分类号 _____ 密级 _____

UDC _____ 编号 _____

中国科学院研究生院
理学博士学位论文

TreeFam数据库的建立

李 恒

指导教师 郑伟谋 研究员

中国科学院理论物理研究所

申请学位级别 理学博士 学科专业名称 理论物理

论文提交日期 2006年5月 论文答辩日期 2006年6月

培养单位 中国科学院理论物理研究所

学位授予单位 中国科学院研究生院

答辩委员会主席 李松岗 教授

TreeFam数据库的建立

李恒（理论物理专业）
导师：郑伟谋
中国科学院理论物理所

摘要：TreeFam同源基因数据库是关于基因家族进化的数据库。它致力于发展成为一个经过人工校正，准确而又翔实的资源，提供所有已知动物基因家族的进化历程以及可靠的直系同源与旁系同源信息。在开发TreeFam数据库的过程中，我们设计了四个新颖的算法以提高构建进化树的准确性及辅助数据库建设。第一个算法是约束邻接法。给定一个特定子树的拓扑，这个算法可以高效的重构整个进化树而不改变给定子树的拓扑结构；这样人工校正结果可以得以保留。第二个是叶节点重排算法。在画树时，这个算法保证叶节点以接近指定的顺序在平面排列，从而使得相似的树画在纸面上有着相似的视觉效果。第三个算法是推断基因倍增/缺失算法。在这一部分中，我们把前人的推断算法纳入更加一般的理论框架，并将其推广于物种树为多分岔的情形。基因倍增/缺失推断算法的概率形式也在这一章得到讨论。第四个算法是树合并算法，它结合多棵从同一序列集构造的基因树，通过减少推断出的倍增/缺失事件和选择较高的自展支持而构建一棵最优树从而超越所有备选树。在文章结尾基于真实数据的评测表明树合并算法可以显著提高建树的准确性；而这一事实也说明：尽管简约法与极大似然法一般而言更加准确，它们仍不可完全替代距离法。各种方法的相互补充是准确建树的关键。TreeFam数据库的网址是http://www.treefam.org，在中国的镜像为http://treefam.genomics.org.cn。

关键词：分子进化，进化树重构，数据库

Constructing the TreeFam database

Li Heng (Major in Theoretical Physics)
Directed by Zheng Wei-mou

The Institute of Theoretical Physics
Chinese Academic of Science

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. TreeFam is freely available at http://www.treefam.org or http://treefam.genomics.org.cn.

Keywords: molecular phylogenetics, tree reconstruction, database

Contents

List of Figures

List of Tables

Chapter 1

背景介绍

系统进化学(phylogenetics)是研究一群物种或基因进化关系(phylogenies)的一门科学 [1]。它的一个最基本的假设是地球上所有物种都有一个共同祖先(common ancestor)，并从这个祖先开始以树状发展。这个假设源自达尔文(Darwin)，并为之后众多的分子生物学证据所证明 [2]。准确的构建生命之树(tree of life)一直是进化学家们的理想和奋斗目标。

起先进化树是通过比较现有生物或化石的形态学特征来实现的。这种做法即使在现在也十分耗时并需要分类者有着充分的经验 [3]。自从在1965年Zuckerkandl和Pauling [4]发表了他们划时代的文章来说明分子序列也可以作为进化特征来建树，分子系统进化学登上了科学的舞台，并随着测序技术的进步成为系统进化学发展最快而又最富有成果的一个分枝。它不仅为生物学家提供了大量有价值的分析结果，也促进了计算机科学和统计学的发展。

1.1 背景介绍

构建物种的生命之树是进化学的核心问题，而构建每一个基因家族的基因进化树则显示了分子进化学在分子生物学领域更实际的作用。基因家族的进化关系不仅给出了基因进化的历史，它也是基因注释的关键：当基因进化关系明确时，一个物种的注释或对一个物种基因的实验结果可以通过进化关系传递到另一个物种；这样，基因进化将不同物种的基因注释结果织成一个整体，这对现有的基因注释尤其是新测序物种的注释至关重要。同时，重构基因的进化关系是发现基因进化规则的前提，而这些规则又为进一步阐释基因功能提供了重要线索。举例来说，基因在不同时期受到的选择压力的不同暗示了基因功能的改变，而有意义的突变，假基因(pseudogene)和互作用网络都可以以进化树为辅助而得以推断[5, 6]。另外，恢复基因家族的进化关系也为包括内含子(intorn)进化

和基因组复制等其它的进化分析铺平了道路。

出于基因进化关系的重要性，许多数据库都给出了基因家族的进化史，它们包括KOG [7], PANTHER [8], SYSTERS [9] PhIGs [10], Ensembl-compara [11], OrthoMCL [12]和HOGENOM [13]。这些数据库都为进化学的发展提供了重要参考，但它们也都面临同样一个问题：怎样重构准确的进化树？而这个问题却是生物信息领域最困难的问题之一[14]。序列质量较低(比如基因注释错误和序列较短)和不切实际的进化模型(比如假设基因在不同种系总以相同速率进化)都会影响建树的质量。没有一个自动算法能在各种情况下准确建树。为了使一个基因进化的数据库发挥更大的实用价值，这个问题必须得以解决。

1.2 论文概览

TreeFam是一个关于(基因)进化树的数据库，进化树是该数据库的主要单元。在开发TreeFam 的过程中，我们试图从两个方向解决建树的困难：首先，我们开发了一个新的算法以提高自动建树的质量；其次，我们实现了约束邻接法以在建树时使用人工校正的信息进一步提高建树的准确性。本文将详细描述这些算法，并且涵盖构建TreeFam数据库相关的大部分细节，包括：基本概念和原理、自动流程、其它各种算法以及实现。

在本章剩下的小节中我们将阐明分子进化学上的生物概念和一些数学记号。第二章描述了TreeFam的结构及自动流程。第三章涵盖了在建树过程中许多相对较小而又很重要的方面，尤其注重讲述了TreeFam独特的两个算法：约束邻接法和叶结点重排算法。在第四章我们给出了对多分叉物种树进行倍增/缺失推断的算法，并将之纳入更一般的理论框架。第五章详细地给出了树合并算法，这也是TreeFam提高自动建树质量的主要算法。在第六章我们使用TreeFam中经过校正的树为测试集对各种建树算法进行了综合的评测，也是对自动构树算法的总结。图1.1给出了各章之间的关系。

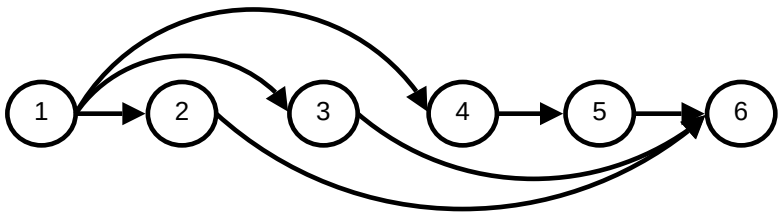

Figure 1.1: 各章之间的关系图。

除了本章，本论文大部分结论都未发表。即使在第二章，我们也对所发表的文章 [15]有了更进一步的阐释。我们期望在下面的几个月中将此论文中一些主要章节予以发表[1]。

1.3 系统进化学基础

进化系统(phylogeny)指物种或基因之间在进化上的联系。这种关系可以表征为一棵树，称做进化树(phylogenetic tree)。系统进化学(phylogenetics)是研究进化系统的科学，是生物学的一个分枝。系统进化学主要研究如何从现有数据复原进化历史和影响进化的原因以及进化对生物的意义。

在本文我们将涉及两种进化树：物种树(species tree)和基因树(gene tree)。严格的说，一个物种是“一群可以在内部杂交或交配而不能和其它物种杂交或交配的生物体的集合”[2]，而物种树则是描绘物种间关系的进化树，它的每一个叶结点(leaf node, external node)都代表着如今可以观测到的现有物种(present species)，而它的每一个内结点(internal node)则代表在历史上存在的祖先物种(ancestral species)。一个祖先物种通常以分类名(taxon)来命名。比如，祖先物种*Eutheria*就是指分类名*Eutheria*所包括涵盖的全部物种的最近共同祖先物种。为生物体命名和分类的科学叫系统分类学(taxonomy)，它也是进化学的一个分枝[3]。

基因(gene)定义为一段有功能或表型的DNA片段，在没有确知功能时一个基因也可以以序列、转录体或存在同源关系来表征[4]。基因在进化上是相互联系的。同源基因(homolog)是历史上从一个基因进化过来的一组基因 [16]；直系同源基因(ortholog)则是从两个物种的最近共同祖先(last common ancestor, LCA)的一个基因进化过来的一组基因 [17]；如果一个物种的m个基因和另一物种的n个基因都从两个物种的LCA的一个基因进化而来，则称这$m+n$个基因为$m:n$直系同源(图1.2)。从定义上看，直系同源基因是亲缘更近的同源基因；不是直系同源的同源基因称做旁系同源基因(paralog)。广义的说，一个基因家族(gene family)是从一个祖先基因进化而来的一组基因[5]。基因家族中基因的进

[1]此论文的中文版事实上是从英文版翻译过来的，即先有英文版而后才有中文翻译。由于本文所述内容较新，多无已有翻译为参照，因此许多术语的翻译可能不很得体，我们尽量给出相应的英文以供参考。另外，由于中文排版问题，索引和PDF格式中的连接只在英文版给出。

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

[3]根据系统分类学，所有的物种都在六个主要层次上划分，按从大到小的顺序，它们是：kingdom、phylum、class、order、family和genus。以人为例，他/她属于下列分类：*Animalia* (kingdom), *Chordata* (phylum), *Mammalia* (class), *Primates* (order), *Hominidae* (family), *Homo* (genus), and *Homo sapiens* (species)。

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

[5]TreeFam中使用基因家族这个概念时略有不同。见第2章。

化关系也可以以树的形式展示，这称做基因树(gene tree)。类似地，基因树的每一个叶结点代表着一个现有基因，而每一个内结点代表了仅在历史上存在的一个基因。

物种树和基因树是相互联系的：物种树可以从基因树中推断，而基因树在某种程度上反映了物种的进化关系。同时，基因树也可能由于基因倍增(duplication)，缺失(loss)和水平基因迁移(lateral gene transfer, LGT)等进化事件而不同于物种树；这些事件也能靠基因树与物种树的树融合(tree reconcili-

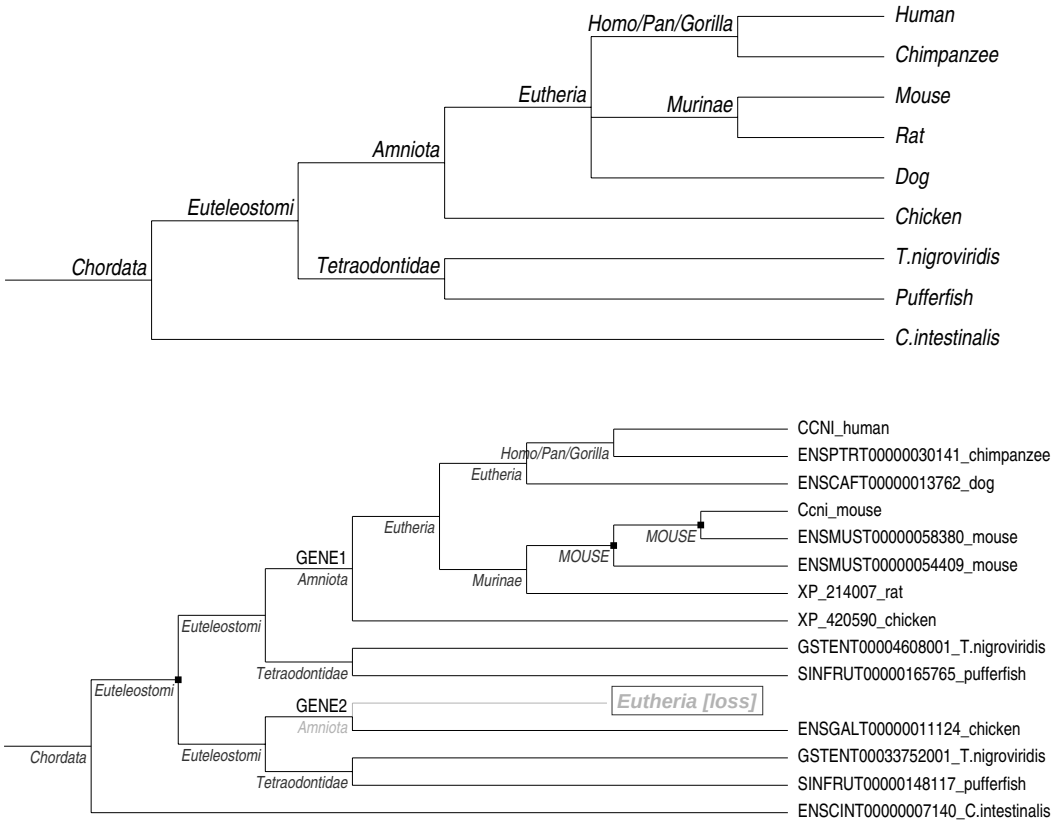

Figure 1.2: 物种树和基因树的例子。上图为*Chordata*纲的一棵物种树，本文在其它章节也会使用这棵物种树。下图为*Cyclin I*基因家族的基因树。内结点旁边的斜体字为包含相应祖先基因的祖先物种名。在这棵基因树中人的基因CCNI_human和鸡的XP_420590_chicken是1:1直系同源，因为它们都是从人鸡共同祖先*Amniota*的一个基因进化而来。但这个人的基因和ENSGALT00000011124_chicken只是普通同源而不是直系同源，这是因为尽管两个基因在*Euteleostomi*中是一个基因，它们在人鸡共同祖先*Aminota*中是两个基因(GENE1和GENE2)。另外，这个基因树也给出了几个1:n直系同源的例子，比如，基因ENSCINT00000007140_C.intestinalis和河豚的两个基因SINFRUT00000165765_pufferfish与SINFRUT00000148117_pufferfish是1:2直系同源。下面的基因树还会出现在第四章，在那里会给出更多的细节。

ation)过程再现。TreeFam数据库——也是本论文——一个与众不同之处是它大量依靠物种树和基因树之间的关系推断进化事件，并用这种关系来提高自动建树质量。

1.4 树的基本概念

在图论中，树是没有圈的连通图，它由结点(node or vertex)和连接结点的边(branch or edge)组成。一个结点可以是只连接一条边的外结点(external node)也可以是连接两条或更多边的内结点(internal node)。外结点也称为叶结点(leaf)。

严格的说，每一棵进化树都是有根树(rooted tree)，代表着进化的方向性。它的根结点(root)代表着树中所有物种或基因的最早的共同祖先。在有根树中每一个内结点都有它的子结点(children or daughter)；除了根结点，每一个结点都有一个父结点(parent)。靠近树根的枝(结点)称为较高的枝(结点)，而靠近叶子的称做较低的枝(结点)。给定一个结点的集合，覆盖这些结点的最低结点称做最近共同祖先(last common ancestor, LCA)；这个集合中所有的结点都从它们的LCA结点演化出来。

一棵真实的进化树也是一棵二岔树(binary tree)，它的每一个内结点有且仅有两个子结点。由于在二岔树中所有的分化都是清楚的，二岔树也称做一棵已决定树(resolved tree)。但是，如果两个连续的分化事件在历史上很短的时间内发生，它们在建树时是很难准确区分的，因此在实用中一个内结点也允许有三个或更多子结点，这样的结点称做多分岔点或未决定点(multifurcated node, unresolved node, or polytomy)，含有多分叉点的树称做多分叉树或未决定树(multifurcated tree or unresolved tree)。现在广泛使用的NCBI物种分类树 [18]是一棵多分叉树。

1.4.1 树的表示

显示一棵树的最直观方法是将它画在纸上。但是，一个图形既不能直接用数学语言来描述也不能很方便的用计算机程序来处理，因此我们有必要将树以抽象的语言来描绘以详细阐述有关树的算法。

在本文我们将使用三种树的表示方法：图表示(graph representation)，字符串表示(string representation)和集合表示(set representation)。前两种表示法将在下面介绍，而第三种将在第五章引入，在那一章，更加抽象的集合表示法更有助于严格地阐明算法和证明。

图表示：定义及记号

除了第五章我们会为了方便树合并算法的介绍而引入新的记号，这一小节的定义和记号将应用于整篇论文。如果我们认为地球上所有生物都只有一个共同祖先，所有进化树都是有根的。除了在个别章节我们特殊声明会考虑无根树，在本文绝大部分只考虑有根树。在第3.3节我们会看到如何为无根树定根。

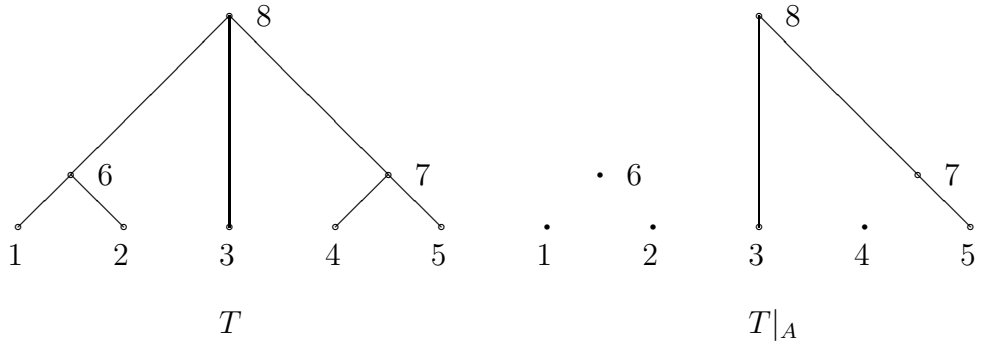

Figure 1.3: 用于解释基本概念的一个例子。在原始的树T中，$\mathrm{lca}(\{4,6\}) = 8$和$\omega(8) = \{1,2,3,4,5\}$成立。如果$A = \{3,5\}$，则有显示在右边限制于$A$的子树$T|_A$，在这棵树中$V_E(T|_A) = \{3,5\}$以及$V_I(T|_A) = \{7,8\}$成立。要注意的是在子树$T|_A$中，结点7只有一个子结点。树$T$可以用这样的一个New Hampshire字符串(见本小节)来表示：`((1,2)6,3,(4,5)7)8`；如果不计叶子顺序，也可以表示成`((5,4)7,(1,2)6,3)8`。它们是同一棵树。类似地，$T|_A$的New Hampshire字串表示为：`((3,(5)7)8)`。

令T是一棵有根树，$V(T)$是它的结点集，$V_E(T)$是T的外结点或叶结点集，而$V_I(T)$为内结点集。给定一个结点v，它的子结点形成集合$\mathrm{chi}(v) \subset V(T)$，它的父结点为$\mathrm{par}(v) \in V(T)$.

