FACTORS AFFECTING THE PERFORMANCE OF PHYLOGENETIC METHODS

A THESIS PRESENTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF PH.D. IN
MATHEMATICS AT
MASSEY UNIVERSITY

Michael A. Charleston

1994

Massey University Library
Thesis Copyright Form

Title of thesis: A Factors Affecting The Performance of Phylogenetic Methods

(1) (a) I give permission for my thesis to be made available to readers in Massey University Library under conditions determined by the Librarian.

(b) I do not wish my thesis to be made available to readers without my written consent for .3. months.

(2) (a) I agree that my thesis, or a copy, may be sent to another institution under conditions determined by the Librarian.

(b) I do not wish my thesis, or a copy, to be sent to another institution without my written consent for .3. months.

(3) (a) I agree that my thesis may be copied for Library use.

(b) I do not wish my thesis to be copied for Library use for .3. months.

Signed M. Charleston

Date 26/1/94

The copyright of this thesis belongs to the author. Readers must sign their name in the space below to show that they recognise this. They are asked to add their permanent address.

NAME AND ADDRESS DATE

Acknowledgements

I am indebted to my supervisors, Michael Hendy and David Penny, who have with patience and wisdom guided me through the last three years' research. Without their enthusiasm, understanding and skill I am certain I would not have achieved this opus.

I am indebted also to my colleagues here in the Department of Mathematics and further afield, who have aided my study in countless ways. In particular, I would like to thank Richard Rayner for his computing support, Mark Byrne for his patience with my inane questions, Todd Cochrane for his tranquility, Peter Frizzell for his appreciation of classical music, Shane Dye for his enthusiasm, Marijcke Vlieg for her tolerance, Mark Johnston for his humility, John Giffin for his cynicism, and Mike Steel for his inspirational insight.

My friends, who have not forsaken me, I thank also. Nigel Green, Maree Sleeman, Lon Teal, Marian Trembath, Kathryn Hurr, and countless others have kept me more or less in touch with reality, and I appreciate it immensely.

I thank Julie Spicer for her love and understanding, treasured forever.

I thank my parents for their love and support throughout this endeavour, and for keeping me almost as sane as I was when I started.

To everyone else who has aided me through this period I give my thanks, and apologies that I cannot mention you all here.

Dedication

This thesis is dedicated to my parents: a small token, but heartfelt.

Abstract

This thesis comprises several computer simulation experiments in which the performance of a selection of phylogenetic methods was assessed. Data were generated according to a known model and used as input for the phylogenetic methods. Some new methods were introduced, and their performance compared with extant methods. Performance was judged by several criteria, being *accuracy*, *consistency*, *efficiency*, *falsifiability* and *robustness*.

The experiments were designed to be biologically relevant, and yet computationally tractible. Hence the models of evolution used were simple, to allow a wide range of parameters to be tested for their effects within the bounds of available computing resources.

The experiments were divided into two main types, the "small n" with up to 10 taxa, and the "large n" with from 10 to 30 taxa. Parameters which were allowed to vary in the "small n" case included number of taxa (n), sequence length, tree topology, edge length probability distribution, and purity of data. In the "large n" case, number of taxa, sequence length, and edge length probability distribution were varied.

The simulation experiments show that the accuracy of phylogenetic methods decreases with increasing n, and that the mean number of internal edges of the generating tree which are incorrectly inferred increases at least linearly with n. The rate at which the sequence length must increase with n, to retain a fixed confidence in the inferred tree, is shown to be at least linear in n.

All the methods are approximately as susceptible as each other to sampling error, which is exacerbated by the generating tree having very short or very long internal edges, and by finite sequence length. All the methods are susceptible to random error such as sequencing error, but provided such error is small, the effect is not great.

One type of method, using edge lengths inferred by the Hadamard conjugation

process, is shown to be much more robust to impure data and to sequencing error than are the other methods.

With $n \geq 10$ only the fastest methods were used. Increasing n again decreased the accuracy of the methods. Varying the "molecular-clockness" of the generating tree was shown to have a much greater effect upon those methods inconsistent with data which do not satisfy the molecular clock hypothesis.

All the methods used are described algorithmically, and their computational complexity is discussed. New proofs are provided of the consistency/inconsistency of several methods with the models of evolution used.

A notation is introduced to characterize all tree topologies, and used throughout this thesis.

Pseudocode is provided for all the major algorithms used in the simulation experiments.

Contents

Acknowledgements ii

Dedication iii

Table of Contents vi

List of Figures xi

List of Tables xii

List of Algorithms xiv

1 Introduction 1

1.1 Organization of this thesis 1
1.2 The Problem 4
1.3 This Study 4
1.4 Definitions 6
1.4.1 Mathematical terminology 6
1.4.2 Phylogenetic terminology 10
1.5 Models of evolution 13

2 Phylogenetic Methods 17

2.1 Introduction 17
2.2 Desirable Characteristics of Phylogenetic Methods 18
2.2.1 Accuracy 18
2.2.2 Consistency 18
2.2.3 Efficiency 18
2.2.4 Falsifiability 20
2.2.5 Robustness 20

2.3 General Classes of Phylogenetic Methods 21
2.4 Constructive methods 23
2.4.1 Unweighted Pair-Group Method with Arithmetic Mean . . . 25
2.4.2 Transformed Distance method (TD) 25
2.4.3 Neighbourliness (ST) 26
2.4.4 Neighbour-joining (NJ) 27
2.5 Search methods 28
2.5.1 Closest Tree (CT) 28
2.5.2 Compatibility method (Co) 29
2.5.3 Maximum Parsimony (MP) 30
2.5.4 Branch and bound implementations 31
2.6 A note on some other methods 39

3 New Methods **43**
3.1 Why find more methods ? 43
3.2 The Distance Spectrum 44
3.3 Compatibility — again 46
3.4 SL 47
3.5 NJa 48

4 Experimental methods **49**
4.1 Models of sequence evolution used 49
4.2 Deviations of the data from a model 50
4.2.1 Inadequate and contradictory models 50
4.2.2 Sampling error 51
4.2.3 "White noise" and "Pink noise" 51
4.3 General approach 52
4.4 Small n 53
4.4.1 Choosing the tree topology and other parameters 53
4.4.2 Choosing the edge lengths 57
4.4.3 Calculation of the expected bipartition frequencies 57
4.4.4 Sampling from the expected bipartition spectrum 58
4.4.5 Distance Calculation 60
4.4.6 Inferring edge lengths 60
4.4.7 Example 62

4.5 Large n 66
4.5.1 Choosing a random tree 67
4.5.2 Deriving the ancestral sequence 69
4.5.3 Growing the data 70
4.5.4 Parameters 71

5 Results 1: "Small n" 75
5.1 Introduction 75
5.2 Representation of findings 76
5.3 Agreement between methods 76
5.4 Sampling error 78
5.5 Tree topology 87
5.6 Edge length distribution 91
5.7 Number of taxa 108
5.7.1 Required growth in sequence length with number of taxa . . 108
5.7.2 Optimal number of taxa for inferring a given edge 111
5.8 Use of the Distance Spectrum 113
5.9 White noise 115
5.10 Pink Noise 126
5.11 Summary 131

6 Results 2: "Large n" 135
6.1 Computational considerations 136
6.2 Sampling error 136
6.3 Number of taxa 138
6.4 Overall evolutionary time 141
6.5 Time to last bifurcation event 142
6.6 Edge length distribution 145
6.6.1 Type of distribution 145
6.6.2 Variance of the distribution 146
6.7 Depth of edges 148
6.8 Summary 150

7 Discussion 153
7.1 Introduction 153
7.2 Summary of results 154

7.2.1 Sampling error 154
7.2.2 Tree topology 155
7.2.3 Number of taxa 156
7.2.4 Edge length probability distribution 157
7.2.5 Treatment of observed data 157
7.2.6 Relationship between phylogenetic methods 158
7.2.7 White noise 158
7.2.8 Pink noise 159
7.3 Comparison with some other studies 159
7.4 That which may be 163
7.4.1 Sources of error in finite data sets 163
7.4.2 Properties of selection criteria 164
7.4.3 Search strategies 165

A Tree Topology Description Notation **171**

B Proofs of theorems **181**

C Pseudocode **195**
C.1 Introduction 195
C.2 General functions 196
C.3 Functions used in `sim.c` 202
C.3.1 Clustering methods 214
C.3.2 Search methods 227
C.4 Main program structure of `sim.c` 242
C.5 Functions used in `big.c` 244
C.6 Main program structure of `big.c` 250

D Dangers of Computer Simulation **253**
D.1 Tied Decisions 253
D.2 Rounding error 255
D.3 Programming errors 256

Bibliography **257**

List of Figures

1.1 Example of a graph 8

4.1 Typical generating tree used in simulations 63
4.2 Example incorrect tree inferred by some methods 63
4.3 Spectra of edge lengths for an example tree 66
4.4 Choosing a rooted binary tree from a given permutation. 68
4.5 Two rooted binary trees on three pendant vertices 69
4.6 An example of a rooted tree used in `big.c`. 72

5.1 The agreement of phylogenetic methods 77
5.2 UPGMA vs. c with $n = 10$ and all trees equally likely 80
5.3 TD vs. c with $n = 10$ and all trees equally likely 81
5.4 NJ vs. c with $n = 10$ and all trees equally likely 82
5.5 Mean proportion of trials in which the correct tree is inferred 85
5.6 Performance of NJ with all tree topologies of the same diameter, with 10 taxa 88
5.7 Effect of varying r with maximum path length $\sigma = 0.112$ 93
5.8 Effect of varying r with maximum path length $\sigma = 0.35$ 95
5.9 Effect of varying r with maximum path length $\sigma = 1.12$ 97
5.10 Effect of varying maximum path length σ with $r = 0.16$ 101
5.11 Effect of varying maximum path length σ with $r = 0.5$ 103
5.12 Effect of varying maximum path length σ with $r = 1$ 105
5.13 Minimum required growth rate in c with n for 85% confidence in inferred tree 109
5.14 The mean number of edges wrongly inferred with increasing n . . . 112
5.15 Compatibility methods with sequencing error rate $e = 0.025$ 117
5.16 Compatibility methods with sequencing error rate $e = 0.1$ 118
5.17 Closest tree methods with sequencing error rate $e = 0.04$ 119

5.18 Closest tree methods with sequencing error rate $e = 0.064$ 120
5.19 Maximum parsimony methods with sequencing error rate $e = 0.1$ . . 121
5.20 Distance Hadamard methods with sequencing error rate $e = 0.064$. 122
5.21 Sequence Hadamard methods with sequencing error rate $e = 0.025$. 123
5.22 Sequence Hadamard methods with sequencing error rate $e = 0.1$ . . 124
5.23 Constructive methods with sequencing error rate $e = 0.064$ 125
5.24 Effect of amalgamating data from two trees 129

6.1 Effect of sequence length with $n = 30$ 137
6.2 Effect of number of taxa on the accuracy of constructive methods . 139
6.3 Distance from T_G of inferred trees 140
6.4 Effect of time from the first bifurcation to the present 141
6.5 Inferred time of divergence of two sequences 143
6.6 Effect of divergence time factor f 144
6.7 Effect of variance of edge length probability distribution 147
6.8 Effect of edge depth on the probability of its correct inference . . . 149

7.1 Possible visualisation of the sources of error in inference of amount of evolutionary change 164
7.2 Possible move in the Hitch-hiking heuristic search strategy 167

A.1 The three binary unrooted trees on 4 pendant vertices. 172
A.2 Example tree T for which the TTDN is sought 173
A.3 Skeleton of tree T in the previous figure. 173
A.4 The first stage in reconstructing a tree from its TTDN. 174
A.5 The second stage in reconstructing a tree from its TTDN. 175
A.6 The third stage in reconstructing a tree from its TTDN. 175

