CONSTRUCTING THE TREEFAM DATABASE

Li Heng

Directed by Zheng Wei-mou

The Institute of Theoretical Physics,
Chinese Academic of Science

May, 2006

Abstract

TreeFam is a database of phylogenetic trees of gene families. It aims to develop a curated resource that presents the accurate evolutionary history of all animal gene families, as well as reliable orthologs and paralog assignment. In developing TreeFam, four novel algorithms were designed to improve the accuracy of tree building or to serve special needs for development. The first is a constrained neighbour-joining that efficiently adds new sequences to an existing tree while maintaining the original topology at the same time. This method is used to expand a seed tree to a full tree without losing any information added by manual curation. The second algorithm is a leaf reordering that orders the leaves of a tree according to the weights of leaves. When it is drawn as a picture, one tree can be displayed in different ways, depending on the order of leaves. This algorithm helps to display trees in a consistent algorithm and facilitates visual examination of trees, which is particularly helpful when comparing two trees. Thirdly, duplication and loss inference is fit into a more general theoretical framework and extended to allow for a multifurcated species tree. A fourth algorithm has also been developed, which is a new algorithm for merging trees. The tree merge algorithm itself is not a tree building algorithm, but it reconstructs an optimal tree from several trees that are built from an identical sequence set with different tree building methods. The resultant tree should combine the advantages of, and so outperform, all the candidates. This is shown to occur successfully in a large-scale benchmark presented in the last chapter. This benchmark is one of the few evaluations that are based on real data in the phylogenetics literature. It also highlights the fact that each tree-building algorithm has its own strength, although ML and parsimonious methods are slightly better in general.

Keywords: molecular phylogenetics, tree reconstruction, database

Acknowledgements

First of all, I am most grateful to my supervisor, Wei-mou Zheng. It was him who took me to the realm of bioinformatics, and it was him who gave me both strict instruction and full flexibility, which brought me a meaningful and colorful PhD life.

Next thanks go to Jun Wang and all my friends and colleagues in Beijing Genomics Institute (BGI), Chinese Academic of Science. I spent most of my PhD in BGI. It was them who gave me three happy yet thrilling years in my life. I particularly thank Jun Wang for his providing me the opportunity to collaborate with the Sanger TreeFam group, and thank Tao Liu for his assist and support over three years even when I overlooked his role at the beginning.

With the same passion I appreciate the generous help and instruction from Sanger TreeFam group including Richard Durbin, Avril Coghlan, Jean-Karim Hériché, Lachlan James Coin, and Alan Moses. Richard is the father of TreeFam. He raised the whole idea about TreeFam. Without him, there would be no TreeFam, not to speak my thesis. Avril, Jean-Karim and Alan read through this thesis and gave me numerous useful remarks and hints on further improvement. I particularly thank them for their patience with my poor English. Lachlan and Avril also helped to revise an earlier version of Chapter 5. Their insightful opinions made this chapter more solid and generalized. In addition, I learnt a lot from the communications with Sanger group, either by e-mail or in TreeFam workshop. TreeFam is always the result of group work.

I also wish to show my gratefulness to my Danish friends, especially to Lars Bolund who provided me a peaceful place and harmonious evironment where I could focus on TreeFam project and my thesis in 2005. I owe a lot to him and to all my Danish friends. At the same time I want to thank to Professor Bai-lin Hao for his kind instructions, to Ms. Ling Guo for her help when I was in BGI, and to all my classmates and roommates of the Institute of Theoretical Physics (ITP) for their help in these years.

At last, a big thank you to my dear prarents and my girlfriend for their silent support all the time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Phylogenetics is the science of studying the evolutionary relationships (phylogenies) of a group of organisms (Baldauf, 2003) or genes. One of the fundamental hypotheses of phylogenetics is that all organisms on Earth had a common ancestor and can be represented as trees. This hypothesis can be dated back to the era of Darwin and is supported by various studies on the similarity of molecular mechanisms of organisms (Durbin *et al.*, 1998). Successfully reconstructing the tree of life has long been the dream, and also the aim, of evolutionists.

Traditionally, phylogenetic trees are reconstructed by examining and comparing morphological characters (both from living and fossilized organisms), which was and still is a time-consuming and specialized endeavor (Cotton, 2003). Since the publication of Zuckerkandl and Pauling's original paper (Zuckerkandl and Pauling, 1965) that showed molecular sequences could also serve as characters, molecular phylogenetics found its own position and has become a distinct and fruitful direction of phylogenetics in recent years with rapidly increasing sequence data. It not only produces an overwhelming amount of information to biologists, but also inspires the development of computer science and statistics.

1.1 Background

Whereas reconstructing the tree of life is one of the central topics of phylogenetics, reconstructing the phylogeny of each gene family shows the strength of phylogenetic methods to most of molecular biologists. Phylogenies of gene families not only present the history of gene evolution, but also hold the key to gene annotations across species. The functional annotations or experimental results of genes in one species can be transferred to another when the history of related gene families is clear. This integrates the efforts of gene annotation in different species and is vital to annotations, especially to the annotations of a newly sequenced species. Of the same importance, reconstructing of gene phylogenies is the premise of discovering the priciples of gene evolution, which further provides clues to interpret gene functions. For example, the difference of selective pressures on genes implies functional shifts between them. Meaningful mutations, pseudogenes and interaction networks may also be inferred with the help of gene phylogenies (Ng and Henikoff, 2003; Coin and Durbin, 2004). In addition, reconstructing

phylogenies of gene families paves the way for other related evolutionary studies, such as intron evolution and genome-wide duplications.

Given the importance of gene phylogenies, many databases have been developed to present the history of each gene family, including KOG (Tatusov *et al.*, 2003), PANTHER (Mi *et al.*, 2005), SYSTERS (Meinel *et al.*, 2005) PhIGs (Dehal and Boore, 2005), Ensembl-compara (Hubbard *et al.*, 2005), OrthoMCL (Li *et al.*, 2003) and HOGENOM (Dufayard *et al.*, 2005). These efforts greatly boost the studies of gene evolution, but they are all faced with a common problem: how can a reliable phylogenetic tree be reconstructed? And this problem is, unfortunately, always one of the hardest topics in the area of bioinformatics (Delsuc *et al.*, 2005). Sequences of poor data quality (such as incorrect gene annotation or few informative sites) and unrealistic underlying evolutionary model (such as assuming that different lineages have evolved at the same rate) may greatly affect the accuracy of tree building. No automatic algorithm can reconstruct an accurate tree all the time. To make a phylogenetic database practically useful, this problem has to be solved.

1.2 Overview of the Thesis

TreeFam is a (gene) tree family database. Phylogenetic trees are the central elements. In developing TreeFam, the tree building difficulties are solved in two directions. First, a new algorithm, tree merge, is developed to improve the quality of automatic trees, and second, mannual curation is applied to further get higher accuracy with the help of the constrained neighbour-joining algorithm. This thesis presents the details to these algorithms and also covers most of topics related to the construction of TreeFam, including basic concepts, pipelines, additional algorithms, and implementations. In attempt to keep this thesis as an academic one, technical issues are placed in the appendix even though these issues might have been of the same importance as academic aspects when TreeFam database was set up.

In the following section of this chapter, the biological backgrounds and mathematical notations will be clarified. Chapter 2 describes the philosophy and structure of the TreeFam database. Chapter 3 covers various small yet important topics on the tree-building process, particularly two algorithms, constrained neighbour-joinining and leaf reordering algorithm, which are inspired by the TreeFam project. Chapter 4 extends the traditional tree-reconciliation procedure so as to allow for a multifurcated species tree and presents discussion about tree mapping. Chapter 5 describes tree merge algorithms, which is the chief effort in improving the quality of automatic trees. And in Chapter 6,

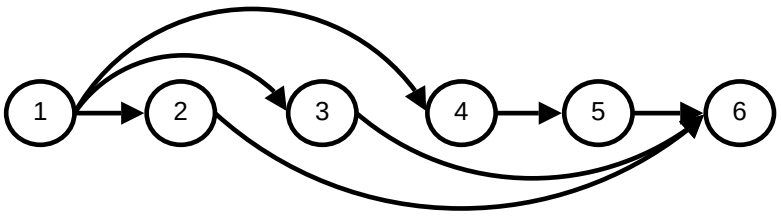

Figure 1.1: Relationships between chapters.

various tree-building algorithms are evaluated, using the curated TreeFam trees as a test set. The relationships between chapters are showed in Figure 1.1.

Except this introductory chapter, most of the results showed in this thesis have not been published yet. Although Chapter 2 is based on our paper (Li *et al.*, 2006), it still differs much from the published works. Some important results in the thesis are expected to be published in the following months.

1.3 Phylogenetic Terminology

Phylogeny denotes the evolutionary relationships of a group of organisms or genes. It can be represented as a tree, called a *phylogenetic tree*. *Phylogenetics* is the field of biology that studies phylogenies. It includes the reconstruction of phylogenies and the study of the factors that affect phylogenies, and the biological meaning that phylogenies reveal.

In this thesis, two types of trees will be involved: species trees and gene trees. Strictly speaking, a *species* is "a group of organisms capable of interbreeding freely with each other but not with members of other species" [1]. A *species tree* (Figure 1.2) describes the evolution relationships between species. Each of its external nodes (or terminal nodes, or leaf nodes, or leaves) stands for a present species that can be observed nowadays, while each internal node for an ancestral species in history. An ancestral species is usually named after the taxon that represents its descendant species. For example, when we say species *Eutheria*, we mean the ancestral species from which all species in the taxon *Eutheria* are descendant. The science of naming and classifying organisms is called *taxonomy*, a branch of phylogenetics. Taxonomists would prefer to term a (present or ancestral) species as a *taxon* (*pl.* taxa)[2]

A *gene* is defined as: "A DNA segment that contributes to phenotype/function. In the absence of demonstrated function a gene may be characterized by sequence, transcription or homology" [3]. Genes are related in the context of evolution. *Homologs* are genes that are descendant from a single gene (Koonin, 2005). *Orthologs* are genes in different species that originate from a single gene in the last common ancestor of these species (Remm *et al.*, 2001); furthermore, we say m genes in one species and n genes in another form $m : n$ orthologs if the $m + n$ genes are all descendant from a single gene in the LCA of the two species (Figure 1.2). By definition, orthologs must be homologs, and they are closer homologs; homologs that are not orthologs are *paralogs*. Homologous genes form a *gene family* which is defined a group of gene that are descendant of a single ancestral gene. The evolutionary relationships between genes in a gene family can also be represented as a tree, named *gene tree*. Similarly, each external node of a gene tree stands for a present gene that can be observed nowadays, while each internal node for an ancestral gene that only existed in history.

Species trees and gene trees are related: species trees can be inferred from gene trees,

[1] http://www.edu.gov.nf.ca/curriculum/teched/resources/glos-biodiversity.html

[2] According to taxonomy, all species are classified at six major levels. In an order from most genernal to most specific, the six levels are kingdom, phylum, class, order, family, genus and species. For example, *human* belongs to such groups: *Animalia* (kingdom), *Chordata* (phylum), *Mammalia* (class), *Primates* (order), *Hominidae* (family), *Homo* (genus), and *Homo sapiens* (species).

[3] http://www.gene.ucl.ac.uk/nomenclature/guidelines.html

and gene trees reflect patterns in species trees. At the same time, gene trees can differ from species trees due to the existence of duplications, losses, and lateral gene transfer (LGT), which can, in turn, be inferred by reconciling a gene tree and a species tree. One of the exceptional features of TreeFam, and therefore of this thesis, is to untangle the relations between species trees and gene trees. Many meaningful conclusions, such as the functional evolution and differentiation of genes, can be made when these relations are resolved. Such relations also help to reconstruct reliable gene trees.

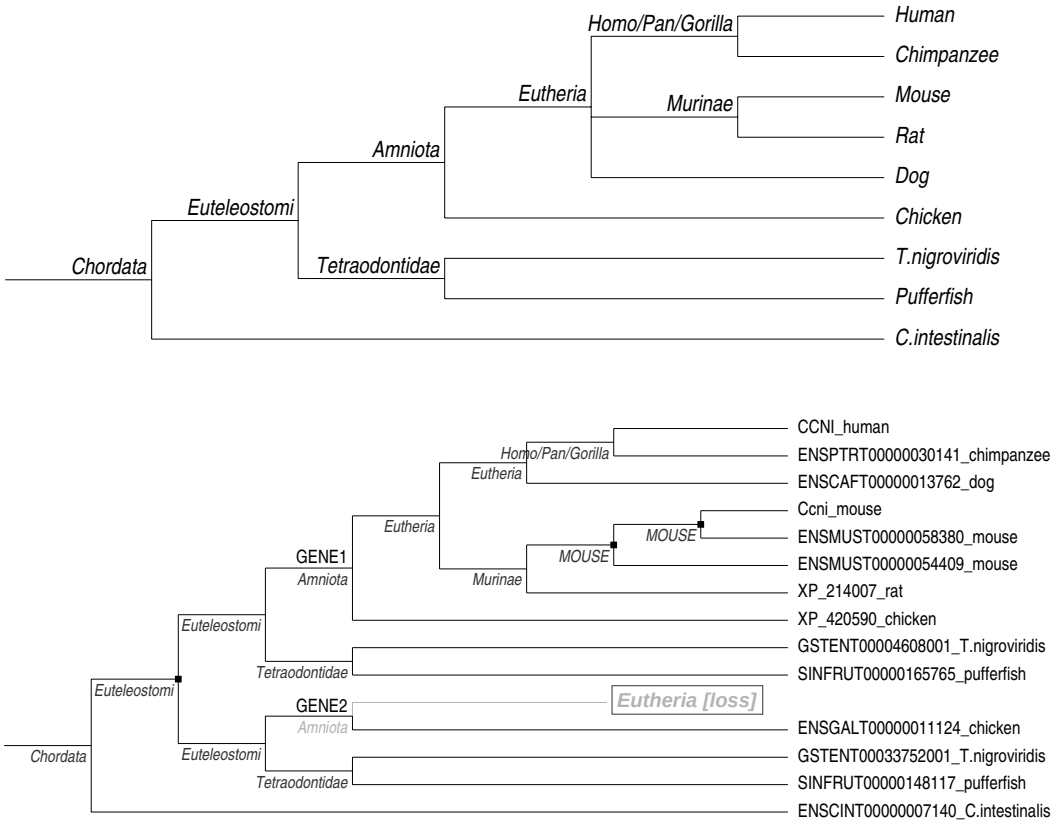

Figure 1.2: Example of a species tree and a gene tree. The top figure shows a species tree of *Chordata* phylum. This tree will be used throughout this thesis. The bottom shows the gene tree of *Cyclin I* sub- gene family. A slant name beside the internal node shows the ancestral species that contained the correponding ancestral gene. In this gene tree, the human gene CCNI_human is 1:1 orthologous to XP_420590_chicken because they are descendant from a single gene in *Amniota*, the LCA of *human* and *chicken*. But this human gene is only homologous to, but not orthologous to, ENSGALT00000011124_chicken because although both genes are descendant from a single gene in *Euteleostomi*, the human gene is a descendant of GENE1 in *Amniota* while ENSGALT00000011124_chicken of GENE2. In addition, this gene tree also shows several 1:n orthologs. For example, gene ENSCINT00000007140_C.intestinalis is an 1:2 ortholog of SINFRUT00000165765_pufferfish and SINFRUT00000148117_pufferfish. The bottom figure will be reexamined in Chapter 4 where more details will be presented.

1.4 Terminology for Trees

1.4.1 Common terminology for trees

In graph theory, a tree is a connected graph that contains no cycle. It is comprised of *nodes* (vertices) and *branches* (edges) which connect nodes. A node can be external (terminal) if it is only attached to one branch, or internal if two or more branches are joined by the node. An external node is also termed as a *leaf*.

Strictly speaking, every true phylogenetic tree is a *rooted tree* that presents the direction of evolution. The root node is regarded as the earliest ancestor of all the species or genes in the tree. In a rooted tree, each internal node has its children; except the root, each node has one parent. The branches close to the root are called higher or deeper branches, while close to the leaves called lower branches. Given a set of nodes, the lowest ancestral node from which all the given nodes are descendant is called LCA (Last Common Ancestor). A node and all the nodes descendant from it form a *clade*.

A true phylogenetic tree is also a *binary tree* in which each internal node has exactly two children. A binary tree is also called a *resolved tree* in the sense that every evolutionary event is presented in the tree. However, if two successive events happened in a short time, they can be hardly discriminated when the tree is reconstructed. In practice, nodes having three or more children are still allowed. These nodes are called *polytomies* or *unresolved nodes* or *multifurcated nodes*. A tree containing polytomies is an *unresolved tree* or *multifurcated tree*. The widely used NCBI taxonomy tree (Wheeler *et al.*, 2005) is a multifurcated tree.

1.4.2 Representation of trees

The most intuitive way to view a tree is to display it as an **image**. However, an image can neither be directly described in mathematical language nor be easily parsed by a computer program. Thus it is necessary to introduce abstract languages to shape a tree and to describe related algorithms as well.

Three types of representations will be used in the thesis: graph representation, string representation and set representation. The first two will be explained here and the third one will appear in Chapter 5 where set representation, being more abstract, helps to clarify the concepts and proof in a strict manner.

Graph representation: definitions and notations

Definitions and notations in this section will be used throughout the thesis, except in Chapter 5 where new notations will be introduced to facilitate specific aims. Only *rooted* trees will be discussed here as any phylogenetic tree is rooted given the assumption that all organisms on Earth had a common ancestor. In Section 3.3, we will come to the topic of how to find the root of an unrooted tree.

Let T be a *rooted* tree and $V(T)$ be the set of nodes in T. Let $V_E(T)$ and $V_I(T)$ be the set of external nodes (or leaves) and internal nodes, respectively. Given a node v, its direct descendants form $\text{chi}(v) \subset V(T)$, v's children set, and v is a child of $\text{par}(v) \in V(T)$, the parent node of v.

For $u, v \in V(T)$, we write $u < v$ or $v > u$ if u is a descendant of v. The set of external nodes that are descendent from v is defined as:

$$\omega_T(v) = \{u \in V_E(T) : u \leq v\}. \tag{1.1}$$

Similarly, the set of external nodes of set $A \subset V(T)$ form the set:

$$\omega_T(A) = \bigcup_{u \in A} \omega_T(u). \tag{1.2}$$

Sometimes, we might abbreviate $\omega_T(v)$ as $\omega(v)$ if T can be understood from the context.

Next, we seek to construct a subtree of T. For a node set A, we denote by $\text{lca}(A)$ the last common ancestor (or LCA) of A, which is the lowest node ancestral to all the nodes in A. Let $T|_A$ be the subtree comprising of only those paths in T that connect $\omega(A)$. Thus, the node set of $T|_A$ is:

$$V(T|_A) = \{v \in V(T) : \omega_T(v) \cap \omega_T(A) \neq \emptyset\} \tag{1.3}$$

The root of $T|_A$ is $\text{lca}(A)$. We also write $T|_{\{v\}}$ as $T|_v$ for simplicity. Figure 1.3 shows an example related to these concepts.

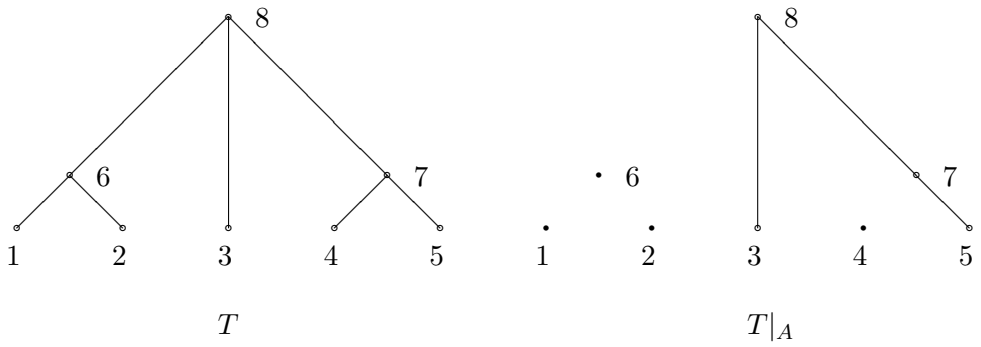

Figure 1.3: Example tree used to illustrate basic concepts. In the original tree T, $\text{lca}(\{4,6\}) = 8$, and $\omega(8) = \{1,2,3,4,5\}$. If $A = \{3,5\}$, the restricted tree $T|_A$ is showed in the right where $V_E(T|_A) = \{3,5\}$ and $V_I(T|_A) = \{7,8\}$. Note that in $T|_A$, node 7 only has one child. Tree T can be represented as a string in New Hampshire format: `((1,2)6,3,(4,5)7)8`, or `((5,4)7,(1,2)6,3)8` regardless of the order of leaves. They represent the same tree.

String representation: New Hampshire format

New Hampshire (NH) format, or Newick format, is the standard computer-readable format to store a *rooted* phylogenetic tree. It makes use of the correspondence between trees and nested parentheses, and represents a tree as a string, called an **NH string**.

NH format is clear and intuitive. Figure 1.3 shows how a tree can be represented as a NH string. The formal grammar of NH format is described as follows:

```
<tree>    →  <cell>
<cell>    →  <nhcell> | <nhcell>[<comment>]
<nhcell>  →  <node> | <node>:<dist>
<node>    →  <id> | (<list>) | (<list>)<id>
<list>    →  <cell> | <list> , <cell>
```

where `<comment>`, `<id>` and `<dist>` denote comment, identifier and distance, respectively. These symbols can be matched by sheer regular expressions and not showed here. Actually, the tree format used by TreeFam is New Hampshire eXtended (NHX) format, which allows for storing additional information in `<comment>` with a specified key-value structure. NHX format was first suggested by Zmasek *et al* (Zmasek and Eddy, 2001*b*).

It is important to note that one tree corresponds to many different NH strings, depending on the order of leaves appearing in the strings. All these equivalent strings reflect the same topology. In practice, we usually take as the preferred choice the string in which leaves appear in an order identical to what is showed in the 'image', and so an NH string has a natural 1:1 relation to the 'image' of a tree.

As discussed above, one tree can be displayed in different 'image' when plotted on a plane, or be represented in different NH strings. Although all these images or strings are equivalent to the same tree, they are quite different in human's eyes. Given two big trees with hundreds of leaves, it is formidable for a human to discern the difference between them if the leaves do not appear in similar orders. To find universal rules to plot a tree is quite necessary, and this is what we will seek to solve in Section 3.5.

1.4.3 Comparing two unrooted trees

Given a set V, a *bipartition* of V is a pair of disjoint sets $A|B = B|A$ that satisfy: (i) $A \cap B = \emptyset$ and (ii) $A \cup B = V$. Let tree $T = (V(T), E(T))$ be an *unrooted* tree whose node (vertex) set is $V(T)$ and branch (edge) set is $E(T)$. Each branch of T can divides T into two disjoint parts. If we let the leaf sets of the two disjoint parts be A and B, $A|B$ is a bipartition of $V(T)$. In this way, each branch of T uniquely corresponds to a bipartition of $V(T)$, and therefore we can construct a set:

$$\tilde{T} = \{A|B : \text{there is a branch corresponding to } V(T)\text{'s bipartition } A|B.\} \tag{1.4}$$

Obviously, $|\tilde{T}| = |E(T)|$. As the maximum value of $|E(T)|$ is $2 \cdot |V(T)| - 3$ when T is a binary tree, the maximum value of $|\tilde{T}|$ is also $2 \cdot |V(T)| - 3$. In the rest of this section, when we say 'a branch $A|B$ exists in T', we actually mean that $A|B \in \tilde{T}$, or equivalently, that there is edge corresponding to the bipartition $A|B$. Thus the topologies of unrooted trees that have the same leaf set V can be compared.

Let T_1 and T_2 be two unrooted trees with $V(T_1) = V(T_2)$. The Robinson-Fould distance measures the topological difference between them, which is defined as:

$$d_T(T_1, T_2) = |(\tilde{T}_1 \cup \tilde{T}_2) \setminus (\tilde{T}_1 \cap \tilde{T}_2)| \tag{1.5}$$

This distance counts the number of branches that exist in one tree but not in the other. Note that branches that connect leaves always exist in both $\tilde{T}_1$ and $\tilde{T}_2$, and therefore the maximum value d_T is $2 \cdot (|V(T_1)| - 3)$, which can be reached given two totally different binary trees.

