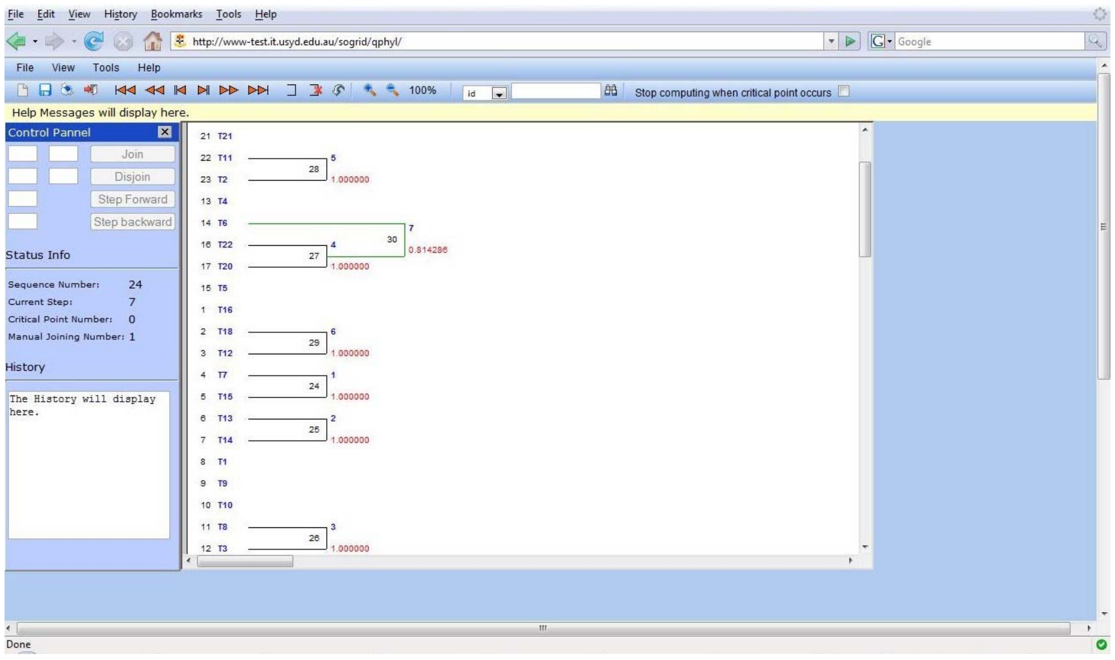
The History will display here.

Manually Joining

Join 14 and 16

Figure 6.11: Screenshot of manually joining (1)

In manually joining operation, user can pick up 2 nodes, which are legal to be joined, by clicking the name of their representative leaf nodes. The sub-trees which will be influenced from this operation will be highlighted by changing the branch color to yellow. And also, the possible combination of selected nodes will be graphically demonstrated in yellow broken lines with confidence value. Figure 6.12 illustrates the result after this operation.



Stop computing when critical point occurs

Help Messages will display here.

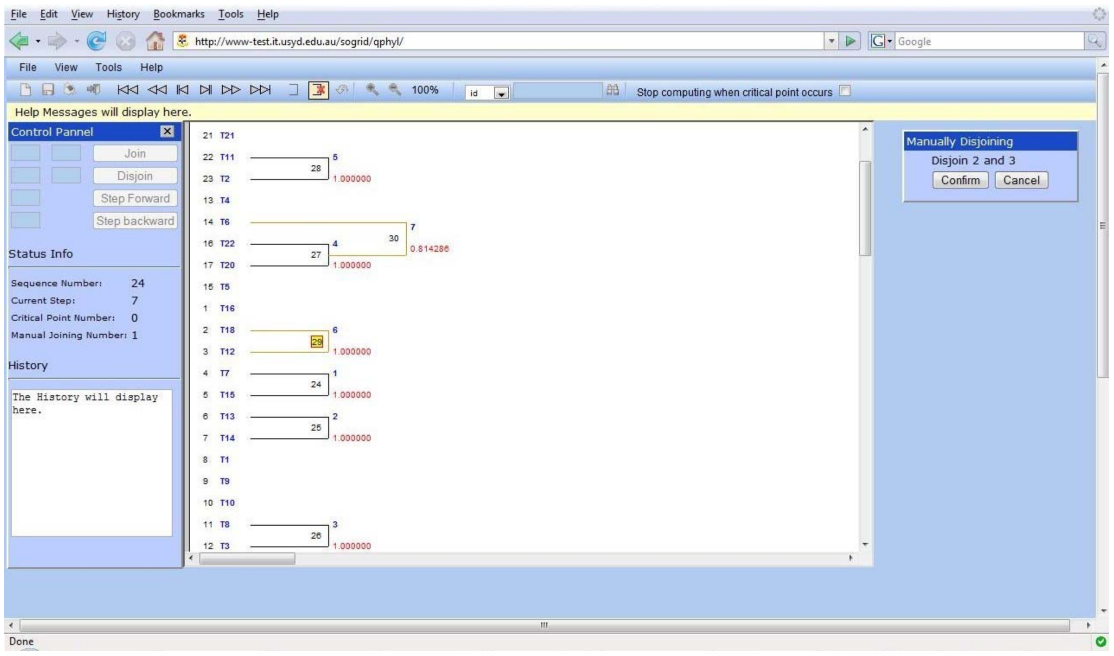
Sequence Number: 24
Current Step: 7
Critical Point Number: 0
Manual Joining Number: 1

The History will display here.