B.1 Tree T used in the proof that TD is inconsistent. 184
B.2 The three unrooted binary trees on four pendant vertices 190

D.1 An example labelled tree 255

List of Tables

2.1 The proportion of trees tested using branch and bound. 40

3.1 A classification of some phylogenetic methods 44
3.2 The bipartitions and even-ordered subsets of $\{1,\ldots,6\}$ 45

4.1 The number of trees with each topology for $4 \leq n \leq 10$. 55
4.2 Edge-length probability distributions for different diameters of generating tree. 57
4.3 Number of operations required to generate a bipartition frequency spectrum. 59
4.4 Typical distance matrices, true, observed and inferred. 64
4.5 Typical expected, observed and inferred edge lengths 65

5.1 Half the mean partition distance between inferred trees of different methods 78
5.2 The net disagreement of some phylogenetic methods 79
5.3 Effect of tree topology on performance of phylogenetic methods . . 90
5.4 Variance of edge lengths as inferred from the distance and bipartition spectra 114

6.1 Effect of edge length probability distribution 146
6.2 Number of edges with depth k over all binary trees on 26 pendant vertices 148
6.3 Effect of depth of edges on their correct inference 150

List of Algorithms

2.1 Clustering process 24
2.2 Branch and bound for Co and CT 35
2.3 Branch and bound for MP 38
4.1 get_distances 60
5.1 variance.c 113
7.1 Example of Hitch-hiking 168
C.1 compare_sets(A, B, max) 196
C.2 Hadamard(**v**, **w**) 197
C.3 **HexpH(inv,outv)** 198
C.4 **HlnH** 198
C.5 permute(x, $perm$) 199
C.6 rough_exp(i) 199
C.7 permutation_to_tree($perm, output\_tree$) 200
C.8 sample_uniform($mean, range$) 201
C.9 sample_normal($mean, std\_dev$) 201
C.10 sample_log_normal($mean, std\_dev$) 202
C.11 compat(x, f, A) 202
C.12 choose_topology($topnumber$, $edge\_set$) 203
C.13 bipartitions_to_distances(**v**) 205
C.14 choose_tree($x, edge\_set$) 206
C.15 correct_distances 207
C.16 get_pathsets(**D**) 208
C.17 number_of_bipartitions 209
C.18 random_edge_lengths(**edge_set, outv**) 210
C.19 sample_bipartitions($length, error\_rate$, **inv, outv**) 211
C.20 sort_vector_descending(**inv, outv**) 213
C.21 NJ($averaging$) 214

C.22 ST 217
C.23 UPGMA(*version*) 220
C.24 TD(*version*) 224
C.25 CT(**v**, *distances*, *use_Hadamard*) 227
C.26 Co(**v**, *distances*, *use_Hadamard*) 231
C.27 set_up_first_tree(*A*) 234
C.28 add_taxon(*taxon*, *position*, *A*, *new_node*) 234
C.29 remove_taxon(*taxon*, *position*, *A*, *node*) 234
C.30 convert_edges_to_tree(*edge_set*, *tree_array*) 235
C.31 convert_tree_to_edges(*tree_array*, *edge_set*) 236
C.32 Fitch(*A*, *nodes*, **v**, *bound*) 237
C.33 MP(**v**, *distances*, *use_Hadamard*) 239
C.34 sim.c 242
C.35 get_bif_times(**output_bif_time**) 244
C.36 ones_count(*z*) 244
C.37 get_whole_tree 245
C.38 grow_data 247
C.39 generalJC 249
C.40 big.c 250
D.1 Typical loop to find $\{i, j\}$ to maximise $f(i, j)$ 254

Chapter 1

Introduction

> There is a theory, which states that, if anyone should ever discover exactly what the Universe is for, and why it is here, it will instantly disappear, and be replaced by something even more bizarrely inexplicable.
> There is *another* theory which states that *this has already happened.*
> There is yet a third theory, which suggests that both of the first two theories were concocted by a wily editor of *The Hitch-Hikers' Guide To The Galaxy*, in order to increase the level of universal uncertainty and paranoia, and so boost sales of the Guide.
>
> [*The Hitch-Hikers' Guide To The Galaxy, Douglas Adams* [1]]

1.1 Organization of this thesis

This thesis summarises my simulation study into the performance of some methods of phylogenetic inference. It is divided into seven chapters, with four appendices. The current chapter introduces the general problem of phylogenetic inference and defines many of the terms which are used throughout this thesis. Other terms are discussed in context. Also discussed in this chapter are the models of molecular evolution used in the data generation for the simulation experiments.

The second chapter describes five desirable properties of phylogenetic methods: accuracy, consistency, efficiency, falsifiability and robustness. A concise description is given of some known methods, and each of these is assessed for its consistency and efficiency. A new method, here called 'Compatibility' is introduced in this chapter,

because of its close relationship with the 'Closest tree' selection criterion [40]. (This method should not be confused with the compatibility method used in connection with maximum parsimony: see Section 2.5.2 and [91].) General algorithms are given in pseudocode of the operation of the phylogenetic methods based on clustering. Algorithms are also provided of the exhaustive search strategies used to evaluate the three types of optimality criteria used in this study. Some other methods, not studied here, are mentioned in the last section of Chapter 2, and reasons for their exclusion from the current investigation are discussed.

The first section of Chapter 3 addresses the question of why new methods should be sought, and discusses some relationships between those methods used in this study. Section 3.2 describes a way of inferring expected frequencies of patterns in sequence or character data from matrices of distances between taxa. Section 3.3 briefly describes the new compatibility method again, and Sections 3.4 and 3.5 describe two more new variants. The consistency and efficiency of these new phylogenetic methods is noted.

The fourth chapter describes the experimental methods used to conduct this study. Section 4.1 discusses in depth the models of sequence evolution which were used to generate data in the experiments. Section 4.2 describes some of the ways in which data can deviate from the model used in the generation of data, these being random noise, sampling error and inadequate or contradictory models. The next section (4.3) describes the methods used to generate data, and the way in which accuracy of each method is assessed in each experiment. The study is divided into two parts, these called the "small n" and "large n", where n is the number of taxa being considered. Section 4.4 describes in detail the generation of data in the "small n" case and provides an example of such data generation. Data generation for the "large n" case is described in Section 4.5.

Chapter 5 describes eight general results pertaining to the performance of phylogenetic methods in this investigation. Section 5.3 discusses the amount of agreement between the phylogenetic methods. Sections 5.4 to 5.7 discuss the effects of sampling error, tree topology, edge length probability distribution and number of taxa on the performance of the phylogenetic methods. Section 5.8 notes the effect of using observed distances to infer edge lengths, rather than frequencies of patterns of character states observed across the taxa. Sections 5.9 and 5.10 show how the phylogenetic methods behave when there is random error in the data: (1)

when the random error is unbiassed, and (2) when the random error is biassed, respectively.

The first section of Chapter 6 discusses some computational considerations which apply when the number n of taxa increases, and how these considerations affect the choice of data generation and phylogenetic inference methods used. The "large n" case is restricted so that only 'clustering methods' of phylogenetic inference are used. Section 6.2 notes the effect of sampling error on the accuracy of the methods used in this part, and Section 6.3 discusses the way the number of taxa affects this accuracy. Sections 6.4 and 6.5 discuss the effects of the overall time from the root of a phylogenetic tree to the present, and the relative time at which the last bifurcation (speciation) event occurred. The edge length probability distribution having been established as having a major influence on the performance of phylogenetic methods, Section 6.6 describes an experiment in which the type of distribution and their spread were assessed for their affect on the performance of the clustering methods. Also in Chapter 6 is discussed the effect that the location within a phylogenetic tree of a given edge of the tree has, upon the probability that it will be correctly inferred by the clustering methods.

Chapter 7 summarises the results of these experiments, and relates them, where possible, to similar experiments that have been carried out by other authors. A brief discussion then follows in Section 7.4 of the possible directions in which research may continue.

The appendices contain information which, it was decided, could not be included within the main text without interrupting the flow of the discussion. Appendix A describes a method by which the shape (topology) can be characterized in a concise way. This notation, the 'Tree Topology Description Notation', is used throughout this thesis.

Appendix B includes mathematical proofs of several theorems described in the main text, concerning the consistency of some methods with respect to models of data generation (Theorems 1 to 4). Also in this appendix are a proof (Theorem 5) that the 'Neighbour-joining' method is optimal in a more general class of methods, that two of the methods used are equivalent in a special case (Theorem 6), and that a simple modification of the Neighbour-joining method yields a method equivalent to the 'Neighbourliness' method (Theorem 7).

Appendix C contains pseudocode listings of all the main algorithms used in this

study. A brief introduction in Section C.1 describes the notation used. In Appendix D are described three problems which I have encountered in the simulations, these being tied decisions, rounding error, and programming errors. Appropriate courses of action concerning each of these potential problems are described.

1.2 The Problem

A central problem of taxonomy is that of inferring phylogenetic relationships, usually described by evolutionary trees [37], between taxa (e.g., families, genera, species, populations). Given data from a set of taxa, we most often want to reconstruct the evolutionary tree which best fits the data set. With the considerable advance of the Polymerase Chain Reaction (PCR) making the acquisition of DNA and RNA sequences relatively easy and cheap, there has appeared a vast quantity of DNA and RNA sequence data. The logical course is to take maximal advantage of these sequence data, and either to use them as inputs to sequence-based methods of phylogeny reconstruction, or indeed to convert sequence data to distance data, for use with distance-based methods. Though, as has been pointed out [87], a large proportion of the information in sequence data is lost when converting from sequences to distances, the high level of understanding of the behaviour of distance-based methods, coupled with their relatively high speed of operation, means that use of distance data is also a sensible course of action.

1.3 This Study

The main aim of this study is to examine some of the factors which can affect the performance of phylogenetic methods. Since the theoretical or analytical calculations required to cover all possible cases with all but the very simplest models and smallest numbers of taxa (typically 4 or 5) are often prohibitively complex, we resort to the empirical approach: we simulate.

As a starting point for further study, simulation is very useful. It is not the 'b-all and end-all' of phylogenetic study, but it can reveal trends in the characteristics of phylogenetic methods which can then perhaps be investigated theoretically [12]. The simulation methods are described fully in Chapter 4.

The simulated data, generated according to given models of evolution and

known phylogeny, were used as inputs to the various phylogenetic methods, and measures were obtained of the ability of these methods to recover the generating phylogeny.

I have split the investigation into two sections, which I have called the "small n" and the "large n" cases. For the "small n" case unrooted trees (defined in the next section) were used, with from 4 to 10 pendant vertices, corresponding to extant taxa. For these unrooted trees, the following parameters were varied:

- Number of taxa;
- Sequence length;
- Tree topology;
- Edge length probability distribution (range);
- Purity of the data.

Varying these parameters it was possible to determine the effect each had on the performance of phylogenetic methods.

For the "large n" case, rooted trees (defined in the next section) were used, with up to 30 pendant vertices. For these rooted trees, the following parameters were varied:

- Number of taxa;
- Sequence length;
- Overall time from the root to the pendant vertices;
- Proportional time to the last bifurcation event;
- Edge length probability distribution (type and range).

The other aspect studied with rooted trees is the effect of the "depth" of each edge (defined in Chapter 6) on the probability that it will correctly be inferred.

1.4 Definitions

There are some special terms used in this thesis, which are discussed in this section. They consist of some basic mathematical terminology and some terms commonly associated with phylogenetic inference. The definitions of some moderately new terms related to the behaviour of phylogenetic inference methods are given in Chapter 2.