对结点$u, v \in V(T)$，如果u由v进化而来我们则记$u < v$或$v > u$。结点v所覆盖的所有叶结点组成集合为：

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

类似地，$A \subset V(T)$覆盖的叶结点的集合为：

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

有时如果在上下文只出现一个树T，我们也将$\omega_T(v)$简记为$\omega(v)$。

下面我们来看如何表示T的一棵子树。对一个结点的集合A，令$\mathrm{lca}(A)$为A的最近共同祖先。令$T|_A$为根在$\mathrm{lca}(A)$包含T中连接$\omega(A)$的路径组成的子树。这样子树$T|_A$的结点集为：

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

我们也简记$T|_{\{v\}}$为$T|_v$。图 1.3给出了关于这些概念的一个具体例子。

树的字符串表示：**New Hampshire**格式

New Hampshire (NH)，或Newick格式是标准的计算机可识别的树的表示方法。它使用相互嵌套的括号来表示树的结点关系，从而将树转化成一个字符串，称做NH字串(NH string)。

NH格式十分简单而直观。图 1.3也给出了一个例子。用正式的形式文法(formal grammar)[6]，NH格式可以严格地写成如下形式：

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

这里<comment>、<id>和<dist>分别表示注释、标识符和距离，它们完全可以用正则表达式(regular expression)来表示。事实上，TreeFam所使用的格式是扩展NH格式(New Hampshire eXtended, NHX)。这种格式将结点的其它信息以一种特定的":键=值"的格式存于<comment>中。NHX格式最早由Zmasek等人提出 [19]。

我们必须注意到同一棵树可以用许多不同的NH字串来表示，这些字串都代表同样的拓扑，它们的不同在于叶结点在字串中出现的顺序不同。在实用中我们习惯上按树画在纸上的图形中叶结点出现的顺序来写NH字串，比如对图1.3中的树T我们习惯上将其表示为((1,2)6,3,(4,5)7)8而不是其它字串。从这一点上说，一个NH字串和一个树的图形有着自然的一一对应关系。

如前一段所指出的，一棵树可以表现为不同的图形或不同的NH字串。尽管这些图形和字串都表示着同一棵树，但在人的眼中它们是很不相同的。给定两棵含有几百棵叶结点的大树，如果两棵树叶结点排列顺序不是很相似，直接在视觉上比较两棵树是极为困难的。设计一个以同一方式画树的算法十分必要。我们将在第三章解决这个问题。

1.4.2 无根树拓扑的比较

给定一个集合V，V的一个二分为一对不相交的集合A和B，表示为$A|B$或$B|A$，它满足：(i) $A\cap B=\emptyset$并且(ii) $A\cup B=V$。令$T=(V(T),E(T))$是

[6]形式文法是计算机科学编译原理中对程序设计语言的常用表示方法。

一棵*无根树*，$V(T)$是它的结点集，$E(T)$是边集，则树T的每一条边都将T分成两个不相交的部分；如果我们令这两个部分所的叶结点的集合分别为A和B，则$A|B$是$V(T)$的一个二分。这样，T的每一条边都对应一个$V(T)$的二分$A|B$，因此我们可以定义：

$$\tilde{T} = \{A|B : \text{存在}T\text{的一条边对应于}V(T)\text{的二分}A|B\} \tag{1.4}$$

显然$|\tilde{T}| = |E(T)|$成立。当T是二岔树时$|E(T)|$达到最大值$2 \cdot |V(T)| - 3$，$|\tilde{T}|$的最值也为$2 \cdot |V(T)| - 3$。在本文，如果我们说“边$A|B$在T中存在”，我们指$A|B \in \tilde{T}$，或者等价地，T中有一条边与$A|B$相对应。以图1.3为例，边$(8, 6)$对应的二分为：$\{1, 2\}|\{3, 4, 5\}$。

集合$\tilde{T}$引入后，有相同叶结点集的两棵树就可以进行比较了。令T_1和T_2是两棵无根树，并且$V(T_1) = V(T_2)$，衡量它们之间拓扑不同的Robinson-Fould距离[20]为：

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

这个距离事实上等于存在与一棵树但不存在于另一棵的边的个数。要注意和叶结点相连的边总同时在两棵树中存在，因此d_T的最大值为$2 \cdot (|V(T_1)| - 3)$，它在当两棵二岔树T_1和T_2完全不同时达到。

Chapter 2

TreeFam数据库的建立

从进化的角度说一个基因家族中的基因是相互关联的。为了在物种间传递基因注释的信息以及为了研究基因进化的规律，分析一个基因家族中基因的进化关系是至关重要的，而实现这一目的的最好方法是构建这个基因家族的进化树。进化树不仅给出了进化的每一细节，也为进一步的功能推断提供了基础。TreeFam [15] (Tree families database)是关于基因家族进化树的数据库。它的目的在于开发经过人工校正的资源以准确给出所有多细胞动物基因家族的进化历史以及基于进化树的各种推断。在这一章我们将介绍TreeFam数据库的内容和结构以及构建TreeFam的自动流程。

2.1 TreeFam概览

2.1.1 什么是TreeFam

首先TreeFam是一个分类数据库。它把基因分成基因家族并为每一个家族和重要的子家族命名。以往的分类数据库，如KOG [7], PANTHER [8]和SYSTERS [9]，都是通过序列之间的相似度来划分基因，但TreeFam则不然。它把一个基因家族定义成从多细胞动物共同祖先中的一个基因，或者从第一次出现在多细胞动物中的一个基因演化出来的一群基因。事实上，以相似性分数为表现的进化速率在不同的基因之间是有很大差异的，基于相似性定义的基因家族不可避免的会对聚类阶段所使用的阈值非常敏感，这种定义也缺乏生物上的意义。相比之下，TreeFam不会遇到这样的问题。正如PhIGs数据库 [10]所指出的，像TreeFam这样从进化角度定义的数据库不会对阈值敏感，更加稳定，也更具生物意义。

其次，TreeFam是直系同源基因(ortholog)的数据库。它从基因家族的进化树上推断直系同源基因。传统方法是从成对物种

基因序列的比对来推断直系同源基因，大部分同源基因数据库都是基于这种方法，包括：NCBI的HomoloGene [18]，Ensembl-compara [11]，OrthoMCL [12]和Inparanoid [21]。尽管这些数据库对同源基因的推断起到了积极作用，但当基因缺失发生时，它们就会出现问题。这是由于成对推断时没有看到其它物种，成对方法都会漏掉一些倍增事件而预测出更多的直系同源基因。更为严重的是，成对方法在不同物种对上的推断可能是相矛盾的。举个例子。假设基因g_1和g_2以及g_2和g_3分别是两对是一一直系同源(1:1 ortholog)，则g_1和g_3理论上也应是一一直系同源。但用成对推断的方法，g_1和g_3可能被推断成不是1:1直系同源，这是因为三对基因(g_1, g_2)、(g_2, g_3)和(g_3, g_1)是分别处理的，成对方法不要求它们之间是自洽的。相比之下，由于基于进化树的方法将许多物种看成一个整体，这些问题就不会存在了。据我们所知，除了TreeFam目前只有HOGENOM [13]从进化树上推断直系同源基因。

最后，TreeFam是一个经过人工校正的数据库。实际数据反复说明在不同种系和不同基因家族间的进化规律是不同的。准确的自动建树始终是生物信息领域最具挑战的课题之一。这也是为什么生物学家即使知道基于进化树推断的优点也不得不退而求其次使用成对方法的原因。尽管TreeFam开发了新的算法以为提高建树的准确性，自动建树还是不能与人工校正相比——只有人能够成功的结合多方面的信息，也只有人能够赋予进化树以生物学的意义。TreeFam特殊之处在于它是经过人工校正的。

2.1.2 TreeFam的基本结构

TreeFam的基本结构很像Pfam[22]——一个人工校正的蛋白质结构域(protein domain)数据库。像Pfam一样，TreeFam也包含两个部分：经过人工校正的TreeFam-A，和自动生成的TreeFam-B；每一部分都包括两种类型的序列：经过人工校正或从PhIGs来的种子序列(seed)，以及在种子序列中加入新序列而形成的全序列(full)。图 2.1(A)给出了这些概念之间的联系。具体的细节将在下面介绍。

2.2 输入数据

2.2.1 序列集

TreeFam-2包含从GenBank [18]、SGD [23]、WormBase [24]、GeneDB [25]和Ensembl [11]等数据库中下载的19个已测序物种的核苷和蛋白序列(表2.1)。如果数据库提供基因在基因组中的坐标和相应的蛋白质结构域，这些数据也将加入TreeFam。在

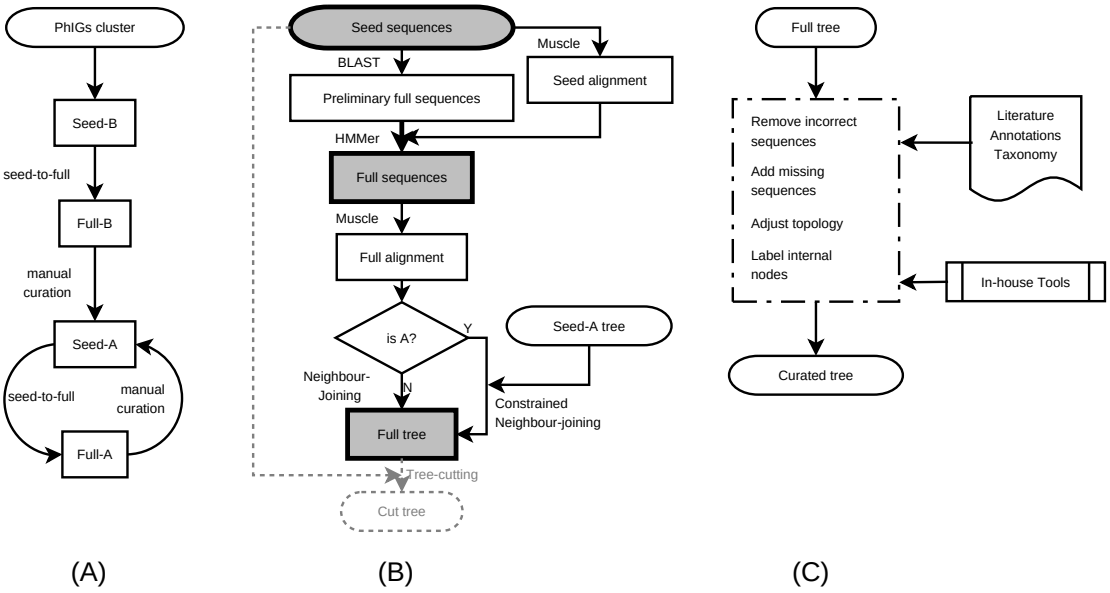

Figure 2.1: TreeFam的流程图。(A) 整体结构。TreeFam-B中的种子序列(seed)来自于PhIGs数据库。这些种子通过(B)中描绘的过程而扩展为完整序列(full)。人工校正将TreeFam-B的基因家族校正为TreeFam-A，而TreeFam-A的基因家族也可以在日后重新校正。(B) 种子扩展到完整的过程。虚线和灰色前景的文字表示这些过程只存在于早期的TreeFam-1.x，而灰色背景的方框表示相关部分的在TreeFam-2中有所改变。要注意的是，完整的种子扩展到完整的过程只用于序列升级的情况。当TreeFam-B校正为TreeFam-A时，新的TreeFam-A的种子由手工生成，完整序列直接取自TreeFam-B。(C) 人工校正概览。这个过程涉及阅读文献的支持和一些TreeFam特有工具的使用。

本章，TreeFam序列集将简称为TFSEQ。

在19个物种中，两种酵母和拟南芥(*A.thaliana*)都作为外群(outgroup)。在基因树中，外群基因能显示出早于多细胞动物最近共同祖先的更古老物种，它们用于判断是否两组动物基因应该分到两个TreeFam基因家族中而不是一个。可以说，外群的加入界定了基因家族的范围。

2.2.2 种子序列集

整个TreeFam自动流程的起点是TreeFam-B种子集合的建立，这必须通过聚类算法。幸运的是PhIGs已经按照我们期望的方式将所有动物的蛋白进行了聚类。在PhIGs数据库中，每一个类中的基因都是从一些指定物种的共同祖先一个基因进化而来的；基因由于进化关系而不是之间的相似度而聚类在一起。目前PhIGs网站提供两种聚类：一种是对所有脊椎动物的基因聚类，一种是对所有多细胞动物的聚类。理想状况下，第二种聚类下的一个类正好对应于一个TreeFam的基因家族。鉴于此，TreeFam跳过了聚类这一步而直接用PhIGs的

类作为TreeFam-B的种子。

2.2.3 其它数据

一棵进化树不仅仅提供了基因家族进化的历史，它也为进一步的进化研究提供了基础，比如：内含子(intron)进化，蛋白结构域进化及基因功能改变的研究。为了辅助这些研究我们争取将其它相关信息加入TreeFam。自TreeFam-2起，我们已经加入剪切和结构域信息，在TreeFam-3中我们将进一步提供表达信息。基因敲除的表型结果和GO注释(GO ontology [26]) 也有可能加入进来。

2.3 TreeFam自动流程

TreeFam-2所使用的自动流程和TreeFam-1.x是不同的。这种不同体现在两点：PhIGs的重聚类和“竞争”方法的使用。

Table 2.1: TreeFam-2中包含的所有完整测序物种。两种酵母(*S.pombe*和*S.cerevisiae*)及拟南芥(*A.thaliana*)作为外群。

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

2.3.1 TreeFam-B种子集的生成

在TreeFam-1.x中，我们直接用PhIGs含有三个或更多动物基因的类来作为TreeFam-B的种子。然而，由于PhIGs 在聚类阶段似乎使用了很严的标准，当家族的成员进化很快时，它有时会把一个基因家族分成若干小类，这样PhIGs的聚类经常偏小，是某个动物基因家族的子家族；只有几个聚类的并集才代表整个家族。在校正过程中这种现象时有发生，这也促使我们考虑在TreeFam-2中在PhIGs的基础上重新聚类。

在TreeFam-2中，PhIGs中的类按如下方式聚类。首先，原始的聚类中的序列用BLAST[1] [27]与TreeFam序列集TFSEQ相比较，期望值(E-value)的阈值设在0.01。比对上的TFSEQ再用HMMER[2] [28]再进行一遍比对搜索，期望值设在0.1；其中HMMER的模型参数是从由MUSCLE[3] [29]对原始PhIGs类中的序列进行的多序列的比对构建的。接着我们从HMMER的结果中构建一个关系图，在这个图中，每一个结点(vertex)对应着一个原始的PhIGs类，每一条加权边(weighted edge)代表着两个类包含有共同的动物基因。更准确地说，给定一个PhIGs类u，令R_u是能和u中序列匹配并且HMMER分数超过所有匹配外群基因的动物基因的集合(TFSEQ的子集)，如果$R_u \cap R_v \neq \emptyset$成立则在$u$与$v$之间加上一条边，这条边的权重为$\frac{|R_u \cap R_v|}{\min\{|R_u|,|R_v|\}}$。建好这个加权图之后，我们就可以用各种基于图论的聚类算法来进行聚类。在TreeFam-2中，我们实现了一个在Zdobnov等人 [30] 的算法基础上改进的一个半经验算法(heuristic algorithm)，这个算法和大多数同类算法一样试图找到内部联系紧密而不与其它类有联系的一个个集团 [4]。最后，聚在一起的PhIGs类被合并在一起作为新的种子。原则上这种聚类的过程可以反复进行直到结果稳定为止；但事实上我们发现两轮或更多的轮的聚类会导致较多的包含几个动物基因家族的超家族(superfamily)，这可能是因为当更多的序列混起来之后HMMER会变得不够敏感。因此我们只对PhIGs进行了一次聚类，其结果被直接用来作为种子。

[1]BLAST (Basic Local Alignment Search Tool) 是用于搜索同源序列的最受欢迎的程序。它将一条序列和一个序列集中的每一条序列进行比对，计算比对分数，并用以一条随机序列与同样大小的随机库比对时出现比观测分数更高的序列数的数学期望(E-value)来衡量同源的显著性。

[2]HMMER也是进行同源比对的工具。它用隐马尔科夫模型(Hidden Markov Model, HMM)为多序列比对建模(alignment profile)，并使用建好的模型来衡量其它序列是否和给定多序列比对相匹配。由于HMMER使用了多条序列，HMMER一般比使用成对比对的BLAST更加敏感也更准确。但它的速度较慢。

[3]MUSCLE是像Clustalw一样的多序列比对程序。它把一组序列比对到一起以使它们共有的同源区域得以匹配。一般来说，MUSLCE比Clustalw更快也更准确。

[4]事实上，MCL算法 [31]也许更加合适。

2.3.2 竞争性的分配序列到家族

在TreeFam-1中，家族成员是靠树剪枝(tree-cutting)算法来决定的。这种算法将从多细胞动物共同祖先另一个基因进化出来的同源基因(homolog)从树上删除。理想状况下，这种做法正是我们所期望的。但是在实际应用中，由于远源基因更容易丢掉以及建树算法在长枝上更容易发生误差，这个算法经常出错而使得一个TreeFam家族包含许多本不该属于这个家族的序列。这样在TreeFam-1.x中一个基因可以同时属于几个甚至几十个家族，这本身就是不自洽的，更给人工校正带来了许多麻烦。另外，由于比对和进化树是在许多不必要序列排除之前建立的，包含超过500条序列的庞大的比对经常可以见到，这极大加重了运算的负担。因此，我们决定在TreeFam-2中开发更好的方案。

在TreeFam-2中，同源基因(homolog)的搜索还是靠结合BLAST和HMMER来完成的，即先用BLAST初筛再用从MUSCLE比对建立的HMMER来精细挑选。在这步之后，每一条序列被强行分配到HMMER分数最大的那个家族中；而且如果一个家族包含一个基因的不同的可变剪切形式，只有HMMER分数最高的那个剪切形式被保留下来。在这这一步后每一个基因家族的成员已经确定了。

根据我们的实际观察，如果种子序列的质量很好HMMER方法的确很准确而有效，大多数动物基因，尤其是脊椎动物基因可以很准确的分配到基因家族中。但当种子只含有几条序列或比对结果很差时，HMMER的分数不再精确并可能导致错误。并且，无同源基因(orphan gene)和种子集的不完整性也会影响到结果：即使一个基因不属于任何已有TreeFam家族，它还是会被强行的分配，而不管BLAST和HMMER的分数有多低。在下一个版本中，我们会对这些问题予以更多的关注。

2.3.3 升级信息的跟踪

TreeFam将人工校正的信息存于TreeFam-A的种子树。然而随着序列集的升级，同一个基因的识别符(identifier)可能会改变从而在旧版本种子树(seed tree)失去踪迹。这时部分校正信息就会缺失，而如果这种信息的缺失持续发生，人工校正的努力将付诸东流。减缓这种信息流失的速度对TreeFam这样的人工校正数据库至关重要。

TreeFam-2中我们在分配家族的序列之后设计了一个专门的流程来跟踪两个版本间不同的基因命名。其基本的想法是，新的命名可以和旧的相匹配应该满足三个条件[5]：(i) 两条序列来自同一物种；(ii) 它们在相匹配的编码区几乎相同；(iii) 没有其它和两条中的任一条更近的其它序列。条件(i)和(ii)可以很容易

[5]这一策略在新的TreeFam-3中会有所改变。

的直接判断，而第三个可以从新旧版本基因混合而建的树中看到。在这棵树中如果新旧两个版本的基因有共同的父结点则它们可判断为同一条基因。图 2.2 给出了例子。

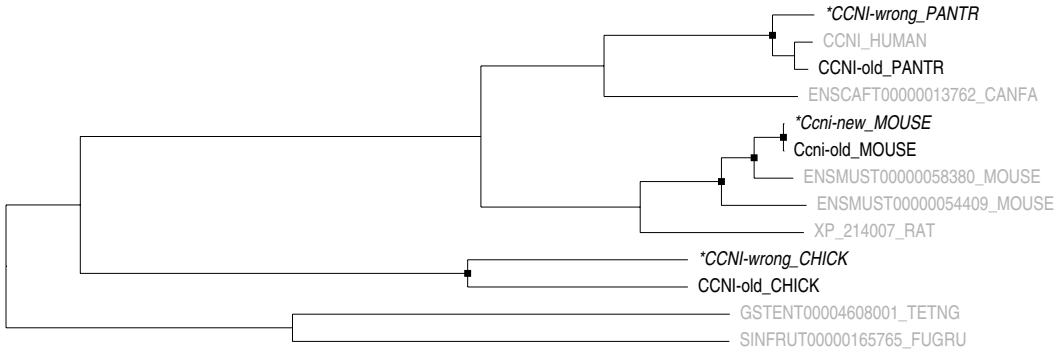

Figure 2.2: 用于解释基因标识跟踪的一个例子。这棵树既包括两个版本间没有发生标识(identifier)改变的序列(灰色字体)，也包括发生标识改变的序列(黑色字体)。只在新版本中存在的序列用斜体显示。这棵树中，序列CCNI-old_PANTR和*CCNI-wrong_PANTR 不应是同一条，这是因为CCNI_HUMAN更加靠近前者；序列CCNI-old_CHICK与*CCNI-wrong_CHICK不应为同一条，因为它们之间的距离太长；只有小鼠的两个基因在新旧版本是同一个基因。

2.3.4 进化树的构建

由于竞争法的应用，TreeFam-2的每一个家族在建树前不再包含无关的远源序列。和TreeFam-1.x相比，建树时所用的序列量要小得多，这也使得使用像极大似然法这种更加准确却更慢的算法成为可能。另外，我们在新版本中也更加频繁的使用树合并算法，这也有助于提高建树的准确性。

在新的TreeFam-B中，完整树是由同义邻接树(synonymous neighbour-joining tree，使用同义突变(synonymous mutation) 距离构建距离矩阵用邻接法(neighbour-joining)建的树) 和非同义邻接树(nonsynoymous neighbour-joining tree)通过树合并算法而得到的。在TreeFam-A中，构建完整树时还要以人工校正的种子树作为约束。至此，自动流程结束。

为了获得更准确的自动建树，我们还尝试对同义邻接树，非同义邻接树，蛋白模型下的极大似然树(maximum-likelihood tree)和核苷极大似然树应用树合并算法(第五章)建树。无论是第六章的评测还是人工校正的过程中的观察都表明这棵合并树的确超过了其它的算法。合并树将在TreeFam-3中发挥更大的作用。