Figure 6.12: Screenshot of manually joining (2)

The branch of internal node which is created by manually operation is green. The "Manual Joining Number" in "Status Info" in "Control Panel" is updated after this operation.

Figure 6.13 demonstrates how to use manual disjoining function: click the "manual disjoining" item and then click the internal node user wants to disjoin. The colors of branches in sub-trees which will be affected are transformed to yellow. Once the disjoining operation is confirmed, all the internal nodes which created after the selected internal node will be eliminated as well. Figure 6.14 is the tree topology after this operation. The intermediate nodes with the index 29 and 30 are deleted.


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Help Messages will display here.
Control Pannel
Join
Disjoin
Step Forward
Step backward
Stop computing when critical point occurs
Manually Disjoining
Disjoin 2 and 3
Confirm
Cancel
Done

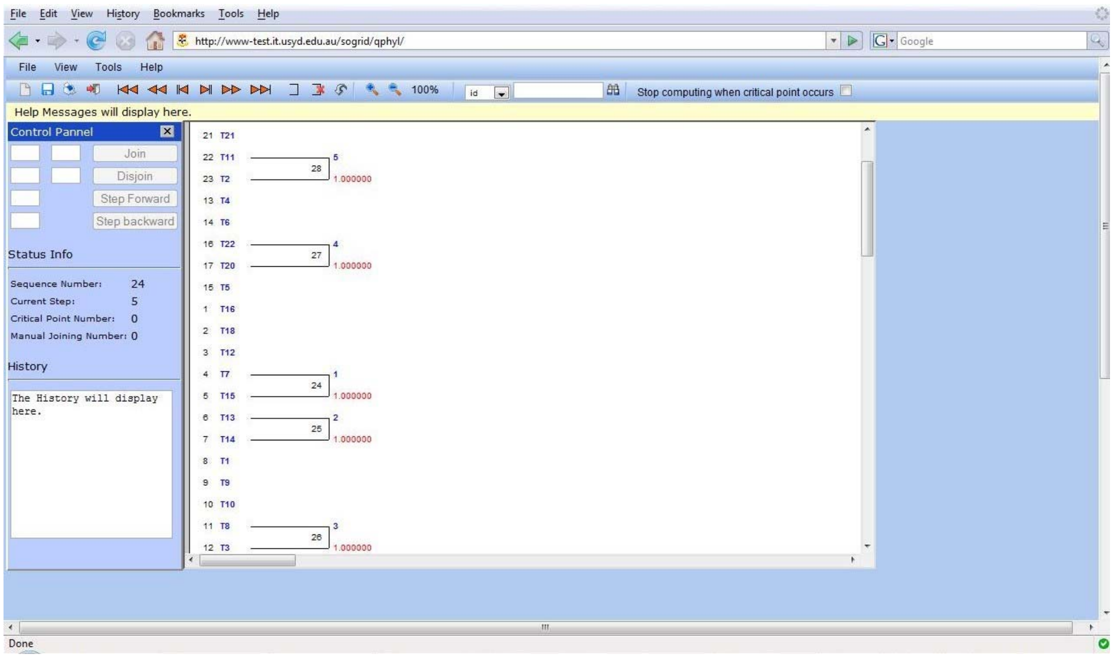
Status Info

Sequence Number: 24
Current Step: 7
Critical Point Number: 0
Manual Joining Number: 1

History

The History will display here.

Figure 6.13: Screenshot of manually disjoining (1)


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Help Messages will display here.
Control Pannel
Join
Disjoin
Step Forward
Step backward
Stop computing when critical point occurs
Done