1.4.1 Mathematical terminology

Graphs and Trees

A *graph* G consists of a set $V(G)$ of *vertices*, also called *nodes*, and a set of *edges* $E(G) \subseteq V(G) \times V(G)$. An edge connecting v_1 and v_2 can be *directed*, in which case it is denoted by the ordered pair (v_1, v_2), or *undirected*, in which case the order of the pair is unimportant, and the edge may be denoted $\{v_1, v_2\}$. The direction of edge (v_1, v_2) is from v_1 to v_2.

We say that edge $e = \{v_1, v_2\}$ is *incident* on vertices v_1 and v_2. Each directed edge $e = (v_1, v_2)$ is said to be *directed out of* v_1 and *directed into* v_2.

The *degree* of a vertex v is the number of edges which are incident on v, and is written $\deg(v)$. The *in-degree* of a vertex v is the number of edges directed into v, and the *out-degree* v is the number of edges directed out of v.

If a vertex has degree 1, it is a *pendant vertex*, and the edge incident on it is a *pendant edge*. In phylogenetics, the pendant vertices correspond to extant taxa, and the internal vertices correspond to hypothetical ancestral taxa. (Note that pendant edges and vertices can also be referred to as 'exterior', 'external' and 'outer': such notation is not used here.)

A *path* between two vertices v_1 and $v_2 \in V(G)$ is a set of edges $\{(v_1, v_{i_1}), (v_{i_1}, v_{i_2}), \ldots, (v_{i_k}, v_2)\}$. The *length* of a path is the number of edges in it. A graph G is *connected* if for each pair of vertices $v_1, v_2 \in V(G)$, there exists a path between them. A *cycle* is a non-empty set of distinct edges in $E(G)$ which describe a path from a vertex v to itself (see Figure 1.1). A *loop* is an edge of the form (v, v) (see Figure 1.1). If G has no cycles or loops, and is connected, then it is a *tree*. All trees considered in this thesis are *finite*, that is, they have a finite number of vertices. Note that in a tree the path $\pi_{i,j}$ between vertices v_i and v_j is unique for each $v_i, v_j \in V(G)$. The maximum of the number of edges in any path in T is the

diameter of T, and is denoted by $d(T)$.

A tree T whose pendant vertices represent taxa is referred to as a ***phylogenetic tree***. Sometimes phylogenetic trees are also known as ***cladograms***, but I do not use such nomenclature here [44].

Given a tree T with edge set $E(T)$ and vertex set $V(T)$, and a tree T' with edge set $E(T')$ and vertex set $V(T')$, T' is said to be a *subtree* of T if $E(T') \subseteq E(T)$ and $V(T') \subseteq V(T)$.

With apologies to terminology purists, I shall henceforth adopt a slightly sloppy approach and refer to edges e and vertices v as simply being in the graph G where this is unambiguous, instead of $v \in V(G)$ and $e \in E(G)$. I will often write an edge as an ordered pair of vertices without assuming any direction, unless otherwise stated, so an edge joining vertices v_1 and v_2 can equivalently be written (v_1, v_2) or (v_2, v_1).

We can distinguish one vertex of a tree T and call it the *root*, in which case the tree is *rooted*. A *rooted binary tree* is a rooted tree in which, when all the edges are directed away from the root, the in-degree of every vertex is either 0, for the root vertex, or 1, for the other vertices. For such a directed rooted binary tree, the out-degree of every internal vertex is 2, and that of every pendant vertex is 0.

Each rooted binary tree T with n pendant vertices has $|V(T)| = (2n - 1)$ vertices in total and $|E(T)| = (2n - 2)$ edges, $(n - 2)$ of which are internal and n of which are pendant.

An *unrooted binary tree* is a tree in which every vertex has either degree 1 or degree 3. Any unrooted binary tree can be changed to a rooted binary tree by inserting a new vertex of degree 2 into an existing edge of the tree. Each unrooted binary tree T with n pendant vertices has $|V(T)| = (2n - 2)$ vertices, $(n - 2)$ of which are internal, and $|E(T)| = (2n - 3)$ edges, $(n - 3)$ of which are internal.

Two trees have the same *topology* if they are indistinguishable when all labels and edge lengths or weights are ignored. This is an equivalence relation on all trees. This definition is at odds with some authors (e.g. [93]), who include the labels of pendant vertices in their definition of tree topology, so we note it specially to avoid confusion.

If a tree has any internal vertices with degree greater than 3, it is said to be *not fully resolved*. All binary trees are fully resolved.

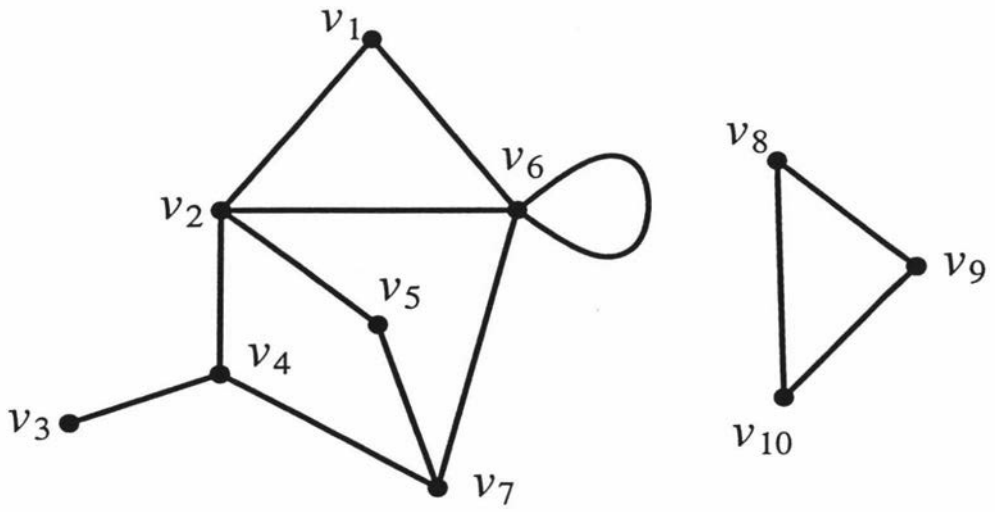


Figure 1.1: Example of a graph

The above graph with 10 vertices, $v_1, \ldots, v_{10}$ has two connected components. The degrees of the vertices are $\deg(v_3) = 1$; $\deg(v_1) = \deg(v_5) = \deg(v_8) = \deg(v_9) = \deg(v_{10}) = 2$; $\deg(v_4) = \deg(v_7) = 3$; $\deg(v_2) = 4$; $\deg(v_6) = 5$. The edge set $\{(v_1, v_6), (v_6, v_2), (v_2, v_1)\}$ is a cycle, and the edge (v_6, v_6) is a loop.

Additivity

Consider a tree T, with edge set $E(T)$, and whose pendant vertices are labelled from the set $L = \{1, \ldots, n\}$. Suppose that for each edge $e \in E(T)$, there is a corresponding *length* (or *weight*) $q_e \in \mathbb{R}$. Suppose also that we have a set of *distances* $d_{i,j} \in \mathbb{R}$ between each pair of vertices $\{i, j\} \in V(T)$. The distances $d_{i,j}$ are said to be *additive on* T, if $\forall\, i, j \in L$, $d_{i,j} = \sum_{e \in \pi_{i,j}} q_e$.

A set D of distances $d_{i,j}$ for which there exists a tree T upon which D is additive, then D is said to satisfy the *additive tree hypothesis.*

Bipartitions

A *bipartition* of a set S is a division of the elements of S into two disjoint subsets, say A and A', whose union is S. The *trivial bipartition* of a set S is $\{S, \emptyset\}$.

Each edge of a tree T induces a natural bipartition of the pendant vertices of T, each pendant vertex being either on one side or other of that edge. If each bipartition in a set of bipartitions can be induced by an edge of a single tree T, then the set of bipartitions is defined to be *compatible* [7]. If the bipartitions induced by a set of edges is compatible, then the set of edges is also said to be compatible.

Each bipartition $\{A, A'\}$ of $\{1, \ldots, n\}$ shall be represented by that subset, either

A or A', which does not contain n. For example, the bipartition of $\{1,2,3,4\}$ into the subsets $\{1,3\}$ and $\{2,4\}$ is represented by $\{1,3\}$. With this representation, two bipartitions, say X and Y, $X \neq Y$, are *compatible* if $X \cap Y \in \{X, Y, \emptyset\}$.

This representation is used in the scheme devised by Hendy [40], by which the bipartitions of the set $\{1,\ldots,n\}$ are labelled by the numbers $0,\ldots,2^{n-1} - 1$. Under this labelling scheme, bipartition X is labelled with $\sum_{i \in X} 2^{i-1}$. In a tree T with n pendant vertices, each edge e has the same label as the bipartition of $\{1,\ldots,n\}$ that e induces. Note that the edges incident on the pendant vertices $\{1,2,3,\ldots,n-1,n\}$ are then labelled $\{1,2,4,\ldots,2^{n-2},2^{n-1} - 1\}$, respectively. This edge-labelling system is used throughout this thesis. For convenience, m is defined to be 2^{n-1} henceforth.

Testing two edges labelled x and y for compatibility is then achieved by the bitwise logical "AND" operation: Edges x and y are compatible if and only if x AND $y \in \{x,y,0\}$. Pendant edges are compatible with all other edges. For example, edges labelled with $x = 19 = 010011_{(base\ 2)}$ and $y = 23 = 010111_{(base\ 2)}$ are compatible, because

$$x \text{ AND } y = 010011_{(base\ 2)} \text{ AND } 010111_{(base\ 2)} = 010011_{(base2)} = y,$$

whereas edges labelled with $x = 19 = 010011_{(base\ 2)}$ and $z = 24 = 011000_{(base2)}$ are *not* compatible, because

$$x \text{ AND } z = 010000_{(base\ 2)} = 16 \notin \{19, 24, 0\}.$$

Complexity

If the number of mathematical operations required by a function f acting on a data set of size k is bounded above by a constant multiple of some function $g(k)$ as k increases, then the function f is defined as being *of order* $g(k)$, and is written $O(g(k))$. Hence the "bubble sort" algorithm, which requires $\binom{n}{2}$ operations, is $O(n^2)$, whereas the faster "Shell sort" algorithm is $O(n\ln(n))$ [70].

The number of unrooted binary trees with n pendant vertices is

$$(2n - 5)!! = (2n - 5)(2n - 7)...(3)(1).$$

. By using Stirling's approximation [5] it can be shown that $(2n - 5)!! \sim (2/e)^n n^{n-2}$, so the number of trees with n pendant vertices is said to be *exponential.*

Throughout this thesis the number of taxa under consideration is n, and the sequence length is c.

1.4.2 Phylogenetic terminology

Character sequences

A *character* is some object, common to some or all of the taxa. Here we use characters which are either from the four nucleotides of DNA or RNA molecules or from a two-element set.

Each character is able to be in one of a finite set of *states*, or *colours*. If the characters are nucleotides, the possible character states are A (adenosine), C (cytosine), G (guanine) and either T (thymine) or U (uracil), depending on whether the characters are taken from DNA or RNA molecules, respectively. If there are just two colours they are 0 and 1, which could correspond to pyrimidines and purines ({C, T/U} and {A, G} respectively).

A *character sequence* is an ordered list of character states. I have assumed throughout this study that the character sequences have been obtained and *aligned* correctly, i.e., that the characters (e.g., nucleotides or amino acids) at the same positions in different sequences are homologous. The alignment problem is a separate (and often difficult) issue: for further detail see [4], [35], [57], [94], [99].