Chapter 2

Constructing the TreeFam

Genes are related in the context of evolution. To transfer functional annotations across different species and to study the principle of gene evolution, it is important to untangle the relationships between the genes in a gene family. The best way to achieve this is to reconstruct a phylogenetic tree, which not only presents most of evolutionary details, but also provides the basis for further studies such as ortholog inference and gene annotation. TreeFam (Li *et al.*, 2006) (Tree families database) is a database of phylogenetic trees of gene families. It aims to develop a curated resource that presents the accurate evolutionary history of all metazoan animal gene families, as well as various inference based on phylogenetic trees.

In this chapter, we will introduce TreeFam database, its contents, structures, and pipelines that are used to construct TreeFam. As detailed procedures used by TreeFam-1.x have already been described in our paper, we will put more emphasis on the difference between release 1.x and 2.0 when presenting the whole pipelines.

2.1 Overview of TreeFam

2.1.1 What is TreeFam?

TreeFam is first a protein classification database. It aims to classify genes into families, and assign a name to each family and significant subfamily. Distinguished from most of other similar databases like KOG (Tatusov *et al.*, 2003), PANTHER (Mi *et al.*, 2005) and SYSTERS (Meinel *et al.*, 2005), which defines families according to the degree of similarities between family members, TreeFam aims to define a gene family as a group of genes that descended from a single gene in the last common ancestor of all metazoan animal, or that first appeared in metazoan animals. Evolutionary rates, reflected by similarity scores, may vary greatly in different groups of genes. Families defined by similarity based methods will be inevitably sensitive to the threshold that is used at clustering stage, and also lack biological meaning. In contrast, TreeFam will not suffer from this problem. As is pointed by PhIGs (Dehal and Boore, 2005), which is one of the basis of TreeFam, families defined in the phylogenetic way are not sensitive to similarity threshold, and therefore more robust and meaningful from the evolutionary angle.

Secondly, TreeFam is an ortholog database. It infers orthologs and paralogs from the phylogenetic tree of a gene family. Traditionally, orthologs are inferred from pair-

wise alignment between two species with a little help of synteny for close related species. Most of ortholog database at present are based on this method, including NCBI HomoloGene (Wheeler *et al.*, 2005), Ensembl-compara (Hubbard *et al.*, 2005), OrthoMCL (Li *et al.*, 2003), and Inparanoid (O'Brien *et al.*, 2005). Although these databases provide useful information about ortholog assignment, they might fall short when gene loss interferes. In the absence of genes from third-part species, pairwise method might overlook the existence of duplications and therefore overestimate orthologs. Furthermore, their inference between different pairs of species might be inconsistent. For example, assume gene g_1 is a 1:1 ortholog of g_2, and g_2 is a 1:1 ortholog of g_3. Then g_1 should also be one-to-one orthologous to g_3 in theory. But in pairwise framework, it is possible that g_1 is not an ortholog of g_3. This is because the three pairs, (g_1, g_2), (g_2, g_3) and (g_3, g_1), are processed separately. It is not required that they should be consistent. In contrast, Tree-based method, considering genes from multiple species as a whole, will not suffer from these problems. It is also more intuitive and informative in comparison with pairwise method. To the best of our knowledge, only HOGENOM (Dufayard *et al.*, 2005) has inferred orthologs from phylogenetic trees.

Finally, TreeFam is a curated database. Evolution diverges in different gene families and also in different lineages. To automatically reconstruct reliable phylogenetic trees is always one of the most challenging topics in the area of bioinformatics. This is why biologists have to reduce to pairwise methods even when they know the advantage of tree-based methods. Although in TreeFam we have developed new algorithms to improve the accuracy, automatic process is not comparable to human curation. Only human being can successfully integrate information from various resources; only human being can endow a tree with biological meaning. TreeFam is unique because it incorporates human curation.

2.1.2 Basic Structures of TreeFam

The basic structure of TreeFam resembles that of Pfam (Bateman *et al.*, 2004), a curated resource for protein domains. Like Pfam, TreeFam is also a biparted database: part A that consists of curated tree families, and part B that consists of automatically generated families. Each part also comprises of two types of sequences: seed that come from either manual curation (for TreeFam-A) or PhIGs clusters (for TreeFam-B), and full sequences that are expanded from seeds. Figure 2.1(A) presents the overall relations between these concepts. Detailed pipelines will be explained in following sections.

2.2 Input Data

2.2.1 Sequence data

TreeFam 2.0 collected protein and coding sequences for 19 fully sequenced species from GenBank (Wheeler *et al.*, 2005), SGD (Balakrishnan *et al.*, 2005), WormBase (Chen *et al.*, 2005), GeneDB (Hertz-Fowler *et al.*, 2004) and Ensembl (Hubbard *et al.*, 2005) (Table 2.1). Protein sequences for species that have not been sequenced were obtained from UniProt (Bairoch *et al.*, 2005). Genomic locations and Pfam (Bateman *et al.*, 2004) domain structures of genes were also included if available. In this chapter, TreeFam

sequence set will be called TFSEQ for simplicity.

Of the 19 species, two yeasts and *A.thaliana* serve as outgroup species. In a gene tree, genes of outgroup speices help to reveal ancestral species earlier than the LCA of metazoan animals, and thus indicate when two groups of animal genes should belong to two TreeFam gene families instead of one. Outgroups set the boundaries of gene families.

2.2.2 Original seeds

The start point of the entire TreeFam pipelines is the construction of seeds of TreeFam-B families, which has to be achieved by clustering methods. Fortunately, PhIGs has already done this in the way we desired. In PhIGs, clusters are inferred to have descended from a single gene from the last common ancestor of a specified group of species. Genes are grouped together following the evolutionary relations between species, instead of sheer similarity scores. At present, two types of clusters are presented in PhIGs website: one for all vertebrates, and the other for all metazoan animals. Ideally, a gene cluster of the latter type is exactly equivalent to a gene family defined in TreeFam, and therefore TreeFam skips the clustering step and directly uses PhIGs clusters as the

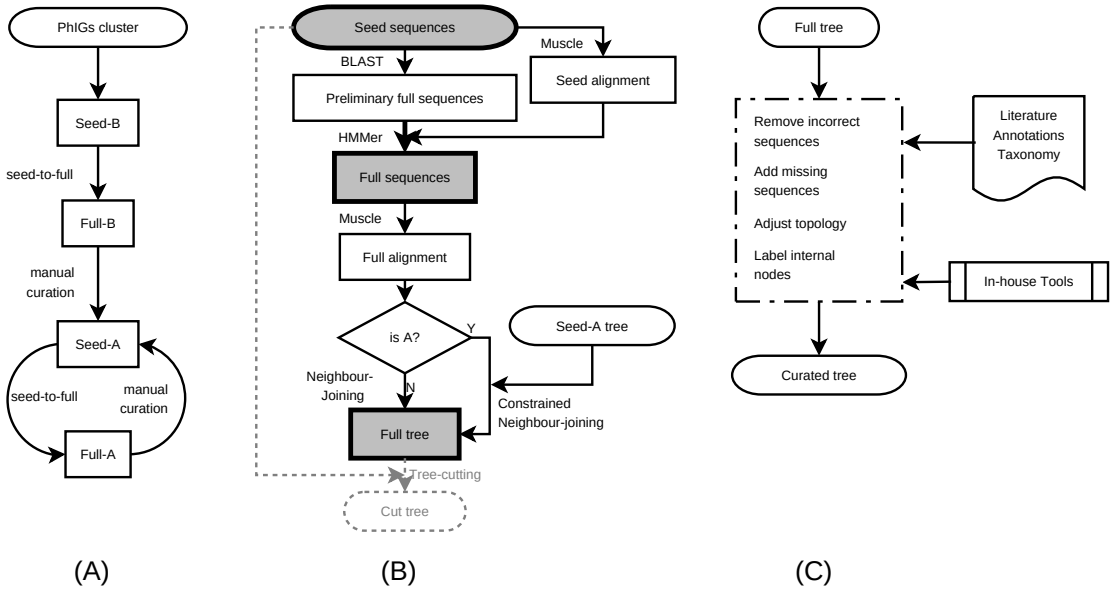

Figure 2.1: Flowchart of TreeFam pipeline. (A) Overall strategy. The seed families for TreeFam-B are taken from PhIGs clusters. They are expanded by a seed-to-full procedure to form full families. Manual curation makes TreeFam-B families become TreeFam-A families, which can also be curated further at a later date. (B) The seed-to-full procedure. Dashed lines and gray texts show the actions only present in TreeFam-1.x. In the boxes with gray background, details have been changed since TreeFam-2. Seed-to-full procedure is used to expand seed sequences to full sequences. Note that the complete procedure is only applied when the sequences sets are updated or a whole new genome is added to TreeFam. For TreeFam-A families created by curation of a TreeFam-B families, the TreeFam-A seed is generated by manual curation, and the full sequences are taken directly from the TreeFam-B that was curated. (C) Manual curation. Various published resources and in-house tools are utilized in this process.

original seed sets.

2.2.3 Miscellaneous data

A phylogenetic tree not only presents the history of a gene family. It also provides the basis for various evolutionary researches, such as studies related to intron evolution, domain evolution, and shift of gene functions. To help these studies and to discover the principles behind evolutionary phenomena, we also integrate several other resources with TreeFam. Since TreeFam-2.0, splicing information and domain structures of each gene have been available. Expressional profiles of genes will be provided in TreeFam-3. Phenotypes from a knockout screen and GO ontology (Ashburner *et al.*, 2000) categories may also be considered in future.

2.3 Automatic Pipelines

Pipelines used for constructing TreeFam 2.0 mainly differ from those for 1.x in two aspects: grouping of PhIGs clusters and use of the competitive method. As TreeFam paper has already presented detailed procedures used in 1.x series, we only focus on the 2.0 pipelines in this section.

Tax ID	Tax Name	*abbr.*	Common Name
9606	*Homo sapiens*	HUMAN	Human
9598	*Pan troglodytes*	PANTR	Chimpanzee
10090	*Mus musculus*	MOUSE	Mouse
10116	*Rattus norvegicus*	RAT	Rat
9615	*Canis familiaris*	CANFA	Dog
9031	*Gallus gallus*	CHICK	Chicken
8364	*Xenopus tropicalis*	XENTR	Western clawed frog
31033	*Fugu rubripes*	FUGRU	Japanese pufferfish
99883	*Tetraodon nigroviridis*	TETNG	Green puffer
7955	*Danio rerio*	BRARE	Zebrafish
7719	*Ciona intestinalis*	CIOIN	
7227	*Drosophila melanogaster*	DROME	Fruit fly
7165	*Anopheles gambiae*	ANOGA	African malaria mosquito
7460	*Apis mellifera*	APIME	Honeybee
6239	*Caenorhabditis elegans*	CAEEL	
6238	*Caenorhabditis briggsae*	CAEBR	
4896	*Schizosaccharomyces pombe*	SCHPO	Fission yeast
4932	*Saccharomyces cerevisiae*	YEAST	Baker's yeast
3702	*Arabidopsis thaliana*	ARATH	Mouse-ear cress

Table 2.1: Fully sequenced species that are included in TreeFam-2. Two yeasts and *A.thaliana* serve as outgroup species that are used to indicate the boundary of a gene family.

2.3.1 Generating seeds of TreeFam-B families

In TreeFam release 1.x series, PhIGs clusters that consisted of three or more vertebrate sequences were directly used as the seeds of TreeFam-B families. However, as PhIGs seemed to apply a very stringent threshold at clustering stage, it sometimes split a gene family into several smaller clusters if family members were too divergent. This has been observed at times in curation processes, and made us consider clustering PhIGs results when TreeFam 2.0 was planned.

Since TreeFam 2.0, PhIGs clusters are grouped as follows. First, sequences of original clusters are aligned against TFSEQ by BLAST[1] (Altschul *et al.*, 1997) with E-value cutoff 0.01. Matched TFSEQs are searched again by HMMER[2] (Eddy, 1998) with E-value cutoff 0.1. HMM profiles in this step are built from MUSCLE[3] (Edgar, 2004) multialignments of original PhIGs clusters. A relational graph is then constructed from HMMER results. In this graph, each vertex is an original PhIGs cluster. A weighted edge is added if the HMMER results of two clusters share some animal genes. More precisely, given a PhIGs cluster u, let R_u be the set of TFSEQ animal genes whose HMMER bit-score are higher than scores of any outgroup genes matched by sequences in u. An edge is added between two clusters u and v if $R_u \cap R_v \neq \emptyset$. The weight of this edge is $\frac{|R_u \cap R_v|}{\min\{|R_u|,|R_v|\}}$. After the construction of this weighted graph, clustering can be achieved by various means based on graph theory. In TreeFam 2.0, we implemented a heuristic algorithm revised from Zdobnov *et al.* (Zdobnov *et al.*, 2002). Like most similar methods, this algorithm tries to find groups in which edges tend to be saturated [4]. Finally, grouped PhIGs clusters are merged to form a larger one, which is regarded as a new cluster in next round. In theory, this grouping process should be iterated for many times until the results are stable, but in practice we found that grouping two or more times will lead to many superfamilies that contain several animal families, seemingly due to the reason that HMMER becomes insensitive when clusters get larger. Consequently, only one round of grouping is applied. The results are then used as the seeds of TreeFam-B families.

2.3.2 Competitively assigning each sequence to one family

In TreeFam 1.x, family members were determined by tree cutting algorithm. This algorithm discarded homologs that are descendants of a different (paralogous) gene in the last common ancestor of animals. Ideally, such a behaviour is exactly what we

[1] BLAST (Basic Local Alignment Search Tool) is the most popular alignment program that finds the homologs of one sequence. It aligns one sequence against each sequence in a sequence set and measures the similarity by an alignment score. Then BLAST evaluates the significance of homology by E-value which is the mathematical expectation of the number of sequences having scores higher than the observed score when a random sequence is aligned against a random sequence set of the same size.

[2] HMMER is also a tool for homolog search. It builds an HMM (hidden Markov model) profile based on a multialignment. It then makes use of the profile to measure how well a sequence matches the given multialignment. Utilizing a set of sequences, HMMER is more sensitive and more accurate than BLAST which only performs pairwise alignment. However, HMMER is much slower than BLAST.

[3] MUSCLE is a multialignment program like Clustalw. It puts all sequences together and aligns them such that their homologous regions match with each other. Generally, MUSCLE is much faster and more accurate than Clustalw (Edgar, 2004).

[4] As a matter of fact, MCL algorithm (Enright *et al.*, 2002) is more sophisticated in this case.

desired. However, due to the tendency of missing distant homologs and the difficulties in reconstructing long branches, tree cutting malfunctioned at times and resulted in families that consisted of excessive members. Thus in TreeFam 1.x, one gene could belong to several or even dozens of families, which by itself was inconsistent and also caused much trouble in curation. In addition, as multialignments and phylogenetic trees had to be built before unnecessary sequences were discarded, huge multialignments that contained over 500 genes were encountered from time to time, which greatly aggregated computational burden. Consequently, we decided to develop a better strategy to assign sequences to TreeFam families.

In TreeFam-2.0, homologous sequences are still collected by a combination of BLAST and HMMER: homologs are first found by BLAST and then refined by HMMER with the profile built from seed multialignment by MUSCLE. After this step, one sequence is competitively assigned to one family which gives the sequence the highest HMMER score; if one family contains several splicing forms of one gene, only the transcript with the highest HMMER score is retained. Family members are determined at this time.

To our observations, HMMER usually works quite well if the quality of seeds is high enough. Most of animal genes, especially vertebrate genes, can be correctly assigned in this way. However, when seed contains few sequences or the seed alignment is too poor, HMMER score will be inaccurate and cause errors. Furthermore, the existence of orphan genes and the incompleteness of the whole seed set also interfere. Even if a gene does not belong to any existing TreeFam families, it will still be forcedly assigned no matter how low the BLAST and HMMER scores are. In the next release of TreeFam, this problem will be paid more attention.

2.3.3 Tracing sequence identifiers for TreeFam-A families

TreeFam stores curated knowledges in TreeFam-A seed trees. However, with the update of sequence sets, genes in previous seed trees might be absent in the latest sequence sets when the identifiers have been changed. In this case, some curation cannot be reserved, and loss of of information occurs. And continuous loss of information will finally wipe out all the effort in curation. To slow down this process is very important to a curated database like TreeFam.

In TreeFam-2.0, a particular pipeline is designed to trace sequence identifiers when they are changed between releases. The basic idea is to map a new identifier to an old one if [5]: (i) the two sequences come from the same species; (ii) the two are almost identical in matched regions; and (iii) no other sequence is closer to either of the two sequences. The condition (i) and (ii) are obvious and can be easily judged. The third condition can be tested by studying the tree that consists of both unchanged sequences and sequences whose identifiers are only present in one release, either old or new. In this tree, condition (iii) is satisfied if a new identifier and an old one share the same parent node. Figure 2.2 shows an example where violation of condition (ii) and (iii) occurs.

[5]These rules will be changed since TreeFam-3.0.

2.3.4 Building trees

With irrelevant distant homologs discarded in TreeFam-2 families, family trees are much smaller than those in TreeFam-1.x, which makes it possible to reconstruct trees with more accurate, yet slower, algorithms such as maximum-likelihood methods. In addition, tree merge algorithm (Chapter 5) is also applied more frequently, and further improves the quality of phylogenetic trees.

In the new TreeFam-B, a full tree is reconstructed, at protein level, by neighbour-joining constrained (Section 3.2) by d_m tree, the synonymous-nonsynonymous merged tree. In TreeFam-A, full trees are further required to be constrained by seed trees. While tree reconstruction at protein level gets more sequences involved, the use of synonymous tree improves the topologies at lower branches. In the effort to build even better automatic trees, we also build a 'clean tree' that is merged from four trees including ML protein tree, ML nucleotide tree, NJ synonymous tree and NJ nonsynonymous tree (Chapter 5). This method really outperforms all the others both in our benchmark and to our experience in manual curation processes (Chapter **??**).

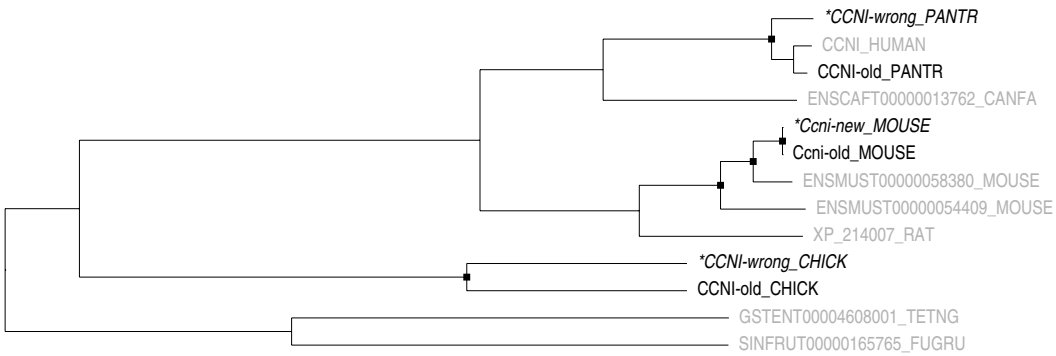

Figure 2.2: Example explaining trace-ID procedure. This tree consists of both unchanged sequences (in gray) and sequences whose identifiers differ between releases (in black). Sequences only present in the new release are highlighted in slanting font. Identifier `CCNI-old_PANTR` shall not be mapped to `*CCNI-wrong_PANTR` because `CCNI_HUMAN` is closer to the former one (violating (iii)). ID `CCNI-old_CHICK` cannot be mapped to `*CCNI-wrong_CHICK` because they are not close enough (violating (ii)) although they share the same parent node. Only the two mouse genes can be mapped.

Chapter 3

Reconstructing Gene Trees

Phylogenetic trees represent the history of evolution instead of measuring the similarity between sequences. Since the history can never be changed, correct trees are always there. Good algorithms must correctly recover the history. On the other hand, as no one knows what exactly happened in history, phylogenetic trees have to be reconstructed based on present observations. While fossils are essential evidence to reconstruct species trees, reconstructing gene trees can only utilize sequence similarity because fossils are usually insufficient and unable to date molecular events. Consequently, how to precisely recover the true history from sequence similarity becomes the eternal topic in evolutionary studies.

This chapter is to describe how TreeFam reconstructs a gene tree provided a multialignment. After an overview of various tree-building algorithms, constrained neighbour-joining, one of the exceptional algorithms developed in TreeFam, will be presented. Rooting a tree will be discussed in the next. Bootstrapping test and leaves reordering algorithm are also introduced here because the two topics are closely related to tree building process. In TreeFam-1.x, neighbour-joining is the only algorithm we used to build trees, but since the release of TreeFam 2.0, more sophisticated algorithms, including maximum-likelihood method and tree merge, are applied. These methods are not covered in this chapter, but are discussed in Chapter 5 and evaluated in Chapter 6.

3.1 Overview of Tree Building Algorithms

Whereas phylogenetic trees can be reconstructed with various evolutionary characters, molecular phylogenetics mainly focuses on tree reconstruction from multiple-sequence alignments. Throughout this thesis, multialignment is the starting point of tree reconstruction. Most of algorithms discussed in this thesis will take a multialignment as the unique input. Only in Chapter 5, auxiliary information will be integrated with tree building process.

Tree building algorithms can be classified into four types: distance based, parsimonious, maximum-likelihood, and Bayesian methods. Each type of algorithm has its own strength and weakness. In meticulous case studies, all kinds of algorithms should be tried in attempt to reconstruct a reliable phylogenetic tree. This has been confirmed by the benchmark done in Chapter 5.

3.1.1 Distance based algorithms

Distance based method is actually the name of a group of algorithms that reconstruct a tree from a distance matrix (d_{ij}) where an element d_{ij} measures a kind of distance between sequence pair i and j. Distances are usually, though not necessarily, calculated from a multialignment. They can be alignment scores, number of mismatches (also called p-distance), or historical substitution frequencies estimated from statistical models.

Of all distance based algorithms, UPGMA (Sokal and Michener, 1958) is the earliest. It mimics hierarchical clustering processes, joining the pair of closest nodes. UPGMA assumes the divergence of sequences occurs at a constant rate at all branches (molecular clock), which, unfortunately, does not always stand. Lateral algorithms do not arbitrarily require this hypothesis any longer, though the existence of a molecular clock help to improve the accuracy of tree building. In 1967, Cavalli-Sforza and Edwards (Cavalli-Sforza and Edwards, 1967) suggested that tree can be built such that the length of the path between any leaf pair in the tree should approach the distance given by the distance matrix. This gave the rise of LS (least square) method. The reconstruction of a LS tree require to search the whole tree space for the optimal tree, which is formidable. Thus, Fitch and Margoliash (Fitch and Margoliash, 1967) further simplified this procedure and presented a algorithm that was practically useful. The breakthrough came in 1987 when Saitou and Nei (Saitou and Nei, 1987) invented NJ (neighbour-joining) algorithm. Instead of joining a pair of closest nodes at each step like UPGMA, NJ joins a neighbour and does not assume the existence of a molecular clock. It is simple, accurate and efficient. Two alternatives, BIONJ (Gascuel, 1997) and Weighbor (Bruno *et al.*, 2000), were also developed by improving the joining procedures. As a matter of fact, neighbour-joining is a greedy approximation of ME (minimum evolution[1]) (Rzhetsky and Nei, 1993). It follows ME rule at each separated step, but does not build a true ME tree. Recently, Desper and Gascuel (Desper and Gascuel, 2004) developed BME framework (balanced minimum evolution) where ME tree can be precisely reconstructed. Their program FASTME was claimed to be both faster and more accurate than NJ and all its offshoots, which is partly confirmed in Chapter 6.

Distance based method is popular mainly for its unparalleled speed. It is the only hope for reconstructing a phylogeny containing as many as 1000 sequences. However, distance method is always criticized for its loss of information: substitutions on each site between two sequences are compacted into a single number. This compression not only affects the accuracy, but also limits the further application. For example, it is not allowed to reconstruct ancestral sequences, or to estimate evolutionary parameters such as transition/transversion ratio and Γ rate in the distance based framework. In those cases, we must fall back to other methods.

[1]Minimum evolution (ME) principle regards that nature favours the tree with minimum sum of the lengths of the branches. In some way, it resembles maximum parsimony principle introduced in next subsection. Rzhetsky and Nei (Rzhetsky and Nei, 1993) prooved that ME tree is correct 'provided that the evolutionary distances used are statistically unbiased and that the branch lengths are estimated by the ordinary least-squares method'.

3.1.2 Maximum parsimony

Given a phylogenetic tree and a multialignment, we can trace back the history presented by the tree and calculate the minimum number of substitutions that are required to explain the given data. Different trees usually yield different numbers. MP (Maximum parsimony) algorithm (Fitch, 1971) works by searching the whole tree space for the tree that minimizes the minimum number of substitutions.