Chapter 3

基因树的重构

从分子序列构造的进化树代表了进化的历史而不是衡量序列之间的相似度。历史是真实存在的，正确的树也就是真实的，好的算法必须能够准确恢复进化历史。但另一方面，历史在很大程度上是未知的，进化树的构建只能依靠现有的观测数据。尽管化石为重构物种树提供了重要证据，但由于化石的发掘很困难，它也不可能描述分子层次上的进化，重构基因树只能依靠序列的相似度。如何从序列的相似度准确恢复真实的进化历史是分子进化学永恒的主题。

这一章将讲述TreeFam是如何在给定一个多序列比对时构建基因树的。在对各种建树算法作完评论后，我们首先介绍TreeFam中第一个与众不同的算法：约束邻接法(constrained neighbour-joining)。无根树的定根和叶结点重排算法也会得以介绍，这两个主题都与建树的后处理紧密相关。在TreeFam-1.x中，我们只使用了邻接法，但自TreeFam-2.0开始，我们在其中应用了更多的优秀算法包括极大似然法和树合并算法。这些算法并未包括在本章。第五章将详细描述树合并算法，第六章将对各种算法作以评测。

3.1 标准建树算法

尽管各种进化的性质都可以用来构建进化树，分子进化学集中研究如何从多序列比对实现树的重构。多序列比对是重构进化树的起点，本章算法都以多序列比对为唯一输入。在第六章中，辅助的信息才会用于建树。

建树算法可以分为四类：距离法，简约法，似然法和Bayes方法。每一种算法都有各自的长处和弱点。在仔细的进化分析中，所有算法都该尝试以得到一棵可靠的进化树。在第六章的评测中，这一点也十分明显。

3.1.1 距离法

距离法事实上是一大类从距离矩阵(d_{ij})建树的算法的总称。这个距离矩阵的每一个元素衡量了一对序列i与j之间的某种进化距离。这种距离一般反映为多序列比对中序列的相似度，它可以是比对的分数，错配的个数(又称做p-distance)，或者从某种统计模型中估计出的残基替换数。

在所有距离法中，UPGMA [32]是最早的。它模仿了系统聚类(hierarchical clustering)的过程，每一步将距离最短的一对联结起来。UPGMA方法假定序列始终以相同的速度进化(分子钟假设, molecular clock hypothesis)，但这并不总是事实；以后的算法一般也不再强行要求这个假设。在1967年，Cavalli-Sforza和Edwards [33]提出正确进化树中任何两叶结点树上的路径长度(即连接两结点路径上的枝长和)应该尽可能与距离矩阵给出的距离相近，这就是建树的最小二乘算法(Least Square, LS)。但构建LS树需要遍历整个树空间以搜索最优树，这种计算量是难以接受的。随后Fitch和Margoliash [34]使用近似简化了最小二乘算法并给出了一个实际中可行的算法。距离法发展的突破发生在1987年，这一年Saitou和Nei [35]提出了邻接法(neighbour-joining, NJ)。邻接法不像UPGMA每步连接一对最近的结点而是连接最小进化(minimum evolution, ME[1])[36]意义下的一对近邻(neighbour)，从而不再需要分子钟的存在。邻接法简单、准确、高效，是最受欢迎的建树算法之一。在邻接法出现10年之后，BIONJ [37]和Weighbor [38]算法也被相继提出，它们都属于邻接法的变形，只不过在连接结点的过程有所改善。事实上，邻接法是最小进化的贪心近似，即它在每一步遵从ME的规则，但并不保证能构造出在全局意义上的ME树。最近，Desper和Gascuel [39]提出了平衡最小进化(balanced minimum evolution, BME)的理论框架，在这种理论下，ME树可以快速而准确的构建。他们提出的算法声称比邻接法及其变种都更准、更快。第六章的评测也部分证实了这一点。

距离法之所以受欢迎是因为它的速度，它是构建超过1000条序列进化树唯一的备选。然而，人们经常批评距离法在建树时丢掉了许多信息——两条序列每一个位点间的替换被压缩成了一个数字。这种压缩造成的信息丢失不但影响了它的准确性，也限制了距离法的进一步应用。比如，使用距离法不能推测祖先序列，也不能估计各种进化参数。在这些情况下我们只能依靠其它方法。

[1]最小进化认为进化倾向于枝长和最短的树，某种程度上它很像下一小节介绍的最大简约原理。Rzhetsky和Nei证明 [36]如果一对序列间进化距离的估计在统计上是无偏的并且树枝长是由最小二乘估计的，则最小进化原理总能构造出正确的进化树。

3.1.2 最大简约法

给定一棵进化树和一个多序列比对，我们能沿着进化树给出的进化史计算出能够解释给定数据的最小替换数；对不同的树这个最小替换数也不相同。最大简约法(Maximum parsimony, MP) [40]则是在树空间中寻找使最小替换数最小的树作为重构树。

基于奥卡姆剃刀(Ockham's razor)原则——预测时简单的理论通常是正确的——简约法以其最简单的假设形式而著称：正确的树含有最少的替换数。它的使用不依赖于任何进化模型。然而，尽管简约法的优点在多数生物学家眼中很有吸引力，统计学家则有另外的看法。他们批评简约法在出现长枝时不能构建正确的树 [41]；他们声称简约法事实上假定了一个进化速率在位点及时间跨度上都相同而不切实际的理论模型 [42]；他们也指出简约法也不能像似然法那样给出更多的信息。虽然面临各种各样的问题，简约法依然为生物学家所爱，并在实用中显示了它的准确性 [43]。

3.1.3 极大似然法和Bayes方法

极大似然法(maximum-likelihood, ML) [44]在树空间中搜索在一个指定进化模型下使得该模型下概率最大的那棵树为重构树。如果不考虑效率上的问题，极大似然法的过程其实很简单：对每一棵树计算概率，再从整个树空间中挑选概率最大的那棵树。极大似然给出了一个理论框架，它的实际应用要依靠各种进化模型，由于进化模型的设计可以十分灵活，似然法也就十分灵活而多产，各种进化信息都可以在统计的框架下自然的推断出来。另外，似然法也被认为是最准确的建树算法 [45, 43]。多数分子进化学家会承认极大似然法的出现在系统进化这个领域是一次革命。

然而，似然法的使用需要极大的计算量。即使在多种改进算法提出来之后 [46, 47, 48] 建一棵有几百个叶结点的树还是极为困难的，更不要说对树进行多次自展(bootstrapping) 检验(第3.4节)了。另外，尽管极大似然法允许各种进化模型的使用，但用户一般只能使用软件预先内部设定好的模型而不能方便的创建模型或将各种模型相结合。这阻碍了极大似然法在更复杂进化中的应用。

Bayes方法可以看做是极大似然法的一种变形。它不是像似然法那样主动遍历整个树空间来寻找极大化概率的树，而是通过MCMC(Markov Chain Monte Carlo)来采样指定模型能产生的进化树，进而找到极大化后验概率的那棵树 [49]。Bayes方法最大的优点是它无与伦比的灵活性。由于它生成树而不是对指定树计算概率，计算树的概率时涉及的许多复杂的解析计算可以得以避免。这样我们可以在Bayes框架下方便的使用或结合各种复杂的模型并从采样的树

上自然的推断各种进化参数，也有可能将更合理的先验分布加入模型。另外，在MCMC过程中的采样树在某种程度上也可以看成是自展树[50]。通过比较这些采样树我们也可以方便的计算显著值而不需要大量的计算资源。不过，要注意的是这些值往往过高于(excessively high)正确的显著值 [51, 52, 53]。

3.2 约束邻接法

所有的传统建树方法都是无监督(unsupervised)算法，它们从来无法利用人提供的已知信息。即使我们知道一簇叶结点A是和另一簇B分开的，在重构树中它们可能还被混在了一起。然而在TreeFam中，我们必须能够在建完整树(full tree)时保留种子树(seed tree)的拓扑，或者说校正者的知识。这是一个带监督(supervised)的过程。我们必须设计新的算法来实现这一要求，否则如果完整树不能利用校正的拓扑，专家的工作就会付之东流。在这一节我们介绍TreeFam的第一个关键算法，约束邻接法。它能够把新的序列加到已知树中而不改变已知拓扑。

3.2.1 标准邻接法

如果存在一棵树使得树上任何一对叶结点间路径长度等于距离矩阵(distance matrix)给出的长度，那么这个距离矩阵称为可加的(additive)。反过来，如果一个距离矩阵是可加的，则必存在使得可加性成立的那棵树。邻接法 [35]就是用于找到这棵树的。

在运用邻接法时，每一步都有一个连接一对近邻(neighbour)的新的内结点被加入树中；而给定一个对称的矩阵d_{ij}，一对近邻对是所有可能的(i,j)中使得D_{ij}最小的那对叶结点，其中D_{ij}定义为：

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

当连接一对近邻(I,J)的新结点K被加入后，它们之间的距离棵计算为：

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

而K和所有其它结点的距离为：

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

随着I和J的删除和K的加入，原来的$n \times n$矩阵化为$(n-1) \times (n-1)$的矩阵。至此一轮迭代结束。这个过程反复进行，直到矩阵仅为3×3时中止[2]；遵循连接的过程我们就可以得到一棵二岔无根树，这就是邻接树。如果距离矩阵满足可加性(additivity)，这棵树就是使得可加性成立的那棵树。Durbin等人的书 [2]中给出了证明。

在每一步邻接法将最小化D_{ij}的一对叶结点相连，而不是将最小化d_{ij}的一对相连，这是邻接法与UPGMA的主要区别之一。分子钟假设(molecular clock hypothesis)的失效常以出现长枝为特征，而当长枝出现时UPGMA方法就容易出现错误。最小化D_{ij}可以避免这样的错误连接，这是邻接法的长处。要注意的一点是邻接法要求举例矩阵是可加的，尽管真实数据很少满足可加性，但如果差别不大，邻接法也可以准确重构正确的进化树。

3.2.2 约束邻接法

约束邻接法总体思想是很简单的，它与普通邻接法唯一的不同是约束邻接法禁止违反约束树拓扑的连接；但由于在每一步所有的叶结点对都要参与判断，判断允许连接必须足够高效。事实上，如果我们注意到一次允许连接只能发生在据有相同父结点的两个叶结点之间，我们就可以在每一次使用一个约束时收缩约束树(constraining tree)，将已经使用的约束删除而把公用的父结点作为一个新的叶结点。由于判断两叶子是否具有共同父结点可在$O(1)$实现，判断是否是允许联结可以也在$O(1)$时间完成。因此约束邻接法和普通邻接法有着相同的时间复杂度$O(N^3)$。图3.1 给出了一个具体例子。算法的详细描述参见表3.1，这张表也给出了标准邻接法。

3.2.3 进一步的讨论

事实上，约束邻接法并不遵循“邻接”这个规则。它不能保证一对近邻(neighbour)总能被联结。这是因为约束邻接法总是假定约束树是一棵有根树，它不会考虑其它有根拓扑而强行的按照给定树结点的后缀顺序(suffix order)[3]来进行连接。为了解释这一点，我们来看一个例子。假定我们有一棵树`((1,2),3,4)`由标准的NJ方法构建。从这个结果上看，唯一的一次连接发生在`1`和它的近邻`2`之间。现在我们继续假定定根后这棵树为`(((3,4),2),1)`并用这个有根树作为约束。那么在应用上述约束邻接法时，尽管真正的近邻`(1,2)`在

[2]因为无根二岔树的每一个内结点都有三个相连接的边

[3]如果以后缀顺序来遍历一棵有根树的所有结点，较低的结点总会先于较高的结点被遍历到。

无根拓扑中是符合约束的，(3,4)也一定会被先连接起来。因此，约束邻接法会打乱连接的顺序而破坏邻接原则。

如果距离矩阵严格满足可加性，或者说如果两个叶结点间路径的长度总与矩阵中的距离相等，那么连接结点的顺序并不重要，无论以怎样的顺序联结都会得到完全相同的树。但在现实中，严格的可加性很少满足，即使把一棵标准的NJ树作为约束，约束邻接法经常会生成一棵拓扑相同而枝长不同的树。

出于这个原因，我们还开发了另一种可以以无根树为约束的约束邻接法。

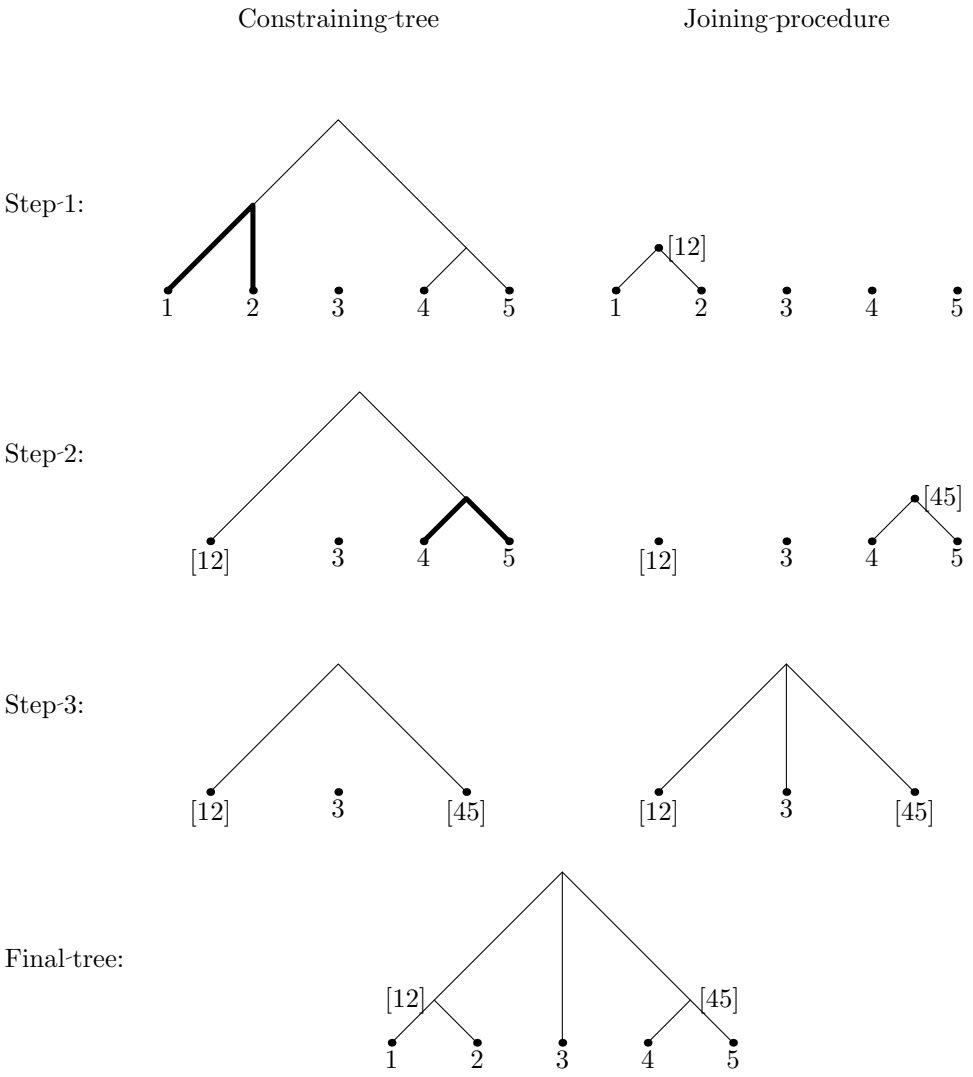

Figure 3.1: 约束邻接法的一个例子。在前三排，左边的树是在本步骤的约束树，右边的是剩余的结点和最好邻接。在第三步只剩下三个结点，它们会被连在一起而形成一个三分岔点。最后的结果显示于最后一排。要注意的是，结点‘3’没有出现在约束树中，如果在某一步有结点和它近邻(neighbour)，它可以自由的和‘3’结点邻接。

在前面给出的例子中，新的算法能够正确的把近邻(1,2)相联结。这也是一个$O(N^3)$的算法，但它的不足是只能应用于二岔约束树，否则它时间复杂度就

Table 3.1: 约束邻接法。除以'$*$'开头的行就可以得到标准的邻接法。

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 $|\mathcal{L}| > 2$ **do**
  for each node $i \in \mathcal{L}$ **do**
    $r_i \leftarrow \frac{1}{|\mathcal{L}-2|} \sum_{k \in \mathcal{L}} d_{ik}$
  for each node pair $(i, j) \in \mathcal{L} \times \mathcal{L}$ **do**
    $M \leftarrow \infty$
    $D_{ij} \leftarrow d_{ij} - (r_i + r_j)$
    if $D_{ij} < M$ **then**
$*$ **if** $\{i, j\} \not\subset \mathcal{L}_C$ **OR** $\mathrm{par}_C(i) = \mathrm{par}_C(j)$ **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** $\mathrm{chi}(\mathrm{par}_C(I)) = \{I, J\}$ **then**
$*$ $\mathrm{par}_C(K) \leftarrow \mathrm{par}_C(\mathrm{par}_C(I))$
$*$ **else if** $I \in \mathcal{L}_C$ **then**
$*$ $\mathrm{par}_C(K) \leftarrow \mathrm{par}_C(I)$
$*$ **else if** $J \in \mathcal{L}_C$ **then**
$*$ $\mathrm{par}_C(K) \leftarrow \mathrm{par}_C(J)$
$*$ $\mathcal{L}_C \leftarrow (\mathcal{L}_C \cup \{K\}) \setminus \{I, J\}$
add a new node K that joins the remaining three nodes in $\mathcal{L}$

会增加。这个算法比前面给出的复杂得多，我们这里不再讨论这个算法。

3.3 树的定根

如果我们认为地球上所有物种都只有一个共同祖先，那么所有的进化树都是有根的，即进化是有方向的。但是，常用的邻接法和极大似然法只能构造无根树，因此我们必须有算法来为无根树定根。定根同建树一样重要，一个错误的根描述着不同的进化历程。

定根有三种方法。最常用的一种是在参与建树的叶结点中选取一个外群，然后把根直接定在连接这个外群序列的枝上。但是，在缺少充分生物证据的情况下，外群的选择很困难，尤其是对于那些复杂的基因家族。为TreeFam的每一个基因家族都指定一个外群是不切实际的，因此该方法未在TreeFam中使用。

第二种方法是从所有有根拓扑中选择高度最矮的有根树，这相当于把根定在叶结点间最长路径的中点。如果序列的进化基本符合分子钟条件，这种算法能够给出正确的根 [2]。Huelsenbeck [54]等人又进一步将这种方法纳入了Bayes理论的框架中。

更好的是第三种算法，这也是TreeFam所使用的算法。这种算法通过选择和物种树尽可能相像的有根树来定根 [55]，而在下面一章我们会看到，这种物种树和基因树的相似性可以从基因树上推断出的最简约倍增/缺失事件数来判断，越小则越相似。如果参与建树的序列确是属于一个基因家族的序列并且在这个家族没有发生很多缺失事件，这个方法一般都能找到正确的树根。值得注意的是倍增和缺失事件都应该被考虑用来定根；只数倍增数经常会导致许多具有相同倍增数的有根拓扑 [56]。

3.4 自展检验

从统计的角度，用于建树的序列可以看成是符合某种进化模型的所有序列的抽样，因此重构树在某种程度上也是一个随机变量，检验这个随机变量的稳定性，即检验一个拓扑是否易受随机事件影响，对重构进化树至关重要。

有几种方法都可以用来检验树的稳定性，但使用最广泛的是自展法(bootstrap)[57, 58]。自展法的输入是一个多序列比对$(X_{ij})_{n\times m}$ 和从这个比对建的一棵树T，其中X_{ij}是比对中第i条序列第j个位置上的残基；自展法的输出是T中的一条枝出现在从X随机抽样生成的比对建得的树的频率。具体来说，在一轮自展中，我们从比对(X_{ij})的m列位点中带放回的重复抽样m次，每次抽一列，并将抽样出的m列位点组成一个新的比对(称做重抽样比对)(X^r_{ij})。我们

再用X^r建树得到T^r(称为重抽样树)，并比较T和T^r：对T的每一条枝$A|B \in \tilde{T}$，如果$A|B$也出现在T^r中，则在枝$A|B$的计数上累加1；否则不加。当我们把这个过程重复比如说1000次后，枝$A|B$上的计数就称为自展值，越大则越好。