Edge lengths

The lengths of edges in phylogenetic trees are usually in $\mathbb{R}^+$, the positive real numbers. However in some methods (e.g., maximum parsimony [26]) a continuous measure is not always appropriate, and discrete (integer) edge lengths are usually used. In the case of maximum parsimony, these inferred edge lengths can be taken as the minimum number of character state changes on each edge which could account for the pattern of character states observed at the pendant vertices of the tree. (This minimum number is evaluated over all possible assignments of character states to the internal vertices of the tree.)

Farris raised an objection to the use of continuous edge lengths [21], but this was based on correlating estimated distances with *actual* events (character state changes), rather than, as was pointed out to be more appropriate by Felsentein [24], [25], with *expected numbers* of events.

Occasionally it is possible to infer edge lengths which are negative (e.g., with the Hadamard conjugation [40]), but these are a theoretical construct, and do not have a readily interpretable biological meaning.

Note that the number of observed character state *differences* between sequences at the ends of a path π or edge e of a tree T is not an additive measure (see above). To obtain additive distances, the number of *changes* of character state must be inferred.

Under the *molecular clock* model of evolution, the distance between the root of a generating tree and each of the pendant vertices is the same. This is equivalent to the rate of evolution being the same on all lineages of the tree, from the root to the extant taxa.

Normally the distance between two character sequences is defined as a weighted sum of the various types of difference, at each site, between the sequences. In a very simple case, the distance between two sequences is the proportion of sites on one sequence which have a different character state at the corresponding site on the other sequence. With four character states, the *type* of difference between sequences is often taken into account: sites at which the difference between corresponding character states require a transition (A $\leftrightarrow$ G, C $\leftrightarrow$ T) can be assigned different weights from those sites which require a transversion (all other changes), depending on the relative rates at which such types of change occur.

In this thesis the distance between two sequences is taken as the proportion of sites at which they differ, whether for two or for four character states. This is also known as the *Hamming distance*.

Under Cavender's model of evolution [10], with just two character states and constant, i.i.d. rates of change, the following formula accounts for multiple character state changes between two sequences:

$$q = -0.5 \ln(1 - 2p),$$

where q is the (inferred) number of character state changes, per character, between the two sequences and $p<0.5$ is the (inferred) number of differences, per character, between the sequences [39].

Under the four-state model, there are several suggested formulae for the estimation of the number of changes. If the rates of character state change are the independent of the states (the Jukes-Cantor 1-parameter model [51]) and the four

states are equally likely, the formula is

$$q = -0.75 \ln(1 - p/0.75)),$$

with p and q defined as before, though now $p < 0.75$.

In general, with r states, exactly one parameter in the transition matrix and constant i.i.d. rates of change,

$$q = \frac{1-r}{r} \ln\left(1 - \frac{rp}{r-1}\right).$$

With the four states having observed proportions which may be unequal between states but are the same for the same states on different sequences, the formula becomes

$$q = -e \ln(1 - p/e),$$

where p and q are as before, and

$$e = 1 - f_A^2 - f_C^2 - f_G^2 - f_T^2,$$

the f's being the proportions of the character states in the two sequences [91]. Note that if $f_A = f_C = f_G = f_T = \frac{1}{4}$, $e = 0.75$, and so the Jukes-Cantor formula is a special case.

There are more complicated formulae for models with more parameters [73], but these models are not used here and are not discussed.

Clusters

A *cluster* is a set of taxa. Often a cluster in a phylogeny is constructed under the hypothesis that it is a monophyletic group with respect to the other taxa under consideration. With unrooted trees, a cluster may also be formed under the hypothesis that its *complement* is monophyletic, with respect to the whole set of taxa. Note that a cluster may also be regarded as a taxon, and that each taxon can be regarded as a cluster of size 1. The *size* of a cluster is the number of taxa in it. A cluster of extant taxa can be represented on a phylogenetic tree T by a subtree T' of that tree, where the pendant vertices of T' correspond to the extant taxa.

Neighbours

The *nearest neighbour* of a taxon in a graph, with respect to a given set of taxa, is that taxon which is the closest taxon to it. The nearest neighbour of a taxon is usually taken from one of the other pendant vertices (i.e., extant taxa).

A *neighbouring pair* of vertices is a pair of vertices of the tree which are adjacent to a single common vertex. In Figure 1.1 vertices v_3 and v_7 are a neighbouring pair.

1.5 Models of evolution

There are many different models of evolution. The models of interest here are those which have some stochastic process operating over time on a set of character sequences (e.g., DNA sequences). Also part of the model is the tree which governs the mechanism by which data were generated, called the *generating tree* and usually denoted here by T_G.

Generally with these models, the probability of a character state change between the vertices at each end of an edge is dependent on the initial state and the final state, this information given in a *transition matrix*. For a transition matrix $\mathbf{M}$ governing the change of state from time t_1 to time t_2 (perhaps on an edge of the tree), entry $m_{i,j}$ of $\mathbf{M}$ is the probability that, given that the character is in state i at time t_1, it will be in state j at time t_2. Such a matrix is called a *stochastic matrix*, and the sum of the elements in each column add to 1. A further constraint is that each entry in these matrices must be non-negative.

These are *Markov models*, and characters changing according to such a model are obeying a *Markov process*.

The transition matrix may vary over the edges of a tree and across sites of the sequences, but in this thesis the transition matrix for a given edge is independent of lineage and sequence site, and is a function of the length of that edge.

For two character states, the transition matrix used here is of the form

$$\begin{bmatrix} 1-x & x \\ x & 1-x \end{bmatrix},$$

while for four character states the transition matrix has been of the form

$$\begin{bmatrix} 1-3x & x & x & x \\ x & 1-3x & x & x \\ x & x & 1-3x & x \\ x & x & x & 1-3x \end{bmatrix}.$$

These models are the Cavender and Jukes-Cantor models ([10] and [51], respectively).

More general models have more parameters, for example Kimura's models, known as the 2P and 3ST models [73], in which there are 2 or 3 arbitrary parameters in each transition matrix, as shown below (the rows and columns are indexed with character states A, C, G, T/U, in that order):

The 2P model:

$$\begin{bmatrix} 1-2x-y & x & y & x \\ x & 1-2x-y & x & y \\ y & x & 1-2x-y & x \\ x & y & x & 1-2x-y \end{bmatrix}.$$

The 3ST model:

$$\begin{bmatrix} 1-x-y-z & x & y & z \\ x & 1-x-y-z & z & y \\ y & z & 1-x-y-z & x \\ z & y & x & 1-x-y-z \end{bmatrix}.$$

The above matrices are all *symmetric*, i.e., the i,j-th entry is identical to the j,i-th entry. The most general symmetric transition matrix is

$$\begin{bmatrix} 1-u-v-w & u & v & w \\ u & 1-u-x-y & x & y \\ v & x & 1-v-x-z & z \\ w & y & z & 1-w-y-z \end{bmatrix}.$$

The most general *asymmetric* transition matrix is

$$\begin{bmatrix} 1-r-u-x & o & p & q \\ r & 1-o-v-y & s & t \\ u & v & 1-p-s-z & w \\ x & y & z & 1-q-t-w \end{bmatrix}.$$

A good discussion of all the above models is to be found in Rodríguez *et al* [73].

In this study the models of evolution used are very simple. In both the "small n" and "large n" cases I have used the Cavender and Jukes-Cantor models as described above, with two and four character states, respectively.

Chapter 2

Phylogenetic Methods

> I am, in point of fact, a particularly haughty and exclusive person, of pre-Adamite ancestral descent. You will understand this when I tell you that I can trace my ancestry back to a protoplasmic primordial atomic globule.
>
> [*The Mikado, Gilbert and Sullivan*]

2.1 Introduction

By the use of phylogenetic methods, we hope to resolve as many as possible of the questions of the relationships between organisms. This chapter first lists some of the properties which are desirable in phylogenetic methods. It is unfortunately not possible for any one method to possess all of the properties listed [66].

The following sections describe the operation of several of the phylogenetic methods studied here. The methods are split into two groups, being *constructive* and *search* methods. General algorithms are provided in pseudocode of the operation of the constructive methods, and of the search methods. Within each description are noted whether the method is consistent with additive data, and its computational complexity.

2.2 Desirable Characteristics of Phylogenetic Methods

As in Penny *et al* [65] we use *consistency*, *efficiency*, *falsifiability*, and *robustness* as desirable characteristics of phylogenetic methods. However, we do not, as Penny *et al* do, use 'power' to measure the ability of phylogenetic methods to correctly infer the tree from which the data were generated, as this term has other meanings in statistics. The probability of a method correctly inferring the generating tree is defined as its *accuracy*.

2.2.1 Accuracy

The *accuracy* of a phylogenetic method is a measure, albeit an imprecise one, of the probability that a method will correctly infer the phylogeny which generated a data set, when using that data set as input. It is imprecise, because the probability that a method will correctly infer the tree which generated a given data set is a complicated, and generally unknown, function of the data. The property of accuracy is the one which is most often sought, since we are usually most concerned with the overall shape of the phylogeny: up to a point, we are willing to wait for a slow but accurate method, as opposed to a fast but inaccurate method.

2.2.2 Consistency

Consistency of a phylogenetic method must be with respect to some model of evolution. It is the property that, when data are generated according to that model, and as sampling error of such data tends to zero, the probability that the correct phylogenetic tree will be inferred tends to unity.

All the methods used here have been tested for consistency or inconsistency (as appropriate) with the molecular clock and additive tree models, described earlier. Consistency is desirable because we do not want to use a method which may give us the wrong tree, if the data fit some known model.

2.2.3 Efficiency

Generally speaking, a method is *efficient* if it is fast, and *inefficient* if it is slow. (Note that this is at odds with some authors, e.g., [50], [80], who deem a method

to be efficient if it has a high probability of returning the correct tree.)

In this general sense, whether a method is efficient or not is determined by its implementation. For example, a search through the $\approx 10^{30}$ trees on 25 pendant vertices may take years on a personal computer, and hence be generally regarded as 'inefficient' in that implementation, or less than a day on a massively parallel supercomputer, hence being regarded as 'efficient'. Speeding up the implementation of an algorithm, for instance by using a branch and bound process to discard more than one tree at a time [28], may well make it more efficient. This has been done for the search methods. In some other studies a conscious decision has been made, to sacrifice accuracy for speed by conducting the search on a proper subset of all potential trees ([16], [80]).

A more precise definition of efficiency can be obtained by considering the computational complexity of each of these methods. The computational complexity of reconstructing phylogenies is described in [34] and [98]. More specific discussions of complexity in tree reconstruction are to be found in [52] and [83]. The computational complexity of each of the methods used in this study is noted within its description.

In general, those methods which require a number of operations which is bounded above by a polynomial function of the size of the input data set are regarded as efficient. These are often referred to *polynomial-time* methods. Those methods for which no implementation is known to make them polynomial-time methods are generally regarded as inefficient.

Note that in the statistical sense, efficiency is related to the amount of information which can be retrieved in a test. An algorithm which is $O(n^n)$ may have a high probability of returning the correct tree, and so may be 'efficient' statistically, but for large n, may take a very long time, and hence would be called 'inefficient' (though accurate) here.

Efficient methods — in the mathematical sense — are of interest because they can provide answers to phylogenetic questions in *a reasonable amount of time.* Though the collection of data may take months or years, the tendency is to infer a phylogeny quickly. However, including the gathering of data, the phylogeny reconstruction can often legitimately take up the bulk of any investigation, particularly if the number of taxa is large. "Large" is a again rather loose term in this case,

as the amount of time required for the inference of a particuar phylogeny is dependent on the method of phylogenetic inference used and its implementation, but most researchers would agree that in the order of 100 taxa or more is "large".