Based on the principle of Ockham's razor [2], MP algorithm is most appealing for its simplest assumption: a true tree tends to contain fewer substitutions. No evolutionary model is in need to build a parsimonious tree. While this advantage attracts many biologists, statistists see it in a different way. They criticized that MP is unable to reconstruct a correct tree when long branches appear (Felsenstein, 1978). They argued that MP is actually based on an irrealistic model where substitutions occur uniformly both across sites and across subtrees (Holder and Lewis, 2003). They claimed that MP yields less information in comparison to max-likelihood methods. However, although being faced with these long-lasting disputes, MP remains popular and shows its power especially for close homologs.

3.1.3 Maximum-likelihood and Bayesian methods

ML (maximum-likelihood) method (Felsenstein, 1981) finds the tree that maximizes the likelihood given a specified evolutionary model. Regardless of efficiency issues, the procedure of ML reconstruction is very simple: calculating the likelihood and finding the best tree. While ML method presents the theoretical framework, its power is actually determined by statistical models, which also makes ML method very versatile due to the flexibility of models. Various evolutionary information can be smoothly inferred in the statistical framework. ML method is definitely the most fruitful method of all the tree building algorithms. Furthermore, ML method was also claimed to be the most accurate tree building algorithms in several studies (Kuhner and Felsenstein, 1994; Hall, 2005). Most of molecular evolutionists will agree that the use of ML method reformed the system of phylogenetics.

However, applying ML method is extremely computation extensive. Even when a number of heuristic algorithms were designed to accelerate tree searching (Olsen *et al.*, 1994; Guindon and Gascuel, 2003; Stamatakis *et al.*, 2005), building a tree with hundreds of leaves is still formidable, not to speak bootstrapping the alignment (Section **??**). In addition, although ML method allows for various evolutionary models in use, these models have to be preset in softwares and cannot be combined at will. This affects its further use in complex phylogenies.

Bayesian method can be regarded as an alternative to ML method. But instead of actively traversing the whole tree space to find the one with maximum likelihood like ML, it finds the one that maximizes posterior probability by sampling phylogenetic trees through MCMC (Markov Chain Monte Carlo) based on the specified model (Rannala and Yang, 1996). The most exciting advantage of Bayesian method is its unparalleled flexibility. As it just generates trees instead of calculating the probability of a given

[2] Given two equally predictive theories, choose the simpler, and the simplest answer is usually the correct answer. *See also* http://en.wikipedia.org/wiki/Occam's_Razor.

tree, complex analytical computations can be avoided. Thus, various complex models can be easily designed and combined together, and by summarizing from a bulk of sampled trees, parameter estimation also becomes painless. Furthermore, sampled trees in MCMC process can be considered as resampled trees to some extend (Larget and L, 1999). Support values are naturally calculated with less computational resources, though these values may be 'excessively high' as pointed out by several works (Suzuki *et al.*, 2002; Cummings *et al.*, 2003; Simmons *et al.*, 2004).

3.2 Constrained Neighbour-Joining

All traditional tree-building algorithms are unsupervised in that they never make use of additional knowledge provided by a human. Even if we know a leaf set A should be separated from B, they can be still mixed up in the resultant tree. In TreeFam, however, we must preserve the topology of the seed tree, or the knowledge of curators, when building a full tree. This is a supervised procedure. New algorithm must be designed for this goal; otherwise the effort of curation would come to naught when the full tree does not utilize the curated topology. In this section we introduce the first key algorithm of TreeFam, constrained neighbour-joining (cNJ), which can add new sequences to an existing tree without changing the given topology. We start with the standard neighbour-joining.

3.2.1 Standard neighbour-joining algorithm

A distance matrix is said to be *additive* if there exists a binary tree from which the length of the path [3] connecting each pair of leaves equals to the their pairwise distance presented by the matrix. In turn, if a distance matrix is additive, we can always reconstruct the tree that makes the additivity. Neighbour-joining (Saitou and Nei, 1987) is the algorithm that finds this tree.

Neighbour-joining works by adding a new internal node that joins a pair of neighbouring leaves at each step. Given a symmetric distance matrix $(d_{ij})_{n\times n}$, a neighbour pair is the pair of leaves that minimizes D_{ij} of all possible (i, j), where D_{ij} is defined as:

$$\begin{aligned} D_{ij} &= d_{ij} - (r_i + r_j) \\ r_i &= \frac{1}{n-2}\sum_{k=1}^{n} d_{ik} \end{aligned}$$

When a new node K is added to join a neighbour (I, J), the distances between them can be calculated as:

$$\begin{aligned} d_{IK} &= \frac{1}{2}(d_{IJ} + r_I - r_J) \\ d_{JK} &= d_{IJ} - d_{IK} \end{aligned}$$

And the distances between K and all the other nodes are:

$$d_{Km} = \tfrac{1}{2}(d_{Im} + d_{Jm} - d_{IJ}), \text{ for } m = 1, 2, \ldots, n,\ m \neq I \text{ and } m \neq J$$

[3] In a tree, the path length equals to the sum of lengths of the edges/branches in the path.

With neighbour I and J removed and K added, the original $n \times n$ distance matrix (d_{ij}) will be reduced to $(n-1) \times (n-1)$. This finishes one round of iteration. When this process is applied repeatedly until the matrix becomes 3×3[4], the three remaining nodes should be joined together. Tracing the joins at each step we can make a binary unrooted tree, which is the standard neighbour-joining tree. If the distance matrix is additive, this tree suffices additivity. Proof can be found in the book written by Durbin *et al.* (Durbin *et al.*, 1998).

Neighbour-joining algorithm joins the pair that minimizes D_{ij} instead of d_{ij}. This is one of the main points where neighbour-joining differs from UPGMA algorithm. Minimizing D_{ij} can avoid false-joining when long branches occur, and the existance of long branches will always break the molecular clock, which will fail UPGMA. This is the strength of neighbour-joining.

3.2.2 Constrained neighbour-joining algorithm

The basic idea of neighbour-joining algorithm is very simple. The only difference between the standard neighbour-joining and the constrained one is that constrained neighbour-joining disallows joining that are conflict with the topology of the given constraining tree. As in each step all pairs of leaves will be checked, check of valid joining must be fairly efficient. As a matter of fact, if we notice the fact that joining is only allowed between two leaves with the same parent node, we can contract the constraining tree by removing joined leaves and making the parent a new external node. Such contract is performed each time when two leaves in the constraining tree are joined together. As checking whether two leaves sharing a same parent can be done in $O(1)$, validating allowed joining can also be achieved in $O(1)$ time, and consequently, constrained neighbour-joining has the same time complexity with that of the original one, that is, $O(N^3)$ where N is the number of leaves. Figure 3.1 also gives an example. Detailed algorithm is described in Table 3.1. This table also presents the standard neighbour-joining algorithm.

3.2.3 Further discussion

As a matter of fact, constrained neighbour-joining presented above does not follow the neighbour-joining rule. It cannot ensure that an allowed pair of neighbours can always be joined. This is because constrained NJ assumes the constraint is a rooted tree. It will not consider other rooted topologies and arbitrarily join the nodes in the suffix order[5] of the given constraining tree. To explain this, let's see an example. Assume we have a tree `((1,2),3,4)` reconstructed by standard NJ. From the given tree, the only joining happens between leaf `1` and its neighbour `2`. Now, we continue to assume that the rooted topology of this tree should be `(((3,4),2),1)` and use this rooted tree as constraint. Then `(3,4)` must be joined first by constrained NJ described above, although the true neighbour `(1,2)` does not violate the constraint in the unrooted topology. Constrained NJ shuffles the order of joining and breaks NJ rule.

[4] This is because in an unrooted binary tree each internal node has three edges attached.

[5] If the nodes of a rooted tree are traversed in suffix order, lower nodes will always be visited earlier than higher nodes.

If the distance matrix was strictly additive, or equivalently, if the distance between any pairs of the leaves always equaled to the length of the path that connects the pair, the order of joining would be unimportant. Joining in any order will lead to the same tree, including the branch length. In the real world, however, strict additivity is rarely held. Even if a standard NJ is used as constraint, constrained NJ will frequently generate a tree with different branch length, though the topology is the same.

Given this fact, we also developed another version of constrained neighbour-joining that regards the constraint as an unrooted tree. On the example above, the new version will correctly join the neighbour `(1,2)` in the first step. This is also an $O(N^3)$ algorithm,

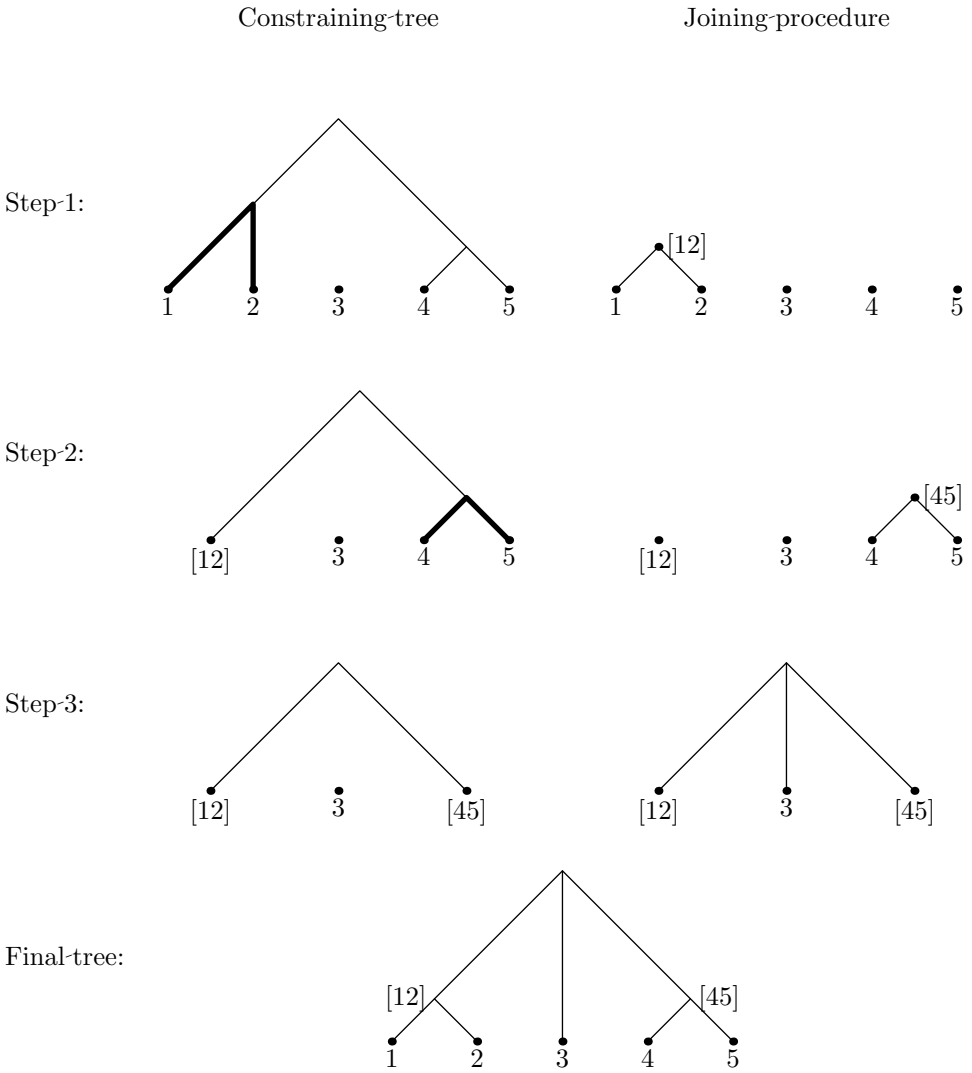

Figure 3.1: Example of constrained neighbour-joining algorithm. In the first three rows, the left is the constraining tree at current step; the right shows the remaining nodes and the best allowed joining. At step 3, only three nodes remains. They are joined together, which makes a trifurcation. The resultant tree is showed in the end. Note that no constraint is applied to node '3'. It can be freely joined to other if node '3' neighbours a certain node at some step.

but it can only be applied when the constraint is a binary tree; otherwise the time complexity will be increased. This algorithm is much more complex than the previous one and will not be described in this thesis.

Input:
Distance matrix (d_{ij})
A rooted constraining tree C with all its leaves appearing in the matrix
Output:
A neighbour-joining tree $T = (V(T), E(T))$
Procedure:
Let $\mathcal{L}$ be the set of nodes that appear in the distance matrix
$\mathcal{L}_C \leftarrow V_E(C)$
$V(T) \leftarrow \mathcal{L}$
$E(T) \leftarrow \emptyset$
while $|\mathcal{L}| > 3$ **do**
  for each node $i \in \mathcal{L}$ **do**
    $r_i \leftarrow \frac{1}{|\mathcal{L}-2|} \sum_{k \in \mathcal{L}} d_{ik}$
  $M \leftarrow \infty$
  for each node pair $(i, j) \in \mathcal{L} \times \mathcal{L}$ **do**
    $D_{ij} \leftarrow d_{ij} - (r_i + r_j)$
    if $D_{ij} < M$ **then**
* **if** $\{i, j\} \not\subset \mathcal{L}_C$ **OR** $\text{par}_C(i) = \text{par}_C(j)$ **then**
        $M \leftarrow D_{ij}$, $I \leftarrow i$ and $J \leftarrow j$
  $V(T) \leftarrow V(T) \cup \{K\}$
  $E(T) \leftarrow E(T) \cup \{(K, I)\} \cup \{(K, J)\}$
  for each node $m \in \mathcal{L}$ **do**
    $d_{Km} \leftarrow \frac{1}{2}(d_{Im} + d_{Jm} - d_{IJ})$
  $d_{IK} \leftarrow \frac{1}{2}(d_{IJ} + r_I - r_J)$
  $d_{JK} \leftarrow d_{IJ} - d_{IK}$
  $\mathcal{L} \leftarrow (\mathcal{L} \cup \{K\}) \setminus \{I, J\}$
* **if** $\{I, J\} \cap \mathcal{L}_C \neq \emptyset$ **then**
* **if** $\text{chi}(\text{par}_C(I)) = \{I, J\}$ **then**
* $\text{par}_C(K) \leftarrow \text{par}_C(\text{par}_C(I))$
* **else if** $I \in \mathcal{L}_C$ **then**
* $\text{par}_C(K) \leftarrow \text{par}_C(I)$
* **else if** $J \in \mathcal{L}_C$ **then**
* $\text{par}_C(K) \leftarrow \text{par}_C(J)$
* $\mathcal{L}_C \leftarrow (\mathcal{L}_C \cup \{K\}) \setminus \{I, J\}$
add a new node K that joins the remaining three nodes in $\mathcal{L}$

Table 3.1: Algorithm for constrained neighbour-joining. Lines labeled by '$*$' are specific to constrained neighbour-joining. By removing these lines, we can get the ordinary NJ algorithm.

3.3 Rooting Trees

Any phylogenetic tree is a rooted tree given the hypothesis that all organisms on Earth had a common ancestor. However, neighbour-joining and likelihood method can only reconstruct an unrooted tree, and so algorithms must be developed to root a tree. In fact, rooting is of the same importance as building a tree. A wrong root leads to a different story.

There are several ways to root a tree. The most widely used one is to select an outgroup sequence among all the leaves, and directly put the root on the branch that connects the outgroup. However, in lack of adequate biological evidence, outgroups are hard to decide especially for some complex gene families. It is impossible to automatically define an outgroup for every TreeFam family, and so this method is not applicable to TreeFam.

Another method is to minimize the height among all the possible rooted trees, which can be achieved by placing the root at the middle point of the longest chain of consecutive edges. It works well if deviations from molecular clock is not that much (Durbin *et al.*, 1998). This method was further extended in Bayesian framework (Huelsenbeck *et al.*, 2002).

A more sophisticated method, the one used by TreeFam, is to minimize the dissimilarity between the gene tree and a species tree (Zmasek and Eddy, 2001*a*), which can be achieved by enumerating all the possible rooted trees and choosing the one that contains fewest duplications and losses under parsimonious species map (Chapter 4). If all sequences in the multialignment belong to one gene familyand the gene family does not suffer from massive losses, this method is usually able to find the correct root. It is worth noticing that both duplications and losses should be counted when rooting a tree. Merely considering duplications will frequently result in several rooted topologies with the same number of duplications (Page and Charleston, 1997).

3.4 Bootstrapping: testing topological stability of unrooted trees

From the statistical point of view, sequences that are used to reconstruct a tree are randomly sampled from whole genome sequences, and therefore the topology of the tree built from these sequences is also a random variable to some extent. It is important to check whether a tree built from current observed sequences are stable in the sense that it is less affected by stochatic evolution events; otherwise the resultant tree will not be useful to represent the history of evolution.

Several methods are available to test the topological stability. The most widely used one is bootstrapping (Efron, 1979; Felsenstein, 1985). The input of bootstrapping are an unrooted tree T and its underlying multialignment $(X_{ij})_{n\times m}$ where X_{ij} is the residue (amino acid or nucleotide) at i-th position of the i-th sequence. Its output are the frequencies of each T's branch appearing in the resampled trees built from randomly generated multialigns from (X_{ij}). During one round of bootstrapping, the columns (sites) of (X_{ij}) is randomly resampled with replacement for m times. These resampled columns (sites) then form a new multialignment (X^r_{ij}), from which a resampled tree T^r

is built. For each branch $A|B \in \tilde{T}$, we add 1 to the counter at $A|B$ if $A|B \in \tilde{T}^r$, and add 0 if not. When we repeat this procedure for many, say 1000, times, we will know the counter at each branch of T. The value of these counters are called bootstrapping values. The higher, the better.

Mathematically, given a multialignment $(X_{ij})_{n\times m}$ and an unrooted tree T built from X, a resampled multialignment (X^r_{ij}) is a random matrix:

$$X^r_{ij} = X_{iJ_j} \tag{3.1}$$

where each random variable J_j, $j = 1, \ldots, m$, follows discrete uniform distribution between $[1, m]$. That is, for all $k = 1, \ldots, m$, $\Pr\{J_j = k\} = \frac{1}{m}$. Let T^r is the tree built from alignment X^r, the bootstrapping support for each bipartition $A|B \in \tilde{T}$ will be:

$$\mathcal{B}(A|B) = \Pr\{\text{bipartition } A|B \in \tilde{T}^r\} \tag{3.2}$$

The procedures described above can be adapted to calculate bootstrapping supports for two rooted trees, but this is seldom used given the difficulty in rooting. In addition, bootstrapping are also helpful to evaluate the stability of other properties of trees. The basic idea is similar to the procedure described above.

3.5 Reordering External Nodes

In mathematical angle, a tree is an abstract entity. Its leaves form a set and never have any sense of order. When the tree is visualized on a paper, however, we must place the leaves in certain order. Although the plotted trees on the plane represent the same tree no matter how the order is specified, these planar trees are quite different to human eyes. When one tree is plotted twice in differeny leaf orders, it is sometimes hard to recognize they are virtually the same; when two large trees are compared, finding inconsistent branches always turns out to be extremely difficult. How to plot the tree in a regulated way is very important to the visualization of phylogenetic trees.

This section aims to plot a tree, or equivalently to order the leaves, in a regulated way that makes similar trees always be plotted in a similar manner. To achieve this goal, we will assign a weight to each leaf, and design an objective function such that when optimized, leaves with smaller weights tend to be placed before those with bigger weights. This optimization can be achieved in $O(N)$ with an intuitive yet correct algorithm.

For simplicity, only *binary rooted* trees will be considered in this section. Generalization to multifurcated rooted trees is basically easy.

3.5.1 Leaf reordering problem and algorithm

Given a binary rooted tree G with n leaves (i.e. $|V_E(G)| = n$), an **order** of G is a one-to-one map $I : V_E(G) \to \{1, \ldots, n\}$, such that there exists an NH string satisfying $v \in V_E(G)$ appears in the $I(v)$-th position of the string. If G is a binary tree, 2^{n-1} different orders exist, representing independent branch switch at $n-1$ internal nodes. As discussed in section 1.4.2, an NH string has a natural 1:1 relation to the 'image' of a tree. An order I actually decides how the tree should be plotted on the paper

(Figure 3.2). If we assign a weight $W(v) \in \Re$ to each leaf $v \in V_E(G)$, **leaf reordering** problem is to find an order I that minimizes the object function:

$$\sum_{v \in V_E(G)} [W(v) - \alpha I(v)]^2 \tag{3.3}$$

where $\alpha > 0$ is a constant. As I is one-to-one, map I^{-1} must exist. We can transform Equation 3.3 as:

$$\begin{aligned}
& \min_I \sum_{v \in V_E(G)} [W(v) - \alpha I(v)]^2 \\
= & \sum_v W^2(v) + \sum_v \alpha^2 I^2(v) - 2\alpha \max_I \sum_v W(v) I(v) \\
= & \sum_v W^2(v) + \alpha^2 \sum_i i^2 - 2\alpha \max_I \sum_v W(v) I(v) \\
\sim & \max_I \sum_v W(v) I(v) \\
= & \max_I \sum_{i=1}^{n} i \cdot W(I^{-1}(i))
\end{aligned}$$

Thus leaf reordering problem is equivalent to find I that maximize

$$\sum_{i=1}^{n} i \cdot W(I^{-1}(i)) \tag{3.4}$$

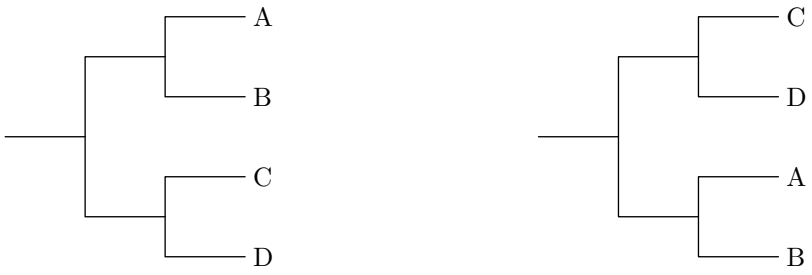

Figure 3.2: Example for explaining the order of a tree. Both trees showed in this figure are exactly the same tree but with different orders. In the left, the corresponding NH string is `((A,B),(C,D))`, and therefore $I(A) = 1$, $I(B) = 2$, $I(C) = 3$, and $I(D) = 4$; in the right, NH string is `((C,D),(A,B))`, and $I'(A) = 3$, $I'(B) = 4$, $I'(C) = 1$, and $I'(D) = 2$.

A naive solution to leaf reordering problem is to enumerate all the possible I. This works, but is of course a bad algorithm. As a matter of fact, branch switch at different internal nodes are independent of each other. This means the switch that lead to the reduction of the cost function 3.3 will always help to reduce the cost no matter how the other branches are switched. As a result, we can always apply $n-1$ times of independent switch to find the order I^* that minimize equation 3.3. Now the remaining issue is how to choose an optimal switching pattern at an internal node. In fact, the rule is quite

simple: at an internal node, switch the branch such that the average leaf weight of its left subtree smaller than the average weight of its right subtree. Let's prove it.