Status Info

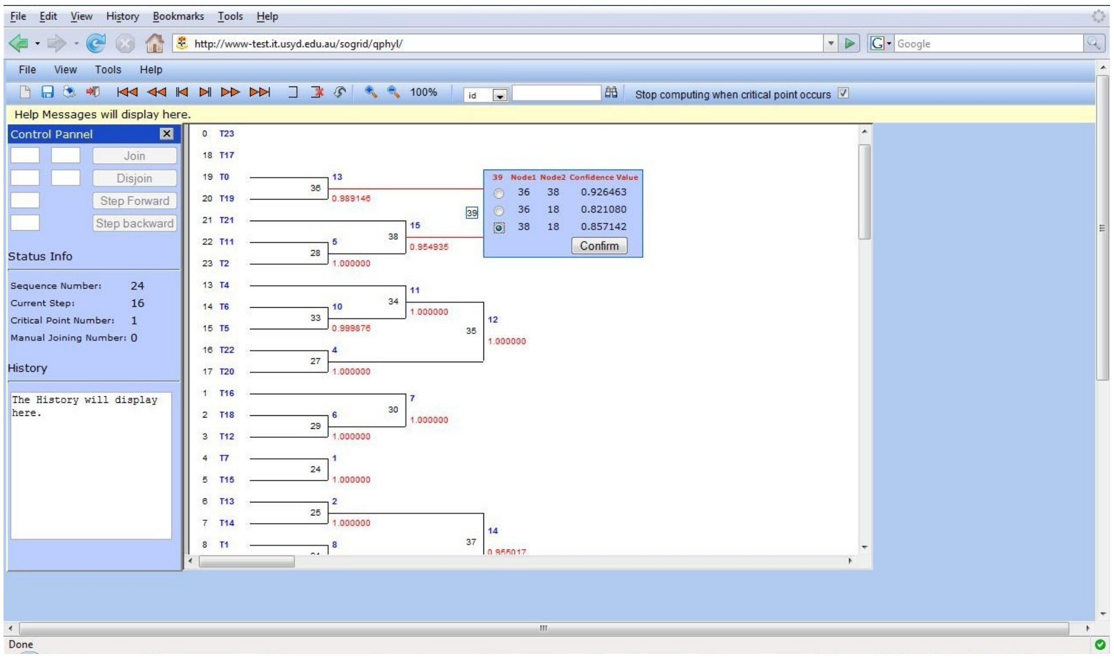
Sequence Number: 24
Current Step: 5
Critical Point Number: 0
Manual Joining Number: 0

History

The History will display here.

Figure 6.14: screenshot of manually disjoining (2)

The stage of tree reconstruction in Figure 6.15 has one critical point (highlight by red color) which occurred when node 38 was generated. Move the cursor over the internal node's id, a dialog box with 3 options, which are related to 3 possible computing directions respectively, will be splashed. The information provided by the dialog box includes the 3 possible combination with corresponding confidence values. By option-selection in this dialog box, user can redirect the tree topology computations. In this example, the third computation direction with the second highest confidence value is selected and the tree topology is transformed to as displayed in Figure 6.16.



File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
File View Tools Help
100%
Stop computing when critical point occurs
Help Messages will display here.
Control Pannel
Join
Disjoin
Step Forward
Step backward
Status Info
History
Done

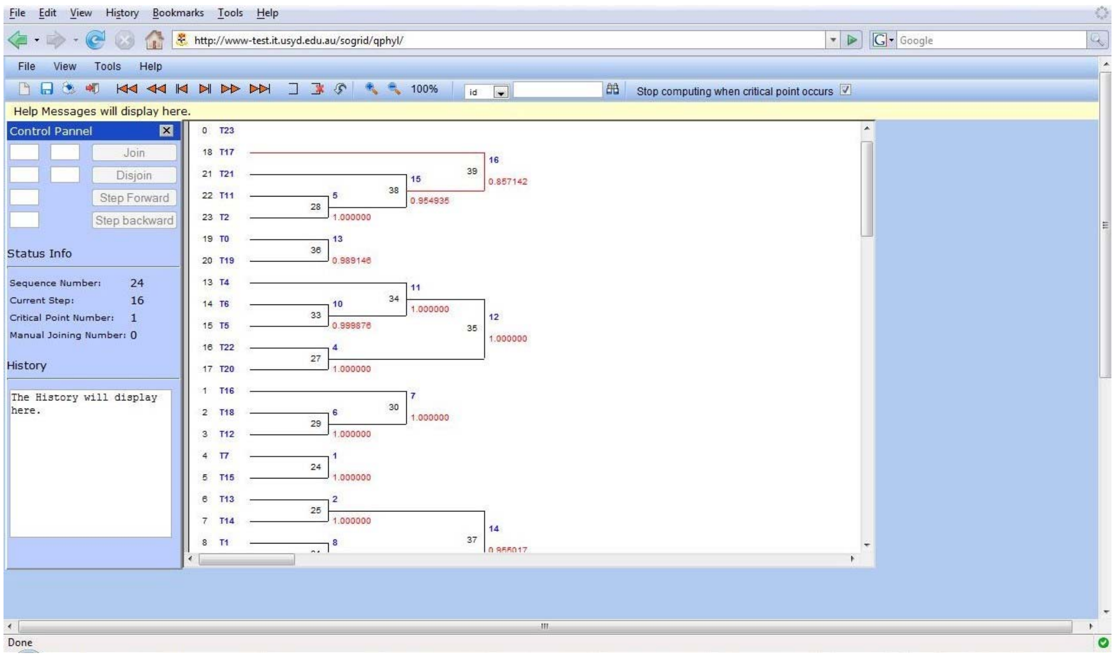
Sequence Number: 24
Current Step: 16
Critical Point Number: 1
Manual Joining Number: 0

The History will display here.

39	Node1	Node2	Confidence Value
○	36	38	0.926463
○	36	18	0.821080
●	38	18	0.857142

Confirm

Figure 6.15: Screenshot of critical point (1)


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
Google
File View Tools Help
100%
id
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
Status Info
History
Done

Help Messages will display here.