2.2.4 Falsifiability

Falsifiability is required if we are to be able to conduct valid experiments in phylogenetic analysis [69]. It is the property that given a data set and an hypothesized model, a method can *reject* the model if it is not appropriate to the data set. For example, suppose we have a data set, consisting of a matrix of pair-wise distances between taxa, which has arisen from a phylogeny where there has been recombination. In this case the underlying phylogeny cannot correctly be described by a tree. We would therefore like our phylogenetic method to reject the hypothesis that the phylogeny is tree-like.

Note that the clustering methods do not have this property; whatever the data set (up to a point: the distances must be semi-metric (i.e., $d_{j,i} = d_{i,j} \geq 0 \; \forall \; i, j$ and $d_{i,j} = 0 \iff i = j$) [5]), each of the clustering methods used here will provide a tree, even if the input data were not generated from a tree.

We find later that the original closest tree and compatibility methods (CTSH and CoSH respectively — see later) do have this property, if the accuracy of the methods can be estimated in some way, and when used in conjunction with other methods: when data generated from more than one tree are used as input to these two methods, they perform relatively better than do the other methods. Knowing this, if we see such relatively high accuracy of CTSH and CoSH (with respect to the other methods) we can hypothesize that the data set is not pure. Judging the accuracy of the methods could in this case be achieved by bootstrapping sequence data, but has not been investigated here.

2.2.5 Robustness

The *robustness* of a method is a measure of its *insensitivity* to variation in the mechanism by which data are generated, or, in other words, to the model of data generation deviating from the evolutionary model which is implicitly or explicitly assumed by the phylogenetic method. This is desirable because in general we do not know the evolutionary mechanism, and so it is more than likely that any

hypothesized model will be violated to a greater or lesser degree. (Tests of the robustness of maximum likelihood (ML) methods have been carried out by Fukami-Kobayashi and Tateno [30]. They varied the type of transition matrix used in the mechanism of evolution, and the relative frequencies of the character states.)

In this thesis, the models of data generation deviate from the hypothesized models used by the phylogenetic methods in the folowing ways:

- Contamination of data;
- Adherence to a molecular clock;
- Random error.

Contamination of data is modelled here by amalgamating data generated from trees representing two independent phylogenies. This is described in Section 4.2.3, and experimental results given in Section 5.10.

Adherence to a molecular clock is investigated in the "large n" case, by varying the spread of the edge length probability distributions about their expected values. With only a small spread for these probability distributions, the edge lengths closely fit the molecular clock hypothesis, but with a greater spread they can deviate from it. This is described in Section 4.5.4, and experimental results given in Section 6.6.2.

2.3 General Classes of Phylogenetic Methods

We can split the methods used here into pairs of distinct classes. The methods are grouped according to the form of their input data, and how they go about selecting a phylogeny or set of phylogenies. If a method uses pairwise distances (i.e., a set of distances between pairs of the taxa) as its input, it is *distance-based*, whereas if it uses sequences of characters directly, it is *sequence-based*. Distances can arise from sets of morphological characteristics which vary over a continuous range, for example, the lengths of spines on the legs of different types of weta [71]. Distances can also be calculated from sequences, as described in the previous chapter. Character sequences are most commonly sets of DNA, RNA or protein sequences taken from extant taxa. (Though some progress has been made with extracting DNA from fossilized organisms, such sequences tend to be rather short, in the order of 100 base-pairs, due to the fragmentation of DNA over time [61].)

It is important to note that while the original data set may be the same, different methods may transform the data in different ways: Here, the sequence-based methods (CoSH, CoSO, CTSH, CTSO, MPSH, and MPSO — see later) begin with a set of partition frequencies, from which are estimated path lengths and then edge lengths of the generating tree T_G. NJ, SL, ST, TD and UPGMA (see later) obtain a matrix of distances, in this study from the set of partition frequencies described above. Note that distances can be obtained from other sources also, e.g., immunological and morphological.

The last group of methods investigated here, which use a "distance spectrum" (CoDH, CTDH and MPDH — see later), convert the above matrix of distances *back* to a set of inferred partition frequencies. These three methods are also referred to as "distance-based" as they do not require a set of sequences from which to calculate the distance spectrum, but only a distance matrix.

Within this thesis the following methods are studied:

- the *Unweighted Pair-Group Method with Arithmetic Mean* [79], abbreviated *UPGMA*.
- The *transformed distance method* of Li [54], abbreviated *TD* [58];
- the *neighbourliness* method [77], abbreviated *ST* after the initials of its inventors, Sattath and Tversky;
- the *neighbour-joining* method [76], [89], abbreviated *NJ*;
- the *closest tree* method [40], abbreviated *CT*;
- the *compatibility* method, which is abbreviated *Co*;
- *maximum parsimony* [26], abbreviated *MP*;
- *SL*, a simplification of Li's TD method;

We describe these methods generally in the following sections, but wait until Section 4.4.7 to give examples of their operation. This is so the method of data generation can be explained before showing a typical data set, which is then used by each method.

For algorithmic descriptions of each method, refer to Appendix C.

2.4 Constructive methods

Methods which build a phylogeny, by successively constructing pieces of a tree, are called *constructive* methods. With such methods the construction is not reversible: once a decision has been made, e.g., a particular pair of taxa being supposed to be monophyletic with respect to the others, that decision is retained for that tree.

The constructive methods studied here are all *heuristic*, in that they find solutions to local optimisation problems (in this case the choice of the next neighbouring pair), but cannot guarantee that the final solution is globally optimal. Constructive phylogenetic methods are generally efficient, typically $O(n^3)$.

All the clustering methods begin with a set of distance data in a matrix $\mathbf{D}$. (Note that the distances need not be additive on any tree: none of the clustering methods have the ability to reject such distance data.)

Each method has a pair of functions $F(i, j, \mathbf{D})$ and $G(i, j, \mathbf{D})$ which depend on the method. Clusters x and y are chosen as a neighbouring pair if $F(x, y, \mathbf{D})$ is optimal. After a pair of clusters $\{x, y\}$ is chosen, it is replaced by the cluster $z = x \cup y$, and the distance matrix $\mathbf{D}$ is replaced by $\mathbf{D}' = G(i, j, \mathbf{D})$, which again is a matrix of distances, now with one row and one column fewer. This process is continued until there is only one choice for the final tree.

We can write the clustering process in an algorithmic form (see Algorithm 2.1).

Methods which use the above approach require $O(n^3)$ operations to construct a tree. This is because the three nested loops ("**for** ... **do**") each run through $O(n)$ repeats. The methods which use this kind of algorithm, and are therefore $O(n^3)$, are NJ, NJa, SL, TD, and UPGMA.

Algorithm 2.1 : Clustering process (to maximise $F(i,j,\mathbf{D})$)

```
variables:
  n,                          { number of taxa, n ≥ 2 }
  A,                          { the set of clusters currently available }
  C_1, C_2, ..., C_2n         { the clusters }
  k,                          { current number of available clusters }
  i, j;                       { counters }
A ← {1, 2, ..., n}
for i = 1 to n do  C_i ← {i}
k ← n
while (k > 1) do
  i ← 1
  while (i ∉ A) do  i ← i + 1
  x ← i                       { x is the smallest element in A }
  i ← i + 1
  while (i ∉ A) do  i ← i + 1
  y ← i                       { y is the next smallest element in A }
  f ← F(x, y, D)
  for i = x to 2n − k − 1 do
    if (i ∈ A) then
      for j = i + 1 to 2n − k do
        if (j ∈ A) then
          if (F(i, j, D) > f) then
            f ← F(i, j, D)
            x ← i
            y ← j
  C_{2n−k+1} ← C_x ∪ C_y
  A ← A ∪ {2n − k + 1}
  A ← A − {x} − {y}
  D ← G(x, y, D)
  k ← k − 1
end.
```

2.4.1 Unweighted Pair-Group Method with Arithmetic Mean

This method, one of the earliest developed, is almost always abbreviated UPGMA, as it is here [79]. It is also known as the Average Linkage method (AL).

It is consistent with the molecular clock model of evolution. Given data generated by such a mechanism with the added restriction that evolutionary rates are i.i.d. across all sites, and in the absence of sampling error, UPGMA will correctly construct the generating tree (a proof is provided in Appendix B, Theorem 1). (However, the example tree T shown in Section 4.4.7 is not of this form; wherever we choose to place a root on T, we cannot thereby make all the distances from the root to the pendant vertices equal.)

UPGMA is known to be inconsistent with additive data without appropriate adjustment of the input distances to account for varying evolutionary rates [12]. Hence we expect UPGMA to perform badly in this experiment, in fact not to improve its performance beyond a certain imperfect level, as we increase the sequence length c. A proof is provided in Appendix B , Theorem 1 of the consistency of UPGMA with data which satisfy the molecular clock hypothesis. Despite the simplicity of the proof, I am unaware of such a proof in the relevant literature.

In UPGMA, the function F(x, y, **D**) is just $d_{x,y}$. Suppose that at some clustering stage clusters x and y are amalgamated to form cluster z. Then we modify **D** according to

- $G(d_{i,j}) = d_{i,j}$ if i and j are distinct from z;
- $G(d_{z,j}) = G(d_{j,z}) = \frac{\alpha_x d_{x,j} + \alpha_y d_{y,j}}{\alpha_x + \alpha_y}$, where α_x is the size of cluster x, α_y that of cluster y.

UPGMA requires $O(n^3)$ operations to construct a tree on n taxa.

An algorithmic description of UPGMA is given in Appendix C, Algorithm C.23.

2.4.2 Transformed Distance method (TD)

TD works by modifying the distance matrix **D** before using UPGMA to infer a tree. The modification is to take into account differing rates of evolution on different lineages, and is achieved by putting each of the taxa into one of two sets (here called L and R), which are supposed to contain all the taxa on the left- and

ight-hand side of the root, respectively, and then modifying the distances so that hey fit the molecular clock model. The root is assumed to lie between L and R.

First TD finds $\{x, y\}$ such that $d_{x,y}$ is maximal. L is set to the singleton $\{x\}$; R is set to $\{y\}$. The set A of available taxa is then $\{1, 2, \ldots, n\} - \{x, y\}$. The following process is repeated until $L \cup R = \{1, \ldots, n\}$:

- Find $i \in A$ such that $d_{L,i}$ or $d_{R,i}$ is minimal, where $d_{L,k} = \sum_{j \in L} d_{j,k}/|L|$ and $d_{R,k} = \sum_{j \in R} d_{j,k}/|R|$.
- If $d_{L,i}$ is minimal, $L = L \cup \{i\}$, otherwise $R = R \cup \{i\}$.
- $A = A - \{i\}$.

The new distance matrix, $\mathbf{D}'$, is calculated as follows: For each $i \in L$, let $r_i = d_{i,y} - d_{x,y}$, and for each $j \in R$, let $r_j = d_{j,x} - d_{x,y}$. Then the entries $d'_{i,j}$ of $\mathbf{D}'$ are calculated by $d'_{i,j} = d_{i,j} - r_i - r_j$.

These distances are then used as the input data for UPGMA.

TD is not consistent with additive data (see Appendix B:Theorem B.1). It requires $O(n^3)$ operations to construct a tree on n taxa.

An algorithm description of TD is given in Appendix C, Algorithm C.24.