Figure 3.3 shows two orders: I, before switching, and I', after switching. Let $w_i \equiv W(I^{-1}(i))$ and $w'_i \equiv W(I'^{-1}(i))$, $i = 1, \ldots, n$. Then w_i is the weight of i-th leaf under the order I and w'_i is the weight of i-th under the order I'. From the caption of the figure, we know such relationships stand:

$$\begin{aligned} w_i &= w'_i && (i = 1, \ldots, j,\ j{+}k{+}l{+}1, \ldots, n) \\ w_{i+j} &= w'_{i+j+l} && (i = 1, \ldots, k) \\ w_{i+j+k} &= w'_{i+j} && (i = 1, \ldots, l) \end{aligned}$$

Then we can compare the cost function 3.4 before and after branch switching:

$$\begin{aligned} & \sum_{i=1}^{n} i \cdot W(I^{-1}(i)) - \sum_{i=1}^{n} i \cdot W(I'^{-1}(i)) \\ = & \sum_{i=1}^{n} i w_i - \sum_{i=1}^{n} i w'_i \\ = & \sum_{i=j+1}^{j+k+l} i w_i - \sum_{i=j+1}^{j+k+l} i w'_i \\ = & \left[\sum_{i=1}^{k} (i{+}j) w_{i+j} + \sum_{i=1}^{l} (i{+}j{+}k) w_{i+j+k} \right] - \left[\sum_{i=1}^{l} (i{+}j) w'_{i+j} + \sum_{i=1}^{k} (i{+}j{+}l) w'_{i+j+l} \right] \\ = & \left[\sum_{i=1}^{k} (i{+}j) w_{i+j} + \sum_{i=1}^{l} (i{+}j{+}k) w_{i+j+k} \right] - \left[\sum_{i=1}^{l} (i{+}j) w_{i+j+k} + \sum_{i=1}^{k} (i{+}j{+}l) w_{i+j} \right] \\ = & \sum_{i=1}^{l} k w_{i+j+k} - \sum_{i=1}^{k} l w_{i+j} \\ = & kl \cdot \left(\frac{1}{l} \sum_{i=1}^{l} w_{i+j+k} - \frac{1}{k} \sum_{i=1}^{k} w_{i+j} \right) \end{aligned}$$

In this formula, $\frac{1}{k} \sum_{i=1}^{k} w_{i+j}$ is the average leaf weight of the left subtree before branch

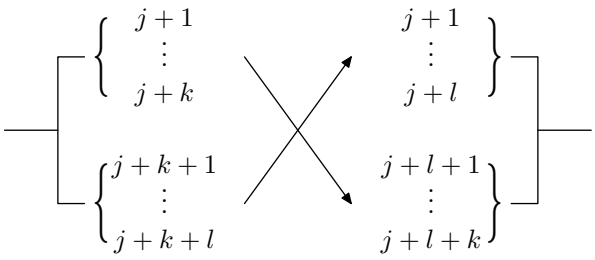

Figure 3.3: Two orders before and after branch switching. After switching, the $(j+1)$-th leaf in the left tree appears at the $(j+l+1)$-th position in the right, while the $(j+k+1)$-th leaf at $(j+1)$-th. Mathematically, $I'(I^{-1}(j{+}i)) = j{+}l{+}i$ when $i = 1 \ldots k$, and $I'(I^{-1}(j{+}k{+}i)) = j{+}i$ when $i = 1 \ldots l$.

switching and $\frac{1}{l}\sum_{i=1}^{l} w_{i+j+k}$ is the weight of the right. Thus a switch is only favoured when the average weight of the right subtree is lower. This proves the statement in the previous paragragh, and therefore proves the tree reordering algorithm.

3.5.2 Defining weight functions

Clearly, weight function $W(\cdot)$ determines how the tree should be ordered. Two forms of weight functions are now used in TreeFam. The first form is calculated based on an ordered (or a plotted) species tree; the second based on an ordered gene tree.

Assume we have a species tree S and an order $I_s : V_E(S) \rightarrow \{1, \ldots, |V_E(S)|\}$. The weight of $g \in V_E(G)$ can be defined as:

$$W_s(g) = I_s(M(g)) \tag{3.5}$$

where $M(g) \in V_E(G)$ is the species of gene g. Such a weight function tries to place the G's leaves in an order that resembles S.

Leaf ordering also helps to compare two trees with the same leaf set. Assume we have a plotted tree G_0 and its order I_0, and we want to display G_1, which is built with a different algorithm, in a similar manner. We can define:

$$W_0(g) = I_0(g) \tag{3.6}$$

If the two trees are identical, objective function 3.3 will equal to zero if $\alpha = 1$. Then the two will be plotted exactly in the same way. Nonetheless, it is worth to be noticed that equation 3.3 may also be zero if the two tree do not differ much (Figure 3.4). This is because the objective function only considers the order of leaves but ignores the changes in internal nodes. To compare the topological difference, counting number of shared branches is preferable. Chapter 6 gives more details.

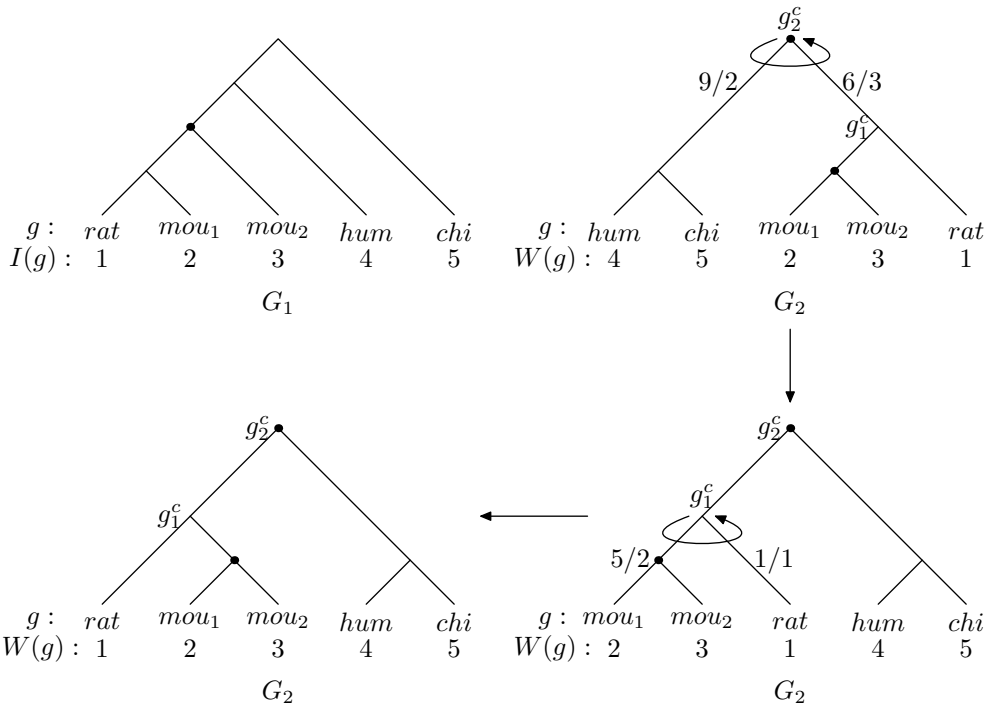

Figure 3.4: Example of leaf reordering algorithm. Tree G_1 is an ordered tree. The goal is to place the leaves of G_2 in an order similar to G_1. The top-right tree shows the original order. At node g_2^c, the average weight of left subtree is 4.5, larger than the weight of right subtree. A branch switch should be applied, which results in the image in the bottom-right. At g_1^c, the weight of left subtree 2.5 is the larger. A branch switch is needed again. In the final tree (in the bottom-left), G_2's leaves appear in the same order as that in G_1, although they have different topologies.

Chapter 4

Inferring Duplications and Losses

The evolution of Eukaryotic genes is largely shaped by three principal factors: speciation, duplication and loss. First of all, speciation differentiates genes into descendants. The story of each gene family also reflects the history of species. When an ancestor species speciates, all of its genes will be separated into two clades at the same time and evolve almost independently. Whereas speciations do not create new genes, duplications add more genes to a species and the descendant species as well, and thus provide candidates for natural selection. When more redundant genes come into being, losses, which tend to occur following a duplication, eliminate copies that are of little importance to the species. Genes evolve, birth and die, telling the complete history of individual gene families[1]. Figure 4.1 shows an example of how duplications and losses shape the evolution of genes. If we take the duplication/loss inference presented by the tree, the history of this gene family follows. In the earliest ancestral species *Chordata* there was only one gene. It underwent a duplication in the ancestral species *Euteleostomi*, and since then it had two copies in each species descendant from *Euteleostomi*. The first copy evolved normally except for two succesive duplications in *mouse*, the second was lost in *Eutheria* subclass. This is the story reflected by this gene tree.

The only way to infer duplications and losses is to compare the gene tree to the species tree (Figure 1.2). This process is usually called *tree reconciliation* which was first suggested by Goodman (Goodman *et al.*, 1979), and implemented and improved by many successors (Page, 1994; Guigo *et al.*, 1996; Eulenstein *et al.*, 1998). Recently, Zmasek and Eddy (Zmasek and Eddy, 2001*a*) have presented an elegant algorithm that greatly simplifies duplication inference. Dufayard *et al.* (Dufayard *et al.*, 2005) further developed algorithms that could handle multifurcated species trees, though no detail is provided. All these algorithms can be classified as parsimonious methods in that the number of duplications and losses is minimized. A Bayesian method (Arvestad *et al.*, 2003) was also developed, which is claimed to be more effective in reducing missing duplication events.

In this chapter, we will introduce a generalized theory for duplication/loss inference, and develop simple algorithms for such inference in the meantime. Our methods

[1]LGT (Lateral Gene Transfer), which is thought to occur between viruses, and to a lesser extent between prokaryotes, is also a possible factor in interpreting a gene tree. However, LGT seldom occurs among Eukaryotes (Stanhope *et al.*, 2001; Salzberg *et al.*, 2001; Roelofs and Van Haastert, 2001; Frickey and Lupas, 2004).

are based on Zmasek and Eddy's work, but generalized in case of loss inference and multifurcated species trees.

4.1 Species Map

To describe the inference of duplication/loss, it is necessary to bridge a gene tree and a species tree. This bridge is called *species map* that maps a gene, whether present or ancestral, to the species that historically possessed the gene. When we know to which species a gene belonged, we could pinpoint species evolution and gene evolution, reconstruct the history, and smoothly infer the duplications and losses.

Mathematically, let G be a gene tree, and S a species trees, both *rooted*. Given $g \in V_E(G)$, a present gene, let $s_g \in V_E(S)$ be the species of gene g. A map $M : V(G) \to V(S)$ is called a *species map* if it satisfies the following conditions for any $g \in V(G)$: (i) $M(g) = s_g$ if $g \in V_E(G)$, (ii) $s_{g'} \leq M(g)$ for all $g' \in \omega_G(g)$ and (iii) $M(g) \leq M(g')$ if $g < g'$. Map M embeds a species tree into a gene tree and describes how the evolution of species shapes gene evolution. In the following section, we will see that M alone determines duplications and losses, and thus how to choose M is the key to duplication/loss inference.

Given a specified gene tree and species tree, many maps will satisfy the two conditions. Likelihood methods choose the one that maximizes the likelihood under certain statistical model. This process is extremely time-consuming. In contrast, parsimony methods simply select the the parsimonious species map M^* that will lead to fewer duplications or losses in the end. Without any proof here, we point out that M^* can be simply constructed as:

$$M^*(g) = \text{lca}(\{s_{g'} : g' \in \omega_G(g)\}) \tag{4.1}$$

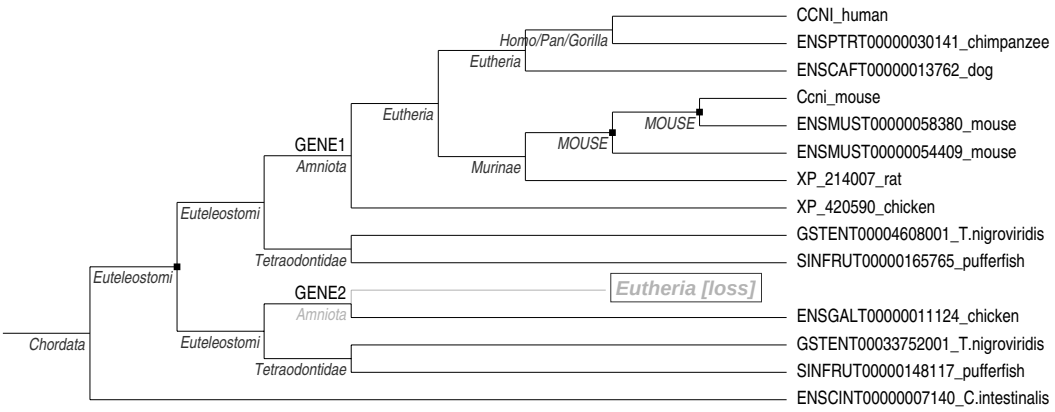

Figure 4.1: Example of gene evolution. A slant name beside an internal node shows the ancestral species that contained the correponding ancestral gene. They are actually $M^*(g)$ for each g under parsimonious species map (Section 4.1). Duplications (the three bold nodes) and loss (the gray boxed name) are inferred using methods that will be described in this chapter.

Obviously M^* is a species map, and in fact, $M^*(g)$ is the most recent possible species to which g belonged in history. Partly, this is also a reason why M^* is called a parsimonious map. Figure 4.2 gives an example of a parsimonious map and a common one.

4.2 Duplication/Loss Inference (DLI)

Ideally, a duplication can be observed if one species appears two or more times in a gene tree. This is true if no loss occur. But when loss occur, this will overlook duplications. We must recover the lost genes before we make such inference. For a binary species tree, we can simply compare $M(g)$ and $M(\mathrm{par}(g))$ (Zmasek and Eddy, 2001*a*), and infer $\mathrm{par}(g)$ as a duplication if $M(g) = M(\mathrm{par}(g))$; for a multifurcated tree, however, this rule will overestimate duplication events. As showed in Figure 4.3, both g and $g_p \equiv \mathrm{par}(g)$ are mapped to *Eutheria*, but g_p still represents speciation event, not duplication. In order to generalize DLI (duplication/loss inference) in case of a multifurcated species tree, it is helpful to explicitly recover the species that are lost in gene trees.

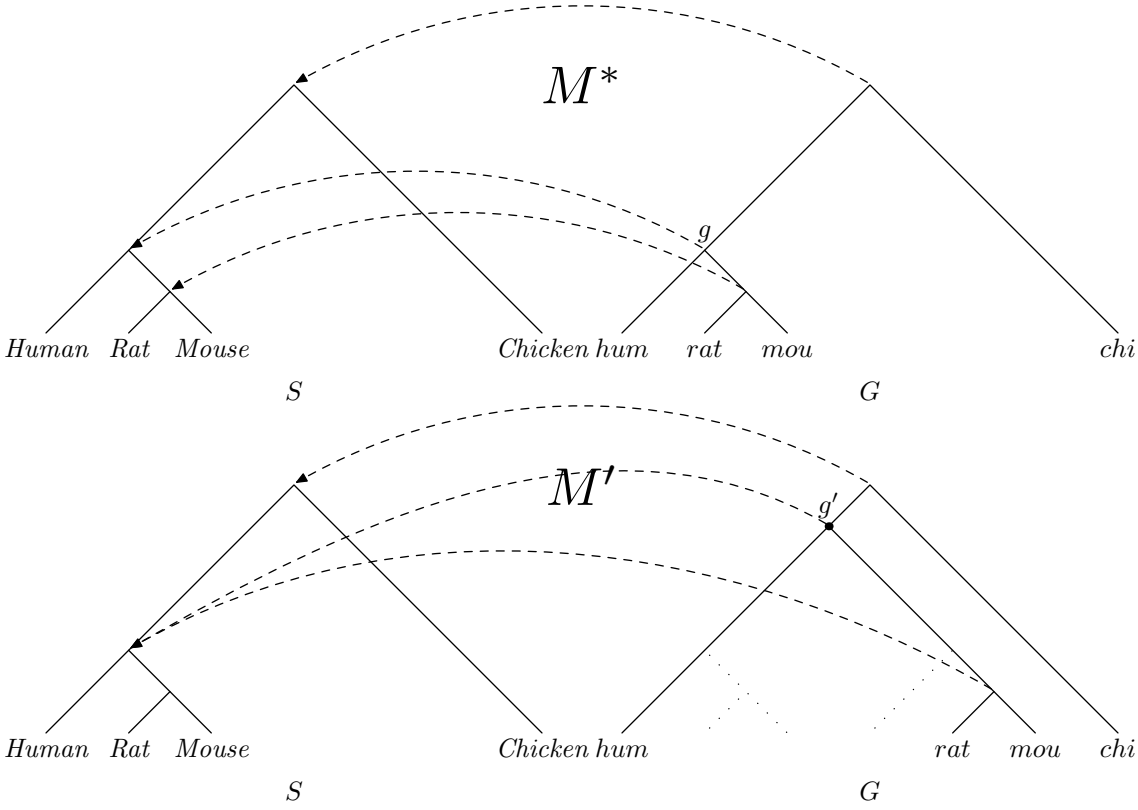

Figure 4.2: Example for illustrating species map. The top figure shows the parsimonious species map M^* between species tree S and gene tree G. Under this map, each gene evolved normally. No duplication or loss occurred. The bottom shows another species map M' for the two trees. Under M', at least one historical duplication (at node g') and two losses (showed in dashed branches) occurred in this gene family. Note that in this example, branch length has no meanings.

Given this fact, we introduce new notations:

$$\sigma'(g) = \{s_{g'} : g' \in \omega_G(g)\} \tag{4.2}$$

and

$$\sigma(g) = \sigma'(g) \cup \{s \in V_E(S) : \exists s' \in \sigma'(g),\ \mathrm{lca}(s, s') < M(g)\} \tag{4.3}$$

While $\sigma'(g)$ only consists of present species that can be observed in the tree, the second set in Equation 4.3 also includes lost species in $G|_g$ subtree. As a consequence, set $\sigma(g)$ consists of the species that should appear in $G|_g$ if no loss occurs in this subtree.

Although the definition of $\sigma(g)$ looks complicated, it is irreplaceable. This can be seen from the relation between the three sets $\sigma'(g)$, $\sigma(g)$ and $\omega_S(M(g))$: $\sigma'(g) \subset \sigma(g) \subset \omega_S(M(g))$ where $\sigma'(g) = \sigma(g)$ stands if no loss occurs, while $\sigma(g) = \omega_S(M(g))$ if $M(g)$ is a binary node. Figure 4.3 also gives a concrete example where the three sets are different. As we will see in the following text, $\sigma(g)$ plays a critical role on DLI.

4.2.1 Inferring duplications and orthologs

If $g \in V_E(G)$ is a duplication and no loss occurs in $G|_{g_1}$ and $G|_{g_2}$, $\mathrm{chi}(g) = \{g_1, g_2\}$, we would expect to see $\sigma'(g_1) = \sigma'(g_2)$. If losses occur, however, it is possible that $\sigma'(g_1) \cap \sigma'(g_2) = \emptyset$. In this case, $\sigma(g)$ will play its role due to its inclusion of the lost species. Formally, g is a *d*uplication (or duplication occurred at g) if $\sigma(g_1) \cap \sigma(g_2) \neq \emptyset$, where $\mathrm{chi}(g) = \{g_1, g_2\}$. Figure 4.4 shows an example.

Orthologs are genes in different species that originate from a single gene in the last common ancestor of these species (Remm *et al.*, 2001). Accordingly, in a gene tree, two present genes g_1 and g_2 are said to be *o*rthologs if $\mathrm{lca}(g_1, g_2)$ is not a duplication. This simple condition exactly capture the meaning of biological definition.

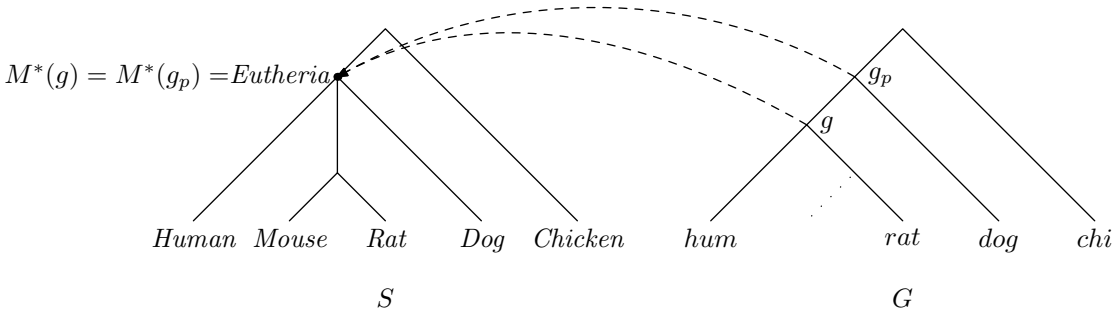

Figure 4.3: Example to explain $\sigma(g)$ set. In the left is a species tree S with a multifurcated node; in the right is a gene tree G with a loss. Node $g \in V(G)$ is mapped to $M^*(g)$ under the parsimonious map M^*, and g_p mapped to $M^*(g_p) = M^*(g)$. In this example, $\sigma'(g) = \{Human, Rat\}$, $\sigma(g) = \{Human, Mouse, Rat\}$ and $\omega_S(M^*(g)) = \{Human, Mouse, Rat, Dog\}$. They are all different from one another: $\sigma'(g)$ is smaller than $\sigma(g)$ because the mouse gene is lost, while $\omega_S(M^*(g))$ is bigger because $M^*(g)$ is a trifurcation.

4.2.2 Inferring losses

We next seek to find a set $\text{loss}(g) \subset V(S)$ that consists of species in which gene losses occur when $g_p \equiv \text{par}(g)$ evolved into g. Note that in this way, $\text{loss}(g)$ will actually be localized at the branch between g_p and g. Thus the calculation at one node will not interfere with the calculation at another one, and the total number of loss events in the tree G is simply the sum of $|\text{loss}(g)|$ of each node $g \in V(G)$.

As duplication and speciation represent different biological events, whether g_p is a duplication or not makes a little difference. If g_p is a duplication, subtree $G|_{g_p}$ and $G|_g$ should in theory consist of same species. When this is violated, losses must happen and $s \in \sigma(g_p) \setminus \sigma(g)$ covers all the species in which gene losses occur. If g_p is a speciation, besides the condition $s \in \sigma(g_p) \setminus \sigma(g)$, a lost s must appear only in $M(g_p)$'s subtree containing $M(g)$, or equivalently, must satisfy $\text{lca}(s, M(g)) < M(g_p)$. We denote by $\text{loss}'(g)$ the set of present species that are lost in the gene tree[2]:

$$\text{loss}'(g) = \begin{cases} \sigma(\text{par}(g)) \setminus \sigma(g) & \text{if par}(g) \text{ is a duplication} \\ \{s \in \sigma(\text{par}(g)) \setminus \sigma(g) : \text{lca}(s, M(g)) < M(\text{par}(g))\} & \text{otherwise} \end{cases} \tag{4.4}$$

Set $\text{loss}'(g)$ consists of present species that cannot be observed due to losses between g and $g_p = \text{par}(g)$. However, this is not good enough. A loss of ancestral species will cause the losses of all the descendant present species. Based on parsimonious rule, we

[2]Given two sets A and B, $A \setminus B = \{a : a \in A, \text{and } a \notin B\}$.

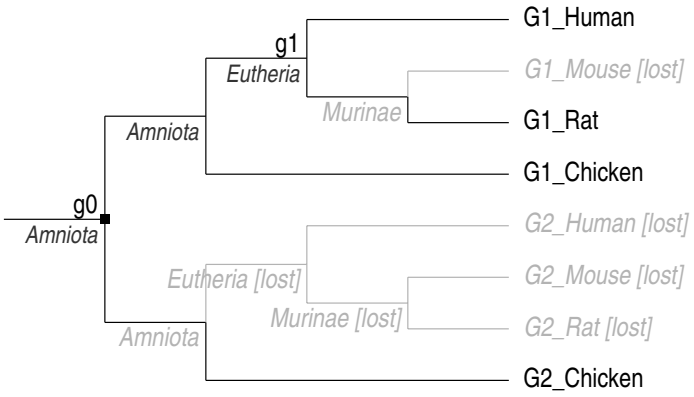

Figure 4.4: Example of duplication/loss inference. A slant name beside each internal node indicates the ancestral speices that contained the corresponding ancestral gene. Gray colour and slant leaves show the genes that cannot be observed nowadays due to loss events. In this gene tree, $\sigma(\mathsf{G1\_rat}) = \{Rat\}$, $\sigma(\mathsf{G2\_chicken}) = \{Chicken\}$, $\sigma(g_0) = \{Huamn, Mouse, Rat, Chick\}$ and $\sigma(g_1) = \{Human, Mouse, Rat\}$. Node g_1 is a speciation, and therefore $\text{loss}(\mathsf{G1\_rat}) = \text{loss}'(\mathsf{G1\_rat}) = \{Mouse\}$, while g_0 is a duplication and $\text{loss}'(\mathsf{G2\_chicken}) = \sigma(g_0) \setminus \sigma(\mathsf{G2\_chicken}) = \{Human, Mouse, Rat\}$ which makes $\text{loss}(\mathsf{G2\_chicken}) = \{Eutheria\}$.

would like to count this as one loss event. For this purpose, we define:

$$\text{loss}(g) = \{s \in V(S) : \omega_S(s) \subset \text{loss}'(g), \omega_S(\text{par}(s)) \not\subset \text{loss}'(g)\} \tag{4.5}$$

Set loss(g) avoids over-counting. It is actually the minimum set $A \subset V(S)$ satisfying $\omega_S(A) = \text{loss}'(g)$. Also take Figure 4.1 as an example. In this gene tree, loss$'$(ENSGALT00000011124_chicken) = $\{human, mouse, rat, dog\}$ and loss(ENSGALT00000011124_chicken) = $\{Eutheria\}$. *Eutheria* is just the species where the loss occured. Figure 4.4 gives another example.