更严格地说，给定一个比对$(X_{ij})_{n\times m}$和从(X)上建的树T，重抽样比对是这样一个随机矩阵：

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

其中随机变量$J_j,\ j = 1,\ldots,m$服从$[1,m]$上的离散均匀分布，即对所有$k = 1,\ldots,m$，$\Pr\{J_j = k\} = \frac{1}{m}$。令$T^r$是从$X^r$中建的树，则$T$一条边$A|B \in \tilde{T}$的自展支持为：

$$\mathcal{B}(A|B) = \Pr\{\text{二分}A|B \in \tilde{T}^r\} \tag{3.2}$$

经过变形，如上描述的的自展过程的计算也可以对有根树计算，但是由于定根的困难，这种用法比较少见。另外，自展法也可以用于关于树的各种统计量的计算，基本思想是类似的。

3.5 叶结点重排算法

从数学上说，树是一个抽象的实体，它的叶结点组成一个集合而没有顺序的说法；但是当把树画在纸上，我们必须把叶子排成一定的顺序。尽管无论这个顺序怎样画在平面上的树在数学上说都是相同的(identical)，但这些树在人的眼中可能大不相同。同一棵树按不同叶子顺序画两遍视觉上可能根本无法分辨；两棵大树进行比较，找到不一致的分枝经常是极为困难的。怎样将树按照指定的方式画出来对树的显示十分重要。

在这一小节将描述怎样将树按照一个指定的方式画出来，或者说怎样将叶结点按指定的方式排列，使得相似的树在视觉上总是相似的。为了达到这一目的，我们要为每一个叶结点赋予一个权重，并设计一个目标函数使得在它得到优化时，权重小的叶结点倾向于排列在权重大的结点之前。这个优化过程可以用一个直观而又准确的$O(N)$算法实现。

简单起见，这一小节只讨论有根二岔树。对一般有根树的推广其实也很容易。

3.5.1 叶结点重排问题及算法

给定一棵有n个叶结点的有根二岔树G，即$|V_E(G)| = n$，定义G的一个顺序(order)为一个一一映射$I : V_E(G) \to \{1,\ldots,n\}$，使得存在一个NH字串(NH string)满足$v \in V_E(G)$出现在字串中的第$I(v)$个位置上。如果$G$是一个二岔树，

共存在有2^{n-1}个不同的顺序，代表着在$n-1$个内结点上不同的两枝交换。如前面所讨论的(Section 1.4.1)，NH字串和树画在纸上的图形有着自然的一一对应关系，一个顺序I事实上决定了树应该以怎样的方式画在纸上(见图3.2)。如果我们进一步赋予每个叶结点v一个权重$W(v) \in \Re$，叶结点重排问题指找到一个顺序I使得如下目标函数最小：

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

其中$\alpha > 0$是一个常数。由于I是一个一一映射，则I^{-1}必然存在，我们就可以将方程3.3 变形为：

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

这样，叶结点重排问题就是寻找一个I使得下面式子极大：

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

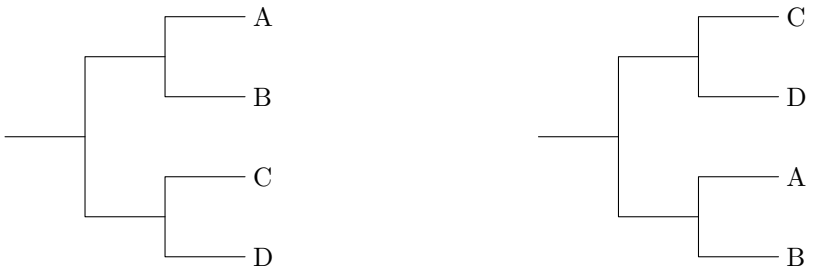

Figure 3.2: 叶结点的顺序。图上显示的两个树据有相同的拓扑但有着不同的顺序(order)。左边的树的NH字串为`((A,B),(C,D))`，因此$I(A) = 1$，$I(B) = 2$，$I(C) = 3$以及$I(D) = 4$；在右边，NH字串为`((C,D),(A,B))`因而$I'(A) = 3$，$I'(B) = 4$，$I'(C) = 1$和$I'(D) = 2$。

叶结点重排问题最直接的算法是枚举所有可能I，但这个算法无疑是最差的。事实上，不同结点的分枝交换是相互独立的，也就是说如果在一个结点上的分枝交换能够降低目标函数(方程3.3)的值，它就总能降低这个值而不管在其它结点如何交换分枝。因此，我们总能应用$n-1$次独立的分枝交换来找到最优的I^*。现在，剩下的问题就是如何在一个结点上选择合适的交换型。这个规则其实也很简单：在一个内结点，交换分枝使得它左子树上平均叶结点权重小于它右子树上的平均权重。我们来证明这一点。

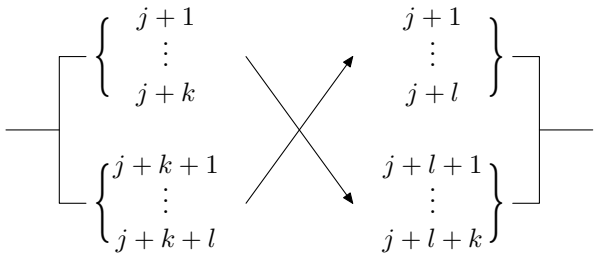

Figure 3.3: 左右树交换后的两个顺序。交换之后在左边树中第$(j+1)$位置上的叶子出现在右边树第$(j+l+1)$位置上；而第$(j+k+1)$个叶子出现在了$(j+1)$。数学化一些，当$i=1\ldots k$时，$I'(I^{-1}(j+i))=j+l+i$，而当$i=1\ldots l$时$I'(I^{-1}(j+k+i))=j+i$。

图3.3给出了两个顺序：交换前的I和交换后的I'。令$w_i \equiv W(I^{-1}(i))$以及$w'_i \equiv W(I'^{-1}(i))$，$i=1,\ldots,n$，则$w_i$是在顺序$I$下的第$i$个叶结点权重，而$w'_i$是$I'$下第$i$个叶结点的权重。从图中给出的叶子顺序的变换我们知道下列关系成立：

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

这样我们就可以在交换前后比较目标函数3.4。

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

在这个式子中，$\frac{1}{k}\sum_{i=1}^{k} w_{i+j}$是交换前左子树平均叶结点权重，而$\frac{1}{l}\sum_{i=1}^{l} w_{i+j+k}$是右子树的平均权重。很明显，只有当左子树平均权重更大时原来的树才需要交换分枝。这就证明了上一段中提出的命题，进而证明了叶结点重排算法。

3.5.2 权重函数的定义

权重函数$W(\cdot)$决定了叶结点该如何重排。TreeFam目前使用两种权重函数。第一种基于一棵已经定好顺序(order)的物种树，第二种则基于已定顺序的基因树。

假定我们有一棵物种树S及其顺序$I_s : V_E(S) \to \{1, \ldots, |V_E(S)|\}$，叶结点$g \in V_E(G)$的权重可定义为：

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

其中$M(g) \in V_E(G)$是基因g的物种。这样的一个权重函数倾向于将基因树G的叶结点排列得类似于物种树。

叶结点重排也有助于比较从同一个序列集构造出来的两棵进化树。假设我们有一棵树G_0和它的一个顺序I_0，我们想把用另一种算法构建的树G_1按相似的形状画出来。我们可以定义：

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

令$\alpha = 1$，当两棵树相同时目标函数3.3会等于零，那么两棵树就会以完全相同的方式在视觉上表现出来。但要注意的是如果两棵树不是很不相同，目标函数3.3也可能为零。这是因为这样的一个目标函数只考虑了叶结点而没有考虑内结点的拓扑(图3.4)。要看两棵树是否据有相同拓扑应该去比较一棵树的每一分枝是否在另一棵树中出现。第一章已经给出了详细的说明。

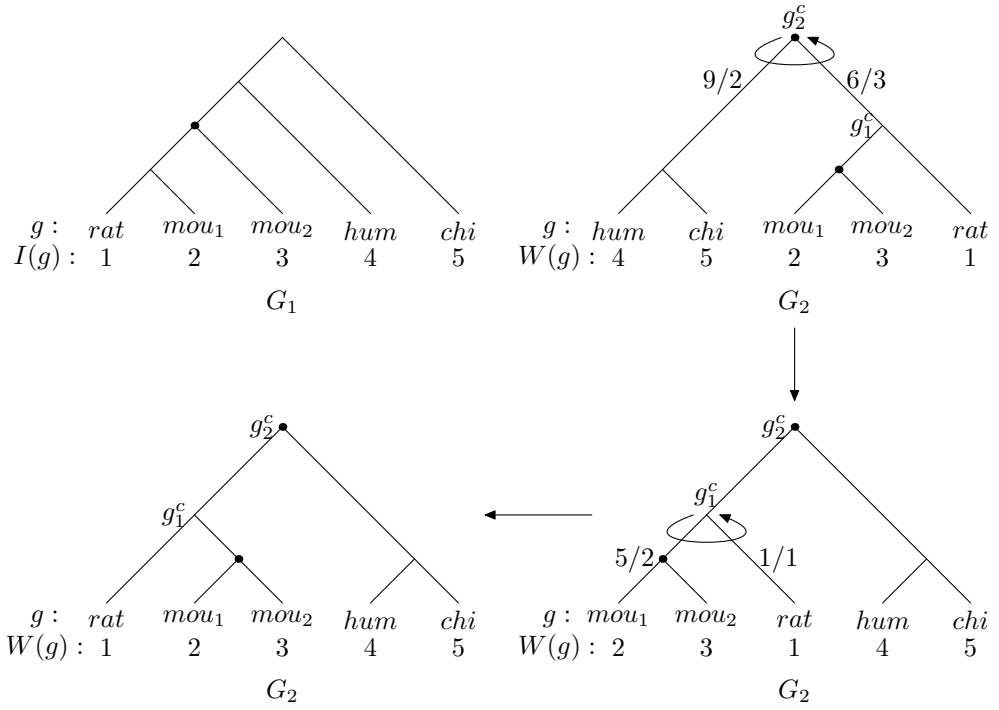

Figure 3.4: 叶结点重排算法的一个例子。树G_1是一棵带顺序的树。我们的目的是将G_2的叶结点排列成与G_1相似的顺序。右上方的树显示了G_2原来的顺序。在g_2^c结点，左子树的平均权重为4.5比右子树的权重大，在这个结点交换左右子树我们得到右下方的树形。而在结点g_1^c，左子树的平均权重也大于右子树，我们交换分枝而得到最终的结果(左下方)。这个例子中，重排后G_2的叶子和G_1的出现顺序相同，尽管它们拓扑不同。

Chapter 4

基因倍增和缺失的推断

真核基因家族的的进化由三个因素决定：物种分化(speciation)、基因倍增(duplication)与基因缺失(loss)。首先，物种分化的同时也使基因分化到不同物种，每一个基因家族的历史都反映着物种进化的历史：当一个祖先物种(ancestral species)分化了，它所有的基因都会分化到它的后代并在后代中相对独立的发展。物种分化不能产生新的基因，但基因倍增为一个物种及其后代注入新的基因，从而提供可供自然选择的备选基因。当多余基因产生后，基因缺失接着抹掉对物种功能帮助不大的新基因。基因演化、产生和消亡，描绘了基因家族的进化史[1]。图 4.1给出了一个例子以说明倍增和缺失事件是怎样控制基因的进化的。如果我们接受树中给出的推断，那么这个基因家族进化的历史是这样的：最早的祖先物种*Chordata*只有一个基因；这个基因在进化到*Euteleostomi*时发生了一次复制；自此这个基因在*Euteleostomi*都有了两个拷贝；第一个拷贝在后续物种中正常进化，只在*mouse*中接连发生两次复制；另一个则在*Eutheria*发生了一次缺失。这就是这棵树所描绘的进化历程。

基因倍增和缺失是通过比较基因树和物种树得出来的，这一过程通常叫做“树融合”(tree reconciliation)。树融合最早由Goodman [63]提出，之后由许多学者相继实现并提高 [64, 65, 66]。最近Zmasek和Eddy [55]给出了一个简洁优美的算法极大简化了树融合过程，而Dufayard等人又将之推广至物种树含有多分叉的情形，尽管他们没有交代细节。以上的各种算法都属于简约法，给定基因树与物种树它们总是作出倍增与缺失数最小的推断。最近，Arvestad等人 [67]设计了Bayes算法来推断倍增与缺失，在两个例子上他们表明这种基于统计的算法可以有效的减少没有推断出来的倍增与缺失事件。

在这一章中，我们将把倍增/缺失推断至于更一般的理论框架并给出一个新

[1]水平基因迁移(Lateral Gene Transfer, or LGT)也是影响基因进化的因素之一，但是LGT经常发生于病毒或原核生物之间，而很少发生于高等多细胞生物 [59, 60, 61, 62]

的实现算法。我们的算法基于Zmasek和Eddy的工作，但允许多分叉的物种树。

4.1 物种映射

为了描述基因倍增/缺失分析，我们需要在物种树和基因树之间建立一个桥梁：物种映射(species map)。物种映射将一个现有基因或祖先基因映射到拥有这个基因的物种(现有或祖先物种)。当我们知道了每一个基因属于哪个物种，我们就能将物种进化和基因进化相接合，重现历史进而实现推断。

我们用数学的记号来描述这一过程。令G是一个基因树，S是物种树。这两者都是有根树。给定一个现有基因$g \in V_E(G)$，令$s_g \in V_E(S)$是含有基因g的现有物种。那么，物种映射$M : V(G) \to V(S)$ 对每一个$g \in V(G)$满足：(i) 如果$g \in V_E(G)$，则$M(g) = s_g$；(ii) 对所有$g' \in \omega_G(g)$，$s_{g'} \leq M(g)$；(iii) 如果$g < g'$，则$M(g) \leq M(g')$。物种$M(g)$是包含有基因g的物种。映射M事实上将一棵物种树嵌入基因树，它体现了物种进化是如何影响基因进化的。在下面的章节中，我们将看到仅依靠物种映射M我们就能推断出历史上所有的倍增与复制事件。因此物种映射决定着整个推断过程，它是本章的关键。

给定一棵物种树和基因树，许多映射都满足如上两个条件。极大似然法选择使某种模型下概率最大的那个映射。这个过程十分耗时。与之不同的是简约法简单的选择最大简约物种映射M^*来进行推断，最大简约映射总是推断出最小数量的倍增/缺失事件。在本文我们不加证明的指出M^*可以如下构造：

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

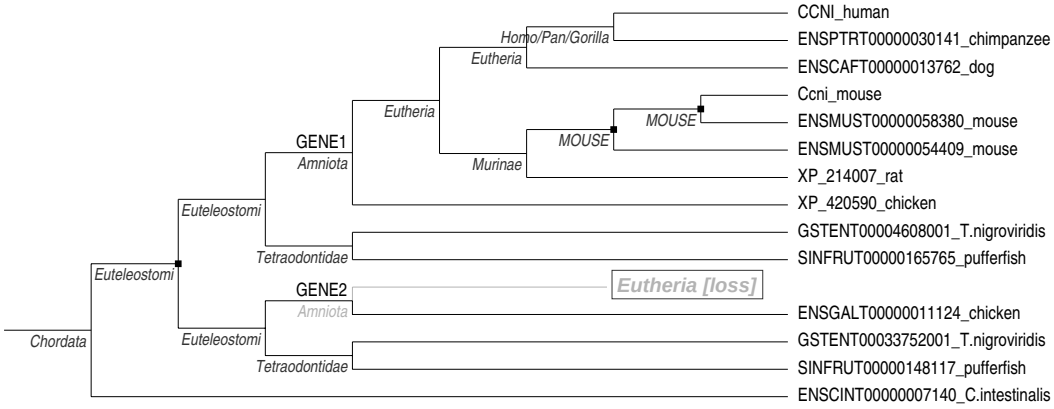

Figure 4.1: 基因进化的一个例子。在内结点边上斜体的名字是包含祖先基因的祖先物种。倍增事件(三个粗体点)和缺失(灰色方框)是用本章下面介绍的方法推断的。

很明显，M^*是一个物种映射，它其实是可能包含基因g的据今最近的物种。这也是M^*被命名为简约物种映射的原因之一。图4.2给出了简约映射和非简约映射的例子。

4.2 基因倍增/缺失的推断(DLI)

在理想情况下，基因的倍增可以通过观察一个物种的基因是否出现多次而得到。如果没有基因缺失的干扰，这是可以做到的。但是如果缺失发生，这种简单的策略就会漏掉部分倍增事件。我们在推断时必须要恢复出丢失的基因才能得到正确的结果。对一棵二分岔的物种树，我们可以简单的比较$M(g)$和$M(\mathrm{par}(g))$ [55]，若$M(g) = M(\mathrm{par}(g))$成立，则在$\mathrm{par}(g)$发生一次复制。但对一棵多岔物种树，这种做法会多估计出一些倍增事件。图 4.3给出了一

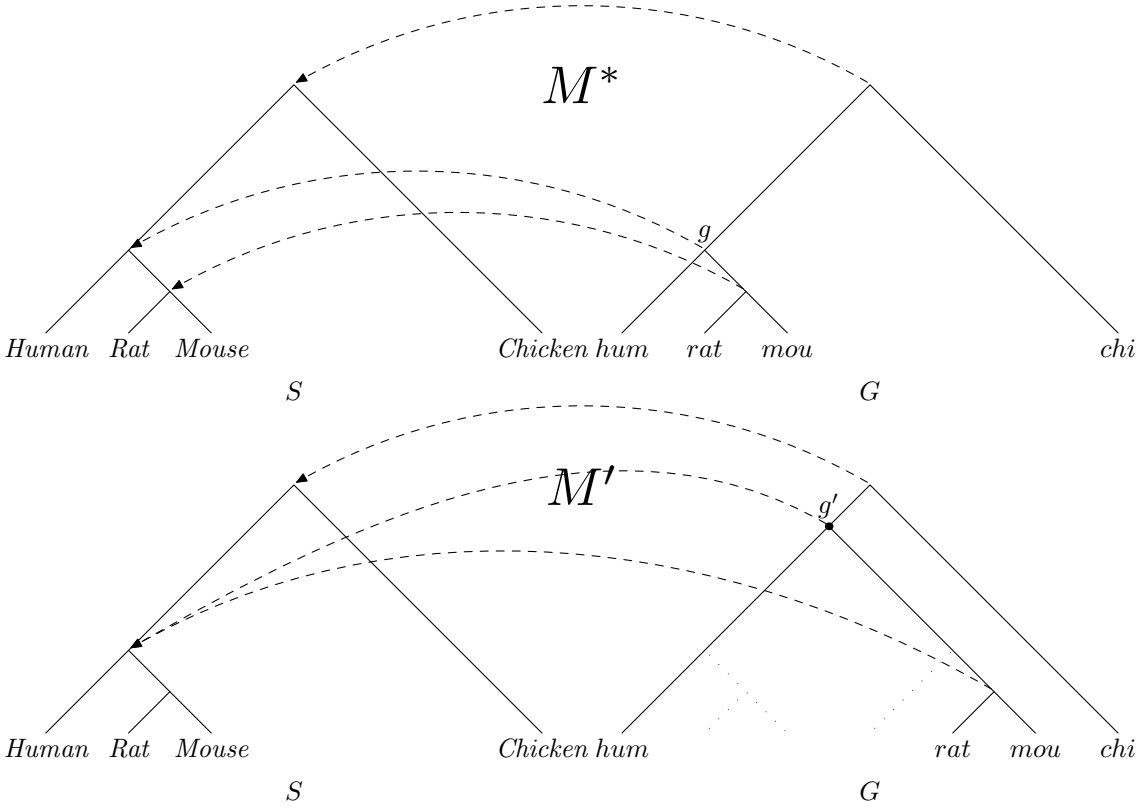

Figure 4.2: 说明物种映射的例子。上面的图显示了物种树S和基因树G之间最简约物种映射M^*。在这个映射下，每个基因都正常演化而没有发生过倍增或缺失事件。下面的图显示了另一个可能的物种映射M'。如果这个物种映射是事实，这个基因家族至少经历了一次倍增(在结点g')和两次缺失(图中的虚线)。在这个例子中，枝长没有任何意义，仅为作图方便。

个例子。在这个例子中，g和$g_p \equiv \mathrm{par}(g)$都被物种映射映射到*Eutheria*，但$g_p$还是一个分化节点，而不是倍增节点。为了将倍增/缺失推断推广到多分叉物种树的情形，我们必须显式的恢复在基因树中缺失的物种。