2.4.3 Neighbourliness (ST)

This method of tree construction was invented by Sattath and Tversky in 1977 [77]. In this method, at each clustering step, each potential neighbouring pair of clusters, say $\{x, y\}$, is evaluated to find the number $f(x, y) = f(y, x)$ of quartets $\{i, j, x, y\}$ for which $d_{i,j} + d_{x,y} < d_{i,x} + d_{j,y}$ and $d_{i,j} + d_{x,y} < d_{i,y} + d_{j,x}$. In the context of this method, we call $f(x, y)$ the *support* for the pair $\{x, y\}$. At each stage the pair of clusters $\{x, y\}$ with the highest support is amalgamated to form a larger cluster, say z. We then modify $\mathbf{D}$ according to

- $G(d_{i,j}) = d_{i,j}$ if i and j are distinct from z;
- $G(d_{z,j}) = G(d_{j,z}) = \frac{d_{x,j} + d_{y,j}}{2}$.

Neighbourliness, referred to as ST after the initials of the authors, measures the support for a given potential neighbouring pair $\{x, y\}$ in a very coarse manner: It counts 'one' for each case where $d_{i,j} + d_{x,y} < d_{i,x} + d_{j,y}$ and $d_{i,j} + d_{x,y} < d_{i,y} + d_{j,x}$,

whether the size of the differences between $d_{i,j} + d_{x,y}$ and $d_{i,x} + d_{j,y}$, and between $d_{i,j} + d_{x,y}$ and $d_{i,y} + d_{j,x}$, are small or large.

Suppose we modify ST to take into account the size of these differences by adding up all the terms $(d_{i,x} + d_{j,y}) - (d_{i,j} + d_{x,y})$ for all possible quartets $\{x, y, i, j\}$. We can show that ST modified in this way produces a method, say 'ST+', which is equivalent to the neighbour-joining method, outlined below. The proof of this assertion is described in Appendix B, Theorem 7. Note that the original description of NJ did not rely on this modification of ST, but was developed independently.

ST is consistent with additive data [77].

Because there are $\binom{n}{4} = (n)(n-1)(n-2)(n-3)/24$ possible quartets of a set of n objects, each clustering stage requires $O(n^4)$ operations, and so the complete method requires $O(n^5)$ operations in this implementation. Note that ST+, if implemented in the same general way as ST has been, would also require $O(n^5)$ operations, while NJ, equivalent to ST+, requires only $O(n^3)$ operations.

An algorithmic description of ST is given in Appendix C, Algorithm C.22.

2.4.4 Neighbour-joining (NJ)

This method was developed by Saitou and Nei in 1987 [76] and a modification was provided by Studier and Keppler in 1988 [89] which rendered a method consistent with additive data. It is this modified method which is referred to throughout this thesis as 'NJ'.

At each clustering stage, the neighbour-joining method chooses $\{x, y\}$ to minimise

$$F(x, y, \mathbf{D}) = d_{x,y} - \frac{1}{n' - 2} \sum_{i \in A} (d_{x,i} + d_{y,i}),$$

where A is the set of clusters currently available and $n' = |A|$. The quantity $\sum_{i \in A} d_{x,i}$ is called the *net divergence* v_x of vertex x.

In Appendix B, Theorem 5, a proof (from Charleston *et al* [12]) is provided that NJ uses the only possible consistent weighting scheme for the net divergence.

Suppose we choose clusters x and y to amalgamate to form cluster z. Then $\mathbf{D}$ is modified according to

- $G(d_{i,j}) = d_{i,j}$ if i and j are distinct from z;
- $G(d_{z,j}) = G(d_{j,z}) = \frac{d_{x,j} + d_{y,j} - d_{x,y}}{2}$.

NJ is consistent with additive data [89], and requires $O(n^3)$ operations to construct a tree on n taxa.

An algorithmic description of NJ is given in Appendix C, Algorithm C.21.

2.5 Search methods

Those methods with an explicit criterion which is optimised over a set of possible phylogenies are called *search* methods.

Search methods evaluate some function, M say, on all or some of the set of possible phylogenies. If M is evaluated on all potential phylogenies, the implementation is an *exhaustive* search. If a proper subset of potential phylogenies is tested, the search is *heuristic*.

Since the number of potential phylogenies grows exponentially with the number of taxa, there comes a point where it is infeasible to conduct an exhaustive search, and then heuristic methods must be used. It is not within the scope of this study to investigate the performance of phylogenetic methods in conjunction with heuristic search methods.

All the search methods described here use as their input data a set of *inferred edge lengths*, which is usually denoted $\mathbf{q}'$. The ways in which $\mathbf{q}'$ is inferred, in this study, from the observed data are described in Section 4.4.6.

2.5.1 Closest Tree (CT)

The Closest Tree method of Hendy [40], [47] minimises the Euclidean distance between an input spectrum of inferred edge lengths $\mathbf{q}'$ and the spectrum of edge lengths which could correspond to a given tree T.

This distance is minimised when

$$M(T) = \sum_{e \in T} q_e'^2 - \frac{1}{2n-2}\left(\sum_{e \in T} q_e' + q_0'\right)^2 \quad \left(q_0' = -\sum_{1 \le i < m} q_i'\right)$$

is maximised.

As originally presented, the inferred edge lengths are obtained from the observed bipartition frequencies using a *Hadamard conjugation* which relates probabilities $\mathbf{s}$ of observing partitions of the taxa to edge lengths $\mathbf{q}$, for a tree T. This relation is described in Section 4.4.3. (Though the method was originally developed to deal with two character states, a four character state version is now available [46].)

Note that the set of inferred edge lengths could be obtained using some other method; the Hadamard conjugation is not peculiar to CT, nor is CT peculiar to the Hadamard conjugation (Co also uses the conjugation).

The three ways in which the observed data are treated to obtain inferred edge lengths $\mathbf{q}'$ are distinguished by the suffixes 'DH' (Distances with Hadamard conjugation), 'SH' (Sequences with Hadamard conjugation) and 'SO' (Sequences, Observed). These ways of inferring $\mathbf{q}'$ are described in detail in Section 4.4.6.

The Hadamard conjugation requires $O(n2^n)$ operations to generate $\mathbf{q}'$, the number of potential trees to test ($(2n-5)!! = (2n-5)(2n-7)...(3)(1)$) tends to a constant multiple of $(2/e)^n n^{n-2}$, and the evaluation of $M(T)$ for each tree T then requires $O(n)$ operations. Hence in the worst possible case CT, using the Hadamard conjugation, could be $O(n2^n + (2/e)^n n^{n-1})$. With branch and bound methods [28], the number of trees for which the evaluation of $M(T)$ must be made can be reduced substantially (see Section 2.5.4, also [64]).

The branch and bound implementation of CT is described in Section 2.5.4, and an example of the saving of computing time thus achieved is shown in Table 2.1. An algorithmic description of CT is given in Appendix C, Algorithm C.25.

2.5.2 Compatibility method (Co)

The Compatibility method is introduced here to take advantage of the spectrum of inferred edge lengths which is provided by the Hadamard conjugation, outlined above.

Co should not be confused with compatibility used in connection with parsimony. In terms of parsimony, each character is defined as compatible *with a phylogeny if it need not arise more than once to give the observed data, and the phylogeny with which the most characters are compatible is taken as the true phylogeny* [91].

In the context of Co, the *support* for each edge is defined as its estimated length (taken directly from the input vector $\mathbf{q}'$), and the support for a set of compatible edges is the sum of the supports of these edges. The set of compatible edges which has maximal support is chosen as the set of edges of the tree. This method is described elsewhere but has not before been applied to these inferred edge lengths as derived by the Hadamard conjugation method. A mathematical description of Co is given in Chapter 3.

With four taxa (but not more than four), compatibility is equivalent to maximum parsimony (they both choose T to maximise the inferred length of the only internal edge)

Co is of the same computational complexity as CT (see Section 2.5.4).

Compatibility is readily amenable to branch and bound search [28], [43], which can make the search of potential trees much faster, as described in Section 2.5.4. The saving of computing time achieved by using branch and bound searching is demonstrated in Table 2.1. An algorithmic description of Co is given in Appendix C, Algorithm C.26.

The consistency of Co is discussed in Section 4.4.6.

Note that the quantity minimised by Co is a 1-norm (the "city-block distance"), and that minimised by CT is a 2-norm (the Euclidean distance) [78]. These optimality criteria could thus be considered to be members of a larger class of methods, which choose T as the 'best' tree if $d_k(\mathbf{q}(T), \mathbf{q}(T_i))$ is minimised, where $\mathbf{q}(T)$ is a set of edge lengths corresponding to tree T, $\mathbf{q}(T_i)$ is the set of edge lengths inferred from the observed data, and d_k is the k-norm for the two sets of edge lengths.

This larger class of methods has not yet been investigated as such.

2.5.3 Maximum Parsimony (MP)

The maximum parsimony criterion (MP) [9] chooses the trees which minimise the overall number of character state changes which would be required on the edges of the tree to account for the observed character sequences. This minimum number of character state changes for a given tree is known as the *parsimony length* of the tree. Given a partition of the taxa into two character states and a tree T, Fitch's algorithm [26] finds the parsimony length of T in $O(n)$ operations. This length is then evaluated for all bipartitions in the data, and, for a complete search, all possible trees. Hence MP is a double-minimization procedure: the minimum number of character state changes is found for each tree T, and the tree or set of trees for which this number is minimised is chosen to estimate the true phylogeny.

Due to the large number of potential trees, some studies have limited the search to testing only those trees which are 1 edge different from the current one, in a "hill-climbing" approach [80].

It has been shown that MP can be consistent with additive data if the observed bipartition spectrum is adjusted to account for multiple changes of character state

on evolutionary paths [88]. When this adjustment is made via the Hadamard conjugation [45] we refer to the method as MPSH (Maximum Parsimony, Sequence Hadamard); when the observed bipartition spectrum is used, the method is referred to as MPSO (Maximum Parsimony, Sequence Observed). MPSO is the maximum parsimony method commonly used in previous studies.

It is known that MPSO is not consistent with sequence data following the Jukes-Cantor model [22], [51].

The Fitch algorithm requires $O(n)$ operations to evaluate the minimum number of character-state changes which could account for each bipartition in the observed data. The maximum number of distinct non-trivial bipartitions which can be observed is $\min\{2^{n-1} - 1, c\}$, and hence with the $O((2/e)^n n^{n-2})$ potential trees the MP methods are of order at most $O((2/e)^n (n)^{n-1} k)$, where $k \leq \min\{2^{n-1} - 1, c\}$ is the number of distinct non-trivial bipartitions. If the Hadamard conjugation is being used to infer expected frequencies of the $m = 2^{n-1} - 1$ non-trivial bipartitions, these expected frequencies may all be positive, in which case the MP methods require at most $O((2/e)^n (2n)^{n-1})$ operations.

In practice the number of operations required is generally much less than the above upper bounds (see Section 2.5.4).

An algorithmic description of MP is given in Appendix C, Algorithm C.33.

2.5.4 Branch and bound implementations

The search methods described above (Co, CT and MP) must effectively search all potential trees to find the tree(s) which is (are) optimal, for the optimality criterion used. Suppose this optimality criterion is evaluated for tree T by a function $M(T)$.

There are $(2n - 5)!! = (2n - 5)(2n - 7)...(3)(1) \sim (2/e)^n n^{n-2}$ possible unrooted binary trees, so if it is necessary to evaluate M on each of these, the search will take an impractically long time for even moderate values of n, but there are alternatives in some cases. If M can be broken down into a linear or other simple combination of functions which can be evaluated on parts of a tree, for example on the edges, then we can use a *branch and bound algorithm* to reduce the number of trees that need be examined. For a general description of branch and bound methods, see [28].