4.3 Duplication Function and Loss Function

Both duplication function and loss function are defined on $V_I(G)$:

$$D_G(g) = \begin{cases} 1 & \text{if } g \text{ is a duplication} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

$$L_G(g) = \sum_{g' \in \text{chi}(g)} |\text{loss}(g')| \tag{4.7}$$

Duplication function $D_G(g)$ measures whether g is a duplication; loss function $L_G(g)$ counts losses when g evolved into its children. Obviously, the total number of duplications and losses in gene tree G simply equal to $\sum D_G(g)$ and $\sum L_G(g)$ respectively. Given a common species map M, the definitions of duplication and loss functions must depend on the topology of the gene tree G, but if parsimonious species map M^* is applied, duplication and loss functions can be topology-independent. This will be re-examined in Section 5.2 when set representation is introduced.

4.4 Towards Statistical Methods

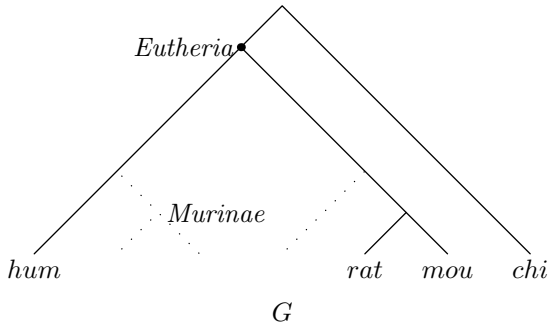

Figure 4.5: Example showing the failure of parsimonious species map. Assume the the true history of gene evolution included one duplication at *Eutheria* and two losses in *Murinae* and *Human*. The human gene and the mouse gene are not orthologs. Arbitrarily applying the parimonious species map M^* will always miss these events and wrongly infer the two genes as orthologs.

In this section, we will come back to the selection of species map M that uniquely determines how the duplication and loss are inferred. Usually we only take the parsimonious species map M^* due to its simplicity, but unfortunately nature does not always follow the parsimonious rule. In a multigene family where duplications and losses tend to occur frequently, arbitrarily choosing M^* might underestimate the numbers and lead to more orthologs (Figure 4.5). More sophisticated method that could possiblely select the species map other than M^* should be applied this time. A Bayesian method has been introduced by Arvestad *et al.* (Arvestad *et al.*, 2003; Arvestad *et al.*, 2004) and cleverly solve this problem. We do not intend to present detailed description, which is far beyond the scope of this thesis. We only add more remarks in comparison of parsimonious M^* and M favoured by Bayesian method.

In Bayesian framework, the desired species map is the one that maximize the posterior probability $\Pr\{M|G,S\}$, which can be calculated as:

$$\begin{aligned}
\Pr\{M|G,S\} &= \frac{\Pr\{M,G|S\}}{\Pr\{G|S\}} \\
&= \frac{1}{\Pr\{G|S\}} \cdot \int_{\mu}\int_{\lambda} \Pr\{M,G,\mu,\lambda|S\}\, d\mu\, d\lambda \\
&= \frac{1}{\Pr\{G|S\}} \cdot \iint \Pr\{M,G|S,\mu,\lambda\} \Pr\{\mu,\lambda|S\}\, d\mu\, d\lambda \\
&= \frac{1}{\Pr\{G|S\}} \cdot \iint \Pr\{M,G|S,\mu,\lambda\} p(\mu,\lambda|S)\, d\mu\, d\lambda
\end{aligned}$$

where λ is the birth rate, μ is the loss rate and $p(\mu,\lambda|S)$ is the prior probability of (μ,λ) given a species tree S. When no prior knowledge is available, we usually take bounded uniform distribution as what Arvestad *et al.* has done. In this equation, $\Pr\{G|S\}$ is a constant given a specified G. Then all we should do is to calculate $\Pr\{M,G|S,\mu,\lambda\}$. If we assume that when separated, whether by speciation or duplication, genes evolved independently, $\Pr\{M,G|S,\mu,\lambda\}$ can be written as the product of a series of independent factors[3]:

$$\Pr\{M,G|S,\mu,\lambda\} = \prod_{s\in V(S)} \prod_{g\in M^{-1}(\mathrm{par}(s))} q_G(g,s) \tag{4.8}$$

where $q_G(g,s)$ is the probability of forming the observed gene tree when a single gene g of $\mathrm{par}(s)$ evolved into its descendants of species s. Thus $\prod_{g\in M^{-1}(\mathrm{par}(s))} q_G(g,s)$ represents the probability of forming the observed gene tree when species $\mathrm{par}(s)$ evolved into s. As to the calculation of $q_G(g,s)$, Arvestad *et al.* presents detailed descriptions base on previous works (Nee *et al.*, 1994; Yang and Rannala, 1997).

Bayesian method establishes an elegant framework for tree reconciliation and also for various complex analyses about gene evolution, including reconstructing species trees and gene trees with the help of each other (Arvestad *et al.*, 2004). However, Bayesian method might be faced with a few theoretical difficulties both in biological angle and in mathematical angle. First of all, separated genes, either by duplication or speciation events, did not evolve independently. Losses tended to occur after massive

[3] $M^{-1} : V(S) \to 2^{V(G)}$ can be regarded ass the reverse of species map M.

duplications. One-copy genes were also less likely to be lost. These facts are not modeled in Bayesian framework at present. Furthermore, Bayesian method has to work with a prior distribution, which can improve the accuracy when correctly formed but may also induce bias when not carefully selected. Unfortunately, bounded uniform distribution of λ and μ is not appropriate. On one hand, from the known gene trees in TreeFam (Li *et al.*, 2006), birth rate λ and death rate μ are centered around small values; on the other hand, in order to deal with families with massive duplications and losses where parsimony fails, bound values must be large enough. This produces the confliction. Whereas the use of parsimonious map never produces false duplications when the gene tree is correct, the use of Bayesian one will, and the use of such a uniform distribution will further aggregate the problem because the higher probabilities at larger λ and μ tend to favour more duplications. False duplications, which might cause more problems than missing some, have to be evaluated before we apply Bayesian method in practice.

In most cases where μ and λ are small, parsimonious species map M^* is accurate enough. Although in a complex gene family where duplications and losses frequently occurred it can be wrong, use of Bayesian method does not give us more confidence when even the gene tree is questionable. Furthermore, under parsimonious map duplication and loss functions also become topological independent. These properties make M^* particularly useful in Chapter 5.

Chapter 5

Tree Merge

Phylogenetic trees are essential to various evolutionary studies. Due to their irreplaceable positions, many algorithms were developed to reconstruct phylogenetic trees, including distance-based methods, parsimony methods (Fitch, 1971), max-likelihood algorithms (Felsenstein, 1981) and the recent Bayesian methods (Rannala and Yang, 1996). Whereas these algorithms establish the fundament of the theory of tree reconstruction, they have to work with certain evolutionary models in practice such as JTT (Jones *et al.*, 1992), WAG (Whelan and Goldman, 2001), HKY (Hasegawa *et al.*, 1985), GY94 (Goldman and Yang, 1994), and many more. Given the variety of algorithms and models, is there an ultimate solution to tree reconstruction problems? Unfortunately, the answer is 'no'. One of the major causes is the evolutionary heterogeneity over time. Shifts in site-specific evolutionary rates, which is usually termed as 'heterotachy', frequently occur (Lopez *et al.*, 2002). A model best fit one lineage or time frame may lead to erroneous inference in another. Although this fact has already been noticed in several previous works (Weiss and von Haeseler, 2003; Susko *et al.*, 2002) and extensively studied in the case of few sequences (Kolaczkowski and Thornton, 2004; Gadagkar and Kumar, 2005; Spencer *et al.*, 2005; Philippe *et al.*, 2005), little was achieved in solving the problem practically. Evolutionists still have to build trees with various algorithms and models, and combine the results with prior knowledges and personal experiences. But how on earth do they assess different trees? Is there a way to achieve this automatically? This chapter aims to answer these two questions.

In order to understand how a tree can be evaluated by evolutionists, let's first see an example. Figure 5.1 shows two trees reconstructed by neighbour-joining, using synonymous distance (d_s) and non-synonymous distance (Nei and Gojobori, 1986; Li, 1993; Goldman and Yang, 1994; Yang and Nielsen, 2000)[1] (d_n), respectively. Even

[1]Nonsynonymous distance (d_n) counts the number of nucleotide changes that contribute to amino acid changes, while synonymous distance (d_s) counts the number that do not contribute to amino acid changes. Both kinds of distances can be only calculated in coding regions (CDS). In theory, the synonymous distance d_s tends to date the true evolutionary history more precisely. Although selection for codon bias might occur, d_s is generally less influenced by heterogeneous selective pressure in different lineages (Ayala, 1999; Baldauf, 2003). However, if the two sequences are separated by a long evolutionary time, multiple substitutions lead to saturation of d_s and make it impossible to estimate d_s accurately. As a consequence, the deep branches of the d_s tree are usually unreliable. Conversely, the nonsynonymous distance d_n is not saturated at long evolutionary time-frames and can more effectively resolve the deep branches of the tree, but if the protein sequences are too similar, too few non-synonymous changes will

without any additional information, we can tell that the deeper branches in d_n tree and lower branches in d_s tree are more likely to be correct. This judgement is based on three rules: (i) reliable branches tend to have higher bootstrapping support; (ii) synonymous trees are more effective between close sequences, while nonsynonymous tree are better at deeper branches; (iii) correct gene trees usually resemble the species tree. If we can

have accumulated to estimate d_n accurately. The lower branches of d_n tree tend to be less accurate.

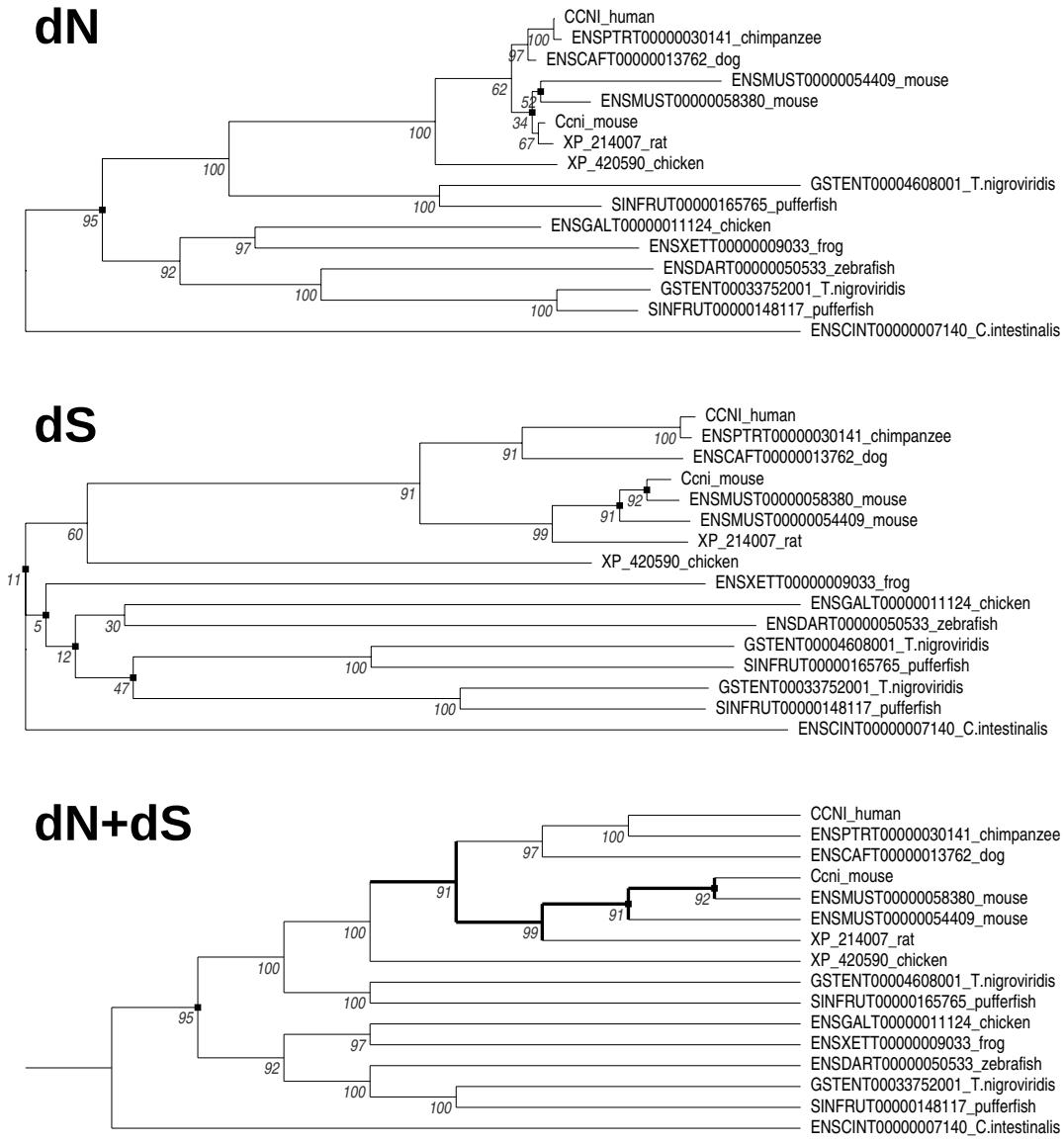

Figure 5.1: Example of tree merge. Non-synonymous tree **dN** was merged with synonymous tree **dS**, both reconstructed with neighbour-joining. Bold lines in the merged tree show the branches coming from **dS**, while the rest from **dN**. The resultant tree contains fewer losses and have higher bootstrapping support.

develop an algorithm that can account of these rules, it is possible to automatically make a better tree by merging several trees built with different methods. This is tree merge algorithm.

Tree merge is a process of choosing optimal children between branches shared by candidate trees. As we want to shape the algorithm in a precise and strict way, many abstract notations have to be used. And therefore, we start this chapter by introducing the third type of representation of trees, set representation. Then the tree merge problem is formally raised. After the description and proof of the algorithm, we analyze the time complexity of this algorithm. In the following chapter, the effectiveness of tree merge will be assessed together with a lot of single-model algorithms.

5.1 Set Representation of Trees

Graph representation is more intuitive, but it is not convenient in comparing rooted trees built from the same sequence set using different algorithms. In graph language, identical trees are judged by comparing the vertex sets and edge sets at the same time. But this is unnecessary. The edges or branches of a rooted tree are naturally determined by the subset of leaves that are descendant from the lower node of the edge; in turn, we can use a set of leaf subset to represent the topology of a rooted tree [2]. Thus, topological comparison can be reduced to set comparison, which is more concise and straightforward.

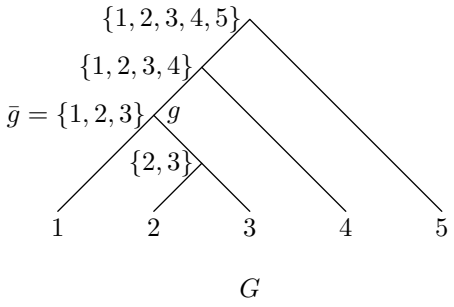

Figure 5.2: Example of set representation. Set $\omega_G(g)$ is labeled at each internal node. The set representation of G is $\bar{G} = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{2,3\}, \{1,2,3\}, \{1,2,3,4\}, \{1,2,3,4,5\}\}$. Conversely, when we get $\bar{G}$, we will also know the topology of G. Furthermore, if we let $\bar{g} = \{1,2,3\}$, $\bar{G}|_{\bar{g}} = \{\{1\}, \{2\}, \{3\}, \{2,3\}, \{1,2,3\}\}$ is the set representation of $G|_g$.

Let V be a leaf node set, and $\Sigma(V) = \{G \text{ is a rooted tree} : V_E(G) = V\}$ be the tree space in which each element is a rooted tree whose external node set is V. For any $G \in \Sigma(V)$, we can construct a set $\bar{G} \subset 2^V$ [3] using the ω_G map (Equation 1.1) as follows:

$$\bar{G} = \{\bar{g} \subset V : \bar{g} = \omega_G(g), g \in V(G)\} \tag{5.1}$$

[2] Similarly, the topology of an unrooted tree can be represented as a set of bipartitions of the leaf set

[3] Given a set V, $2^V = \{A : A \subset V\}$ is the set of all the subsets of V including V itself and empty set $\emptyset$. 2^V is also called the σ-set of V. Note that $\left|2^V\right| = 2^{|V|}$ always stands. This is why it is written as 2^V.

Obviously, $\bar{G_1} = \bar{G_2}$ only stands when the two trees G_1 and G_2 are topologically equivalent; when G_1 differs G_2, $\bar{G_1}$ and $\bar{G_2}$ cannot be the same. As a consequence, Equation 5.1 builds an one-to-one relation between a set $\bar{G}$ and a graph G, and therefore G can be represented by a set $\bar{G}$. This is the set representation of a rooted tree. When we know $\bar{G}$, we will always construct a tree G under graph representation. We also introduce:

$$\bar{G}|_{\bar{g}} = \{\bar{g}' : \bar{g}' \in \bar{G}, \bar{g}' \subset \bar{g}\} \tag{5.2}$$

Set $\bar{G}|_{\bar{g}}$ represents subtree $G|_g$. Figure 5.2 shows an example.

Throughout this chapter, notations like $\bar{G}$ and $\bar{g}$ are always under the set representation.

5.2 Set Forms of Duplication and Loss Functions

In Section 4.3, the duplication and loss functions are defined as (Equation 4.6):

$$D_G(g) = \begin{cases} 1 & \text{if } g \text{ is a duplication} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

$$L_G(g) = \sum_{g' \in \text{chi}(g)} |\text{loss}(g')| \tag{5.4}$$

The two functions defined in this way are dependent on the topology of the gene tree G. As a matter of fact, if the parsimonious species map M^* is applied to infer duplications and losses, the two functions can be defined in a way independent of G. Let's re-examine the deduction of duplication and loss under set representation.

In the following text, we always let $\bar{g}$, $\bar{g}_1$ and $\bar{g}_2$ be a subset of $V_E(G)$. From Equation 4.1 and 4.3, we define:

$$\sigma'(\bar{g}) = \{s_{g'} : g' \in \bar{g}\} \tag{5.5}$$

$$M^*(\bar{g}) = \text{lca}(\sigma'(g)) \tag{5.6}$$

$$\sigma(\bar{g}) = \sigma'(g) \cup \{s \in V_E(S) : \exists s' \in \sigma'(g),\ \text{lca}(s, s') < M^*(\bar{g})\} \tag{5.7}$$

Thus the parsimonious duplication function is:

$$D^*(\bar{g}_1, \bar{g}_2) = D^*(\bar{g}_2, \bar{g}_1) = \begin{cases} 1 & \text{if } \sigma(\bar{g}_1) \cap \sigma(\bar{g}_2) \neq \emptyset; \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

Parsimonious loss function can be defined in a similar way:

$$L^*(\bar{g}_1, \bar{g}_2) = L^*(\bar{g}_2, \bar{g}_1) = |\text{loss}^*(\bar{g}_1, \bar{g}_2)| + |\text{loss}^*(\bar{g}_2, \bar{g}_1)| \tag{5.9}$$

where

$$\text{loss}'^*(\bar{g}_1, \bar{g}_2) = \begin{cases} \sigma(\bar{g}_1 \cup \bar{g}_2) \setminus \sigma(\bar{g}_1) & \text{if } D^*(\bar{g}_1, \bar{g}_2) = 1 \\ \{s \in \sigma(\bar{g}_1 \cup \bar{g}_2) \setminus \sigma(\bar{g}_1) : \text{lca}(s, M^*(\bar{g}_1)) < M^*(\bar{g}_1 \cup \bar{g}_2)\} & \text{otherwise} \end{cases} \tag{5.10}$$

$$\text{loss}^*(\bar{g}_1, \bar{g}_2) = \{s \in V(S) : \omega_S(s) \subset \text{loss}'(\bar{g}_1, \bar{g}_2), \omega_S(\text{par}(s)) \not\subset \text{loss}'(\bar{g}_1, \bar{g}_2)\} \tag{5.11}$$

They correspond to Equation 4.4 and 4.5. Now, parsimonious duplication and loss functions can be calculated given two set of leaves. Such calculation is topology-independent in the sense that the topologies on the two sets will not affect the calculation process. This property is particularly useful in constructing the objective function for tree merge algorithm 5.3.2.

5.3 Tree Merge

5.3.1 Tree merge problem

Given a set of *binary rooted* trees $\Phi = \{\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n\}$, define a permitted branch set:

$$\mathcal{G}(\Phi) = \bigcup_{\bar{G}' \in \Phi} \bar{G}' \tag{5.12}$$

Let $\Omega(\mathcal{G})$ be the set of binary rooted tree $\bar{G}$ that satisfies $\bar{G} \subset \mathcal{G}$. Being a subset of $\Sigma(V)$, $\Omega(\mathcal{G})$ is the tree space spanned by $\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n$. It is also useful to define a permitted branch set $\mathcal{G}|_{\bar{g}}$ that is restricted to a node $\bar{g}$:

$$\mathcal{G}|_{\bar{g}} = \bigcup_{\bar{G}' \in \Phi} \bar{G}'|_{\bar{g}} \tag{5.13}$$

Then $\Omega(\mathcal{G}|_{\bar{g}})$ is the tree space spanned by $\bar{G}_1|_{\bar{g}}, \bar{G}_2|_{\bar{g}}, \ldots, \bar{G}_n|_{\bar{g}}$.

Let F be a function defined on $\Sigma(V)$. The *general tree merge problem* is to find a *binary rooted* tree $\bar{G} \in \Omega(\mathcal{G})$ that optimizes $F(\cdot)$. For common F, the only solution is to enumerate every tree in Ω and then search for the optimum one. But for a class of special objective functions, the exhaustive search can be replaced by a simple precise algorithm. In this thesis, we only consider functions that take the form:

$$F(\bar{G}) = \sum_{\bar{g} \in \bar{G}} f(\mathrm{chi}(\bar{g})) = \sum_{\bar{g} \in \bar{G}} f(\bar{g}_1, \bar{g}_2) \tag{5.14}$$

Such an $F(\bar{G})$ has two critical properties. First, it is additive, and second, function f is defined on $2^V \times 2^V$, which means that the calculation of $f(\mathrm{chi}(\bar{g}))$ is only dependent on the elements in $\bar{g}$'s children set but independent of the topologies of $\bar{g}$'s children. As we will see in the following section, these two properties, especially the second one, guarantee that the optimum tree can be found in $O(|\Phi| \cdot |V|^2)$ time. The (special) *tree merge problem* is to find the optimal *binary rooted* tree in $\Omega(\mathcal{G})$ when $F(\bar{G})$ follows Equation 5.14.

5.3.2 Constructing objective functions

According to Equation 5.14, the effectiveness of tree merge algorithm totally relies on function f. Before detailed algorithm is described, we should make sure that f is present and biologically reasonable.

Based on the discussion in Section 5.2, f can be defined as:

$$f(\bar{g}_1, \bar{g}_2) = f(\bar{g}_2, \bar{g}_1) = \alpha D^*(\bar{g}_1, \bar{g}_2) + \beta L^*(\bar{g}_1, \bar{g}_2) - \gamma B^*(\bar{g}_1, \bar{g}_2) \tag{5.15}$$

where $\alpha, \beta, \gamma > 0$, $D^*(\bar{g}_1, \bar{g}_2)$ and $L^*(\bar{g}_1, \bar{g}_2)$ are the duplication and losses functions, respectively, and $B^*(\bar{g}_1, \bar{g}_2)$ denotes the highest bootstrapping supports among all trees containing $\{\bar{g}_1, \bar{g}_2\}$. Note that the star in in D^* and L^* indicates the parsimonious species map M^* is used in DLI inference. These two functions measure the similarity between a gene tree and the species tree. The smaller, the better. As to $B^*(\bar{g}_1, \bar{g}_2)$, another possible definition is also available, which will be discussed in the end of this chapter. In addition, if we know that certain algorithm is good at particular branches, we can deliberately raise the bootstrapping supports. Consequently, all the three rules addressed in the introduction section of this chapter have been considered.