Sequence Number: 24
Current Step: 16
Critical Point Number: 1
Manual Joining Number: 0

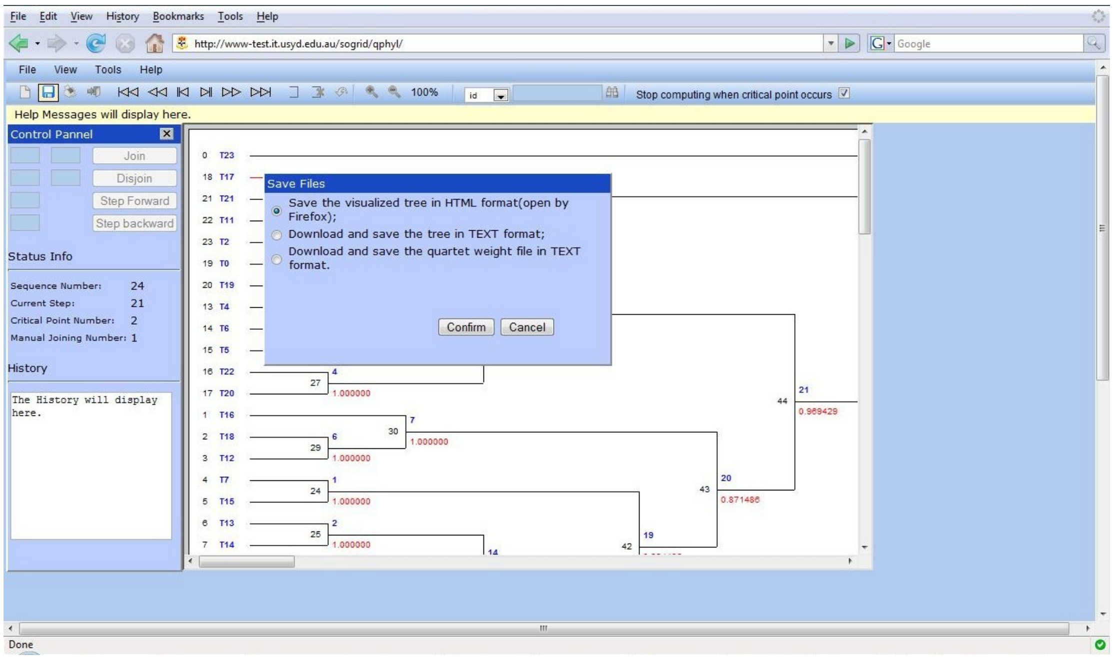
The History will display here.

Figure 6.16: Screenshot of critical point (2)

Undo operation is available if latest operation was not “undo”. Using “undo” user can cancel the last operation and recover the tree to previous stage.

The basic operations of interaction, including manually-joining, manually-disjoining, step forward and step backward, can also be implemented via the control panel options. For manually joining or disjoining two nodes, user should input the node ids into the corresponding text area on the control panel and then press the “Join” or “Disjoin” button. To control the tree reconstruction directions, user should input the number of steps he/she wants the computation go forward or back off into the control panel and then make a confirmation by pressing the “Step Forward” or “Step Backward” button.

All the stages of tree reconstruction can be saved in either an HTML file (visualized tree topology) or a text file (in Newick format). To save a file, press the "Save" button (💾) or select "Save" in the "File" menu.


File Edit View History Bookmarks Tools Help
http://www-test.it.usyd.edu.au/sogrid/qphyl/
File View Tools Help
100%
Stop computing when critical point occurs
Control Pannel
Join
Disjoin
Step Forward
Step backward
History
Save Files
Confirm
Cancel
Done

Help Messages will display here.

Status Info

Sequence Number: 24
Current Step: 21
Critical Point Number: 2
Manual Joining Number: 1

The History will display here.

Save Files

- [✓] Save the visualized tree in HTML format(open by Firefox);
- Download and save the tree in TEXT format;
- Download and save the quartet weight file in TEXT format.

Figure 6.17: Screenshot of wizard to save file

As displayed in figure 6.17, the first option allows user to save the graphically-represented tree topology in HTML format. The second one permits user to download a text file with the content of tree topology in Newick format. The third one enables user to download a text file of generated quartet weights. Figure 6.18 shows a visualized tree opened in a new window.