CT, Co and MP are amenable to this treatment, which can speed up the process considerably [43], depending on the input data. The next section describes the

branch and bound method used in this study for the compatibility and closest tree methods, and the subsequent section describes the branch and bound method used for maximum parsimony. Note that both techniques effectively consider all potential trees, so are equivalent to exhaustive searches. Exhaustive search, even with branch and bound techniques, is generally not possible for n more than 20-30, depending on the data and the implementation.

Note also that the input vector may be either a set of observed bipartition frequencies, the bipartition frequencies inferred by the Hadamard conjugation, or bipartition frequencies inferred from distances. The algorithms are the same for each type of input data: the input vector is simply referred to here as $\mathbf{q}'$.

Branch and bound for compatibility and closest tree

With the closest tree methods the quantity to be maximised,

$$\sum_{e \in T} q_e'^2 - \frac{1}{2n-2}\left(\sum_{e \in T} q_e' + q_0'\right)^2,$$

will be maximised when

$$M(T) = \sum_{e \in \text{ internal edges of } T} q_e'^2 - \frac{1}{2n-2}\left(\sum_{e \in T} q_e' + q_0'\right)^2$$

is maximised. Note that

$$\sum_{e \in T} q_e' + q_0' = \sum_{e \in \text{ internal edges of } T} q_e' + \sum_{e \in \text{ pendant edges of } T} q_e' + q_0'.$$

The latter two terms in the above expression are calculated at the outset, as they are independent of T, and their sum included in the algorithms as the variable "*essentials*".

In the compatibility methods, the quantity to be maximised is

$$\sum_{e \in T} q_e',$$

which is maximised when

$$M(T) = \sum_{e \in \text{ internal edges of } T} q_e'$$

is maximised.

The first tree considered, say T_0, is found by a "greedy algorithm": The vector of inferred edge lengths is sorted in descending order, discarding the pendant edges (in practise, the input edge lengths $\mathbf{q}'$ are copied to a temporary array, and the pendant edge lengths set to -1 in this array). Let the set of internal edges of the current tree (initially T_0) be S. The edge whose input length is largest is included in the internal edges of T_0, so is the first element of S. Subsequent edges from the sorted list are included in S if they are compatible with all the edges already in S. This constructs a set of $(n - 3)$ compatible edges, which are the internal edges of T_0. Let the greatest value of $M(T)$ found be B, so B is initially $M(T_0)$.

Note that the lengths of the internal edges of T_0 are stored in decreasing order.

The general principle of the branch and bound process for closest tree and compatibility is of keeping a 'core' array of edges S and stepping a candidate edge, say e, through the sorted array of inferred edge lengths (of those edges not already in the 'core'). At each step, an upper bound, say b, on the value of $M(T)$ with $(S \cup \{e\}) \in E(T)$ is calculated. For brevity, let

$$M(S) = max(M(T) : S \subseteq E(T)),$$

with the maximization over all possible trees T.

If $b < B$, or if no compatible edge with positive inferred length can be found, the last element of S is removed, and stepped through the array of inferred edge lengths, as above. If $b \geq B$ and $|S| < (n - 3)$, e is appended to S. If $b > B$ and $|S| = (n - 3)$ (i.e., the tree is completely resolved), then S is stored (as "*best*S") and B is set to b. When $|S| = 0$, the search is complete.

(In practise, the entries of S are the *positions of the edge labels* in the sorted vector of inferred edge lengths.)

Note that when the edge lengths are short, and few multiple changes of character state occur on the internal edges of T_G, the inferred q'_i's which do not correspond to the internal edges of T_G will be small. Hence the entries in the sorted $\mathbf{q}'$ vector will decrease rapidly after the first $(n - 3)$. This means that the branch and bound process will be able to reject larger sets of candidate trees than if the edge lengths were longer and the decrease in the sorted $\mathbf{q}'$ vector less rapid. Therefore with shorter edge lengths, we expect the branch and bound search to be relatively faster than with long edge lengths.

Algorithm 2.2 describes the branch and bound process for closest tree and compatibility. The saving in computational cost for an example case is shown in

Table 2.1.

Algorithm 2.2 : Branch and bound for Co and CT

variables:

b,	{ the best possible value for $M(T)$ with the current edge set S }
bestS	{ the set of $n-3$ edges which gives the best $M(T)$ }
B,	{ the best value found for $M(T)$ with a fully resolved tree }
i, j, k,	{ counters }
S,	{ current set of edge labels, stored as array $[S_1, ..., S_{n-3}]$ }
S',	{ temporary storage for S }
optimal,	{ flag }
q',	{ vector of input edge lengths }
r,	{ vector of ranks of **q'**, so q'_{r_i} is the i-th largest input edge length }
v.	{ a temporary vector of the input edge lengths }

copy input edge lengths **q'** into **v**
set pendant edge lengths in **v** to -1
sort **v** in descending order, and put the ranks of **v** into **r**.
$S_1 \leftarrow 1$ { note that r_1 is the label of the edge with the largest inferred length }
$j \leftarrow 2$
for $i = 2$ **to** $n-3$ **do**
  while (r_j is not compatible with S) **do** $j \leftarrow j+1$
  $S_i \leftarrow j$
$bestS \leftarrow S$
$S' \leftarrow S$
$B \leftarrow M$(**bestS**) { here $M(X)$ is max$\{M(T)$ such that X is the set of internal edges of $T\}$ }
optimal $\leftarrow$ FALSE
$fixed \leftarrow n-4$

while (*optimal* is FALSE) **do**

```
    k ← fixed
    while (k ≥ 0) do
    j ← k + 1
    do
      j ← j + 1
      b ← M(S ∪ {r_j})
    while (b ≥ B) and (j < 2^(n-1)) and (r_j is not compatible with S)
    if ((b ≥ B) and (j < 2^(n-1))) then
      S_k ← j
      if ((k = n − 3) and (S' ≠ S)) then
        S' ← S
        B ← b
        bestS ← S
      k ← k + 1
    if ((b ≤ B) or (k > n − 3)) then
      fixed ← fixed − 1
      break
  if (fixed < 0) then optimal ← TRUE
end.
```

Branch and bound for maximum parsimony

Branch and bound with maximum parsimony in this thesis uses a different approach: The evaluation of the objective function $M(T)$ cannot be split into simple functions of the inferred edge lengths, but rather must be evaluated for a complete (and in this case fully resolved) tree.

The process begins with the tree on pendant vertices labelled $(n-2), (n-1)$ and n. Initially the internal vertex is labelled $(n+1)$. All internal vertices of the current tree are labelled from $(n+1)$ up to $(2n-2)$.

Pendant vertices are added sequentially to the tree in all possible positions and the parsimony length of the new tree found. If this length is more than the previous best bound, the next position is tried for the insertion. If there are no more positions available, the pendant vertex with the lowest label is removed, and the pendant vertex with the next lowest label is moved along.

This process continues until no pendant vertices can be moved any further.

Note that the edge lengths which are used by compatibility and closest tree are inferred numbers of character state changes on edges of a tree. They are therefore equivalent to bipartition frequencies, so the same data sets used by compatibility and closest tree can be used by maximum parsimony also.

In fact, complete evaluation of the parsimony length of the current tree T is not necessary if it is known that the length will exceed the shortest length found previously: if that is the case, the T is already 'too long' and will be rejected. Hence the spectrum of bipartition frequencies which is fed into MP is sorted in decreasing order, as it is with the branch and bound method for compatibility and closest tree, above, and the ranks of the bipartition frequencies stored in r. The contribution of a given bipartition to the parsimony length of T will in general, though not uniformly, decrease with decreasing frequency of the bipartitions. Therefore the major contribution to the parsimony length of T will be from the first few bipartitions indexed by r. The Fitch algorithm [26] is evaluated on the bipartitions indexed by $r_1, r_2, \ldots$, until either the parsimony length of T exceeds the current best bound or the last entry of r is reached.

Algorithm 2.3 describes the branch and bound process for maximum parsimony.

Algorithm 2.3 : Branch and bound for MP

local variables:

T,	{ current tree, stored as an array of pointers }
bestT,	{ best tree found so far }
B,	{ current shortest length of a tree }
length,	{ current parsimony length of the tree }
taxon,	{ number of the taxon being added, initially $(n - 4)$ }
newnode,	{ label of the new internal vertex we create by inserting a taxon }
optimal,	{ flag to tell when to stop }
$insertpos[N]$,	{ current position of insertion of each taxon }
i, j,	{ counters }
u.	{ number of bipartitions in input data with positive frequency }

$B \leftarrow$ parsimony length of T_G
sort **q** in descending order, and put the ranks of **q** into **r**.
$u \leftarrow$ number of bipartitions
set up first tree
$taxon \leftarrow n - 3$ { begin by including this taxon }
$insertpos_{taxon} \leftarrow n + 1$ { insert in positions in decreasing order }
$newnode \leftarrow n + 2$
{ the name of the new node we'll create by the insertion }
for $i = 1$ **to** $n - 4$ **do** $insertpos_i \leftarrow 2n - 2 - i$
$optimal \leftarrow$ FALSE
while (*optimal* is FALSE) **do**
  $length \leftarrow 0$
  $i \leftarrow 1$
  while $(length < B)$ **do**
    $length \leftarrow length +$ (Fitch length of T with bipartition r_i) $\times\ q'_{r_i}$
    $i \leftarrow i + 1$
  if $(length \leq B)$ **then**
    if $((taxon = 1)$ **and** $(length < B))$ **then**

```
    bestT ← T
    B ← length
  { Now we need to get the next tree: }
  if ((length > B) or (taxon = 1)) then
    while ((insertpos_taxon = taxon + 1) and (taxon < n − 2)) do
      remove taxon from T
      taxon ← taxon + 1
    if (taxon = n − 2) then optimal ← TRUE
    else
      newnode ← 2n − 1 − taxon
      if (insertpos_taxon > taxon + 1) then
        remove taxon from T
        find the next available insert point: put this into insertpos
        insert taxon at position insertpos_taxon in T
  else
    taxon ← taxon − 1
    newnode ← 2n − 1 − taxon
    insertpos_taxon ← 2n − 2 − taxon
    do
      insertpos_taxon ← insertpos_taxon − 1
    while (tree_insertpos_taxon = taxon + 1) and (insertpos_taxon > taxon)
    insert taxon at position insertpos_taxon in T
  if (taxon = n − 2) then optimal ← TRUE
end.
```

Savings from branch and bound

The above branch and bound algorithms afford a substantial saving in the number of times the function M need be evaluated (see Table 2.1, below).

2.6 A note on some other methods

There are numerous other methods which are not studied here. Some of these are *maximum likelihood* (ML) methods [33]. Another related group are the Fitch-Margoliash (F-M) methods [17], [27]. These have not been included due to the

Table 2.1: The proportion of trees tested using branch and bound.
In this table are shown the proportion of times, out of the number of potential trees, that M was evaluated for a single branch and bound search on each of several generating trees. The tree topologies used were UB2, UB3, ..., UB8, being the "caterpillar" topologies on 4, 5, ..., 10 pendant vertices, respectively. The sequence length used was 1000 characters. Note that the MP methods require more evaluations than do Co and CT of the function M: this is due to the branch and bound algorithm for MP beginning at a certain fixed tree and requiring to finish at another fixed tree, abandoning some trees on the way, while the branch and bound for Co and CT begins with the "greedy tree", which is often very close to the generating tree T_G.