5.3.3 Tree merge algorithm

The basic idea of tree merge algorithm resembles that of dynamic programming. It reduces exhaustive search to the calculation of a function $F^*(\bar{g})$ for each $\bar{g} \in \mathcal{G}$. F^* is defined as:

$$F^*(\bar{g}) = \min_{\bar{G}|_{\bar{g}} \in \Omega(\mathcal{G}|_{\bar{g}})} F(\bar{G}|_g) \tag{5.16}$$

Obviously, $F^*(V) = \min F(\bar{G})$ always stands. Thus searching optimal tree $\bar{G}$ is equivalent to calculating $F^*(V)$. When $F(\cdot)$ follows Equation 5.14, $F^*(V)$ can be calculated recursively. To achieve this goal, we need to construct the set of permitted children pairs given a node $\bar{g}$:

$$\mathcal{C}(\bar{g}) = \{\{\bar{g}_1, \bar{g}_2\} : \bar{g}_1, \bar{g}_2 \in \mathcal{G},\ \bar{g}_1 \cap \bar{g}_2 = \emptyset \text{ and } \bar{g}_1 \cup \bar{g}_2 = \bar{g}\} \tag{5.17}$$

Set $\mathcal{C}(\bar{g})$ consists of $\bar{g}$'s candidate children set. That is, each element $\{\bar{g}_1, \bar{g}_2\} \in \mathcal{C}(\bar{g})$ can be $\bar{g}$'s children set. Then,

$$\begin{aligned}
& F^*(\bar{g}) \\
= & \min_{\bar{G}|_{\bar{g}} \in \Omega(\mathcal{G}|_{\bar{g}})} F(\bar{G}|_g) \\
= & \min_{\bar{G}|_{\bar{g}} \in \Omega(\mathcal{G}|_{\bar{g}})} \sum_{\bar{g}' \in \bar{G}|_{\bar{g}}} f(\mathrm{chi}(\bar{g}')) \\
= & \min_{\{\bar{g}_1, \bar{g}_2\} \in \mathcal{C}(\bar{g})} \left\{ \min_{\bar{G}|_{\bar{g}_1} \in \Omega(\mathcal{G}|_{\bar{g}_1})} \sum_{\bar{g}'_1 \in \bar{G}|_{\bar{g}_1}} f(\mathrm{chi}(\bar{g}'_1)) + f(\bar{g}_1, \bar{g}_2) + \min_{\bar{G}|_{\bar{g}_2} \in \Omega(\mathcal{G}|_{\bar{g}_2})} \sum_{\bar{g}'_2 \in \bar{G}|_{\bar{g}_2}} f(\mathrm{chi}(\bar{g}'_2)) \right\}
\end{aligned}$$

From Equation 5.16 and 5.14, we know that:

$$F^*(\bar{g}) = \min_{\{\bar{g}_1, \bar{g}_2\} \in \mathcal{C}(\bar{g})} \{F^*(\bar{g}_1) + f(\bar{g}_1, \bar{g}_2) + F^*(\bar{g}_2)\} \tag{5.18}$$

Thus $F^*(V)$ can be calculated recursively.

The two equations, 5.17 and 5.18, established the basis of tree merge algorithm. Table 5.1 presents more details. In practical implementation, hash technique should be applied to judge whether $\bar{g}$ has been inserted to $\mathcal{G}$. If the hash function is perfect, only $O(|V|)$ time is needed to locate and insert a set to $\mathcal{G}$. This will greatly accelerate the algorithm.

Hash technique makes tree merge very efficient. In procedure ConstructG, all the hash values can be calculated in $O(n \cdot |V|(|V|-1))$. The following $n \cdot (|V|-1)$ times of set comparisons and insertions take $O(n \cdot |V|^2)$ time at most. Procedure OptimizeF traverses all possible $\bar{g} \in \mathcal{G}$ and $\bar{g}$'s bipartition $\bar{g} = \bar{g}_1 \cup \bar{g}_2$. The time complexity in this part is also $O(n \cdot |V|^2)$. Both BuildTree and TreeMerge can be achieved in $O(|V|)$. Consequently, the time complexity of tree merge is $O(n \cdot |V|^2)$.

Figure 5.3 gives an example of tree merge process. In this example, $\mathcal{G} = \{\{hum\}, \{mou_1\}, \{mou_2\}, \{rat\}, \{chi\}, \bar{g}_1^c, \bar{g}_2^c, \bar{g}_3, \bar{g}_4, \bar{g}_5, \bar{g}_6\}$, $\mathcal{C}(\bar{g}_1^c) = \{\{\bar{g}_3, \{mou_2\}\}, \{\{rat\}, \bar{g}_5\}\}$ and $\mathcal{C}(\bar{g}_2^c) = \{\{\bar{g}_4, \{chi\}\}, \{\bar{g}_1^c, \bar{g}_6\}\}$. Only $\mathcal{C}(\bar{g}_1^c)$ and $\mathcal{C}(\bar{g}_2^c)$ consist of more than one node pairs. Other $\mathcal{C}(\cdot)$ just has one element. If we let $\alpha = \beta = 1$ and $\gamma = 0$ in Euqation 5.15, we know that $f(\bar{g}_3, \{mou_2\}) = 1 + 1 = 2$, $f(\{rat\}, \bar{g}_5) = 0$, $F^*(\bar{g}_5) = 1$ and $F^*(\bar{g}_3) = 0$. Thus $F^*(\bar{g}_1^c) = \min\{0 + 2 + 0, 0 + 0 + 1\} = 1$. The children pair $\{\{rat\}, \bar{g}_5\}$ is preferable. Similarly, as $f(\bar{g}_4, \{chi\}) = 0$, $f(\bar{g}_1^c, \bar{g}_6) = 3$, $F^*(\bar{g}_4) = 1 + 0 + 0 = 1$ and $F^*(\bar{g}_6) = 1$, $F^*(\bar{g}_2^c) = \min\{1+0+0, 1+3+1\} = 1$. The children pair $\{\bar{g}_4, \{chi\}\}$ is preferred. Consequently, $\bar{G}^* = \{\{hum\}, \{mou_1\}, \{mou_2\}, \{rat\}, \{chi\}, \bar{g}_1^c, \bar{g}_2^c, \bar{g}_4\}$ gives the final tree and $F(G^*) = F^*(\bar{g}_2^c) = 1$.

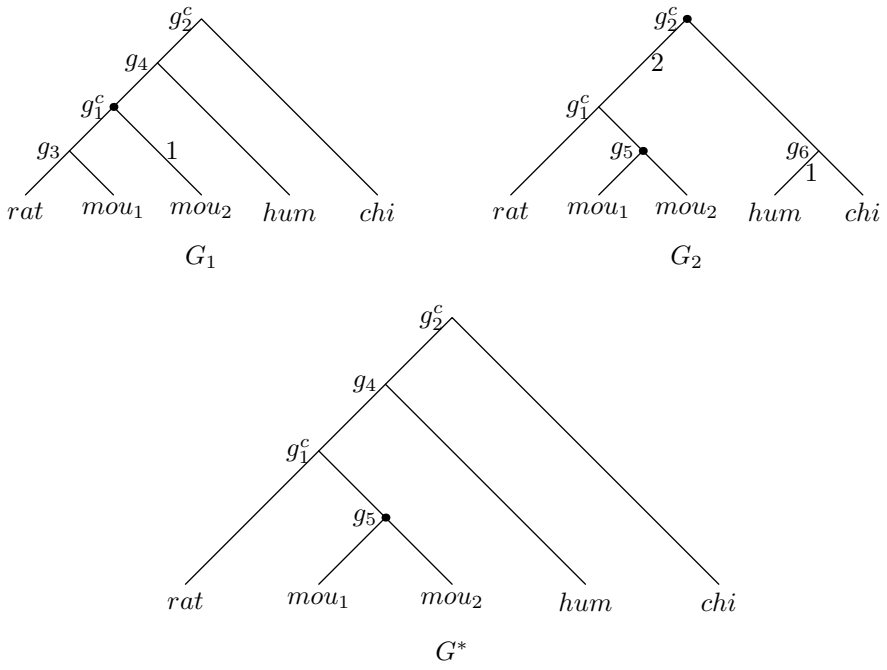

Figure 5.3: Example of tree merge algorithm. Tree G_1 and G_2 are merged into G^*. Bold nodes show the duplications and the number beside a branch is the number of losses occuring at that branch. For example, $|\text{loss}(\{mou_1\}, \bar{g}_3)| = |\{Rat\}| = 1$ and $|\text{loss}(\bar{g}_1^c, \bar{g}_6)| = |\{Human, Chicken\}| = 2$.

5.4 Discussions

Tree merge is able to capture the thinking of evolutionists. Although not a tree-building algorithm by itself, tree merge can combine the advantages of different tree-building algorithms and models, and automatically choose suitable methods in different lineages. It is most useful when candidate trees can compensate for the deficiency of one another. However, just as evolutionists cannot reconstruct correct trees without reasonable candidate trees, tree merge is less useful or even wrong if artefact candidates are provided. Sometimes a human being can detect the artefacts and make a tree outside Ω tree space, but tree merge cannot achieve. This is the theoretical limitation. Furthermore, the performance of tree merge is determined by the objective function. An evolutionary history that breaks the optimization process will fail tree merge, either. When using tree merge algorithm, we should pay attention to these points.

Tree merge algorithm can also be considered as an alternative algorithm to make a consensus tree (Margush and Mcmorris, 1981) from a set of resampled trees. All we need is to modify the definition of $B^*(\bar{g}_1, \bar{g}_2)$ in Equation 5.15. Assume we have n resampled binary rooted gene trees $\Phi = \{\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n\}$. Given $\bar{g}_1, \bar{g}_2 \in \mathcal{G}(\Phi)$, we can define $B^*(\bar{g}_1, \bar{g}_2)$ as the number of times when $\{\bar{g}_1 \cup \bar{g}_2\} \subset \bar{G}_i$, $i = 1, \ldots, n$[4]. In this way, $B^*(\bar{g}_1, \bar{g}_2)$ is the bootstrapping supports for a branch $\{\bar{g}_1, \bar{g}_2\}$. And if we set $\alpha = \beta = 0$, tree merge process is actually the same as that of making a consensus tree from a set of binary rooted trees. In conclusion, when B^* is defined as above, tree merge extends (rooted) consensus tree algorithms by incorporating species evolution, and may help to improve the quality of the consensus tree.

Tree merge is designed to work with a binary rooted gene tree where the phylogeny of species is clear. How to root a tree was discussed in 3. In principle, it is also possible to extend the algorithm in case of unrooted trees, but most of concepts in this chapter should be modified accordingly. Resolving a multifurcated tree is also tactable. Durand *et al.* (Durand *et al.*, 2006) implies an algorithm that may be adapted to resolve polytomies by minimizing duplications and losses. In contrast, the requirement of species phylogeny is critical. To our experience, no other criterion that satisfies Equation 5.14 can be as effective as this standard.

[4] B^* can also be defined in other similar, but different, ways.

Function:
TreeMerge$(V, \bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n)$
Input:
n gene trees $\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n \in \Sigma(V)$;
Output:
Tree $\bar{G}^*$ by merging $\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n$
Procedures:

```
G ← ∅
C ← ∅
ConstructG(Ḡ_1, ..., Ḡ_n)
    for i = 1 to n do
        for each ḡ ∈ Ḡ_i do
            G ← G ∪ {ḡ}
            C(ḡ) ← C(ḡ) ∪ {chi(ḡ)}

selected ← ∅
OptimizeF(ḡ)
    if F*(ḡ) has been calculated then
        return F*(ḡ)
    min ← ∞
    for each {ḡ_1, ḡ_2} ∈ C(ḡ) do
        score ← OptimizeF(ḡ_1) + f(ḡ_1, ḡ_2) + OptimizeF(ḡ_2)
        if min > score then
            min ← score
            minpair ← {ḡ_1, ḡ_2}
    F*(ḡ) ← min
    selected(ḡ) ← minpair

BuildTree(ḡ)
    if selected(ḡ) = ∅ then
        return {ḡ}
    {ḡ_1, ḡ_2} ← selected(ḡ)
    return BuildTree(ḡ_1) ∪ {ḡ} ∪ BuildTree(ḡ_2)

TreeMerge(V, Ḡ_1, Ḡ_2, ..., Ḡ_n)
    ConstructG(Ḡ_1, ..., Ḡ_n)
    for each v ∈ V do
        F*({v}) ← 0
    OptimizeF(V)
    return BuildTree(V)
```

Table 5.1: Tree merge algorithm. Procedure ConstructG constructs $\mathcal{G}$ and $\mathcal{C}(\bar{g})$; OptimizeF recursively calculates $F^*(\bar{g})$ and stores optimum children in *selected*, from which BuildTree builds the merge tree. BuildTree generates a tree in set presentation. It can also be easily adapted to generate a tree $G^* = (V(G^*), E(G^*))$ in graph presentation.

Chapter 6

Evaluation of Tree Building Methods

Whereas the variety of tree-building algorithms give evolutionists the opportunities to choose the one that best fits the data, the disagreements between these algorithms also confuse researchers at times. Given a set of sequences, it is not always clear what method is most effective on a certain condition. In case of one or a few sets of sequences, it is possible to manually curate the resultant trees. But in the case of large-scale analysis for hundreds of gene families, such a process is formidable. Even if massive manual curation can be achieved in the long run, which is major goal of TreeFam (Li *et al.*, 2006), guild-lines on successful tree-building are definitely beneficial to reduce manual work and meanwhile avoid artifacts in curation.

A number of studies have been done to investigate the performance of these algorithms. Based on the characteristics of the data sets in use, these studies can be classified into three types: theoretical analysis on 4- or 5-leaf trees (Gaut and Lewis, 1995; Huelsenbeck, 1995; Yang, 1996), computational simulation on large data sets (Kuhner and Felsenstein, 1994; Hall, 2005; Hollich *et al.*, 2005), and real data from experimental manipulations (Hillis *et al.*, 1992; Hillis *et al.*, 1994; Bull *et al.*, 1997). Although these studies did provide invaluable information on evaluating different algorithms, they may still suffer a number of problems respectively: theoretical analysis can only be applied to very small trees; computational simulation has to work with proposed evolutionary models or processes; experiment-manipulated data are relatively small and usually developed on a special condition that may deviate from real evolutionary processes. The most realistic evaluation is expected to be done with large-scale data that are formed in true evolutionary history, which, unfortunately, is unknown to us. Is there a way out?

In this chapter, we try to present a novel evaluation on various tree-building algorithms. It differs from previous studies on two aspects: the use of large-scale curated real data from TreeFam, and the use of model-independent indication: duplications and losses tend to be minimized. Limited to the potential curation errors in real data and the possibility that true evolution might favour more duplications and losses in some cases, it is still too early to make a final conclusion about what algorithm is superior. We only intend to show some data that will help evolutionists to choose the most appropriate algorithms in practical use.

6.1 Evaluated Algorithms and Evolutionary models

Three classical algorithms, distance-based method, parsimony (Fitch, 1971) and ML (Maximum Likelihood) (Felsenstein, 1981) method, were tested at both nucleotide level and amino acid level. Two types of merged trees were also included. Table 6.1 shows the 17 types of trees used in this benchmark. More details will be explained as follows.

Distance-based method is actually the name of a group of algorithms including Fitch-Margoliash (Fitch and Margoliash, 1967), ME (Minimum Evolution) (Rzhetsky and Nei, 1993), UPGMA (Sokal and Michener, 1958), NJ (Neighbour-Joining) (Saitou and Nei, 1987) and several alternatives to standard NJ algorithms. In this benchmark, only standard NJ and ME algorithms were tested in consideration of their popularity and accuracy. NJ tree was built by NJTREE and ME tree by FastME in GME framework (Desper and Gascuel, 2004). Distance matrix was calculated by TREE-PUZZLE (Schmidt *et al.*, 2002). For nucleotide alignment, HKY (Hasegawa *et al.*, 1985) model was applied with transition/transversion ratio[1] estimated from data; for amino acids alignment, WAG (Whelan and Goldman, 2001) was used. In both cases, we fixed the shape factor $\alpha = 1.0$ of Γ distribution which was approximated by 4 discrete substitution rate categories (Yang, 1994). Alignment gaps were regarded as missing data[2] in TREE-PUZZLE. As TreeFam-1.x trees were built from NJ with p-distance[3], a distance without correction for in-site multiple substitutions[4], this simplest model was also tested this time.

Parsimonious trees were reconstructed by **dnapars** and **protpars**, respectively. Both programs are included in PHYLIP package (Felsenstein, 1989). Tree merge was applied if several optimal trees were given by the programs. In our test, **protpars** only presents binary trees, but **dnapars** may build multifurcated trees. As both duplication/loss inference and tree merge algorithm only work with binary trees, some trees built by `dnapars` could not be processed.

PHYML (Guindon and Gascuel, 2003) built ML trees. Parameters of evolutionary models are identical to the ones used in TREE-PUZZLE. That is, HKY for nucleotide and WAG for amino acids; substitution rates were divided into 4 discrete categories that approximated a Γ distribution with $\alpha = 1.0$. Two or one categories were also evaluated to confirm the role of Γ distribution. It seems that PHYML and TREE-PUZZLE implemented models in slightly different ways. This can be observed when they were used to optimize branch length given a specified tree. When we assume substitutions occur uniformly across sites (one category), PHYML and TREE-PUZZLE can always

[1] Transitions denote the nucleotide changes between purines (A↔G) or between pyrimidines (C↔T), while other types of nucleotide changes are all transversions. Biologically, transitions occur more frequently than transversions, and therefore they are modeled differently.

[2] Likelihood methods are capable of treating gaps as missing data while calculating the distances. This is more robust.

[3] P-distance is also called mismatch distance. It actually equals to the percent mismatches in the matched regions of two sequences.

[4] In-site multiple substitutions denote multiple substitutions occurring at the same site. For example, at a certain site nucleotide Adenine (A) was changed into nucleotide Guanine (G) 100 million years ago, but 50 million years ago, the Guanine (G) was changed back to Adenine (A). Then two substitutions should be counted even if no substitution is observed nowadays. A sophisticated evolutionary model can estimate how often this is the case. But when sequences are too diverged, the variance of such estimation will be very large, and therefore less accurate.

achieve almost the same results. However, when more than one categories are applied, their results differ more or less. But as PHYML is unable to give a distance matrix while TREE-PUZZLE cannot make standard ML trees, this minor inconsistency has to be tolerated.

Two trees generated by tree merge algorithm were also assessed. The first one was built by merging a synonymous tree and a non-synonymous tree; the second by merging two ML trees and also synonymous and non-synonymous trees. As to the objective function f (Equation 5.15), we set $\alpha = 100000$, $\beta = 1000$ and $\gamma = 1$. Bootstrapping values B was scaled between 0 and 100. This setting always prefers less duplications, and less losses only if duplications are the same. Bootstrap only works when both D^* and L^* are identical. Due to efficiency issue, bootstrapping was not applied to ML trees. We arbitrarily set bootstrapping support as 80 at each internal branches (branches that do not connect leaves) given the fact that ML trees tend to be more reliable (Kuhner and Felsenstein, 1994). Here, bootstrapping values are least favoured because a single tree-building algorithm might also have high bootstrapping supports at wrong branches. Duplications are better than losses because in evolution, duplications occur more rarely while losses more frequently, especially following a duplication event.

Name	Method
`CUR`	Curated trees from TreeFam
`NJ-NT-HKY4`	Neighbour-Joining, HKY model, $C = 4$, $\alpha = 1.0$, κ estimated
`NJ-AA-WAG4`	Neighbour-Joining, WAG model, $C = 4$, $\alpha = 1.0$
`NJ-NT-dN`	NJ, non-synonymous distance, no correction for multi-substitutions
`NJ-NT-dS`	NJ, synonymous distance, no correction for multi-substitutions
`NJ-AA-MM`	NJ, p-distance (or mismatch) distance, no correction
`NJ-AA-Kmr`	NJ, with Kimura correction
`ME-NT-HKY4`	FastME, HKY model, $C = 4$, $\alpha = 1.0$, κ estimated
`ME-AA-WAG4`	FastME, WAG model, $C = 4$, $\alpha = 1.0$
`PAR-NT`	Parsimony (dnapars)
`PAR-AA`	Parsimony (protpars)
`ML-NT-HKY4`	PHYML, HKY, $C = 4$, $\alpha = 1.0$, κ estimated
`ML-NT-HKY2`	PHYML, HKY, $C = 2$, $\alpha = 1.0$, κ estimated
`ML-NT-HKY1`	PHYML, HKY, $C = 1$, $\alpha = 1.0$, κ estimated
`ML-AA-WAG4`	PHYML, WAG, $C = 4$, $\alpha = 1.0$
`MG-NT-dM`	Tree merge from `NJ-NT-dS` and `NJ-NT-dN`
`MERGE`	Tree merge from `ML-NT-HKY4`, `ML-AA-WAG4`, `NJ-NT-dN` and `NJ-NT-dS`

Table 6.1: Evaluated algorithms and models. Parameter κ is transversion/transition ratio, α is the shape parameter of Γ distribution which is approximated by C discrete substitution rate categories.

6.2 Construction of Test Sets

Two test sets, TestSet1 and TestSet2, were constructed in favour of curated TreeFam families. TestSet1 consists of 1041 trees curated in TreeFam-1. These trees were initially built by `NJ-AA-MM`. They contain genes from 12 sequenced species: `ARATH`, `SCHPO`, `YEAST`, `HUMAN`, `MOUSE`, `RAT`, `CHICK`, `BRARE`, `FUGRU`, `DROME`, `CAEBR` and `CAEEL`. The 113 trees in TestSet2 were curated based on TreeFam-2 where seven more species were added: `PANTR`, `CANFA`, `XENTR`, `TETNG`, `CIOIN`, `ANOGA` and `APIME`. They were curated from `MERGE` trees. Besides the difference in respects of number of species and automatic methods, trees in TestSet2 were deliberately selected such that they were neither too simple nor too complex. In contrast, TestSet1 contains some trees that are hard to curate, and also a lot of simple trees added at the early stage of TreeFam. This fact can be seen from Figure 6.1. Generally, TestSet2 is of higher quality.

For both test sets, original protein multialignments were directly retrieved from TreeFam. They were converted to nucleotide alignments by matching each codon to the corresponding amino acid. Sequences from unsequenced species were discarded. NJTREE were used to truncate poorly aligned regions and to mask low-scoring segments. Thompson *et al.* (Thompson *et al.*, 1997) presented more details.

6.3 Measuring the Quality of Trees

Two criteria were used to evaluate the quality of automatic trees. First, automatic trees were compared with curated trees. Topological differences between them were measured by topological distance dT (Robinson and Foulds, 1981), which counts the number of branches that exist in one tree but not in the other (Kuhner and Felsenstein, 1994). Topological distance dT can be calculated given unrooted multifurcated trees. It equals to zero if two trees are identical; it reaches its highest value $2n - 6$, where n is the number of leaves in either tree, if two trees are completely different. Based on dT, the

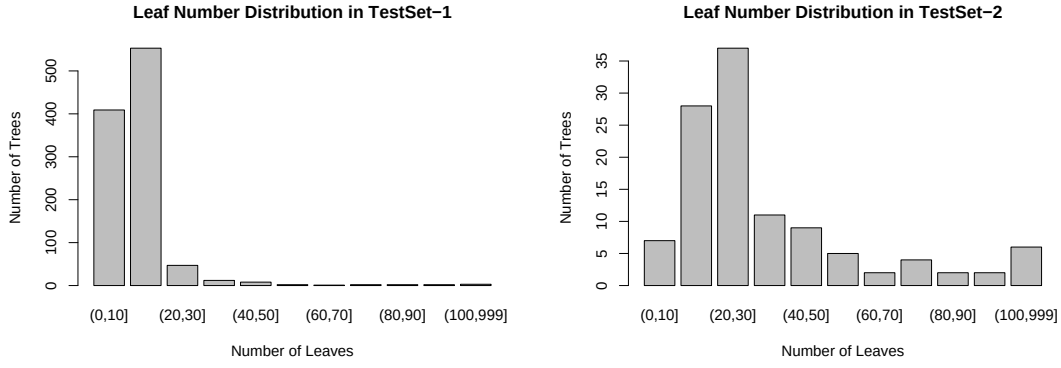

Figure 6.1: Distribution of number of leaves in TestSet1 and TestSet2.

distance between two methods M_1 and M_2 can be defined given m multialignments:

$$d_M(M_1, M_2) = \frac{\sum_{i=1}^{m} dT_i(M_1, M_2)}{\sum_{i=1}^{m} 2n_i - 6} \tag{6.1}$$

where $dT_i(M_1, M_2)$ is the topological distance between two trees reconstructed by the two methods based on i-th multialignment, and n_i is the number of sequences in the alignment. Rescaled between 0 and 1, method distance d_M is comparable between different test sets. It, in fact, represents the percentage of branches that can only be reconstructed by one method. In this thesis, differences between manual curation and automatic algorithms are also measured by d_M.

Curated trees were processed from automatic methods. Their accuracy is inevitablely affected by automatic methods in use. In addition, curation is a subjective process more or less, which might also make curated trees deviate from the true history. To objectively measure the quality of automatic methods, we also use percent duplications p_D and average losses p_L which are defined as:

$$p_D(M) = \frac{\sum_{i=1}^{m} D_i^*(M)}{\sum_{i=1}^{m} n_i - 1} \tag{6.2}$$

$$p_L(M) = \frac{\sum_{i=1}^{m} L_i^*(M)}{\sum_{i=1}^{m} n_i - 1} \tag{6.3}$$

where $D_i^*(M)$ is the number of duplications inferred from the tree reconstructed by method M from i-th alignment, and $L_i^*(M)$ is the number of losses. If we accept the hypothesis that true phylogenies tend to contain fewer duplications and losses, better methods must correspond to smaller p_D and p_L. Although this hypothesis might be violated in a few gene families, the overall trend should still stand.