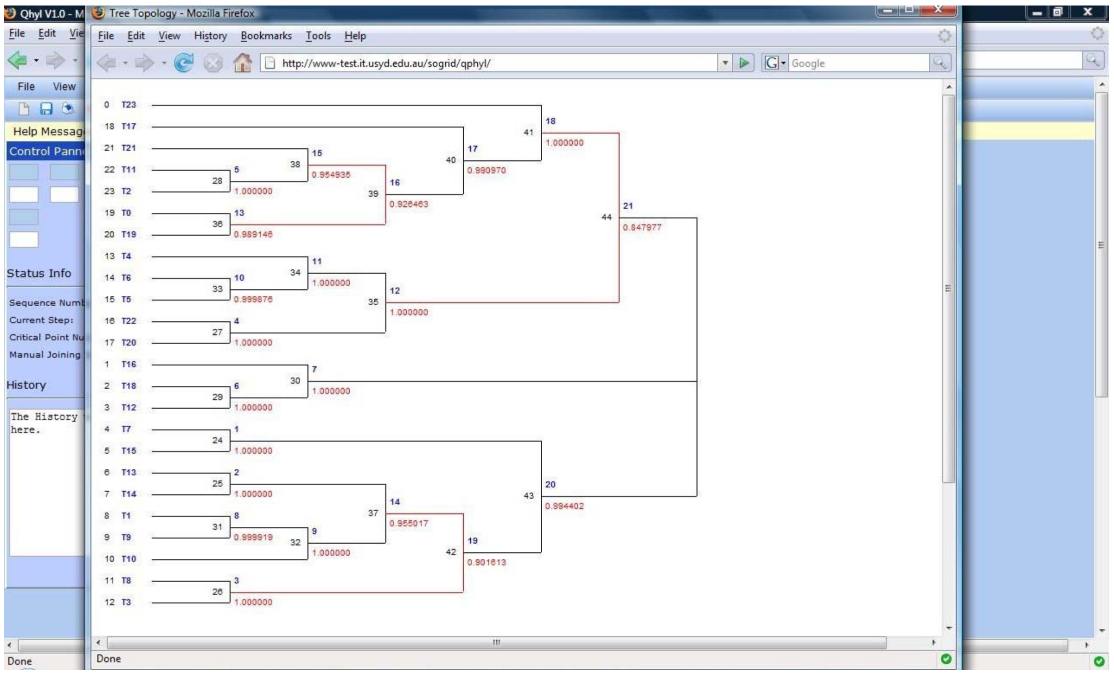


Figure 6.18: Screenshot of tree file saving

The visualized tree topology can also be directly printed in high-quality resolution after pressing the “Print” button (🖨) or selecting the “Print” option from “File” menu if user’s computer is connected to a working printer.

6.4.2 Two Phylogenies Generated by Qphyl

The following are two trees generated by the Qphyl System.

A 24-taxon tree from a set of random 600-bp homologous sequences:

In Newick format:

(T23,(T16,(T18,T12)),(T7,T15),(T13,T14),((T1,T9),T10),(T8,T3),T4,(T6,T5),(T22,T20),T17,T0,T19,T21,(T11,T2));

Visualized:

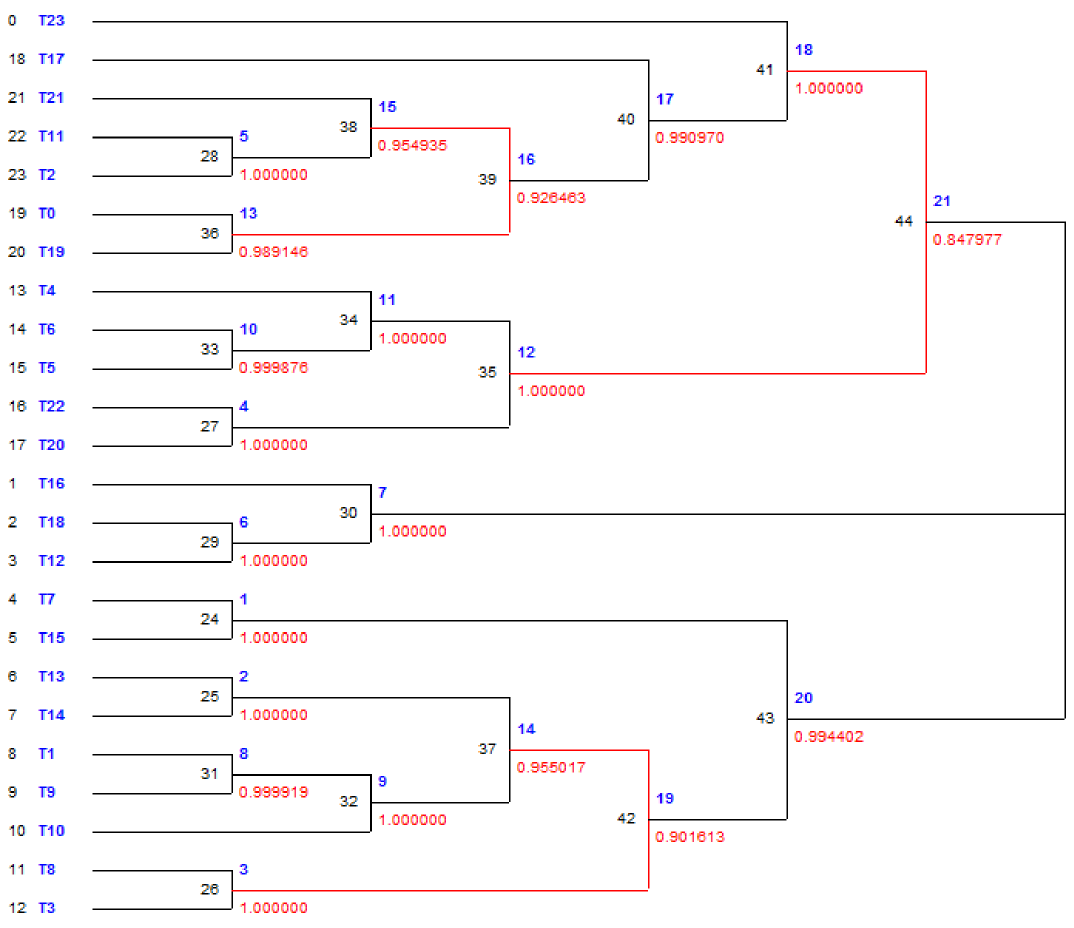


Figure 6.19: A 24-taxon tree from a set of random 600-bp homologous sequences

A 40-taxon tree from a set of real 500-bp homologous sequences:

In Newick format:

((((((((((Tax8,(Tax16,Tax17)),(Tax4,(((Tax28,((Tax34,Tax35),Tax31)),Tax27),Tax23)))
,((Tax5,(Tax10,Tax11)),Tax3)),(Tax0,Tax1)),((Tax38,Tax39),(Tax19,Tax20))),(Tax12,
Tax13)),((Tax6,((((Tax36,Tax37),Tax9),Tax14),Tax15)),Tax7)),(((Tax21,(Tax22,(Tax2
6,(Tax32,Tax33)))),(Tax29,Tax30)),Tax18)),Tax2,(Tax24,Tax25));

Visualized:

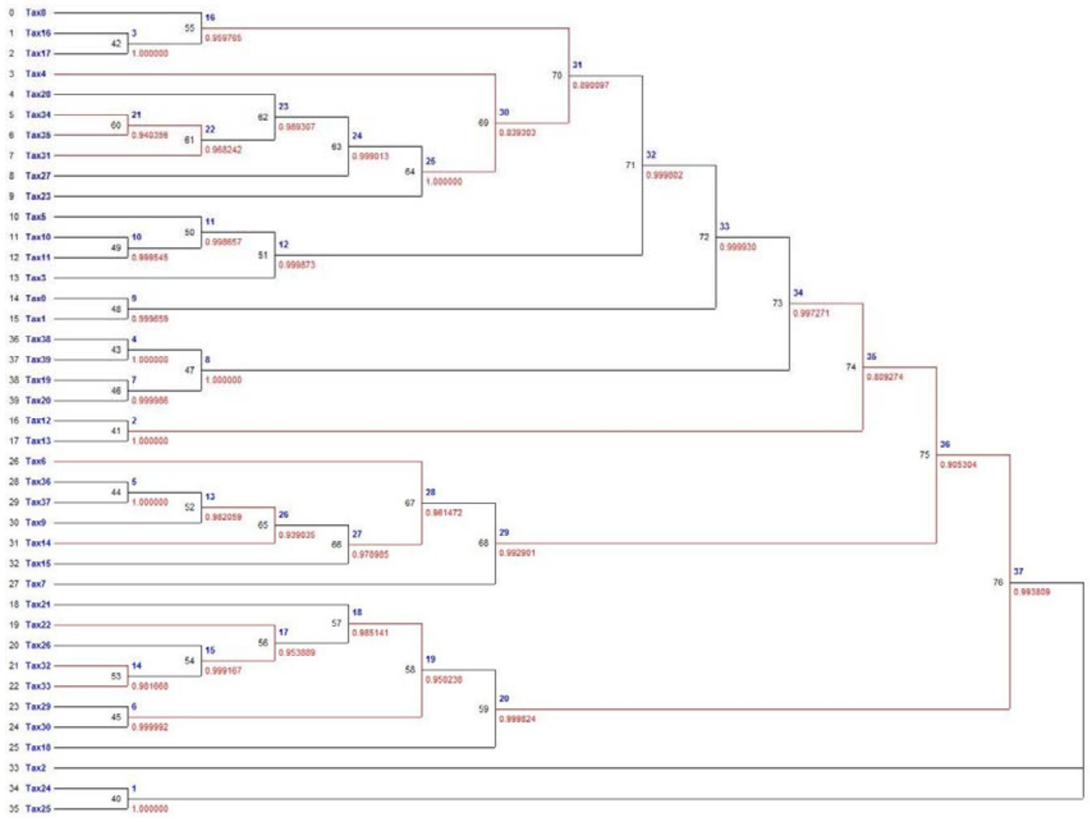


Figure 6.20: A 40-taxon tree from a set of real 500-bp homologous sequences

Chapter 7 Conclusion

The Qphyl System is the first web-based interactive phylogenetic analysis system. Compared to the other packages and systems, the Qphyl System enables biologists to dynamically and effectively interact with the computation, and install their biological-background knowledge into the computation to make better decision when tree reconstruction ambiguity occurs. For the first time, the procedure of phylogenetic analysis is not a “black box” anymore, every step of tree reconstructing procedure is visible and accessible.

By using technologies such as Ajax, the Qphyl System dynamically, integrally and graphically represents the procedure of phylogeny reconstruction to biologists. With this integrated visualization tool, the Qphyl System allows biologists to save every step of the phylogeny reconstruction for further comparisons.

The future work focuses on the two following improvements: the accuracy of the phylogenetic analysis, and the usability of the system.