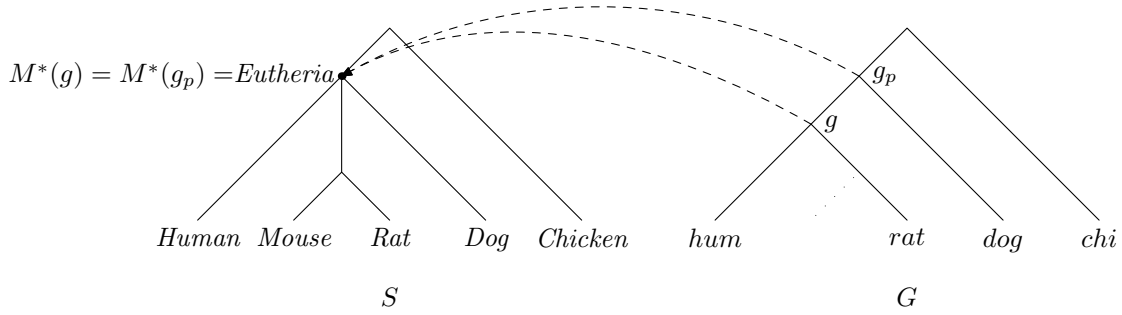

Figure 4.3: 说明$\sigma(g)$作用的例子。左边是一棵有多分岔点的物种树S。右边是带有一个缺失的基因树G。结点$g \in V(G)$在最简约物种映射下映射于$M^*(g)$，而g_p映射于$M^*(g_p) = M^*(g)$。在这个例子中$\sigma'(g) = \{Human, Rat\}$，$\sigma(g) = \{Human, Mouse, Rat\}$以及$\omega_S(M^*(g)) = \{Human, Mouse, Rat, Dog\}$。这三个集合彼此不同：由于小鼠(mouse)有一个基因缺失，$\sigma'(g)$比$\sigma(g)$小；而由于$M^*(g)$是一个三分岔点，$\omega_S(M^*(g))$比$\sigma(g)$大。

为了达到这一点，我们引入两个新的集合：

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

以及

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

集合$\sigma'(g)$只包含可从树中直接观测到的现有物种，而方程4.3的第二个集合中还包含了在$G|_g$中缺失的物种。因此集合$\sigma(g)$包含所有无缺失情况下$G|_g$中应该出现的物种。

尽管集合$\sigma(g)$的定义看起来十分复杂，但这是必要的。这一点可以从$\sigma'(g)$、$\sigma(g)$和$\omega_S(M(g))$这三个集合的关系看出来：$\sigma'(g) \subset \sigma(g) \subset \omega_S(M(g))$总是成立的，并且当无缺失发生时$\sigma'(g) = \sigma(g)$，而当$M(g)$是二岔节点时$\sigma(g) = \omega_S(M(g))$成立。图 4.3说明了这三个集合的不同。我们在下文会看到$\sigma(g)$在倍增/缺失推断中发挥着重要作用。

4.2.1 基因倍增和直系同源基因的推断

如果在$g \in V_I(G)$发生了一次倍增事件而没有发生后继的缺失，我们总会看到$\sigma'(g_1) = \sigma'(g_2)$ $(\mathrm{chi}(g) = \{g_1, g_2\})$。但是，如果缺失发生了，$\sigma'(g_1) \cap \sigma'(g_2) =$

$\emptyset$也可能会成立；在这时，$\sigma(g)$的作用就发挥出来了，这是因为它不但包括了能在树中观测到的物种，也包含了缺失的物种。从数学上说，如果$\sigma(g_1) \cap \sigma(g_2) \neq \emptyset$成立，则说结点$g$是一个倍增结点(duplication node)或说在结点g发生了倍增事件。图4.4给出了例子。

如果不同物种的两个基因在这两个物种的最近共同祖先中是一个基因，则称这两个基因为直系同源基因(ortholog) [17]；相应的，在一棵基因树中，两个现有基因g_1和g_2是直系同源基因应满足条件：$\mathrm{lca}(g_1, g_2)$不是一个倍增结点。以上这个简单的条件如实的捕捉了生物学上给出的定义。

4.2.2 基因缺失的推断

下面我们定义一个缺失集$\mathrm{loss}(g) \subset V(S)$，这个集合包含所有在$g_p \equiv \mathrm{par}(g)$进化到$g$ 时缺失的物种。值得注意的是，在这种定义下，$\mathrm{loss}(g)$是定义在边(g, g_p)上的，在一个结点上的推断不会影响到其它结点。一棵基因树所包含的总缺失数为对所有$|\mathrm{loss}(g)|$的求和。

由于物种分化和基因倍增代表着不同的生物学事件，g_p是否为一个倍增结点会使得推断有所不同。当g_p是一个倍增结点时，子树$G|_{g_p}$和$G|_g$应该含有同

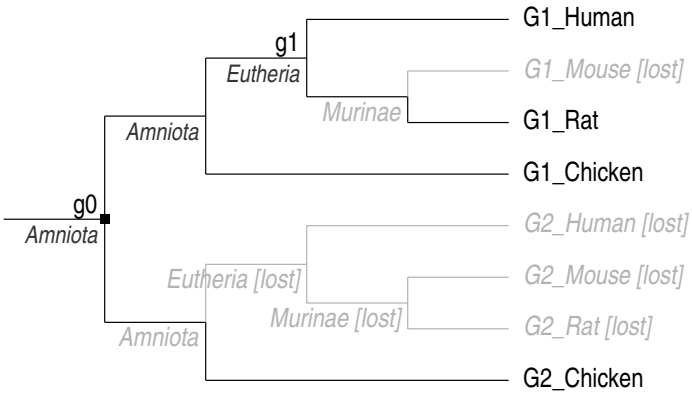

Figure 4.4: 基因倍增/缺失推断的具体例子。只有g_0是一个倍增结点。每一个内结点旁边的斜体字给出了对应祖先基因的祖先物种。灰色字体和边显示了由于基因缺失而不能在现在观测到的基因。在这棵基因树上，一些σ集合如下：$\sigma(\mathsf{G1\_rat}) = \{Rat\}$, $\sigma(\mathsf{G2\_chicken}) = \{Chicken\}$, $\sigma(g_0) = \{Huamn, Mouse, Rat, Chick\}$以及$\sigma(g_1) = \{Human, Mouse, Rat\}$。节点$g_1$是分化并因此在G1_rat和$g_1$间发生的缺失为$\{Mouse\}$；$g_0$是一个倍增节点因而G2_chicken和$g_0$间发生缺失的现有物种为$\sigma(g_0) \setminus \sigma(\mathsf{G2_chicken}) = \{Human, Mouse, Rat\}$，这其实由一次缺失$\{Eutheria\}$这一个缺失事件引起。

样的物种；这一条件的违背意味着缺失的发生，$s \in \sigma(g_p) \setminus \sigma(g)$包含所有发生基因缺失的物种。如果$g_p$是一个分化结点，除了$s \in \sigma(g_p) \setminus \sigma(g)$这个条件，一个发生缺失的物种$s$还应出现在$M(g_p)$包含$M(g)$的子树内，或等价的，$s$必须满足$\mathrm{lca}(s, M(g)) < M(g_p)$。令$\mathrm{loss}'(g)$为$g_p$进化到$g$发生缺失的物种，则$\mathrm{loss}'(g)$可计算为：

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

集合$\mathrm{loss}'(g)$只包含丢掉的现有物种，但这还不够好。缺失一个祖先物种会造成所有后代物种的缺失，基于简约原则，我们应该只记一次缺失事件。因此我们引入：

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

集合$\mathrm{loss}(g)$避免了上述问题。它其实是满足$\omega_S(A) = \mathrm{loss}'(g)$的最小$A \subset V(S)$。还拿图4.1为例子，在那棵树中，$\mathrm{loss}'(\mathsf{ENSGALT00000011124_chicken}) = \{human, mouse, rat, dog\}$ 并且$\mathrm{loss}(\mathsf{ENSGALT00000011124_chicken}) = \{Eutheria\}$成立，而$Eutheria$正是缺失发生的物种。图4.4给出了另一个例子。

4.3 倍增函数和缺失函数

倍增函数和缺失函数都是定义在$V_I(G)$上的。它们定义为：

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

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

倍增函数$D_G(g)$衡量了g是否为一个倍增结点，缺失函数$L_G(g)$记录了当g演化到它的子结点发生的缺失事件的个数。很明显，一棵基因树发生的所有的倍增和缺失事件总数分别为$\sum D_G(g)$与$\sum L_G(g)$。对一般的物种映射M，倍增函数和缺失函数的定义依赖于基因树G的拓扑，但如果我们使用简约物种映射M^*，这两个函数可以与树的整体拓扑无关。我们在第五章介绍完树的集合表示之后会重新审视这个问题。

4.4 计算物种映射的统计方法

物种映射M唯一决定了复制与缺失的推断，在这一小节，我们继续讨论如何选择M。出于简约物种映射的简洁与方便，一般情况下我们只选择M^*，但是如果在一个多基因家族中发生了许多倍增与缺失事件，M^*就会低估事件发生的次数从而产生错误，这时我们应该用更可靠的模型。Arvestad等人 [67, 68]针对此情况设计了一个Bayes算法来解决这个问题。在本文，我们不会过多的讨论这个Bayes算法的细节，而只对简约法与Bayes方法的比较作一些评论。

在Bayes的框架下，正确的物种映射应该最大化后验概率$\Pr\{M|G,S\}$，它可以计算为：

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

其中λ是产生速率，μ是缺失速率，而$p(\mu,\lambda|S)$是给定一棵物种树S时(λ,μ)的先验概率密度。当没有先验知识可供参考时，我们一般取带边界的均匀分布，Arvestad等人也是这样做的。在上面的方程中，当G给定时$\Pr\{G|S\}$是一个常数而不参与比较。那么我们只要去算$\Pr\{M,G|S,\mu,\lambda\}$。如果我们假设基

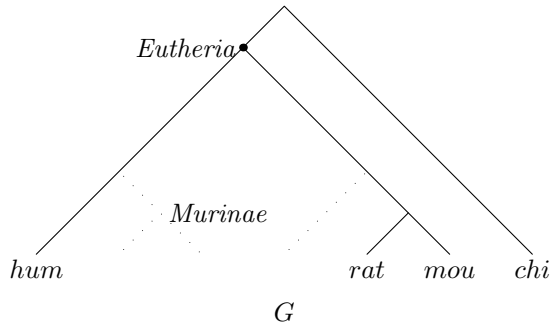

Figure 4.5: 简约物种映射失效的一个例子。假设真实的进化历史包含一次倍增(在*Eutheria*)和两次缺失(分别在*Murinae*和*Human*)事件，这样人和小鼠(mouse)的基因不是直系同源。强行使用简约物种映射M^*会漏掉这些事件并错误地推断出人和小鼠的基因是直系同源。

因在分离后——不管由于物种分化还是倍增事件——就相互独立的演化，那么$\Pr\{M, G|S, \mu, \lambda\}$可以写成一系列相互独立因子的乘积：

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

其中，$q_G(g, s)$是$\mathrm{par}(s)$进化到s时$\mathrm{par}(s)$中的一个基因g进化而形成现有基因树的概率。则$\prod_{g \in M^{-1}(\mathrm{par}(s))} q_G(g, s)$ 是物种$\mathrm{par}(s)$进化到s而形成现有基因树的概率。基于前人的一些工作 [69, 70]，Arvestad给出了计算$q_G(g, s)$的细节。

Bayes方法为树融合过程奠定了一个优美的理论框架，它为各种复杂的基因进化分析奠定了基础，在这种框架下我们也能利用物种树与基因树之间的关系互为辅助从而重构更准确的进化树[68]。但是，Bayes方法在生物和数学方面面临着一些理论上的问题。首先，不管是由于物种分化还是基因倍增，基因在分离后并不是独立进化：缺失更经常紧接着复制的发生，单拷贝基因很难会发生缺失[71]。这些事实还没有在目前的Bayes模型中体现。另外，Bayes方法必须假定一个先验分布，合理的先验可以提高推断的准确性但不合适的先验却可能造成偏差。带边界均匀分布是不恰当的。一方面，从TreeFam [15]的基因树上看，产生率λ和消亡率μ都是很小的；另一方面，为了能应用于使简约法失效的那些带有许多倍增和缺失的基因家族，边界值必须很大，即λ和μ的选择范围很广。这就造成了矛盾。对一棵正确的基因树简约法从来不会预测出错误的倍增事件，但Bayes方法会，而边界均匀分布的使用会加重这一问题，这是因为较大的λ与μ概率偏大会倾向于更多的倍增/缺失事件。错误的倍增/缺失可能造成更坏的后果，如果想在实用中大范围使用Bayes方法，我们必须对这一点也进行评测。

绝大多数情况下，μ和λ都是很小的，这时简约法已经足够准确。尽管在一个频繁发生倍增/缺失的基因家族中简约法会发生错误，但由于这时构建基因树都很成问题，Bayes方法的使用并不会为我们增加信心。另外，最简约物种映射的使用也使得倍增函数和缺失函数是拓扑无关的。在第六章我们将看到它的重要性。

Chapter 5

树合并算法

进化树(phylogenetic tree)是分子进化学(molecular phylogenetics)的核心。出于它不可替代的地位，进化学家们设计了许多算法来构建进化树以恢复进化历史，这些方法包括：距离法(distance based)，简约法 [40](maximum parsimony)，极大似然法 [44](maximum likelihood) 和Bayes方法 [49]；这些方法奠定了方法学基础，但应用中它们也需要具体的进化模型和参数来协同工作包括JTT [72]，WAG [73]，HKY [74]和GY94 [75]。既然我们有这么多算法和模型，树重构问题被最终解决了吗？答案是否定的，其中一个主要原因是：进化速率和方式在不同时间不同种系间是有差异的 [76]。一个最适合某一时间段或进化方式的模型可能在别的时段作出错误的推断；并且针对性越强，造成这种错误的可能性也越大。尽管一些作者已经注意到了这一现象 [77, 78]，并针对简单情况进行了仔细分析 [79, 80, 81, 82]，但在实际应用中生物学家们还是不得不用各种算法和模型建树，仔细观察并靠个人经验手工将各种结果结合成为一棵树。但生物学家是如何做到的？我们有办法实现一个算法自动完成这一点吗？这一章将回答这两个问题。

为了理解生物学家是如何评价和合并几棵进化树的，先看一个例子。图 5.1 给出了两棵树及其合并结果。图中的两棵树，同义树d_s与非同义树d_n[1]，都是由邻接法构造的。不用其它的信息，生物学家很快就可以凭经验判断同义树较低的分支和非同义树较高的分支是正确的。这种判断基于三个规则：(a) 可靠的分支通常具有较高的自展支持；(b) 同义树对较近的物种更有效，而非同义树对较

[1]非同义距离(nonsynonymous distance, d_n) 计量了影响氨基酸(amino acid)变化的核苷(nucleotide)突变，而同义距离(synonymous distance, d_s)计量了不影响氨基酸变化的突变 [83, 84, 75, 85]。这两种距离只能用于编码序列(CDS)。理论上d_s能更准确的衡量序列的进化距离，这是因为尽管密码子倾向性(codon bias)是存在的，d_s通常不受不均匀进化的选择[86, 1]；然而，当分化时间较远时，d_s会倾向于饱和(saturation)从而不能准确估计，因此d_s树较高的分枝不可靠。相反地，d_n即使在分化时间较长时也不容易饱和，但它对相似序列效果却不总是很好，因此d_n树较低的分枝相对而言不够准确。

远物种有效；(c) 正确的基因树经常相似于物种树。如果我们能实现一种策略或算法来综合考虑这些规则，我们就可以自动合并几棵用不同方法构造的进化树而达到更好的效果。这就是树合并算法(tree merge algorithm)。

原则上树合并算法并不复杂。它只是在每一个内节点选择一对能在备选

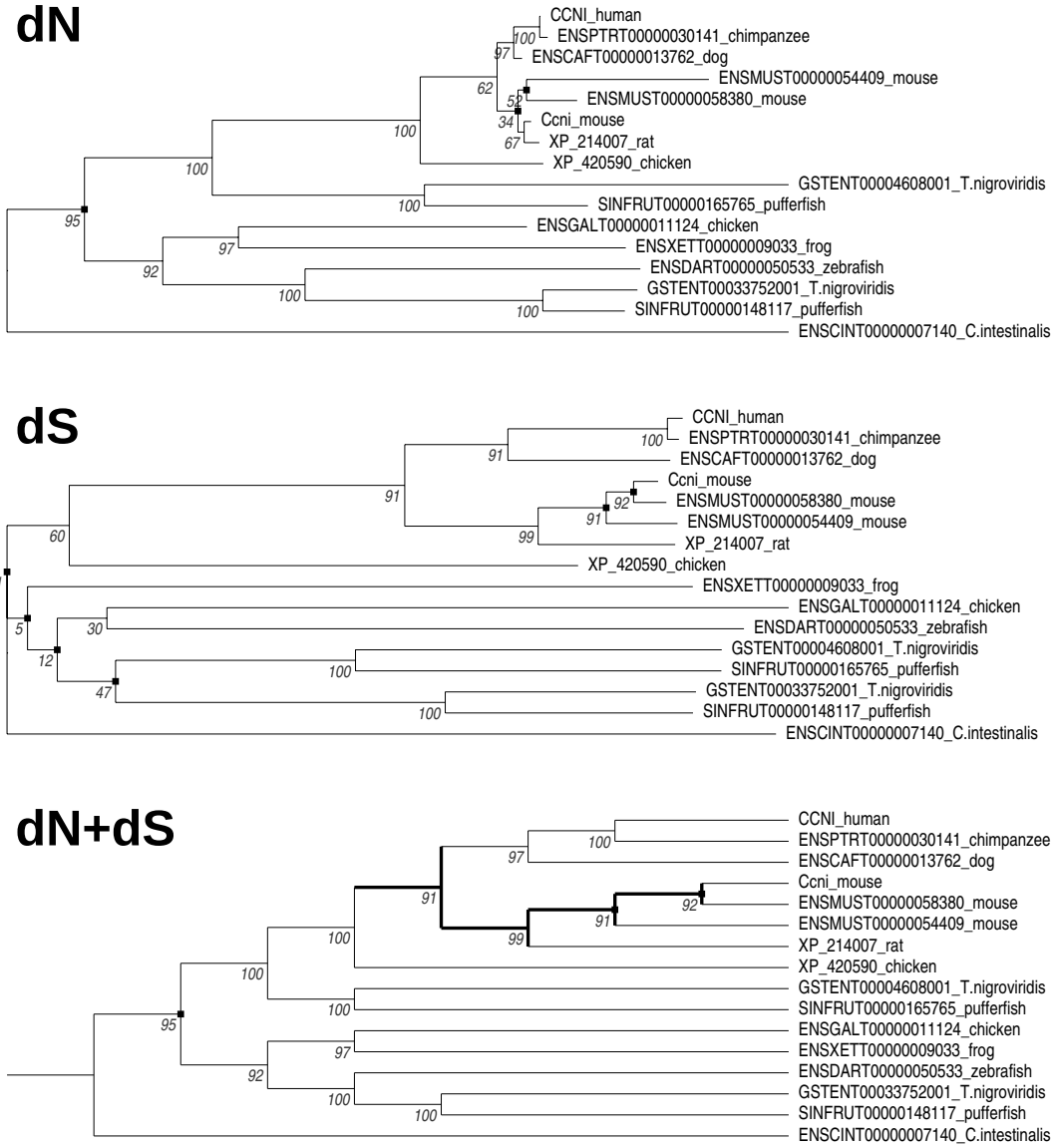

Figure 5.1: 树合并算法的例子。从非同义距离矩阵构造的非同义邻接树(nonsynonymous tree, dN tree)和从同义距离矩阵构造的同义邻接树(synonymous tree, dS tree)均由标准邻接法(neighbour-joining)构建。这两棵树合并后得到最下面的合并树。加粗的线为从同义树dS中选取的枝，而剩下的取自dN。合并树含有较少的缺失数，而具有更高的自展(bootstrap)支持。

树中找到的最优子节点。但为了能严格而准确的描述这个算法，我们还是要引入许多抽象的概念。在本章我们先介绍树的第三种表示法，集合表示(set representation)。接着，树合并算法将以严格的形式提出。在证明及描述这个算法之后我们还要分析它的时间复杂性。在下一章我们将给出一个评测以证明树合并算法的有效性。

5.1 树的集合表示

在比较两棵从同一序列集不同方法构造出的两棵树时，树的图表示并不方便。用图论的语言，我们必须同时比较节点集与边集才能确定两棵树是否相同。但这个过程其实并不必要。在有根树上，每一条边可以唯一地用它的较低节点覆盖的叶节点的子集合来确定；我们可以用一个叶节点的子集来表示一条边[2]，这样树的拓扑比较可化成集合的比较，更简洁也直接。