The two test sets only represent a small part of all Eukaryotic gene families. To investigate the statistical reliability of our results, we designed a bootstrapping procedure to calculate the standard deviation of each statistics, p_D, p_L and d_M. In TestSet2, 113 families were resampled with replacement for B times, which forms B sets of families with each set containing 113 families. Criterion r_i was calculated from 113 resampled families for i-th round of resampling. Mean value μ and SD (Standard Deviation) σ were then estimated as:

$$\mu_r = \frac{\sum_i r_i}{B}$$

$$\sigma_r = \sqrt{\frac{\sum_i r_i^2 - B\mu_r^2}{B-1}}$$

A very small σ_r means that a r estimated from the original 113 families is similar to the r estimated from all animal families on the condition that these 113 families can represent all animal families. In most cases, μ_r is almost identical to r estimated from the original data, and therefore μ_r will not be showed. Such calculation can also be applied to TestSet1. They are not showed as σ_r on TestSet1 are similar to those on TestSet2.

6.4 Accuracy of Tree Building Algorithms

Table 6.2 shows the performance of tree-building algorithms and models. Generally, the small SD reveals that each criterion is quite stable. Due to the bias between TestSet2 and TestSet1, percent duplication p_D significantly differs between the two sets. This, fortunately, does not affect the evaluation of these algorithms. Between the two test sets, the Pearson's correlation coefficient of p_D is 0.965, and of p_L is 0.970, which means that the results from the two sets highly agree with each other.

Judged from the topological distance d_M to the curated tree `CUR`, tree `MERGE` outperforms all the other methods on both test sets. On TestSet2, this may be due to the fact that tree `CUR` was curated from `MERGE`, but such an interpretation cannot account for the good performance on TestSet1 where `CUR` was curated from `NJ-AA-MM` instead of `MERGE`. The similarity between `CUR` and `MERGE` on TestSet1 manifests that human experts also favoured `MERGE` even if they started from a less accurate tree. Tree merge algorithm successfully captures the thinking of a biologist.

Judged from percent duplication p_D and average loss p_L, `MERGE` is still the winner. The number of duplications and losses inferred from this tree is significantly less. The role of tree merge algorithm is also evident from `NJ-NT-dM`. Merged from `NJ-NT-dN` and `NJ-NT-dS`, two less accurate trees, this tree is even as accurate as those built by ML

	TestSet2						TestSet1		
Type	d_M	σ	p_D	σ	p_L	σ	d_M	p_D	p_L
`CUR`	0.000	0.000	0.249	0.017	0.388	0.029	0.000	0.175	0.494
`NJ-NT-HKY4`	0.225	0.017	0.313	0.017	0.786	0.042	0.133	0.204	0.630
`NJ-AA-WAG4`	0.256	0.016	0.322	0.016	0.901	0.050	0.154	0.220	0.756
`NJ-NT-dN`	0.201	0.011	0.308	0.015	0.796	0.040	0.135	0.211	0.714
`NJ-NT-dS`	0.527	0.017	0.431	0.014	1.761	0.041	0.464	0.345	1.561
`NJ-AA-MM`	0.225	0.012	0.308	0.016	0.823	0.042	0.137	0.213	0.740
`NJ-AA-Kmr`	0.275	0.018	0.333	0.017	0.988	0.065	0.165	0.226	0.790
`ME-NT-HKY4`	0.200	0.015	0.307	0.017	0.720	0.035	0.131	0.202	0.620
`ME-AA-WAG4`	0.232	0.014	0.317	0.017	0.858	0.047	0.151	0.217	0.744
`PARS-NT`	0.203	0.013	-	-	-	-	0.151	-	-
`PARS-AA`	0.181	0.013	0.286	0.016	0.614	0.035	0.150	0.207	0.684
`ML-NT-HKY4`	0.152	0.017	0.300	0.016	0.676	0.041	0.145	0.206	0.636
`ML-NT-HKY2`	0.147	0.018	0.301	0.017	0.677	0.041	0.143	0.206	0.639
`ML-NT-HKY1`	0.172	0.017	0.306	0.016	0.693	0.038	0.143	0.208	0.647
`ML-AA-WAG4`	0.185	0.015	0.299	0.016	0.705	0.044	0.159	0.213	0.709
`NJ-NT-dM`	0.165	0.011	0.291	0.016	0.687	0.037	0.121	0.195	0.622
`MERGE`	0.092	0.015	0.259	0.016	0.457	0.033	0.111	0.178	0.503

Table 6.2: Performance of tree builders. In this table, d is the topological distance between tree `CUR` and each other tree. p_D is percent duplication and p_L the average loss. Standard deviation σ was calculated from 1000 times of resampling in TestSet2. Both p_D and p_L are not available for `PARS-NT` because it contains unresolved nodes. For p_L, the Pearson's correlation coefficient between the two sets is 0.970.

with complex model.

So far as single-method trees are concerned, parsimony and ML methods show their power. They are usually better than distance-based methods, except in one case where `ME-NT-HKY4`, the tree built by FASTME, outperforms all the others on TestSet1. Tree `PARS-AA` is the best on TestSet2. But as it was merged from several parsimonious trees given by **protpars**, part of its high accuracy should attribute to the tree merge algorithm. ML methods worked smoothly well on both sets, and on each criterion as well. The role of applying Γ distribution is revealed but very subtle.

As to distance-based methods, nucleotide model HKY is uniformly better than all amino acid models. FASTME definitely outperforms standard neighbour-joining, which is in line with Desper and Gascuel (Desper and Gascuel, 2004). Notably, using complex evolutionary models at protein level does not guarantee higher accuracy at all. This confirms the observation by Hollich *et al.* (Hollich *et al.*, 2005).

Figure 6.2 shows the 'phylogenies' of tree builders. 'Signature' of different tree building algorithms are evident: proteins-level methods and nucleotide-level methods tend to be separated into two classes, while distance-based ones be grouped together. Similar methods share similar properties. This also implies that tree merge can be more helpful if trees with different properties are provided; otherwise common flaws shared by a group of algorithms will never be overcome.

6.5 Discussion

Properties of test data sets, such as divergence of sequences, number of sequences in a tree, number of available sites in alignments, and heterogeneity of site-specific rates, predetermine the performance of each tree builder. This has been observed by many previous studies (Kuhner and Felsenstein, 1994; Hall, 2005; Hollich *et al.*, 2005). Consisting of real data from TreeFam, our test sets represent a small number of families that date back to the last common ancestor of Eukaryotes. Limited to the size and characteristics of data, we cannot extensively study how each algorithm performs on various context. But as our benchmark were carried on real data, the results may be of more practical importance.

Our work shows that tree merge algorithm is capable of capturing the knowledge of biologists. It can greatly improve the accuracy of tree reconstruction when the phylogeny of species is clear. The success of tree merge also manifests that each class of single-model algorithm is able to build part of correct branches that cannot be reconstructed by others; otherwise there must be an algorithm that approaches the accuracy of tree `MERGE`. As no single-method algorithm can uniformly outperform all the others, it is always necessary to combine trees with different algorithms if higher accuracy is desired.

On the other hand, tree merge only works well with Eukaryotic gene trees, where duplications and losses can be inferred without the interference of LGT. On reconstruction of species tree or bacterial gene trees, single-method algorithms are the only solution. Our results suggest that each class of algorithms have their own strength. Although parsimonious tree based on protein data and ML trees seem better in general, distance-based method such as `ME-NT-HKY4` can still outperform them at times. Even when tree merge is not applicable, comprehensively investigating trees built from different classes

of algorithms is still always helpful.

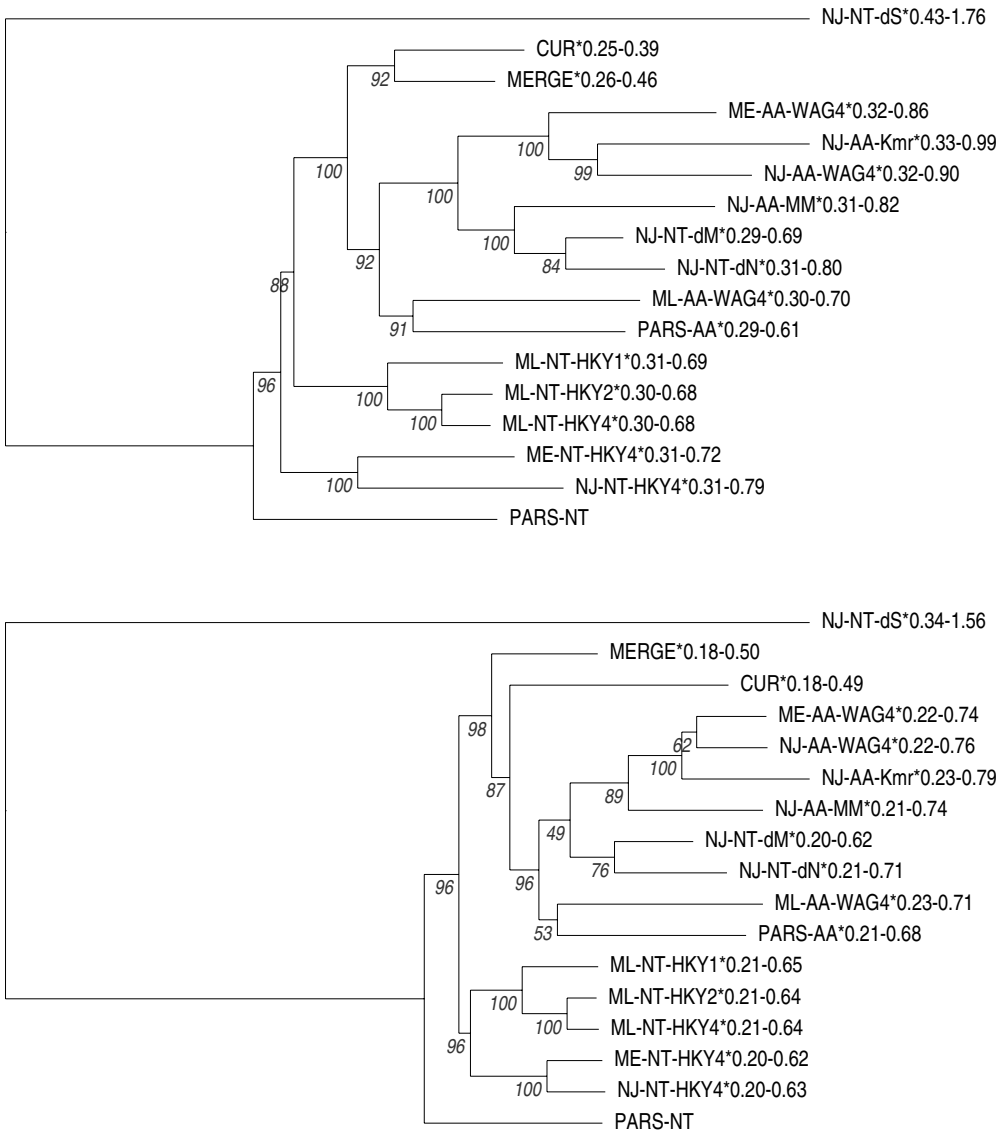

Figure 6.2: Relationships between tree builders. The top tree was built based on TestSet2, and the bottom on TestSet1. Following the name of each leaf, the first number is percent duplication p_D and the second average loss p_L. To build these two tree, topological distance d between each pair of tree builders was first calculated. Resultant distance matrix was then fed to NJTREE, which built the tree by neighbour-joining. Bootstrapping procedure have been described above. 100 resampled trees were finally combined together by **CONSENSE** in **PHYLIP** package. This gave the supporting values on each branch.

Appendix A

Technical Issues

A.1 NJTREE Software

NJTREE is the core engine of the whole TreeFam database. It realizes almost all the algorithms described in this thesis, including the cNJ algorithm (Section 3.2) for both rooted and unrooted constraining trees, rooting and bootstrapping, leaf reording algorithm (Section 3.5), duplication/loss inference for multifurcated species trees (Chapter 4), and tree merge algorithm (Chapter 5. It also provides a lot of utilities facilitating the construction of TreeFam (Chapter 2) and the benchmark done in Chapter 6. In addition, NJTREE incorporates source codes from PHYML (Guindon and Gascuel, 2003), which makes it capable of utilizing the power of ML methods. To some extent, most parts of the thesis is describing the priciples behind this software.

NJTREE is efficient. It is much faster than any other neighbour-joining tree builders, and can calculate max-likelihood distance in a speed not compared by other similar softwares. Even NJTREE-revised PHYML codes runs 20% faster than the original ones. In addition, NJTREE comes with a nice graphical user interface (GUI), FLNJTREE, which is built upon FLTK (fast light-weighted toolkit), a cross-platform widget library. FLNJTREE can be compiled in both UNIX (including LINUX) and Windows. In theory, it should also work in Mac OSX, though this has not been tested. Figure A.1 shows a snapshot of FLNJTREE.

A.2 MySQL Structures

TreeFam database is supported on MySQL, probably the most popular open source database in the world. Except the original PhIGs alignment, all the other data are stored in the relational database. The structures of TreeFam MySQL schema are very intuitive. Most the tables can be classified into three groups: tables for describing gene attributes, tables for recording family information, and tables connecting the two parts. Table A.1 gives a brief list of key tables in TreeFam database, and Figure A.2 shows the relations between them. Note that in TreeFam MySQL tables, gene identifiers and family accessions are directly used as index. This is clearer than using integer ID, but is ineffiecient from a pure technical angle [1]. As efficiency seems not a serious problem

[1] http://www.informit.com/articles/article.asp?p=377652&seqNum=1

at present, we can live with this for the moment. TreeFam MySQL database is open to public at **db.treefam.org:3308**. The read-only account is **anonymous** with empty password.

A.3 Perl API

It is recommended to connect TreeFam MySQL with perl API, which provides convenient interface for the retrieval of various TreeFam data. TreeFam API also implements a light-weighted parser for NHX format, a versatile tree plotter and a simple alignment

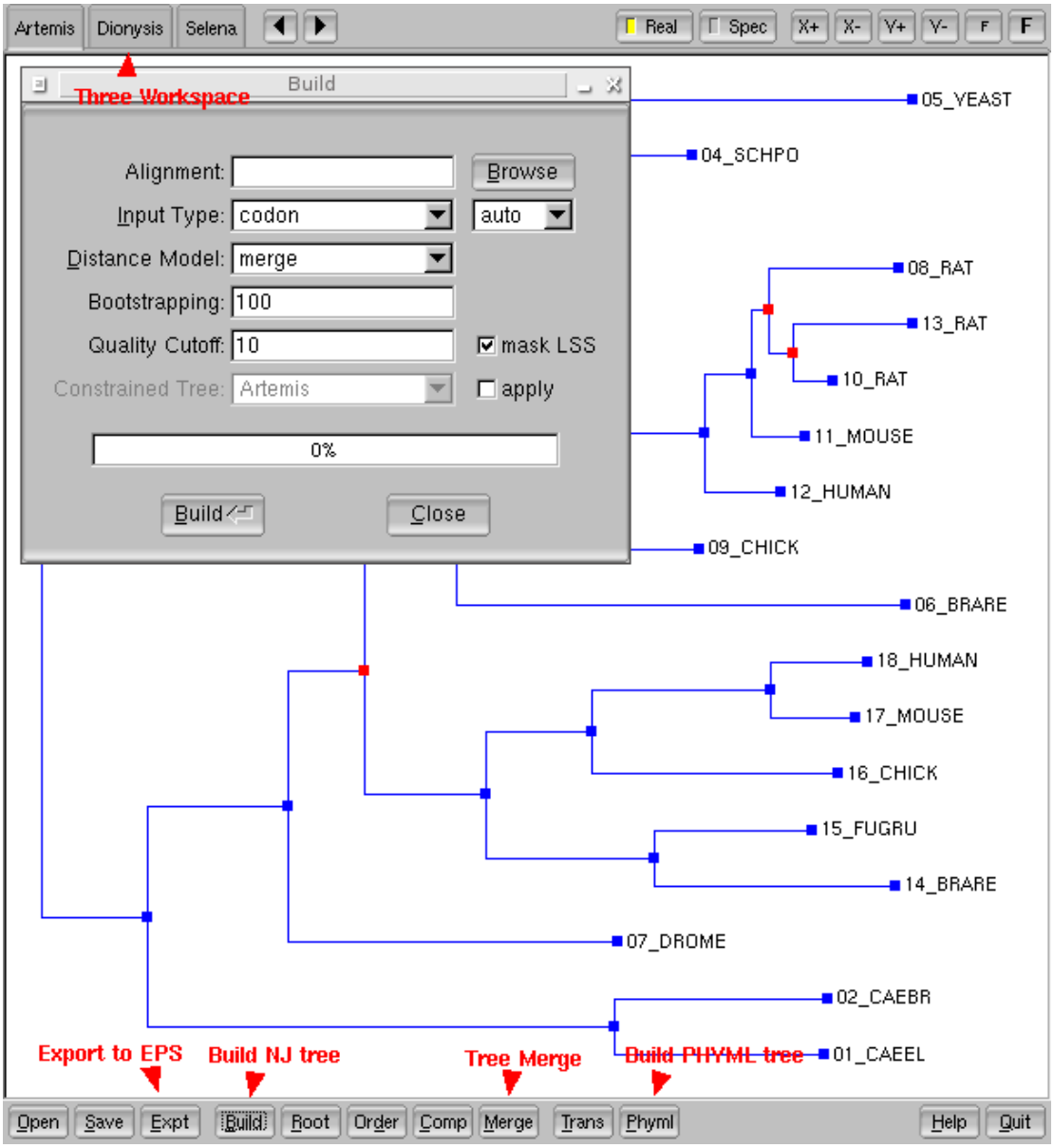

Figure A.1: Screenshot of FLNJTREE software.

plotter. Full documentation is available at http://www.treefam.org/api/.

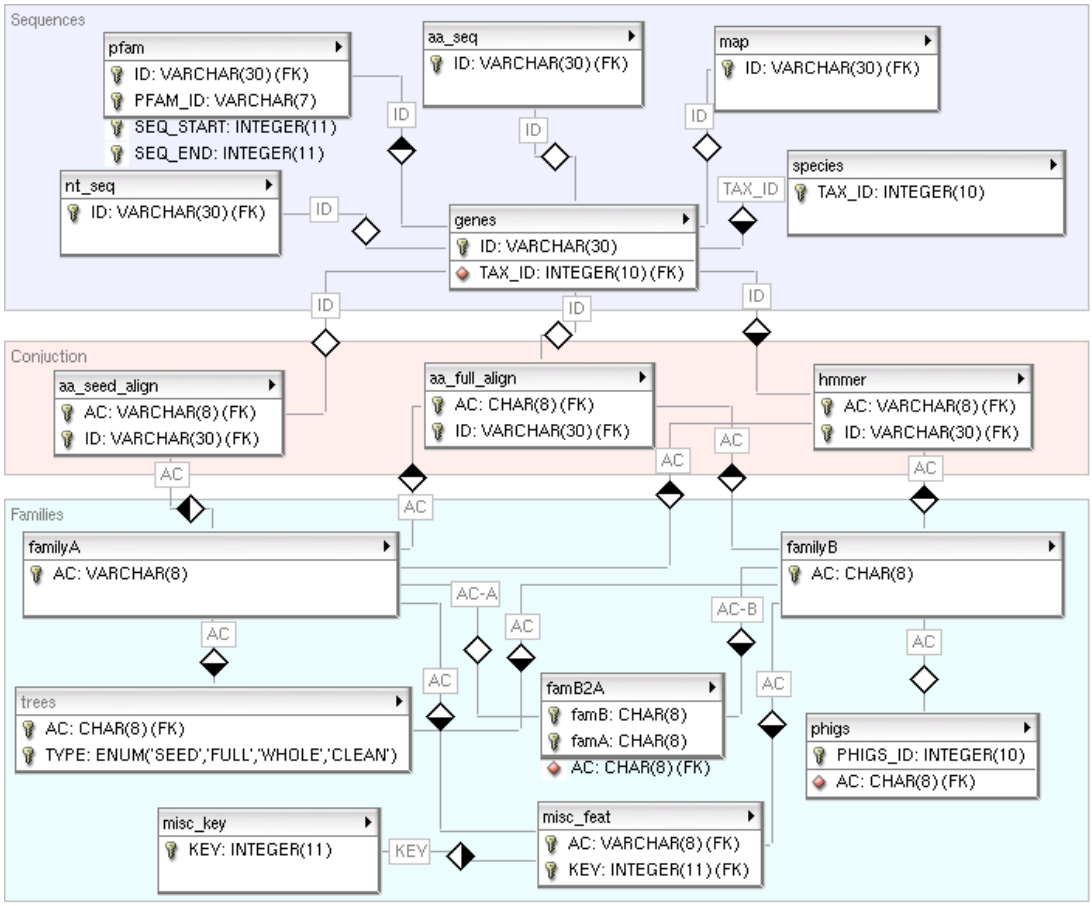

Figure A.2: Schema of TreeFam database. This figure is generated by DBDesigner4.

Table	Description
genes	sequence ID, gene name, transcript name, symbol, description and so on
species	tax ID, taxonomy name, *abbr.* name, and common name of species
map	genomic locations of transcripts; in UCSC format
pfam	Pfam predictions for each sequence
aa_seq	amino acid sequences
nt_seq	nucleotide sequences
familyA	accessions, symbols and names of TreeFam-A families
familyB	basic information on TreeFam-B families
famB2A	relation between curated TreeFam-A and original TreeFam-B families
phigs	PhIGs accessions of TreeFam-B families
trees	phylogenetic trees in NHX format
misc_feat	symbols and names of B families; curators of A families
misc_key	descriptions of 'key' used in 'misc_feat' table
aa_seed_align	amino acids multialignment for TreeFam-A seeds in CIGAR format
aa_full_align	full multialignment for both A and B families in CIGAR format
hmmer	HMMer scores of matched sequences

Table A.1: Description of key TreeFam MySQL tables. Trivial or obsolete ones are not included.