The super-quartet method [12], which places a sub-tree (which may contain a number of taxa) on each vertex of a quartet tree to form a so-called super quartet, has been designed and implemented. The experimental results show that the evaluation of super-quartets at critical points can be very effective. Thus, we will integrate it with our original quartet-based algorithm into the Qphyl System to dynamically make merge decisions when critical points occur.

In the Qphyl System, we have already used a Grid platform to reduce the ML computational complexity and handle the multiple requests. However, for a specific task, the compute and memory intensive is still significant. A parallel environment will be employed into our system to divide the ML-based quartet weight generation task into smaller tasks, in order to improve the robustness and usability of the system.

References

1. Berry, V., et al., *Quartet Cleaning: Improved Algorithms and Simulations*, in *Lecture Notes in Computer Science*. 1999. p. 313-324.

2. Felsenstein, J., *Counting Evolutingary Changes*, in *Inferring Phylogenies*. 2004. p. 11-18.

3. Swofford, D.L. and G. Olsen, *Phylogeny Reconstruction*, in *D.M. Hillis and G. Moritz (eds.), Molecular Systematics*. 1990, Sinauer Associates. p. 411-501.

4. Wang, C., B.B. Zhou, and A.Y. Zomaya, *Quartet-based Phylogenetic Inference: a Grid Approach*, in *Lecture Notes in Computer Science*. 2005, Springer Berlin/Heidelberg. p. 387-391.

5. Beckmann, J., et al. *Common Data File Definitions for Neutron Inelastic Scattering Instruments Using NeXus*. in *Proceedings of NOBUGS2002/03*. 2002.

6. Berry, V. and D. Bryant. *Faster Reliable Phylogenetic Analysis*. in *Proceedings of 3rd Annual International Conference on Comp. Mol. Biol.* 1999.

7. Berry, V., et al. *a Practical Algorithm for Recovering the Best Supported Edges in an Evolutionary Tree*. in *Proceedings of Symposium on Discrete Algorithms*. 2000. San Francisco.

8. Geyer, C.J. *Markov Chain Monte Carlo Maximum Likelihood*. in *Computing Science and Statistics: Proceedings of the 23rd Symposium of the Interface*. 1991.

9. Huson, D.H., S. Nettles, and T. Warnow. *Obtaining Highly Accurate Topology Estimates of Evolutionary Trees from Very Short Sequences*. in *Proceedings of the 3rd Annual Int. Conf. Comp. Mol. Biol.* 1999.

10. Jiang, T., P. Kearney, and M. Li. *Orchestrating Quartets: Approximation*

and Data Correction. in *Proceedings of the 39the IEEE Symposium on Foundations of Computer Science*. 1998.

11. Strimmer, K. and A. Von Haeseler. *Likelihood-mapping: a Simple Method to Visualize Phylogenetic Content of a Sequence Alignment*. in *Proceedings of the National Academy of Science*. 1997. USA.

12. Wang, P., et al. *a Global Maximum Likelihood Super-quartet Phylogeny Method*. in *Proceedings of the 5th Asia-Pacific Bioinformatics Conference*. 2007.

13. Zhou, B.B., et al. *Parallel Implementation of a Quartet-based Algorithm for Phylogenetic Analysis*. in *Proceedings of 20th International Parallel and Distributed Processing Symposium*. 2006. Rhodes Island.

14. Zhou, B.B., et al. *On a New Quartet-based Phylogeny Reconstruction Algorithm*. in *Proceedings of the 2006 International Conference on Bioinformatics and Computational Biology*. 2006. Las Vegas, US.

15. Zhou, B.B., et al. *a Novel Quartet-based Method for Phylogenetic Inference*. in *Proceedings of IEEE International Symposium on BIBE*. 2005. Minneapolis.

16. Adachi, J. and M. Hasegawa, *Instability of Quartet analyses of molecular Sequence Data the Maximum Likelihood Method: the Cetacean/Artiodactyla Relationships. Cladistics*, 1999. **Vol. 5**: p. 164-166.

17. Bandelt, H.J. and A. Dress, *Reconstructing the Shape of a Tree from Observed Dissimilarity Data.* Adv. Appl. Math, 1986. **Vol. 7**: p. 309-343.

18. Berry, V. and O. Gascuel, *Inferring Evolutionary Trees with Strong Combinatorial Evidence.* Theoret Comput. Sci., 2000. **240(2)**: p. 271-298.

19. Eigen, M., et al., *How Old Is the Genetic Code? Statisical Geometry of tRNA Provides an Answer.* Science New Series, 1989. **244**: p. 673-679.

20. Erdos, P., et al., *Constructing Big Trees from Short Sequences.* Lecture Notes Computer Science, 1997. **1256**: p. 827-837.

21. Felsenstein, J., *Evolutionary Trees from DNA Sequences: a Maximum Likelihood Approach.* J. Mol. Evol., 1981. **Vol. 17**: p. 368-376.

22. Fitch, W.M., *a Non-sequential Method for Constructing Trees and Hierarchical Classifications.* J. Mol. Evol., 1981. **18**: p. 30-37.