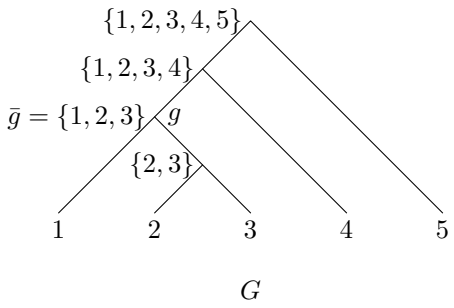

Figure 5.2: 树的集合表示的例子。在这棵树上，集合$\omega_G(g)$显示在每一个内结点的旁边，而G的集合表示为$\bar{G} = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{2,3\},\{1,2,3\},\{1,2,3,4\},\{1,2,3,4,5\}\}$；反过来，如果我们知道$\bar{G}$我们也就知道了$G$的拓扑。更进一步，如果我们令$\bar{g} = \{1,2,3\}$，则有$\bar{G}|_{\bar{g}} = \{\{1\},\{2\},\{3\},\{2,3\},\{1,2,3\}\}$，这是子树$G|_g$的集合表示。

令V是一个叶节点集合，$\Sigma(V) = \{G\text{是一棵有根树} : V_E(G) = V\}$是所有叶节点集为$V$的有根树的集合。对所有$G \in \Sigma(V)$，我们可以利用$\omega_G$映射(方程1.1)构造$\bar{G} \subset 2^V$ [3]：

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

很明显，如果G_1和G_2相同则必有$\bar{G_1} = \bar{G_2}$；而若两者不同则$\bar{G_1}$和$\bar{G_2}$也不同。因此方程 5.1事实上在图G和集合$\bar{G}$间建立了一个一一映射，图G总可以

[2]类似的，无根树的边可以用叶节点集的二分(bipartition)来表示。

[3]给定一个集合V，$2^V = \{A : A \subset V\}$是$V$所有的子集组成的集合，包括$V$本身和空集$\emptyset$。$2^V$也称做$\sigma$-集合。注意到$\left|2^V\right| = 2^{|V|}$总是成立的，这也是$2^V$记号的由来。

用$\bar{G}$来表示，这就是有根树的集合表示。为描述方便，我们还引入：

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

集合$\bar{G}|_{\bar{g}}$是子树$G|_g$的集合表示。图 5.2给出了一个例子。

在本章像$\bar{G}$和$\bar{g}$这样的记号总是代表集合表示。

5.2 倍增与缺失函数的集合形式

在前一章的小节4.3中，倍增与缺失函数定义为(方程4.6)：

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

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

这种方式定义下的两个函数是和基因树G的拓扑相关的，即只有给定一棵树才能计算它们。事实上，如果我们用简约物种映射M^*来推断倍增与缺失事件，这两个函数可以写成拓扑无关的形式。让我们在树的集合表示下重新审视它们的推导。

在下文中，记号$\bar{g}$，$\bar{g}_1$和$\bar{g}_2$总为$V_E(G)$的一个子集。类似于方程4.1和4.3 我们定义:

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

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

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

则简约倍增函数为：

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

简约缺失函数为：

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

其中：

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

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

这两个函数分别对应于方程4.4和4.5。现在，简约倍增函数和缺失函数都是定义在两个叶结点集合上的函数，这样它们的计算是拓扑无关的，即两个叶结点集内部的拓扑结构与倍增和缺失函数的计算无关。这个性质在树合并算法目标函数的构建中非常重要(第5.3.2节)。

5.3 树合并算法

5.3.1 树合并问题

给定一个二岔有根树的集合$\Phi = \{\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n\}$，定义允许节点集：

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

令$\Omega(\mathcal{G})$为所有满足$\bar{G} \subset \mathcal{G}$的*有根二岔树*$\bar{G}$的集合，那么$\Omega(\mathcal{G})$ 一定是$\Sigma(V)$的子集，它可以看作是$\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n$在$\Sigma(V)$张成的树子空间。为下文叙述方便，我们类似的引入：

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

则集合$\Omega(\mathcal{G}|_{\bar{g}})$可以看成是$\bar{G}_1|_{\bar{g}}, \bar{G}_2|_{\bar{g}}, \ldots, \bar{G}_n|_{\bar{g}}$张成的树子空间。

令F是定义在$\Sigma(V)$上的函数，广义树合并问题(general tree merge problem)指：在$\Omega(\mathcal{G})$中找到*有根二岔树*$\bar{G}$使得函数F最优。对一般的目标函数F，唯一的解决方案是枚举Ω中的所有树，计算F并搜索最优；但对于一类特殊的F，这种穷举式的搜索可以用高效而准确的算法来替代。在本文中，我们仅考虑如下形式的目标函数：

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

其中对一个$\bar{g}$，$\bar{g}_1$和$\bar{g}_2$代表$\bar{g}$的两个子结点。函数$F(\bar{G})$有两个十分重要的特性：第一，可加性；第二，函数f定义在$2^V \times 2^V$，这意味着$f(\text{chi}(\bar{g}))$的计算只与$\bar{g}$的孩子$\bar{g}_1$与$\bar{g}_2$本身有关，而与$\bar{g}_1$与$\bar{g}_2$ 内部的拓扑无关。在下面的小结中我们将看到，这两个性质保证最优树可以在$O(|\Phi| \cdot |V|^2)$时间找到。(狭义)树合并问题(special tree merge problem)指：当目标函数满足方程 5.14时在$\Omega(\mathcal{G})$中找到*有根二岔树*$\bar{G}$使得函数F最优。

5.3.2 目标函数的构造

依据方程 5.14，树合并算法的有效性完全取决于函数f，在给出详细的算法描述之前，我们应该确定这样的f存在并且在生物上有意义。事实上，f可以这样定义：

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

其中，$\alpha, \beta, \gamma > 0$，$D^*(\bar{g}_1, \bar{g}_2)$和$L^*(\bar{g}_1, \bar{g}_2)$分别为倍增函数和缺失函数，而$B^*(\bar{g}_1, \bar{g}_2)$等于所有含有$\{\bar{g}_1, \bar{g}_2\}$这种分岔的树给出的最好自展支持。$D^*$和$L^*$中的星号表明简约物种映射$M^*$被用于计算倍增数和缺失数，这两个函数衡量了基因树与物种树的相似性，越小越好。至于B^*的定义，我们还会在在最后一节讨论其它的可能性。另外，如果我们事先知道在某个分支某种算法较好，我们也可以强行的调高自展值。至此，在本章前言里提到的三个规则都被考虑到了。

5.3.3 树合并算法

树合并算法某种程度上很象动态规划算法，它把穷举式的搜索转化为对每一个$\bar{g} \in \mathcal{G}$计算$F^*(\bar{g})$：

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

很显然$F^*(V) = \min F(\bar{G})$总是成立，这样寻找最优树问题实际上就是如何求出$F^*(V)$，而当F服从方程 5.14时这个值可以递归的来求解。为了描述这一过程，我们要对每一个$\bar{g} \in \mathcal{G}$构建允许子节点集：

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

集合$\mathcal{C}(\bar{g})$是$\bar{g}$的备选子结点集，即它的每一个元素$\{\bar{g}_1, \bar{g}_2\}$都是可以作为$\mathrm{chi}(\bar{g})$。这样，我们可以计算：

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

从方程5.16和5.14我们知道：

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

这样$F^*(V)$就可以递归的求解了。

方程5.17和5.18奠定了整个树合并算法的基础。表 5.1给出了更多的细节。在实用中，我们要使用哈希(hash)技术来实现集合的插入与比较，如果针对集合元素计算的哈希函数非常理想，我们只用$O(|V|)$时间就可以完成一次集合的查找与插入。这样做会大大提高算法的速度。

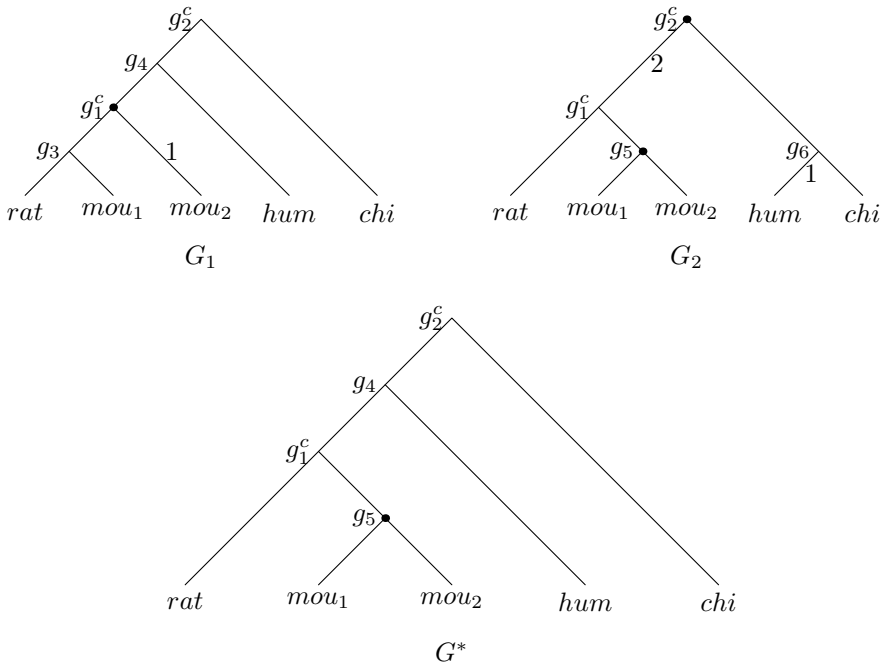

Figure 5.3: 树合并算法的具体例子。基因树G_1和G_2合并成为G^*。加粗节点为倍增点而树枝旁边的数字是发生在这条边上的缺失数。比如，$|\text{loss}(\{mou_1\},\bar{g}_3)| = |\{Rat\}| = 1$，$|\text{loss}(\bar{g}_1^c,\bar{g}_6)| = |\{Human, Chicken\}| = 2$而$|\text{loss}(\bar{g}_6,\bar{g}_1^c)| = 0$。

哈希的使用使得树合并算法非常高效。在过程ConstructG中，所有的哈希值可以在$O(n\cdot|V|(|V|-1))$ 时间内计算。接下来的$n\cdot(|V|-1)$次集合比较和插入最多用$O(n\cdot|V|^2)$时间。过程OptimizeF 要遍历$\mathcal{G}$中所有元素及其所有可能的子节点划分。这一步也是$O(n\cdot|V|^2)$。剩下的两个过程BuildTree和TreeMerge都可在$O(|V|)$完成。因此整个树合并算法的时间复杂度是$O(n\cdot|V|^2)$。

图 5.3给出了树合并的一个具体例子。在这个例子中，$\mathcal{G} = \{\{hum\},\{mou_1\},\{mou_2\},\{rat\},\{chi\},\bar{g}_1^c,\bar{g}_2^c,\bar{g}_3,\bar{g}_4,\bar{g}_5,\bar{g}_6\}$，$\mathcal{C}(\bar{g}_1^c) =$

$\{\{\bar{g}_3, \{mou_2\}\}, \{\{rat\}, \bar{g}_5\}\}$ 而$\mathcal{C}(\bar{g}_2^c) = \{\{\bar{g}_4, \{chi\}\}, \{\bar{g}_1^c, \bar{g}_6\}\}$。只有$\mathcal{C}(\bar{g}_1^c)$和$\mathcal{C}(\bar{g}_2^c)$这两个集合包含两个节点对，其它的$\mathcal{C}(\cdot)$都只含有一个元素。如果我们令方程5.15中$\alpha = \beta = 1$和$\gamma = 0$，我们可以计算出$f(\bar{g}_3, \{mou_2\}) = 1 + 1 = 2$, $f(\{rat\}, \bar{g}_5) = 0$, $F^*(\bar{g}_5) = 1$以及$F^*(\bar{g}_3) = 0$，因此$F^*(\bar{g}_1^c) = \min\{0 + 2 + 0, 0 + 0 + 1\} = 1$，结点对$\{\{rat\}, \bar{g}_5\}$更优。类似地我们知道$f(\bar{g}_4, \{chi\}) = 0$, $f(\bar{g}_1^c, \bar{g}_6) = 3$, $F^*(\bar{g}_4) = 1 + 0 + 0 = 1$以及$F^*(\bar{g}_6) = 1$，那么$F^*(\bar{g}_2^c) = \min\{1+0+0, 1+3+1\} = 1$，结点对$\{\bar{g}_4, \{chi\}\}$更优。最终的合并树为$\bar{G}^* = \{\{hum\}, \{mou_1\}, \{mou_2\}, \{rat\}, \{chi\}, \bar{g}_1^c, \bar{g}_2^c, \bar{g}_4\}$，而$F(G^*) = F^*(\bar{g}_2^c) = 1$。

5.4 讨论

树合并算法可以捕捉生物学家的思维，尽管它本身并不是一个建树算法，但它可以结合不同构树算法和模型的优点，自动在不同的种系间选择最合适的方法。如果备选算法和模型优势互补，树合并算法便能发挥出最大效率。但是，正如不给出近似正确的备选树生物学家不能校正出一棵正确的树一样，如果备选树有严重错误，树合并算法可能会导致偏差；有时人能辨认出输入错误而找出在Ω树空间之外的拓扑，但树合并不能，这是理论上的限制。另外，树合并算法的表现取决于目标函数。违背这个最优目标的进化历史也会令树合并产生错误。这些都是我们在使用树合并算法时应该注意的。

如果我们修改方程5.15中B^*的定义，树合并算法也可以看作是求一致树(consensus tree)算法[87]的变形。假设我们有n棵重抽样树(resampled trees)：$\Phi = \{\bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n\}$，给定$\bar{g}_1, \bar{g}_2 \in \mathcal{G}(\Phi)$我们可以将$B^*(\bar{g}_1, \bar{g}_2)$定义为满足$\{\bar{g}_1 \cup \bar{g}_2\} \subset \bar{G}_i$的$\bar{G}_i$的个数，这时$B^*(\bar{g}_1, \bar{g}_2)$就是$\{\bar{g}_1, \bar{g}_2\}$所代表的枝的自展支持，而如果我们令$\alpha = \beta = 0$，树合并算法事实上就是应用于有根二岔树的一致树构造算法。因此，如果B^*如上定义，树合并算法是对有根一致树构造算法在考虑物种进化时的扩展，它有助于提高一致树的准确性。

树合并算法是用来处理物种进化关系明确的二岔基因树的。在第三章我们讨论了如何定根。通过修改Durand等人 [88]的方法我们也可以利用物种树将基因树中的多分叉节点(multifurcated node)分离成二岔节点(binary node)，因此多分叉树问题也可解决。然而，“物种进化关系明确”这一条件是十分关键的。以我们的经验，其它符合方程 5.14的标准还不能像这一条件这么有效。

Table 5.1: 树合并算法。过程ConstructG构造集合$\mathcal{G}$和$\mathcal{C}(\bar{g})$；OptimizeF 递归地计算$F^*(\bar{g})$并把最优的子结点对结果存于*selected*；BuildTree过程用前一步的信息构建合并树。在这个算法中BuildTree生成一棵集合表示下的树。我们也可以很容易的修改这一部分使之生成图表示下的树$G^* = (V(G^*), E(G^*))$。

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:
$\mathcal{G} \leftarrow \emptyset$
$\mathcal{C} \leftarrow \emptyset$
ConstructG$(\bar{G}_1, \ldots, \bar{G}_n)$
  for $i = 1$ **to** n **do**
    for each $\bar{g} \in \bar{G}_i$ **do**
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{\bar{g}\}$
      $\mathcal{C}(\bar{g}) \leftarrow \mathcal{C}(\bar{g}) \cup \{\text{chi}(\bar{g})\}$

$selected \leftarrow \emptyset$
OptimizeF$(\bar{g})$
  if $F^*(\bar{g})$ has been calculated **then**
    return $F^*(\bar{g})$
  $min \leftarrow \infty$
  for each $\{\bar{g}_1, \bar{g}_2\} \in \mathcal{C}(\bar{g})$ **do**
    $score \leftarrow$ OptimizeF$(\bar{g}_1) + f(\bar{g}_1, \bar{g}_2) +$ OptimizeF$(\bar{g}_2)$
    if $min > score$ **then**
      $min \leftarrow score$
      $minpair \leftarrow \{\bar{g}_1, \bar{g}_2\}$
  $F^*(\bar{g}) \leftarrow min$
  $selected(\bar{g}) \leftarrow minpair$

BuildTree$(\bar{g})$
  if $selected(\bar{g}) = \emptyset$ **then**
    return $\{\bar{g}\}$
  $\{\bar{g}_1, \bar{g}_2\} \leftarrow selected(\bar{g})$
  return BuildTree$(\bar{g}_1) \cup \{\bar{g}\} \cup$ BuildTree$(\bar{g}_2)$

TreeMerge$(V, \bar{G}_1, \bar{G}_2, \ldots, \bar{G}_n)$
  ConstructG$(\bar{G}_1, \ldots, \bar{G}_n)$
  for each $v \in V$ **do**
    $F^*(\{v\}) \leftarrow 0$
  OptimizeF(V)
  return BuildTree(V)

Chapter 6

建树算法及模型的评测

尽管众多建树算法与模型为进化学家重构进化树时提供了大量备选，但各种算法与模型间的差异往往令他们迷惑。给定一个同源基因(homolog)的序列集，究竟哪一种算法合适至少开始时一般是未知的。序列较少的情况下生物学家们可以付诸实验或查阅大量文献来确定准确的进化关系，但对于大规模数据，这样费时费力的工作是难以想象的；即使这样的工作可以在以年为周期的长时间内完成——比如TreeFam项目——准确地自动建树也必将有助于减少工作量，同时也可以有效避免由于错误建树而引起的误差。对于TreeFam数据库，这样的评测结果有着重要意义。

许多作者对建树算法及模型的准确性作出了评测。根据测试集类型的不同，这些评测可以分成三类：a) 基于四到五个叶节点简单进化树的纯理论分析 [89, 90, 91]；b) 基于计算模拟数据的大规模分析 [45, 43, 92]；以及c) 基于实验操控的真实数据的评测 [93, 94, 95]。尽管这些评测为算法与模型的选择提供了宝贵的准则，它们都存在着各自的缺陷：理论分析只能应用于叶节点很少的进化树，而实际树往往有几十甚至上百个节点；模拟数据可以避免这一不足，但序列产生模型很可能有悖于真实历史从而造成评测上的偏差；基于实验的评测通过实验控制进化历程，但这一人工操控的过程也不能保证和真实历史相符。真正合理的数据来源于大规模的真实数据，但问题是我们很难确切的获知真实的进化历史。

在这一章中我将给出一个新颖的评测。它在两点上不同于以往的所有评测：第一，测试数据集来源于经人工校正的大规模真实数据；第二，使用基因倍增/缺失数来衡量建树的准确性，而这个量是模型无关的。受限于专家校正的准确性和个例下真实进化倾向于更多倍增/缺失数的可能性，这个评测也许还不能对哪种算法更优秀作一个最终定论。但这次评测必将对算法的选择提供有益的参考，也将揭示算法固有的一些特性。

6.1 参与评测的算法及模型

本次评测覆盖三大类经典算法：距离法(distance-based method)，简约法(maximum parsimony) [40]与极大似然法(maximum likelihood) [44]，包括常用的核苷及氨基酸模型。两种合并树也参与了评测。表 6.1给出了参与评测的17种进化树。下面几段给出了更多的细节。

距离法实际上是一大类算法的总称，包括：Fitch-Margoliash [34] ME (Minimum Evolution，最小进化) [36]，UPGMA [32] NJ (Neighbour-Joining，邻接法) [35] 和一些邻接法的变形。出于标准邻接法的普遍性和最小进化法的准确性 [39]，在本文我们只评测了这两种距离法。评测中，邻接法用NJTREE软件构建，最小进化树以基于GME框架的FASTME软件构建。距离矩阵的计算用TREE-PUZZLE [96]完成。对核苷序列，我们使用了HKY [74]模型。颠换/转换比(transition/transversion ratio)[1]从数据中估计；氨基酸序列用WAG [73]模型。Γ分布被用来模拟位点间不同的进化速率，其形状因子α固定于1.0。出于计

[1]转换指发生在purine (A or G)之间或pyimidine (C or T)之间的核苷替换；其它种类的替换都称为颠换。在生物上，转换比颠换更容易发生，并因而在对它们的建模上也有所区别。

