第1章 引言

1.1 研究背景

生物信息学（bioinformatics）是生物学与计算机科学以及应用数学等学科相互交叉而形成的一门新兴学科。它通过对生物学实验数据的获取、加工、存储、检索与分析，进而达到揭示数据所蕴含的生物学意义的目的。它涉及生物学、数学、计算机科学和工程学，依赖于计算机科学、工程学和应用数学的基础，依赖于生物实验和衍生数据的大量储存。生物信息学不只是一门为了建立、更新生物数据库及获取生物数据而联合使用多项计算机科学技术的应用性学科，也不仅仅是只限于生物信息学这一概念的理论性学科。事实上，它是一门理论概念与实践应用并重的学科。过去二十年，随着人类基因组工程的实施和深入，生物学数据获得前所未有的增加，数据的内容也从生理生化数据向遗传、结构、功能及其相互关系等数据发展。同一时期，计算机微处理器芯片、半导体存储器和系统软件也在按照指数方式增长。如何有效利用组合学、统计学等数学方法和现代计算机的强大计算能力，从这些生物学数据中提取有用的知识、发现重大的科学规律，成为生物学家、数学家、计算机学家们面临的巨大挑战，从而导致了计算生物学、生物信息学的产生和发展。

计算进化生物学是生物信息学的主要研究方向之一。进化生物学研究物种的起源和演化。引入信息学到进化生物学中，使得研究者能够：

- 通过度量DNA序列的改变研究众多生物体间的进化关系（超越了以前基于身体和生理特征观察的研究方法）
- 通过整个基因组的比对，研究更为复杂的进化论课题，如基因复制，基因水平转移等
- 为种群进化建立复杂的计算模型，以预测种群随时间的演化
- 保存大量物种的遗传信息

未来的研究工作包括重建现已相当复杂的进化树，并进行相关的研究。

1.2 研究意义

进化生物学在生物学中是一个非常重要的研究领域，其中进化树，又称系统发生树，是用来研究一群物种进化历史的标准模型，是进化生物学中不可缺少的重要工具。由于生物进化中可能产生的杂交、基因水平转移、基因重组等网状事件，导致一个物种的基因可能来源于多个祖先。多年来对生物进化历史的研究发现，杂交事件分别在植物、鸟类、鱼类的种群中均有发生。自然的杂交事件在哺乳动物，甚至是灵长类动物中都被发现过。研究显示，25%的植物和10%的动物，尤其是年轻的物种，都与杂交事件相关。

已有许多方法能够从一组物种中构建起他们对应的进化树，但由于进化过程中网络事件的存在，一组相同物种基于不同基因的分析可能产生不同的进化树，反之，通过对比这些进化树的相似性是帮助人们发现这些网络事件的重要手段。许多计算生物学家都对此问题非常感兴趣，他们常用的衡量标准有子树剪切再接距离（subtree prune and regraft distance, rSPR）和杂交数（hybridization number, HN）两种。Baroni等人证明了rSPR距离为进化过程中网状事件的数目给出了一个下限。[?]而对这些衡量标准的计算可以很好地抽象为具体的数学问题，因此借助计算机来研究进化树具有十分重要的意义。

1.3 固定参数算法概述

对于NP完全问题（NP难问题），因为当问题的规模增长时，算法执行所需的时间会随着问题的规模呈指数级增长，所以寻找一种有效的精确算法被广泛认为是不可能的。然而许多问题可以得到一个时间复杂度仅随问题的某个固定参数呈指数增长的算法。当问题在$O(f(k)\cdot n^c)$的时间内可解，$f(k)$是一个与n无关且c为常数时便属于固定参数可解类（fixed parameter tractable, FPT）。如果k远小于问题规模n，那么我们便可以获得该问题一个相对更有效的精确算法。例如，对于一个常见的NP问题——顶点覆盖问题，存在一个复杂度为$O(kn+1.274^k)$的固定参数算法[?]，其中n是顶点总数，k是顶点覆盖的大小。这就意味着顶点覆盖问题可以用该问题的解参数化。本文也是根据类似的思路利用固定参数算法来研究进化树的对比问题。

1.4 论文组织架构

本文采用了如下的结构：

第一章：引言。本章主要介绍本论文课题的研究背景，研究意义和主要的研究方法，最后对论文的章节进行了一个合理的逻辑安排。

第二章：问题建模和描述。本章先介绍本论文研究过程中所使用到的相关概念，包括进化树的定义和相关操作的定义，然后对所研究的问题给出了具体的形式化描述，最后简单介绍该问题的研究现状。

第三章：算法思想与设计。本章先介绍Whidden的固定参数算法，随后给出本文基于此算法的改进思路和具体方法和步骤。

第四章：数据结构设计及算法实现。本章给出改进后的固定参数算法的具体实现方法，包括数据结构设计以及具体的算法实现。

第五章：实验分析。本章通过随机生成的数据集和真实的数据集对算法进行实验分析，验证了其改进效果。

第六章：总结及展望。本章对本文研究的课题进行了归纳和总结，指明了改进和完善的方向。

第2章 数据结构设计及算法实现

```
class TreeNode{

private:
    TreeNode * parent; // 父节点的指针
    vector<TreeNode *> children; // 子节点的指针
    int id; // 节点id
    int label; //
    int pairId;

public:
    // TreeNode的构造函数与析构函数
    TreeNode();
    TreeNode(int id);
    TreeNode(int id,int label);
    TreeNode(int id,int label,int pairId);
    TreeNode(TreeNode *p);
    ~TreeNode();

    void AddChild(TreeNode *pChild);
    void SetId(int x) {id = x;}
    void SetLabel(const int str) {label = str;}
    void SetPairId(const int id) { pairId = id;}
    bool IsLeaf() const { return children.size() == 0;}
    bool IsRtLeaf() const { return pairId != -1;}
    bool Is2ndRtLeaf();
    bool IsRoot() const { return parent == NULL;}
    bool IsSibling(const TreeNode * p) const ;
    void removeChild(TreeNode * p);
    int GetChildrenSize() { return (int)children.size();}
    TreeNode * GetChild(int i) { return children[i];}
    TreeNode * GetParent() { return parent;}
    TreeNode * GetSiblingNode();
    TreeNode * GetRoot();
    int GetId() { return id;}
    int GetLablel() { return label;}
    int GetPairId() { return pairId;}
    string ToString();

```

```
39      TreeNode * Clone(); // 克隆以当前的为根的子树
40  };
```