23. Gambette, P. and D.H. Huson, *Improved Layout of Phylogenetic Networks.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2004. **1(3)**: p. 109-115.

24. Gascuel, O., *BIONJ: an Improved Version of the NJ Algorithm Based on a Simple Model of Sequence Data.* Mol. Biol. Evol., 1997. **14**: p. 685-695.

25. Kloepper, T.H. and D.H. Huson, *Drawing Explicit Phylogenetic Networks and Their Integration Into Splits Tree.* BMC. Evl. Biol., 2008. **8**: p. 22.

26. Kumar, S., *A Stepwise Algorithm for Finding Minimum Evolution Trees.* Mol. Biol. Evol., 1996. **13**: p. 584-593.

27. Lindsey, G., *Evolution--Useful or Useless?* Impact, 1985. **148**.

28. Morrison, D.A., *Phylogenetic Analyses of Parasites in the New Millennium.* Adv. Parasitol, 2006. **63**: p. 1-124.

29. Nieselt-Struwe, K. and A. Von Haeseler, *Quartet-mapping, a Generalization of the Likelihood-mapping Procedure.* Mol. Biol. Evol., 2001. **18(7)**: p. 1204-1219.

30. Olsen, G., et al., *a Tool for Construction of Phylogenetic Trees of DNA Sequences Using Maximum Likelihood.* Comput. Appl. Biosci, 1994. **10**: p. 41-48.

31. Philippe, H. and E. Douzery, *the Pitfalls of Molecular Phylogeny Based on Four Species, as Illustrated by the Cetacean/Artiodactyla*

Relationship. J. Mamm. Evol., 1994. **2**: p. 133-152.

32. Rambaut, A. and N. Grassly, *Seq-Gen: an Application for the Monte Carlo Simulation of DNA Sequence Evolution along Phylogenetic Trees.* Comput. Appl. Biosci, 1997. **13**: p. 235-238.

33. Ramnwez, V. and O. Gascuel, *Quartet-based Phylogenetic Inference: Improvements and Limits.* Mol. Biol. Evol., 2001. **18(6)**: p. 1103-1116.

34. Saitou, N. and M. Nei, *the Neighbor-joining Method: a New Method for Reconstructing Phylogenetic Trees* Mol. Biol. Evol., 1987. **Vol. 4**: p. 406-425.

35. Steel, M., *the Complexity of Reconstructing Trees from Qualitative Characters and Subtrees.* J. Classification, 1992. **9**: p. 91-116.

36. Steel, M., *the Maximum Likelihood Point for Phylogenetic Tree Is Not Unique.* Syst. Biol., 1994. **43**: p. 560-564.

37. Strimmer, K. and A. Von Haeseler, *Quartet Puzzling: a Quartet Maximum-likelihood Method for Reconstructing Tree Topologies.* Mol. Biol. Evol., 1996. **13(7)**: p. 964-969.

38. Swofford, D.L. and S.H. Berlocher, *Inferring Evolutionary Trees from Gene Frequency Data under the Principle of Maximum Parsimony.* Systematic Zoology, 1987. **Vol. 36**: p. 293-325.

39. Wieland, C., *Evolution and Practical Science.* Creation, 1998. **20(4)**.

40. Willson, S., *Measuring Inconsistency in Phylogenetic Trees.* J. Theoret. Biol., 1998. **190**: p. 15-36.

41. *Long Branch Attraction*. [cited; Available from: http://en.wikipedia.org/wiki/Long_branch_attraction.

42. *Sequence Format*. [cited; Available from: http://emboss.sourceforge.net/docs/themes/SequenceFormats.html.

43. *The Newick Tree Format*. [cited; Available from: http://evolution.genetics.washington.edu/phylip/newicktree.html.

44. *The LIRMM Methods and Algorithms in Bioinformatics Research Group*. [cited; Available from: http://www.lirmm.fr.

45. *Substitution Model*. [cited; Available from: http://en.wikipedia.org/wiki/Substitution_model.

46. *Tree Traversal*. [cited; Available from: http://en.wikipedia.org/wiki/Tree_traversal.

47. *N-Body Algorithm*. [cited; Available from: http://trond.hjorteland.com/thesis/node34.html.

48. *Maximum Parsimony*. [cited; Available from: http://en.wikipedia.org/wiki/Maximum_parsimony.

49. *Taxonomy and Systematics at Glasgow, TreeView, Tree Drawing Software for Apple Macintosh and Windows*. [cited; Available from: http://taxonomy.zoology.gla.ac.uk/rod/treeview.html.

50. *PAUP*. 2002 [cited; 4.0:[Available from: http://paup.csit.fsu.edu/.

51. *PHYML*. 2007 [cited; 1.1:[Available from: http://atgc.lirmm.fr/phyml/.

52. Felsenstein, J. *PHYLIP*. [cited; 3.66:[Available from: http://evolution.gs.washington.edu/phylip.html.

53. Schmidt, H.A., K. Strimmer, and A. Von Haeseler. *TREE-PUZZLE*. 2004 [cited; 5.2:[Maximum Likelihood Analysis for Nucleotide, Amino Acid, and Two-state Date].