Table 6.1: 参与评测的算法和模型。其中，参数κ 是颠换/转换比，α是Γ分布的形状因子。Γ分布用C个离散变量来近似。

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, sysnonymous 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, $C = 4$, $\alpha = 1.0$
`MJ-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`

算的方便，我们使用四个离散变量来近似连续的Γ分布 [97]。TREE-PUZZLE将比对的空位(gap)看成缺失数据[2]。由于TreeFam-1.x使用最简单的p-distance[3](一种没有考虑重复突变(multi-substitution)[4] 的距离)来构建邻接树，这种模型也被纳入这次评测。

最大简约树(Maximum Parsimonious tree)由PHYLIP [98]软件包中的**dnapars**和**protpars**来构建。如果程序输出多个同优的进化树，我们使用树合并算法将其结合。值得注意的是，**dnapars**会构建出多分岔树。由于基因倍增/缺失推断(duplication/loss inference, DLI)算法和树合并(tree merge)算法都只能应用于二叉树，部分核苷水平上的简约树不用来分析。氨基酸简约树始终是二岔树。

PHYML [47]用于构建极大似然树。进化模型及参数与TREE-PUZZLE使用的相同，即是：核苷序列用HKY模型，氨基酸用WAG；位点间进化速率的差异用形状因子为1.0的Γ分布来模拟。为了检验离散Γ分布的有效性，我们将Γ分布分别近似为四个和两个离散变量；不使用Γ分布的模型也参与了评测，以检验Γ分布的有效性。需要注意的是，PHYML和TREE-PUZZLE在模型的实现上有微小的差异。当我们分别用这两个软件估算树的枝长时，这一点就很明显：如果我们不使用Γ分布，两个软件给出几乎相同的结果；但当我们引入Γ分布时，两个结果就会有差异。但是，由于PHYML只能用来构树而不能计算距离矩阵，而TREE-PUZZLE只能算矩阵而不能构建标准的极大似然树 [5]，我们只能同时使用这两个软件而忽略这个微小的差别。

两种由树合并算法生成的进化树也参与了评测。第一种树结合了从同义进化距离和非同义进化距离构建的两棵邻接树；第二种树结合了四棵单模型进化树：核苷模型和氨基酸模型的极大似然树，以及同义树与非同义树。对于树合并算法中的目标函数f (方程5.15)，我们令$\alpha = 100000$，$\beta = 1000$以及$\gamma = 1$。自展值(bootstrapping values)度规于0至100之间。在这种配置下，倍增次数少的总是优先考虑，只有当倍增相同时才比较丢失；自展值只有在前两者都相同时才起作用。由于极大似然法所需计算量很大，自展检验没有用于极大似然树。在极大似然树的内节点上我们强制令自展值为80。之所以赋以如此高的自展值是因为极大似然树通常比距离法准确 [45]。在这里，自展值的作用最小的原因是

[2]似然法在计算距离时能将比对的空位(gap)看成缺失数据，这种处理更加可靠。

[3]P-distance也称做错配距离，它等于两条序列匹配区域上的错配位点的百分比。

[4]Multi-substitution指在同一位点发生多次突变。比如一千万年前核苷A变为G，但五百万年前G又变回A，这时即使目前在两条序列观测到两个A，我们也应该记两次突变。好的模型是能够考虑这个因素的。但是，如果两条序列分化很久远，最后的估计的方差就会很大并因而很不准确。

[5]TREE-PUZZLE可以构建“四极近似”下的极大似然树。

单一的构树算法仍可能对错误的边给出较高的自展值；倍增数的作用大于缺失数是因为进化上倍增较少发生，而缺失较多尤其是在发生一次倍增之后。

6.2 测试数据集的构建

为这次评测我们构造了两个测试数据集：TestSet1与TestSet2。数据集TestSet1包含1041个经过校正的基因家族。这些数据是在TreeFam-1阶段从NJ-AA-MM自动树校正的，包括12个完全测序的物种：ARATH（拟南芥），SCHPO（酵母），YEAST（酵母），HUMAN（人），MOUSE（小鼠），RAT（大鼠），CHICK（鸡），BRARE（斑马鱼），FUGRU（河豚鱼），DROME（果蝇），CAEBR（线虫）与CAEEL（秀丽线虫）。数据集TestSet2包含113个在TreeFam-2阶段校正的基因家族，增加了七个完全测序物种：PANTR（黑猩猩），CANFA（家狗），XENTR（青蛙），TETNG（河豚鱼），CIOIN，ANOGA（安蚊）和APIME（蜜蜂）。这113棵树是基于MERGE树校正的。除了数量与自动方法的不同，TestSet2倾向于包含容易校正的树，并且这些树复杂程度适中；与之相比，TestSet1既包含一些很难正确校正的进化树，也包括一些在TreeFam早期阶段加入很简单的树。一般而言，TestSet2更加可靠，尽管数量偏少。

构建两个集合时，原始的氨基酸多序列比对都是直接从TreeFam数据库中获取。这些比对通过氨基酸与三连码(codon)的对应转换成核苷比对。未完全测序的物种也排除在外。我们使用NJTREE来截掉比对较差的部分和屏蔽低分片段。Thompson等人[99]的文章给出了详细算法。

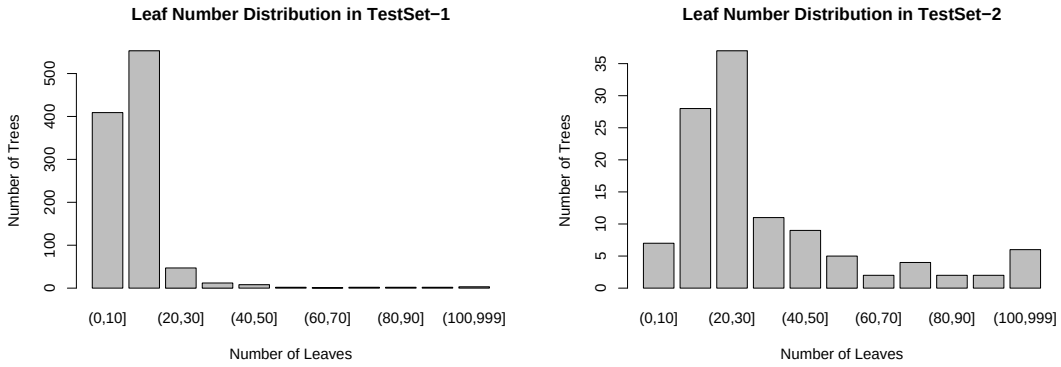

Figure 6.1: 测试集的叶节点数分布

6.3 评价标准

我们用两个指标来衡量建树的准确性。首先，将每一棵自动树分别与校正树相比，它们之间的拓扑距离(Robinson-Foulds距离，第1.4.2节)d_T [20]定义为两棵树间只存在于一棵树的内部分枝(不连接叶结点的边)的个数[45]。Robinson-Foulds距离d_T可以用于无根多分叉树。如果被比较的两棵树完全相同，它的值为0，如果完全不同d_T达到最大值$2n-6$。这里n是其中一棵树的叶节点数。基于Robinson-Foulds距离，我们可以进一步定义两种方法M_1与M_2间的距离：

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

这里m是参与评测的基因家族总数，$d_T^i(M_1, M_2)$为基于第i个家族两个算法构建的进化树或自动树与校正树之间的拓扑距离，而n_i是相应的序列数目。算法距离d_M始终介于0到1之间，$1-d_M$代表了能同时被两种算法重构的内部分支的比例。重新度规后，d_M也可以在不同数据集之间进行比较。

校正树是基于自动树构建的，它的正确性不可避免的会受到起初的自动方法的影响。同时，校正是一个主观过程，校正者的经验与状态或多或少会影响校正树的客观性。为了客观的评价各种自动构树算法，我们使用“百分倍增(percent duplication)”p_D和“平均缺失(average loss)”p_L来衡量准确性：

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

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

其中$D_i^*(M)$是给定方法M建的树发生基因倍增事件的总数，而$L_i^*(M)$是基因缺失事件数。如果我们接受进化倾向于较少的倍增/丢失事件的假设，好的方法的p_D和p_L一定较低。尽管在特定情况下这个假设并不总是成立，但整体的趋势应该是存在的。

这两个数据集只代表了一小部分真核基因家族。为了研究结果的稳定性，对TestSet2我们设计了一个自展过程以对p_D, p_L和d_M计算标准偏差(standard deviation)：首先，113个基因家族被重抽样B次，每次带放回的抽取113个家族；对第i次重抽样，我们计算评测指标r_i。评测指标r的均值μ_r和标准偏差σ_r则估计为：

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

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

由于多数情况下μ_r几乎等同于从原始数据中估计出来的r，在下表中将不再显示。对TestSet1我们也可以同样计算，但由于σ_r在两个集合上结果相似，简单起见，我们也不再给出了。

6.4 构树的准确性

Table 6.2: 建树算法及模型的评测结果。在这张表中，d_M是校正树CUR和其它树的拓扑距离，p_D是百分倍增(percent duplication)而p_L是平均缺失(average loss)。标准偏差σ是从对TestSet2的1000次重抽样而计算得到的。由于PARS-NT含有多分岔结点，p_D和p_L不能从这棵树计算。对p_L，两个测试集间的Pearson相关系数为0.970，对p_D为0.965。

	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

表6.2给出了不同算法与模型在两个测试数据集上的表现。总体上，各个指标相对稳定，这可以从较小的标准偏差上看出来。但由于TestSet1和TestSet2在构造时就是有偏的，这两个集合表现出了不同的性质，发生基因倍增的内节点比例p_D与平均基因缺失p_L明显不同。尽管数据集存在差异，各种方法之间的比较相对稳定。对p_D指标，两个集合的Pearson相关系数为0.965，p_L 的为0.970。这即是说如果算法M_1在Testet1上表现比M_2好，则M_1也倾向于在TestSet2上表现更好。

从到校正树CUR的拓扑距离d_M上判断，合并树MERGE比其它方法表现的更出色。在测试集TestSet2上，这可能因为CUR是从MERGE树校正过来的，但这种解释不能说明MERGE在TestSet1上的表现。在TestSet1上，校正是基于NJ-AA-MM的。这说明生物学家即使从不太准确的树出发也倾向于校正出一棵接近MERGE的树。这说明树合并算法可以捕捉部分生物学家的思维。

从发生基因倍增的几率p_D和平均缺失的个数p_L上，MERGE也有好的表现。从MERGE推断出的这两个数值显著小于基于其它方法的推断。树合并算法的作用也表现在NJ-NT-dM。这棵树是由两个相对简单的距离法构造的树NJ-NT-dN和NJ-NT-dS合并的，但它表现出了同极大似然法相近的准确性，甚至在某些标准上超过了极大似然树。

就单一模型的算法而言，氨基酸序列的简约法和极大似然法总体上超过了距离法，但在TestSet1上ME-NT-HKY4有着更好的表现。虽然在TestSet2上PARS-AA是最出色的，但由于树合并算法也用于PARS-AA树的构造，这种好的表现应该多少归功于树合并算法。在Γ分布的应用上，PHYML-NT-HKY4的确好于PHYML-NT-HKY1，但优势并不十分明显。

至于距离法，核苷HKY模型一致的超过了氨基酸模型。FASTME也优于相应的标准邻接法，这与Desper和Gascuel [39]的结论是相一致的。值得注意的是，在氨基酸水平使用复杂模型往往比简单模型差。同样的现象也在Hollich等人的工作中发现 [92]。

图 6.2给出了建树算法的关系图。同一类算法明显带有相似的印记：从氨基酸序列建的树和核苷酸序列的被分成两组；距离法生成的树聚在一起；最小进化与标准邻接法的关系也十分紧密。相似的算法有共同的特性，这一点也暗示了更多类型的备选树更有助于发挥树合并算法的效果；否则同一类算法的不足很难避免。

6.5 讨论

测试集的特性会影响建树算法的表现，这些特性包括：序列的分歧度、一个多序列比对中的序列数、比对的长度和不同位点变异速率的差异，并已经在许多评测中表现出来 [45, 43, 92]。我们的测试集基于TreeFam构建的真实数据。由于集合相对较小，收集的序列范围固定，我们还无法研究各种算法在其它特定条件下的表现，但在多细胞真核动物基因树的重构上，相信我们的评测更具实际意义。

在这次评测中，树合并算法表现出色，它的确能捕捉到生物学家在校正时所用到的信息。当物种进化关系确定时，树合并算法可以明显提高构树的准确

性。该算法的成功也表明每一类单模型算法都能构建出一些其它类算法不能构建的拓扑；否则的话一定会有一类算法明显接近MERGE的精度。由于没有单模型算法能一致的超越所有其它算法，树合并是不能被单一算法所取代的。

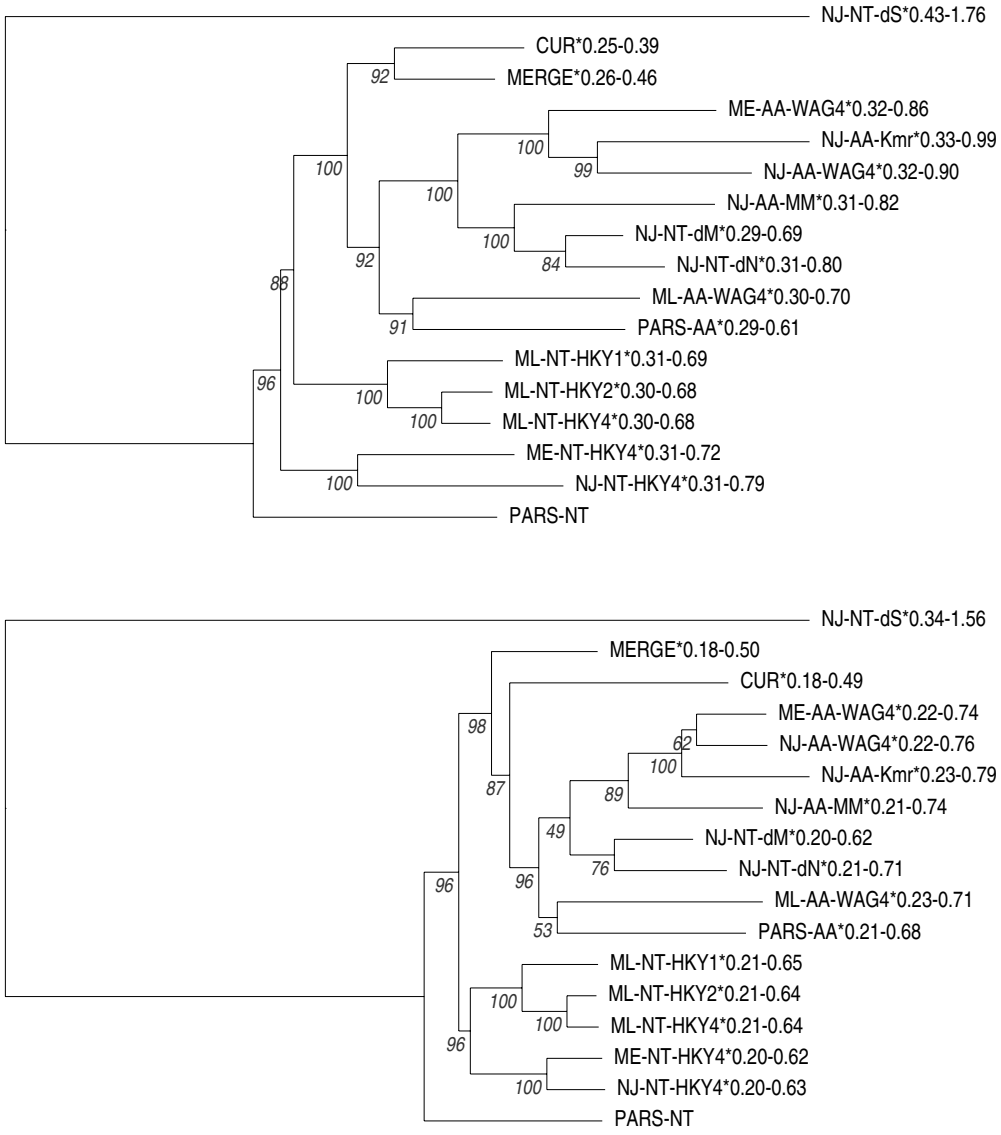

Figure 6.2: 不同建树算法间的关系。上面的树从TestSet2构建，下面的从TestSet1构建。叶结点名字后的第一个数字为百分倍增(percent duplication)p_D，第二个为平均缺失(average loss)p_L。建树过程描述如下：首先计算各种建树算法及模型之间的两两距离，所得结果距离矩阵交给NJTREE用标准的邻接法(neighbour-joining)建树。我们使用了上文介绍的自展过程以检验稳定性，最后100棵自展树用**PHYLIP**包中的**CONSENSE**程序结合到一起，这给出了分枝旁边给出的自展支持值。

另一方面，树合并算法需要利用基因倍增/缺失事件，而这种信息只有从物种进化相对确定的高等真核生物中获取。该算法不能用于构建物种树；由于水平基因迁移的存在也不能用于构建细菌的基因树。构建物种树和细菌的基因树时单模型的传统算法是唯一的方案。我们的评测表明各种算法都有利弊。尽管简约树和极大似然树整体上较好，距离法仍可能在一些情况下表现的更出色。即使树合并算法无法使用，综合各类算法的结果总是成功建树的保证。

Chapter 7

结论及展望

本文介绍了整个TreeFam数据库构建的整体思路和流程，并着重描述了在建设TreeFam过程中为获得更准确的进化树而作出的努力。截至论文定稿，在线版本TreeFam-2(图7.1)已经覆盖了约93%哺乳动物基因，它包含有15154个自动构建的多细胞动物基因家族和1174个经过人工校正的基因家族，已校正的家族包括包括细胞周期相关蛋白、多种激酶和翻译起始因子等重要基因。同时，TreeFam小组也正在紧张的构建新的版本TreeFam-3。在新的版本中，更多的物种会被加入进来，原有流程将继续得以改进，基因表达谱的数据也会结合到数据库中。新的TreeFam-3已于今年6月发布。

TreeFam数据库的最独特之处是人工校正的使用，而人工校正信息的利用则是通过约束邻接法来实现的，从这个角度说，整个TreeFam数据库的算法基础是约束邻接法。目前文章给出了一种假定输入约束树是有根树的实现，并提到了可以应用于无根二岔约束树的算法，但事实上，对无根多岔约束树的约束邻接法也是可行的，并且可以继续保持标准邻接法的时间复杂度$O(N^3)$，这一算法正处于实现阶段。另外，约束邻接法存在的一个问题是邻接法建树的准确性一般不如最大简约法(MP)和极大似然法(ML)，不使用这两种更可靠方法的原因仅仅是实现上的困难，从长远来看，将约束用于更好建树算法是必要的。

TreeFam数据库使用树合并算法来提高自动建树的精度，这个算法也是本文的另一个主要贡献和大部分篇幅之所在。该算法可以捕捉生物学家重构基因树时所用的信息，自动在局部拓扑上选择合适的算法和模型。评测显示树合并算法比所有单一模型的建树算法更加准确，而这一结果进一步揭示了不同单一模型建树算法尽管在整体上存在着准确性的差异，但它们或多或少起着补充作用；这即是说，有些单一模型算法也许整体准确度不是很好，但对特定的情形可能优于其它算法。尽可能的尝试各种单一模型建树算法始终是准确建树的保证。另外，树合并算法的推导也为树的拓扑优化问题给出了一种比较普适的模