Appendix B

Publications

Li,H., Coghlan,A., Ruan,J., Coin,L.J., Hériché,J.K., Osmotherly,L., Li,R., Liu,T., Zhang,Z., Bolund,L., Wong,G.K., Zheng,W., Dehal,P., Wang,J. and Durbin,R. (2006) Treefam: a curated database of phylogenetic trees of animal gene families. *Nucleic Acids Res,* **34** (Database issue), 572–580. (first author)

Yu,J., Wang,J., Lin,W., Li,S., Li,H., Zhou,J., Ni,P., Dong,W., Hu,S., Zeng,C., Zhang,J., Zhang,Y., Li,R., Xu,Z., Li,S., Li,X., Zheng,H., Cong,L., Lin,L., Yin,J., Geng,J., Li,G., Shi,J., Liu,J., Lv,H., Li,J., Wang,J., Deng,Y., Ran,L., Shi,X., Wang,X., Wu,Q., Li,C., Ren,X., Wang,J., Wang,X., Li,D., Liu,D., Zhang,X., Ji,Z., Zhao,W., Sun,Y., Zhang,Z., Bao,J., Han,Y., Dong,L., Ji,J., Chen,P., Wu,S., Liu,J., Xiao,Y., Bu,D., Tan,J., Yang,L., Ye,C., Zhang,J., Xu,J., Zhou,Y., Yu,Y., Zhang,B., Zhuang,S., Wei,H., Liu,B., Lei,M., Yu,H., Li,Y., Xu,H., Wei,S., He,X., Fang,L., Zhang,Z., Zhang,Y., Huang,X., Su,Z., Tong,W., Li,J., Tong,Z., Li,S., Ye,J., Wang,L., Fang,L., Lei,T., Chen,C., Chen,H., Xu,Z., Li,H., Huang,H., Zhang,F., Xu,H., Li,N., Zhao,C., Li,S., Dong,L., Huang,Y., Li,L., Xi,Y., Qi,Q., Li,W., Zhang,B., Hu,W., Zhang,Y., Tian,X., Jiao,Y., Liang,X., Jin,J., Gao,L., Zheng,W., Hao,B., Liu,S., Wang,W., Yuan,L., Cao,M., McDermott,J., Samudrala,R., Wang,J., Wong,G. and Yang,H. (2005) The Genomes of Oryza sativa: a history of duplications. *PLoS Biol,* **3** (2). (co-first author)

Li,H., Liu,J., Xu,Z., Jin,J., Fang,L., Gao,L., Li,Y., Xing,Z., Gao,S., Liu,T., Li,H., Li,Y., Fang,L., Xie,H., Zheng,W. and Hao,B. (2005) Test data sets and evaluation of gene prediction programs on the rice genome. *J Comput Sci & Technol,* **20** (4), 446–453. (first author)

Wong,G., Liu,B., Wang,J., Zhang,Y., Yang,X., Zhang,Z., Meng,Q., Zhou,J., Li,D., Zhang,J., Ni,P., Li,S., Ran,L., Li,H., Zhang,J., Li,R., Li,S., Zheng,H., Lin,W., Li,G., Wang,X., Zhao,W., Li,J., Ye,C., Dai,M., Ruan,J., Zhou,Y., Li,Y., He,X., Zhang,Y., Wang,J., Huang,X., Tong,W., Chen,J., Ye,J., Chen,C., Wei,N., Li,G., Dong,L., Lan,F., Sun,Y., Zhang,Z., Yang,Z., Yu,Y., Huang,Y., He,D., Xi,Y., Wei,D., Qi,Q., Li,W., Shi,J., Wang,M., Xie,F., Wang,J., Zhang,X., Wang,P., Zhao,Y., Li,N., Yang,N., Dong,W., Hu,S., Zeng,C., Zheng,W., Hao,B., Hillier,L., Yang,S., Warren,W., Wilson,R., Brandström,M., Ellegren,H., Crooijmans,R., van der Poel,J., Bovenhuis,H., Groenen,M., Ovcharenko,I., Gordon,L., Stubbs,L., Lucas,S., Glavina,T., Aerts,A., Kaiser,P., Rothwell,L., Young,J., Rogers,S., Walker,B., van Hateren,A., Kaufman,J.,

Bumstead,N., Lamont,S., Zhou,H., Hocking,P., Morrice,D., de Koning,D., Law,A., Bartley,N., Burt,D., Hunt,H., Cheng,H., Gunnarsson,U., Wahlberg,P., Andersson,L., Kindlund,E., Tammi,M., Andersson,B., Webber,C., Ponting,C., Overton,I., Boardman,P., Tang,H., Hubbard,S., Wilson,S., Yu,J., Wang,J. and Yang,H. (2004) A genetic variation map for chicken with 2.8 million single-nucleotide polymorphisms. *Nature,* **432** (7018), 717–722.

Xia,Q., Zhou,Z., Lu,C., Cheng,D., Dai,F., Li,B., Zhao,P., Zha,X., Cheng,T., Chai,C., Pan,G., Xu,J., Liu,C., Lin,Y., Qian,J., Hou,Y., Wu,Z., Li,G., Pan,M., Li,C., Shen,Y., Lan,X., Yuan,L., Li,T., Xu,H., Yang,G., Wan,Y., Zhu,Y., Yu,M., Shen,W., Wu,D., Xiang,Z., Yu,J., Wang,J., Li,R., Shi,J., Li,H., Li,G., Su,J., Wang,X., Li,G., Zhang,Z., Wu,Q., Li,J., Zhang,Q., Wei,N., Xu,J., Sun,H., Dong,L., Liu,D., Zhao,S., Zhao,X., Meng,Q., Lan,F., Huang,X., Li,Y., Fang,L., Li,C., Li,D., Sun,Y., Zhang,Z., Yang,Z., Huang,Y., Xi,Y., Qi,Q., He,D., Huang,H., Zhang,X., Wang,Z., Li,W., Cao,Y., Yu,Y., Yu,H., Li,J., Ye,J., Chen,H., Zhou,Y., Liu,B., Wang,J., Ye,J., Ji,H., Li,S., Ni,P., Zhang,J., Zhang,Y., Zheng,H., Mao,B., Wang,W., Ye,C., Li,S., Wang,J., Wong,G. and Yang,H. (2004) A draft sequence for the genome of the domesticated silkworm (Bombyx mori). *Science,* **306** (5703), 1937–1940.

Wang,J., Zhang,J., Zheng,H., Li,J., Liu,D., Li,H., Samudrala,R., Yu,J. and Wong,G. (2004) Mouse transcriptome: neutral evolution of 'non-coding' complementary DNAs. *Nature,* **431** (7010), 1–1.

Li,C., Ni,P., Francki,M., Hunter,A., Zhang,Y., Schibeci,D., Li,H., Tarr,A., Wang,J., Cakir,M., Yu,J., Bellgard,M., Lance,R. and Appels,R. (2004) Genes controlling seed dormancy and pre-harvest sprouting in a rice-wheat-barley comparison. *Funct Integr Genomics,* **4** (2), 84–93.

Bibliography

Altschul,S.F., Madden,T.L., Schäffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res,* **25** (17), 3389–3402.

Arvestad,L., Berglund,A., Lagergren,J. and Sennblad,B. (2003) Bayesian gene/species tree reconciliation and orthology analysis using MCMC. *Bioinformatics,* **19 Suppl 1**, 7–15.

Arvestad,L., Berglund,A., Lagergren,J. and Sennblad,B. (2004) Gene tree reconstruction and orthology analysis based on an integreted model for duplications and sequence evolution. In *Proceeding of the Eighth International Conference on Computational Biology (RECOMB04)* pp. 326–335.

Ashburner,M., Ball,C.A., Blake,J.A., Botstein,D., Butler,H., Cherry,J.M., Davis,A.P., Dolinski,K., Dwight,S.S., Eppig,J.T., Harris,M.A., Hill,D.P., Issel-Tarver,L., Kasarskis,A., Lewis,S., Matese,J.C., Richardson,J.E., Ringwald,M., Rubin,G.M. and Sherlock,G. (2000) Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet,* **25** (1), 25–29.

Ayala,F. (1999) Molecular clock mirages. *Bioessays,* **21** (1), 71–75.

Bairoch,A., Apweiler,R., Wu,C.H., Barker,W.C., Boeckmann,B., Ferro,S., Gasteiger,E., Huang,H., Lopez,R., Magrane,M., Martin,M.J., Natale,D.A., O'Donovan,C., Redaschi,N. and Yeh,L.S. (2005) The universal protein resource (uniprot). *Nucleic Acids Res,* **33** (Database issue), 154–159.

Balakrishnan,R., Christie,K.R., Costanzo,M.C., Dolinski,K., Dwight,S.S., Engel,S.R., Fisk,D.G., Hirschman,J.E., Hong,E.L., Nash,R., Oughtred,R., Skrzypek,M., Theesfeld,C.L., Binkley,G., Dong,Q., Lane,C., Sethuraman,A., Weng,S., Botstein,D. and Cherry,J.M. (2005) Fungal blast and model organism blastp best hits: new comparison resources at the saccharomyces genome database (sgd). *Nucleic Acids Res,* **33** (Database issue), 374–377.

Baldauf,S. (2003) Phylogeny for the faint of heart: a tutorial. *Trends Genet,* **19** (6), 345–351.

Bateman,A., Coin,L., Durbin,R., Finn,R.D., Hollich,V., Griffiths-Jones,S., Khanna,A., Marshall,M., Moxon,S., Sonnhammer,E.L., Studholme,D.J., Yeats,C. and Eddy,S.R. (2004) The pfam protein families database. *Nucleic Acids Res,* **32** (Database issue), 138–141.

Bruno,W.J., Socci,N.D. and Halpern,A.L. (2000) Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. *Mol Biol Evol,* **17** (1), 189–197.

Bull,J.J., Badgett,M.R., Wichman,H.A., Huelsenbeck,J.P., Hillis,D.M., Gulati,A., Ho,C. and Molineux,I.J. (1997) Exceptional convergent evolution in a virus. *Genetics,* **147** (4), 1497–1507.

Cavalli-Sforza,L.L. and Edwards,A.W. (1967) Phylogenetic analysis. models and estimation procedures. *Am J Hum Genet,* **19** (3).

Chen,N., Harris,T.W., Antoshechkin,I., Bastiani,C., Bieri,T., Blasiar,D., Bradnam,K., Canaran,P., Chan,J., Chen,C.K., Chen,W.J., Cunningham,F., Davis,P., Kenny,E., Kishore,R., Lawson,D., Lee,R., Muller,H.M., Nakamura,C., Pai,S., Ozersky,P., Petcherski,A., Rogers,A., Sabo,A., Schwarz,E.M., Van Auken,K., Wang,Q., Durbin,R., Spieth,J., Sternberg,P.W. and Stein,L.D. (2005) Wormbase: a comprehensive data resource for caenorhabditis biology and genomics. *Nucleic Acids Res,* **33** (Database issue), 383–389.

Coin,L. and Durbin,R. (2004) Improved techniques for the identification of pseudogenes. *Bioinformatics,* **20 Suppl 1**, 94–94.

Cotton,J. (2003). *Vertebrate phylogenomics and gene family evolution.* PhD thesis, University of Glasgow.

Cummings,M.P., Handley,S.A., Myers,D.S., Reed,D.L., Rokas,A. and Winka,K. (2003) Comparing bootstrap and posterior probability values in the four-taxon case. *Syst Biol,* **52** (4), 477–487.

Dehal,P. and Boore,J.L. (2005) Two rounds of whole genome duplication in the ancestral vertebrate. *PLoS Biol,* **3** (10).

Delsuc,F., Brinkmann,H. and Philippe,H. (2005) Phylogenomics and the reconstruction of the tree of life. *Nat Rev Genet,* **6** (5), 361–375.

Desper,R. and Gascuel,O. (2004) Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. *Mol Biol Evol,* **21** (3), 587–598.

Dufayard,J., Duret,L., Penel,S., Gouy,M., Rechenmann,F. and Perrire,G. (2005) Tree pattern matching in phylogenetic trees: automatic search for orthologs or paralogs in homologous gene sequence databases. *Bioinformatics,* .

Durand,D., Halldórsson,B.V. and Vernot,B. (2006) A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J Comput Biol,* **13** (2), 320–335.

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press.

Eddy,S.R. (1998) Profile hidden markov models. *Bioinformatics,* **14** (9), 755–763.

Edgar,R.C. (2004) Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res,* **32** (5), 1792–1797.

Efron,B. (1979) Bootstrap methods: Another look at the jackknife. *The Annals of Statistics,* **7**, 1–26.

Enright,A.J., Van Dongen,S. and Ouzounis,C.A. (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res,* **30** (7), 1575–1584.

Eulenstein,O., Mirkin,B. and Vingron,M. (1998) Duplication-based measures of difference between gene and species trees. *J Comput Biol,* **5** (1), 135–148.

Felsenstein,J. (1978) Cases in which parsimony of compatibility methods will be positively misleading. *Syst Zool,* **27**, 401–410.

Felsenstein,J. (1981) Evolutionary trees from dna sequences: a maximum likelihood approach. *J Mol Evol,* **17** (6), 368–376.

Felsenstein,J. (1985) Confidence limites on phylogenies: an approach using bootstrap. *Evolution,* **39**, 783–791.

Felsenstein,J. (1989) Phylip - phylogeny inference package (version 3.2). *Cladistics,* **5**, 164–166.

Fitch,W.M. (1971) Toward defining the course of evolution: minimum change for a specific tree topology. *Syst Zool,* **20**, 406–416.

Fitch,W.M. and Margoliash,E. (1967) Construction of phylogenetic trees. *Science,* **155**, 279–284.

Frickey,T. and Lupas,A. (2004) PhyloGenie: automated phylome generation and analysis. *Nucleic Acids Res,* **32** (17), 5231–5238.

Gadagkar,S.R. and Kumar,S. (2005) Maximum likelihood outperforms maximum parsimony even when evolutionary rates are heterotachous. *Mol Biol Evol,* **22** (11), 2139–2141.

Gascuel,O. (1997) Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Mol Biol Evol,* **14** (7), 685–695.

Gaut,B.S. and Lewis,P.O. (1995) Success of maximum likelihood phylogeny inference in the four-taxon case. *Mol Biol Evol,* **12** (1), 152–162.

Goldman,N. and Yang,Z. (1994) A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol Biol Evol,* **11** (5), 725–736.

Goodman,M., Czelusniak,J., Moore,G., Romero-Herrera,A. and Matsuda,G. (1979) Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructured from goblin sequences. *Syst Zool,* **28**, 132–168.

Guigo,R., Muchnik,I. and Smith,T. (1996) Reconstruction of ancient molecular phylogeny. *Mol Phylogenet Evol,* **6** (2), 189–213.

Guindon,S. and Gascuel,O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol,* **52** (5), 696–704.

Hall,B.G. (2005) Comparison of the accuracies of several phylogenetic methods using protein and dna sequences. *Mol Biol Evol,* **22** (3), 792–802.

Hasegawa,M., Kishino,H. and Yano,T. (1985) Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *J Mol Evol,* **22** (2), 160–174.

Hertz-Fowler,C., Peacock,C.S., Wood,V., Aslett,M., Kerhornou,A., Mooney,P., Tivey,A., Berriman,M., Hall,N., Rutherford,K., Parkhill,J., Ivens,A.C., Rajandream,M.A. and Barrell,B. (2004) Genedb: a resource for prokaryotic and eukaryotic organisms. *Nucleic Acids Res,* **32** (Database issue), 339–343.

Hillis,D.M., Bull,J.J., White,M.E., Badgett,M.R. and Molineux,I.J. (1992) Experimental phylogenetics: generation of a known phylogeny. *Science,* **255** (5044), 589–592.

Hillis,D.M., Huelsenbeck,J.P. and Cunningham,C.W. (1994) Application and accuracy of molecular phylogenies. *Science,* **264** (5159), 671–677.

Holder,M. and Lewis,P.O. (2003) Phylogeny estimation: traditional and bayesian approaches. *Nat Rev Genet,* **4** (4), 275–284.

Hollich,V., Milchert,L., Arvestad,L. and Sonnhammer,E.L. (2005) Assessment of protein distance measures and tree-building methods for phylogenetic tree reconstruction. *Mol Biol Evol,* **22** (11), 2257–2264.

Hubbard,T., Andrews,D., Caccamo,M., Cameron,G., Chen,Y., Clamp,M., Clarke,L., Coates,G., Cox,T., Cunningham,F., Curwen,V., Cutts,T., Down,T., Durbin,R., Fernandez-Suarez,X.M., Gilbert,J., Hammond,M., Herrero,J., Hotz,H., Howe,K., Iyer,V., Jekosch,K., Kahari,A., Kasprzyk,A., Keefe,D., Keenan,S., Kokocinsci,F., London,D., Longden,I., McVicker,G., Melsopp,C., Meidl,P., Potter,S., Proctor,G., Rae,M., Rios,D., Schuster,M., Searle,S., Severin,J., Slater,G., Smedley,D., Smith,J., Spooner,W., Stabenau,A., Stalker,J., Storey,R., Trevanion,S., Ureta-Vidal,A., Vogel,J., White,S., Woodwark,C. and Birney,E. (2005) Ensembl 2005. *Nucleic Acids Res,* **33** (Database issue), 447–453.

Huelsenbeck,J.P. (1995) The robustness of two phylogenetic methods: four-taxon simulations reveal a slight superiority of maximum likelihood over neighbor joining. *Mol Biol Evol,* **12** (5), 843–849.

Huelsenbeck,J.P., Bollback,J.P. and Levine,A.M. (2002) Inferring the root of a phylogenetic tree. *Syst Biol,* **51** (1), 32–43.

Jones,D.T., Taylor,W.R. and Thornton,J.M. (1992) The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci,* **8** (3), 275–282.

Kolaczkowski,B. and Thornton,J.W. (2004) Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature,* **431** (7011), 980–984.

Koonin,E.V. (2005) Orthologs, paralogs, and evolutionary genomics. *Annu Rev Genet,* **39**, 309–338.

Kuhner,M.K. and Felsenstein,J. (1994) A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol Biol Evol,* **11** (3), 459–468.

Larget,B. and L,S.D. (1999) Markov chain monte carlo algorithms for the bayesian analysis of phylogenetic trees. *Mol Biol Evol,* **16** (6), 750–759.

Li,H., Coghlan,A., Ruan,J., Coin,L.J., Hériché,J.K., Osmotherly,L., Li,R., Liu,T., Zhang,Z., Bolund,L., Wong,G.K., Zheng,W., Dehal,P., Wang,J. and Durbin,R. (2006) Treefam: a curated database of phylogenetic trees of animal gene families. *Nucleic Acids Res,* **34** (Database issue), 572–580.

Li,L., Stoeckert,C.J. and Roos,D.S. (2003) Orthomcl: identification of ortholog groups for eukaryotic genomes. *Genome Res,* **13** (9), 2178–2189.

Li,W. (1993) Unbiased estimation of the rates of synonymous and nonsynonymous substitution. *J Mol Evol,* **36** (1), 96–99.

Lopez,P., Casane,D. and Philippe,H. (2002) Heterotachy, an important process of protein evolution. *Mol Biol Evol,* **19** (1), 1–7.

Margush,T. and Mcmorris,F. (1981) Consensus n-tree. *Bull Math Biol,* **43**, 239–244.

Meinel,T., Krause,A., Luz,H., Vingron,M. and Staub,E. (2005) The systers protein family database in 2005. *Nucleic Acids Res,* **33** (Database issue), 226–229.

Mi,H., Lazareva-Ulitsky,B., Loo,R., Kejariwal,A., Vandergriff,J., Rabkin,S., Guo,N., Muruganujan,A., Doremieux,O., Campbell,M.J., Kitano,H. and Thomas,P.D. (2005) The panther database of protein families, subfamilies, functions and pathways. *Nucleic Acids Res,* **33** (Database issue), 284–288.

Nee,S., May,R.M. and Harvey,P.H. (1994) The reconstructed evolutionary process. *Philos Trans R Soc Lond B Biol Sci,* **344** (1309), 305–311.

Nei,M. and Gojobori,T. (1986) Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol Biol Evol,* **3** (5), 418–426.

Ng,P.C. and Henikoff,S. (2003) Sift: predicting amino acid changes that affect protein function. *Nucleic Acids Res,* **31** (13), 3812–3814.

O'Brien,K., Remm,M. and Sonnhammer,E. (2005) Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Res,* **33 Database Issue**, 476–480.

Olsen,G.J., Matsuda,H., Hagstrom,R. and Overbeek,R. (1994) fastdnaml: a tool for construction of phylogenetic trees of dna sequences using maximum likelihood. *Comput Appl Biosci,* **10** (1), 41–48.

Page,R. (1994) Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst Biol,* **43**, 58–77.

Page,R. and Charleston,M. (1997) From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Mol Phylogenet Evol,* **7** (2), 231–240.

Philippe,H., Zhou,Y., Brinkmann,H., Rodrigue,N. and Delsuc,F. (2005) Heterotachy and long-branch attraction in phylogenetics. *BMC Evol Biol,* **5**, 50–50.

Rannala,B. and Yang,Z. (1996) Probability distribution of molecular evolutionary trees: a new method of phylogenetic inference. *J Mol Evol,* **43** (3), 304–311.

Remm,M., Storm,C. and Sonnhammer,E. (2001) Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J Mol Biol,* **314** (5), 1041–1052.

Robinson,D.F. and Foulds,L.R. (1981) Comparison of phylogenetic trees. *Math. Biosci.,* **53**, 131–147.

Roelofs,J. and Van Haastert,P. (2001) Genes lost during evolution. *Nature,* **411** (6841), 1013–1014.

Rzhetsky,A. and Nei,M. (1993) Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Mol Biol Evol,* **10** (5), 1073–1095.

Saitou,N. and Nei,M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol,* **4** (4), 406–425.

Salzberg,S., White,O., Peterson,J. and Eisen,J. (2001) Microbial genes in the human genome: lateral transfer or gene loss? *Science,* **292** (5523), 1903–1906.

Schmidt,H.A., Strimmer,K., Vingron,M. and von Haeseler,A. (2002) Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics,* **18** (3), 502–504.

Simmons,M.P., Pickett,K.M. and Miya,M. (2004) How meaningful are bayesian support values? *Mol Biol Evol,* **21** (1), 188–199.

Sokal,R. and Michener,C. (1958) A statistical method of evaluating systematic relationships. *University of Kansas Scietific Bulletin,* **28**, 1409–1438.

Spencer,M., Susko,E. and Roger,A.J. (2005) Likelihood, parsimony, and heterogeneous evolution. *Mol Biol Evol,* **22** (5), 1161–1164.

Stamatakis,A., Ludwig,T. and Meier,H. (2005) Raxml-iii: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics,* **21** (4), 456–463.

Stanhope,M., Lupas,A., Italia,M., Koretke,K., Volker,C. and Brown,J. (2001) Phylogenetic analyses do not support horizontal gene transfers from bacteria to vertebrates. *Nature,* **411** (6840), 940–944.

Susko,E., Inagaki,Y., Field,C., Holder,M.E. and Roger,A.J. (2002) Testing for differences in rates-across-sites distributions in phylogenetic subtrees. *Mol Biol Evol,* **19** (9), 1514–1523.

Suzuki,Y., Glazko,G.V. and Nei,M. (2002) Overcredibility of molecular phylogenies obtained by bayesian phylogenetics. *Proc Natl Acad Sci U S A,* **99** (25), 16138–16143.

Tatusov,R.L., Fedorova,N.D., Jackson,J.D., Jacobs,A.R., Kiryutin,B., Koonin,E.V., Krylov,D.M., Mazumder,R., Mekhedov,S.L., Nikolskaya,A.N., Rao,B.S., Smirnov,S., Sverdlov,A.V., Vasudevan,S., Wolf,Y.I., Yin,J.J. and Natale,D.A. (2003) The cog database: an updated version includes eukaryotes. *BMC Bioinformatics,* **4**, 41–41.

Thompson,J., Gibson,T., Plewniak,F., Jeanmougin,F. and Higgins,D. (1997) The CLUSTALX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res,* **25** (24), 4876–4882.

Weiss,G. and von Haeseler,A. (2003) Testing substitution models within a phylogenetic tree. *Mol Biol Evol,* **20** (4), 572–578.

Wheeler,D.L., Barrett,T., Benson,D.A., Bryant,S.H., Canese,K., Church,D.M., DiCuccio,M., Edgar,R., Federhen,S., Helmberg,W., Kenton,D.L., Khovayko,O., Lipman,D.J., Madden,T.L., Maglott,D.R., Ostell,J., Pontius,J.U., Pruitt,K.D., Schuler,G.D., Schriml,L.M., Sequeira,E., Sherry,S.T., Sirotkin,K., Starchenko,G., Suzek,T.O., Tatusov,R., Tatusova,T.A., Wagner,L. and Yaschenko,E. (2005) Database resources of the national center for biotechnology information. *Nucleic Acids Res,* **33** (Database issue), 39–45.

Whelan,S. and Goldman,N. (2001) A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Mol Biol Evol,* **18** (5), 691–699.

Yang,Z. (1994) Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: approximate methods. *J Mol Evol,* **39** (3), 306–314.

Yang,Z. (1996) Phylogenetic analysis using parsimony and likelihood methods. *J Mol Evol,* **42** (2), 294–307.

Yang,Z. and Nielsen,R. (2000) Estimating synonymous and nonsynonymous substitution rates under realistic evolutionary models. *Mol Biol Evol,* **17** (1), 32–43.

Yang,Z. and Rannala,B. (1997) Bayesian phylogenetic inference using dna sequences: a markov chain monte carlo method. *Mol Biol Evol,* **14** (7), 717–724.

Zdobnov,E.M., von Mering,C., Letunic,I., Torrents,D., Suyama,M., Copley,R.R., Christophides,G.K., Thomasova,D., Holt,R.A., Subramanian,G.M., Mueller,H.M., Dimopoulos,G., Law,J.H., Wells,M.A., Birney,E., Charlab,R., Halpern,A.L., Kokoza,E., Kraft,C.L., Lai,Z., Lewis,S., Louis,C., Barillas-Mury,C., Nusskern,D., Rubin,G.M., Salzberg,S.L., Sutton,G.G., Topalis,P., Wides,R., Wincker,P., Yandell,M., Collins,F.H., Ribeiro,J., Gelbart,W.M., Kafatos,F.C. and Bork,P. (2002)

Comparative genome and proteome analysis of anopheles gambiae and drosophila melanogaster. *Science,* **298** (5591), 149–159.

Zmasek,C. and Eddy,S. (2001*a*) A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics,* **17** (9), 821–828.

Zmasek,C.M. and Eddy,S.R. (2001*b*) Atv: display and manipulation of annotated phylogenetic trees. *Bioinformatics,* **17** (4), 383–384.

Zuckerkandl,E. and Pauling,L. (1965) Molecules as documents of evolutionary history. *J Theor Biol,* **8** (2), 357–366.

Index