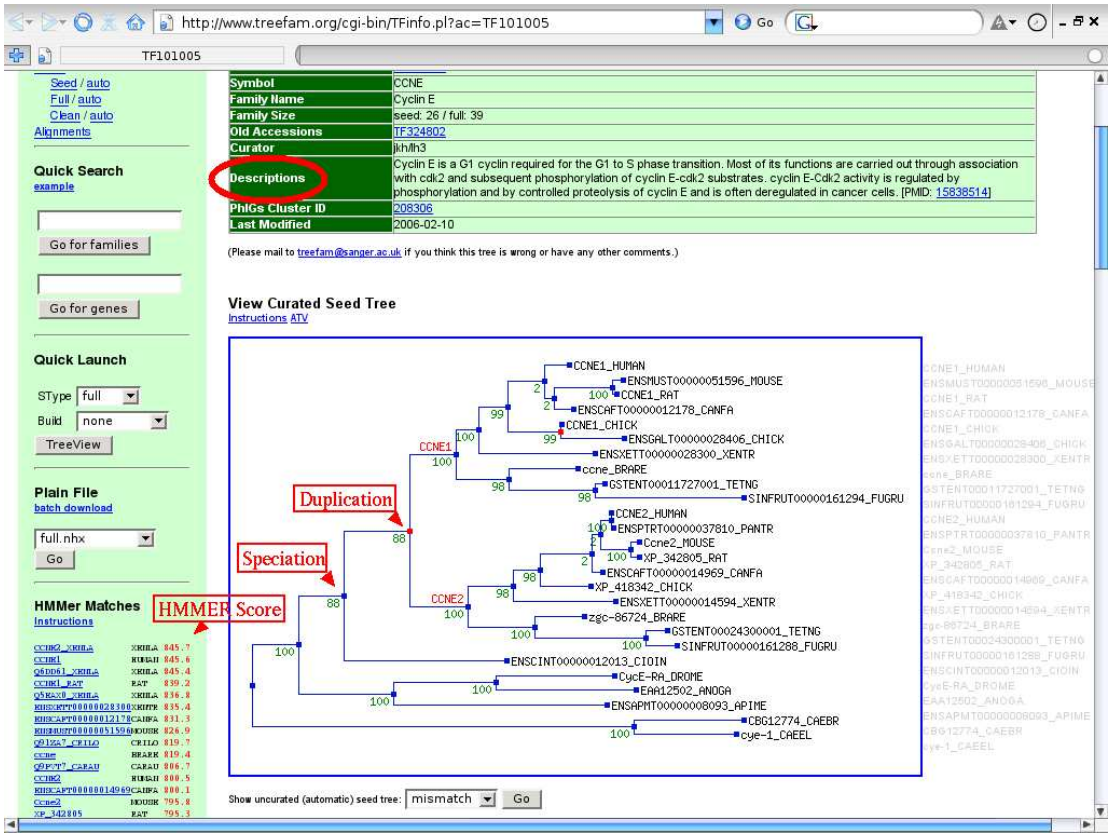

Figure 7.1: TreeFam数据库网站截屏

板，它事实上是一致树算法的更一般形式，这也具有一定的方法学意义。树合并算法的不足是它只能用于物种关系确定的基因树的构建，尽管它的算法本身并不要求这一点，但实用中似乎只有基因树和物种树相似这一条件可以发挥树合并算法的最大效能。要指出的是，经过适当修改，树合并算法也可以用于无根树的合并，但由于目前对无根树没有合适的目标函数以用来优化拓扑，这个算法暂无很实际的意义。

树合并算法基于倍增/缺失推断算法，这个概念早在近30年前就被提出并在10年前实现，而本文对这一方法的改进有二：一是将其纳入了一个更普适的理论框架，这为树合并算法和倍增/缺失推断的进一步应用奠定了理论基础；二是将推断推广到了物种树包含多分岔的情形，由于物种树的多分岔是十分常见的，只有考虑了这种情形的算法才是实用的。

类似地，叶结点重排算法也是一个实用算法。该算法首次提出了有根进化树画法的问题，并以一个简单的方式有效地解决了这个问题。尽管叶结点重排算法不会改进树的拓扑，但它能改进树的视觉效果，这在从同一序列集构建的树

的比较中有着重要作用，它也使得TreeFam中的树总能以确定的规则显示于网页。

本文所描述的四个新算法都是在开发TreeFam数据库时应运而生的，它们既有方法学的意义又很有实用价值。但是，尽管本文着力讲述各种算法，TreeFam数据库的意义却并不仅在于此。TreeFam更重要的意义在于为分子生物学家提供绝大部分基因的进化关系，辅助他们在实用中传递基因注释的信息，并为基因功能的发掘提供新的线索。TreeFam的意义还在于为分子进化学家提供一个可靠的资源，以使他们能利用TreeFam的数据进行更进一步的进化分析。在今后的一段时间，我们除了致力于开发数据更准确、信息更全、使用更方便的进化树的数据库，还会利用TreeFam的数据探索一些进化问题，比如内含子起源和进化以及发生复制后基因特性的变化等问题。解决生物问题才是生物信息学的最终目的。

Appendix A

技术细节

A.1 NJTREE软件

NJTREE软件是整个TreeFam数据库的核心。它实现了本文所讲述的几乎所有算法，包括对有根树和无根树的约束邻接法，树的定根和自展检验，叶结点重排算法，多分叉物种树的倍增/缺失推断和树合并算法。NJTREE也为TreeFam自动流程的建立和第六章的评测提供了大量的工具。另外，NJTREE也囊括了PHYML [47]的源码，这使得它可以进行许多极大似然相关的运算。在某种程度上说，整篇论文实际是在介绍NJTREE软件的工作原理。

NJTREE软件是非常高效的。它比任何其它的实现了邻接法的程序都更快，它也能以极高的速度计算极大似然距离，经我们修改过的PHYML算法也比原来快了20%。另外，我们也为NJTREE设计了一个图形界面(graphical use interface, GUI)，称做FLNJTREE。这个软件基于跨平台的FLTK图形库，可以在UNIX（包括LINUX）和Windows下编译。图 A.1 给出了FLNJTREE的一个抓屏。

A.2 MySQL结构

在服务器端，TreeFam基于MySQL关系数据库——世界上最受欢迎的开源数据库引擎。除了原始的PhIGs比对信息没有入库，其它所有数据都存于MySQL中。TreeFam数据库的整体结构相对简单。大部分表(table)可以分成三类：存储基因属性的表，存储基因家族信息的表和建立两者关系的表。表 A.1给出了TreeFam数据库所使用的关键的一些表，图 A.2则显示了它们之间的关系。要提出的是，TreeFam直接使用基因和家族的索引号(accession)作为表

的索引，这是会影响效率的 [1]，但由于目前来看TreeFam数据库的效率已经足够，我们还是暂用这种更易理解的结构。TreeFam的MySQL数据库已经对外公开，地址为**db.treefam.org:3308**，只读帐号为无密码的**anonymous**。

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

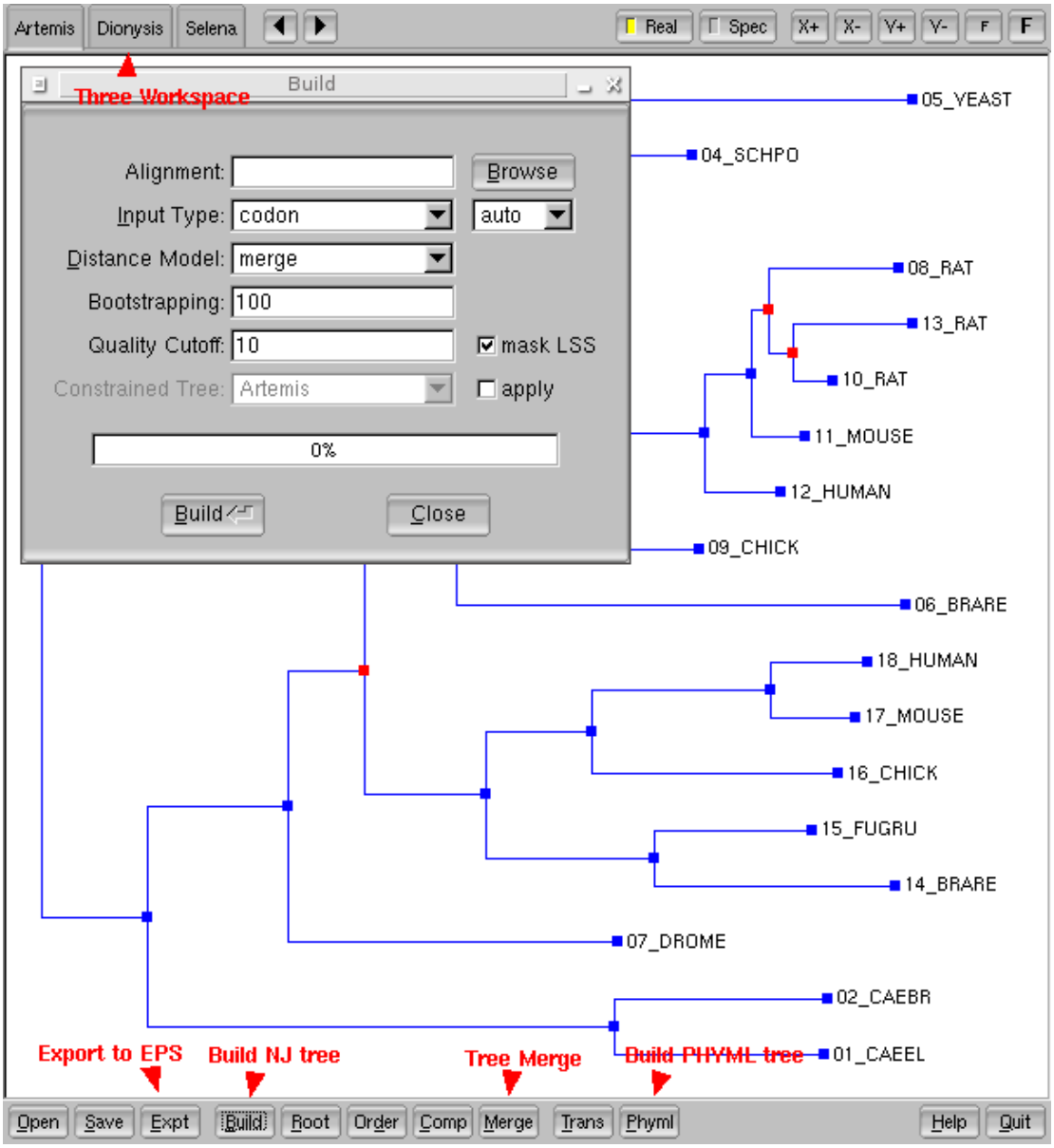

Figure A.1: FLNJTREE软件的截屏。

A.3 Perl API

我们推荐使用TreeFam的Perl-API (application programming interface)来连接数据库以取得TreeFam的各种信息。TreeFam-API也实现了一个简单的NHX格式的解析器，和图形化显示树和比对的工具。完整的文档请参见http://www.treefam.org/api/。

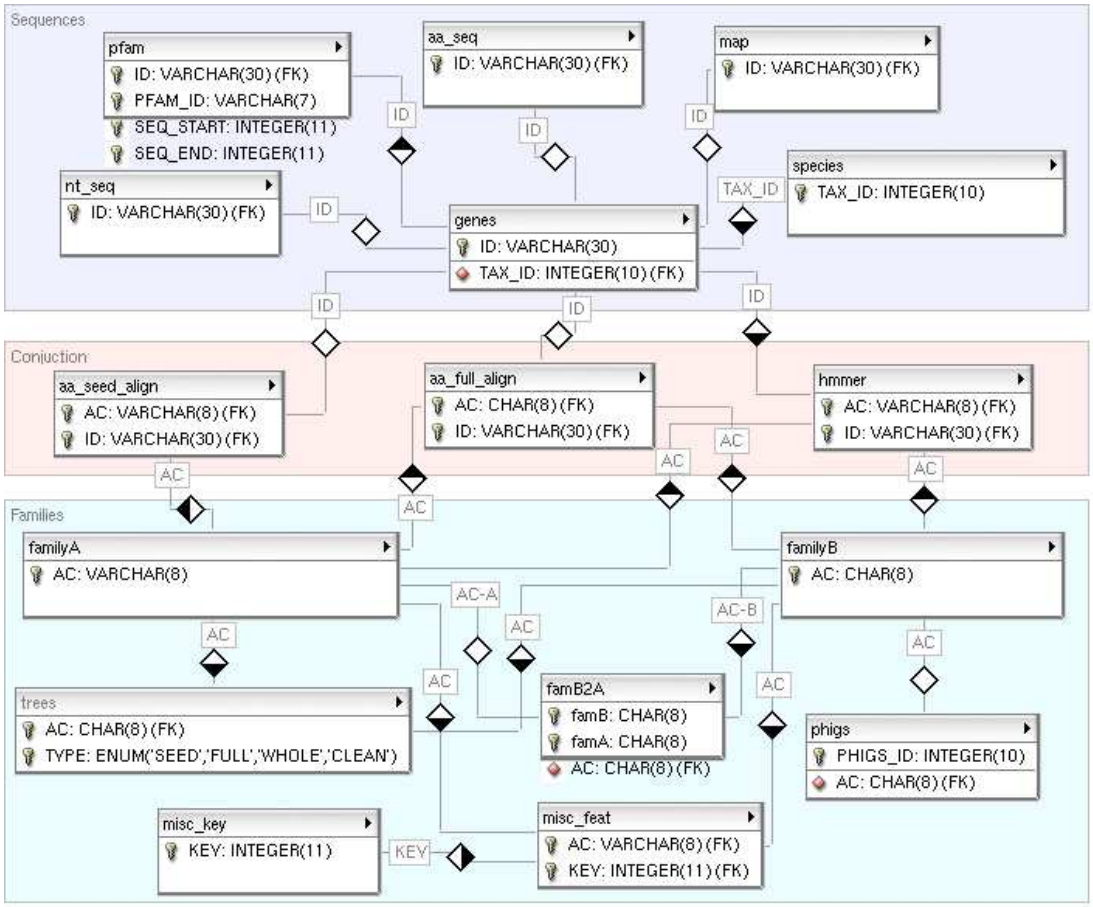

Figure A.2: TreeFam表结构关系图。这张图是由http://www.fabforce.net/dbdesigner4/生成的。

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: TreeFam数据库表的内容。不太重要的和向后兼容的表没有列在其中。

Bibliography

[1] SL. Baldauf. Phylogeny for the faint of heart: a tutorial. *Trends Genet*, 19(6):345–351, Jun 2003.

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

[3] J Cotton. *Vertebrate phylogenomics and gene family evolution*. PhD thesis, University of Glasgow, 2003.

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

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

[6] L Coin and R Durbin. Improved techniques for the identification of pseudogenes. *Bioinformatics*, 20 Suppl 1:94–94, Aug 2004.

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

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

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

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

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

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

[13] JF. Dufayard, L. Duret, S. Penel, M. Gouy, F. Rechenmann, and G. Perri猫re. Tree pattern matching in phylogenetic trees: automatic search for orthologs or paralogs in homologous gene sequence databases. *Bioinformatics*, Feb 2005.

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

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

[16] E V Koonin. Orthologs, paralogs, and evolutionary genomics. *Annu Rev Genet*, 39:309–338, 2005.

[17] M. Remm, CE. Storm, and EL. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J Mol Biol*, 314(5):1041–1052, Dec 2001.

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

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

[20] D F Robinson and L R Foulds. Comparison of phylogenetic trees. *Math. Biosci.*, 53:131–147, 1981.

[21] KP. O'Brien, M. Remm, and EL. Sonnhammer. Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Res*, 33 Database Issue:476–480, Jan 2005.

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

[23] R Balakrishnan, K R Christie, M C Costanzo, K Dolinski, S S Dwight, S R Engel, D G Fisk, J E Hirschman, E L Hong, R Nash, R Oughtred, M Skrzypek, C L Theesfeld, G Binkley, Q Dong, C Lane, A Sethuraman, S Weng, D Botstein, and J M Cherry. 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, Jan 2005.

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

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

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

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

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

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

[30] E M Zdobnov, C von Mering, I Letunic, D Torrents, M Suyama, R R Copley, G K Christophides, D Thomasova, R A Holt, G M Subramanian, H M Mueller, G Dimopoulos, J H Law, M A Wells, E Birney, R Charlab, A L Halpern, E Kokoza, C L Kraft, Z Lai, S Lewis, C Louis, C Barillas-Mury, D Nusskern, G M Rubin, S L Salzberg, G G Sutton, P Topalis, R Wides, P Wincker, M Yandell, F H Collins, J Ribeiro, W M Gelbart, F C Kafatos, and P Bork. Comparative genome and proteome analysis of anopheles gambiae and drosophila melanogaster. *Science*, 298(5591):149–159, Oct 2002.

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

[32] RR. Sokal and CD. Michener. A statistical method of evaluating systematic relationships. *University of Kansas Scietific Bulletin*, 28:1409–1438, 1958.

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

[34] W M Fitch and E Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967.

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

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

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

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

[39] R Desper and O Gascuel. 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, Mar 2004.

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

[41] J. Felsenstein. Cases in which parsimony of compatibility methods will be positively misleading. *Syst Zool*, 27:401–410, 1978.

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

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

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

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

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

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

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

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

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

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

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

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

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

[55] CM. Zmasek and SR. Eddy. A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics*, 17(9):821–828, Sep 2001.

[56] RD. Page and MA. Charleston. From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Mol Phylogenet Evol*, 7(2):231–240, Apr 1997.

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

[58] J Felsenstein. Confidence limites on phylogenies: An approach using bootstrap. *Evolution*, 39:783–791, 1985.

[59] MJ. Stanhope, A. Lupas, MJ. Italia, KK. Koretke, C. Volker, and JR. Brown. Phylogenetic analyses do not support horizontal gene transfers from bacteria to vertebrates. *Nature*, 411(6840):940–944, Jun 2001.

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

[61] J. Roelofs and PJ. Van Haastert. Genes lost during evolution. *Nature*, 411(6841):1013–1014, Jun 2001.

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

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

[64] RD. Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst Biol*, 43:58–77, 1994.

[65] R. Guigo, I. Muchnik, and TF. Smith. Reconstruction of ancient molecular phylogeny. *Mol Phylogenet Evol*, 6(2):189–213, Oct 1996.

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

[67] L. Arvestad, AC. Berglund, J. Lagergren, and B. Sennblad. Bayesian gene/species tree reconciliation and orthology analysis using MCMC. *Bioinformatics*, 19 Suppl 1:7–15, 2003.

[68] L Arvestad, A Berglund, J Lagergren, and B Sennblad. 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)*, pages 326–335, 2004.

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

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

[71] C Seoighe, C R Johnston, and D C Shields. Significantly different patterns of amino acid replacement after gene duplication as compared to after speciation. *Mol Biol Evol*, 20(4):484–490, Apr 2003.

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

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

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

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

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

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

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

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

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

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

[82] H Philippe, Y Zhou, H Brinkmann, N Rodrigue, and F Delsuc. Heterotachy and long-branch attraction in phylogenetics. *BMC Evol Biol*, 5:50–50, Oct 2005.

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

[84] WH. Li. Unbiased estimation of the rates of synonymous and nonsynonymous substitution. *J Mol Evol*, 36(1):96–99, Jan 1993.

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

[86] FJ. Ayala. Molecular clock mirages. *Bioessays*, 21(1):71–75, Jan 1999.

[87] T. Margush and FR. Mcmorris. Consensus n-tree. *Bull Math Biol*, 43:239–244, 1981.

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

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

[90] J P Huelsenbeck. 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, Sep 1995.

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

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

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

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

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

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

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

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

[99] JD. Thompson, TJ. Gibson, F. Plewniak, F. Jeanmougin, and DG. Higgins. The CLUSTALX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Res*, 25(24):4876–4882, Dec 1997.

发表文章

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. (第一作者)

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). (合作第一作者)

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. (第一作者)

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.

致谢

首先，我要感激我的导师郑伟谋，是他将我带入生物信息的王国，也是他既予以我遵遵教诲又给了我充分的发展空间，使我度过了积极而又多彩的博士生活。

下面我要感谢王俊和我在中国科学院北京基因组研究所(BGI)的所有朋友和同事。我在BGI度过了我的大部分博士生活，是他们给了我三年快乐而又激动人心的时光。我要特别感谢王俊给了我和Sanger中心的TreeFam小组合作的机会，也感谢刘涛对我一直以来的帮助和支持。

我同样感谢来自Sanger中心TreeFam小组的帮助和指导，他们是：Richard Durbin, Avril Coghlan, Jean-Karim Heriche, Lachlan James Coin, 和Alan Moses。Richard是TreeFam之父，它提出了TreeFam的基本概念和整体上的实现方案，没有他就没有TreeFam项目也就没有我的这篇论文。Avril, Jean-Karim和Alan仔细阅读了这篇论文并给出了许多有用的评论和修改意见。我要特别感谢他们对我并不流利的英文展现出的极大耐心。Lachlan和Avril对我第五章的早期版本提出了许多改进意见，他们深刻的评述使得这一章更加充实和具有普遍性。另外，我也从和Sanger TreeFam小组的交流上学到了许多东西。TreeFam始终是集体努力的结晶。

在此我也要感谢我的丹麦朋友，尤其是对Lars Bolund表示特殊感谢。我在丹麦时，他为我提供了平静而又和谐的环境使我能够集中于TreeFam项目和论文的写作，我欠他许多。同时我也要感谢郝柏林老师给我的教导，感谢郭玲老师在我不在理论物理所时给予的帮助，以及感谢我在理论物理所的所有室友和同学对我的帮助。

最后，我要谢谢我的父母和妻子，感谢他们始终默默的支持我的工作。