T_G	**CoDH**	**CoSH**	**COSO**	**CTDH**	CTSH
UB2	0.33333333	0.33333333	0.33333333	0.33333333	0.33333333
UB3	0.13333333	0.13333333	0.13333333	0.13333333	0.13333333
UB4	0.04761905	0.02857143	0.02857143	0.03809524	0.02857143
UB5	0.00740741	0.00740741	0.00952381	0.00529101	0.00529101
UB6	0.00105820	0.00115440	0.00115440	0.00105820	0.00096200
UB7	0.00012580	0.00013320	0.00016280	0.00008880	0.00008880
UB8	0.00000839	0.00001184	0.00001628	0.00000641	0.00000641

T_G	**CTSO**	**MPDH**	**MPSH**	**MPSO**
UB2	0.33333333	1.00000000	1.00000000	1.00000000
UB3	0.13333333	0.20000000	0.20000000	0.46666667
UB4	0.02857143	0.06666667	0.06666667	0.37142857
UB5	0.00740741	0.03068783	0.03068783	0.03068783
UB6	0.00105820	0.00394420	0.00933141	0.01240981
UB7	0.00008880	0.00216820	0.00447700	0.01275021
UB8	0.00000691	0.00005377	0.00007351	0.00010607

large amount of computation time which is required for ML and F-M methods [91].

In general, ML methods attempt to optimise for each tree a set of edge parameters, which are allowed to vary continuously, to find those which would have the highest probability of yielding the observed data. It is implicitly (and sometimes explicitly) assumed in many ML computer programs that the edge parameter space is unimodal for a given tree, with respect to this probability. This would mean that simple hill-climbing methods would find the maximum likelihood point P correctly [30], [91]. However, this is now known to be false [86] and so this approach cannot guarantee to find the ML tree.

F-M methods minimise

$$\sum_{i,j} w_{i,j} \, |d_{i,j} - p_{i,j}|^{\alpha_{i,j}},$$

where $d_{i,j}$ is the observed distance between taxa i and j, $p_{i,j}$ is the distance between taxa i and j which would be predicted by the tree, $\alpha_{i,j}$ is some real power (1 or 2 in practise), and $w_{i,j}$ is the weighting assigned to the pair $\{i, j\}$. The summation extends over all pairs of taxa $\{i, j\}$.

There are algorithms which can be implemented to minimise the above sum in a reasonable amount of time ($O(n^2)$ per tree), but these are special cases, in which $w_{i,j} = 1 \; \forall \, i, j$ or $w_{i,j} = 1/d_{i,j}$. The choice of $w_{i,j}$ is made according to the assumed characteristics of the variance of observed distances; this variance must be estimated or inferred from sources other than the distances themselves (for example, bootstrapping the observed sequences [20]), as we only have one distance value for each pair of taxa.

The choice of $\alpha_{i,j}$ is somewhat arbitrary; there seems little reason to suppose that the best results should come from using either $\alpha_{i,j} = 1$ or $\alpha_{i,j} = 2$; in fact in different circumstances each may have advantages over the other.

Perhaps for the most consistently reliable results some fractional value should be used for $\alpha_{i,j}$; this question is not easily answerable with available computing resources, and is not studied here.

Chapter 3

New Methods

> In developing our methods, it is vital that we nourish that intellectual variability in approach without which, as every biologist knows, evolution cannot proceed. The computer represents the gravest threat yet to that variability, and at the same time as we enjoy the tremendous advances in calculating power that it provides, we must consciously strive to defend our subjects from the dead hand of automation and the narrow logic of programmers.
>
> [*A. W. F. Edwards* [19]]

3.1 Why find more methods ?

It is important to investigate the properties of current methods, and to attempt if possible to improve upon them. By developing new methods and evaluating the relationships between them, more can be understood about their behaviour. Felsenstein, in 1979, studied the interrelationships among a set of maximum likelihood methods: the interested reader is directed to [23].

There is a danger of producing too many methods though — there are over 100 available currently [48] — and "new" methods may easily be simple modifications of others. For this reason too it is important to understand how proposed methods fit in with those already in existence. Some of the relationships between the methods studied in this thesis are shown in Table 3.1.

Some 'new methods' are modifications to the treatment of the original data, for example the Hadamard conjugation, the use of the distance spectrum, and the

Table 3.1: A classification of some phylogenetic methods

The relationships between some phylogenetic methods are shown. The methods in italics are inconsistent with the evolutionary model used in Chapter 5. All are consistent with the molecular clock model with additive distances. Those methods which have not previously been described are in bold type. Where one method can be considered to be a modification of another, this is indicated by an arrow in the direction of the most obvious modification.

Input data	Search methods			Constructive methods	
partition spectrum	CTSH *CTSO*	**CoSH** *CoSO*	**MPSH** *MPSO*		
distance matrix	CTDH	**CoDH**	**MPDH**	*UPGMA* (↓) *TD* (↓) **SL**	ST (↓) NJ (↓) **NJa**

Jukes-Cantor correction for multiple changes [41], [51].

Other modifications are new methods of selecting a tree based on inferred edge lengths, for example the closest tree, compatibility, Fitch-Margoliash and maximum parsimony methods. These methods have different optimality criteria, to be evaluated on a set of trees. Hence while maximum parsimony here can take three different types of input data, the essential operation of MP is the same.

3.2 The Distance Spectrum

The bipartitions B_i of $\{1, 2, \ldots, n\}$ can be listed in order, using the indexing system outlined in Section 1.4.1 and described fully in [40].

For each B_i we define A_i such that

$$A_i = \begin{cases} B_i \text{ if } |B_i| \equiv 0 \pmod 2; \\ B_i \cup \{n\} \text{ if } |B_i| \equiv 1 \pmod 2 \end{cases}$$

For example, with $n = 6$, this gives the $m = 2^{6-1} = 32$ bipartitions listed in Table 3.2.

The A_i's are all the subsets of $\{1, \ldots, n\}$ which are *even-ordered*, i.e., they have an even number of elements. For each non-empty even-ordered subset A, it is possible to match the elements of A in pairs. Let such a matching of pairs of elements of a given A with $|A| = 2p > 0$ be denoted $\{\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_p, b_p\}\}$, and suppose that $d_{x,y}$ is a real-valued function defined on the elements of A. The *minimum pathset distance* for a given even-ordered subset $A : |A| = 2p$ is then

Table 3.2: The bipartitions and even-ordered subsets of $\{1,\ldots,6\}$

This table lists the 32 bipartitions B_i of $\{1,\ldots,6\}$ in order, according to the indexing scheme of Hendy [40] (see Section 1.4.1), and the subsets A_i of $\{1,\ldots,6\}$ which have an even number of elements.

i	B_i	A_i
0	$\emptyset$	$\emptyset$
1	$\{1\}$	$\{1,6\}$
2	$\{2\}$	$\{2,6\}$
3	$\{1,2\}$	$\{1,2\}$
4	$\{3\}$	$\{3,6\}$
5	$\{1,3\}$	$\{1,3\}$
6	$\{2,3\}$	$\{2,3\}$
7	$\{1,2,3\}$	$\{1,2,3,6\}$
8	$\{4\}$	$\{4,6\}$
9	$\{1,4\}$	$\{1,4\}$
10	$\{2,4\}$	$\{2,4\}$
11	$\{1,2,4\}$	$\{1,2,4,6\}$
12	$\{3,4\}$	$\{3,4\}$
13	$\{1,3,4\}$	$\{1,3,4,6\}$
14	$\{2,3,4\}$	$\{2,3,4,6\}$
15	$\{1,2,3,4\}$	$\{1,2,3,4\}$
16	$\{5\}$	$\{5,6\}$
17	$\{1,5\}$	$\{1,5\}$
18	$\{2,5\}$	$\{2,5\}$
19	$\{1,2,5\}$	$\{1,2,5,6\}$
20	$\{3,5\}$	$\{3,5\}$
21	$\{1,3,5\}$	$\{1,3,5,6\}$
22	$\{2,3,5\}$	$\{2,3,5,6\}$
23	$\{1,2,3,5\}$	$\{1,2,3,5\}$
24	$\{4,5\}$	$\{4,5\}$
25	$\{1,4,5\}$	$\{1,4,5,6\}$
26	$\{2,4,5\}$	$\{2,4,5,6\}$
27	$\{1,2,4,5\}$	$\{1,2,4,5\}$
28	$\{3,4,5\}$	$\{3,4,5,6\}$
29	$\{1,3,4,5\}$	$\{1,3,4,5\}$
30	$\{2,3,4,5\}$	$\{2,3,4,5\}$
31	$\{1,2,3,4,5\}$	$\{1,2,3,4,5,6\}$

defined to be

$$\min_{\text{all matchings } \{\{a_1,b_1\},\ldots,\{a_p,b_p\}\} \text{ of A}} \left(\sum_{j=1}^{p} d_{a_j,b_j} \right).$$

The *distance spectrum* g is defined such that for $i = 1,\ldots,m = 2^{n-1}-1$, g_i is the minimum pathset distance of $A_i \subseteq \{1,\ldots,n\}$. If the set of distances $d_{x,y}$ used is taken from observed data, the distance spectrum is denoted by g′.

The calculation of g can be achieved in less time than that which may at first be apparent from the above description. For $n = 6$, $A_{31} = \{1,2,3,4,5,6\}$, and when matching the elements of A_{31}, 1 can be paired with 2, 3, 4, 5 or 6. If we fix the pair $\{1,2\}$, the other elements can be paired up in 3 ways, and the component of g which corresponds to the minimum pathset distance of $\{3,4,5,6\}$ is g_{28} (as $A_{28} = \{3,4,5,6\}$). Hence we find that $g_{31} = min\{d_{1,2} + g_{28}, d_{1,3} + g_{26}, d_{1,4} + g_{22}, d_{1,5} + g_{14}, d_{1,6} + g_{30}\}$.

We exploit this principle in Algorithm C.16 ("`get_pathsets`"), listed in Appendix C, which calculates the minimum pathset distances of all the sets B_k for $k = 1,\ldots,m-1$ in one pass, hence requiring $O(2^n)$ operations. This algorithm is adapted from a procedure in Hendy and Penny's "`HadTree`" program.

An important property of the Hadamard conjugation is that multiplication by the Hadamard matrix $\mathbf{H}_n$ converts from a vector of *edge lengths* s′ to a vector of *path lengths* ρ. The entries of ρ are the minimum pathset distances of the even subsets of $\{1,\ldots,n\}$, as inferred from a bipartition spectrum, whereas the entries of g are minimum pathset distances as inferred from a set of observed pair-wise distances. Either ρ or g can be used to infer a set of edge lengths [45] of the generating tree T_G, and these inferred edge lengths can be used as the input data for Co, CT and MP.

3.3 Compatibility — again

Compatibility is introduced to take advantage of the bipartition and quadripartition spectra which are generated using the Hadamard conjugation. Instead of minimising the Euclidean distance between the spectrum s calculated from the observed data, and that which would be expected if the generating tree were, say, T, Co simply finds T to maximise the sum of the inferred edge lengths of T. This optimisation function is slightly quicker to evaluate than CT (though both are $O(n)$

for each tree).

The Co method is consistent with an appropriate estimation of the edge lengths of the generating tree, as shown in Appendix B.

In Co, the function $M(T)$ is the sum of the estimated lengths of T, so

$$M(T) = \sum_{e_i \in T} q'_i.$$

This maximises the sum of the inferred edge lengths , and like CT, can take a distance spectrum (CoDH) or the original sequence spectrum (CoSH or CoSO) as its input, or a set of inferred edge lengths obtained in some other way.

3.4 SL

This is proposed as a modification to TD, and is consistent with additive data (see Appendix B, Theorem 4), whereas TD is not (see Appendix B, Theorem 2). It is a simplification of Li's original method, hence the name. (TD is described in Section 2.4.2.)

As in TD we have sets L and R which we aim to compose from taxa which are on the left- and right- hand sides, respectively, of the root. The only difference between TD and SL is in this assignment of taxa to L and to R. First SL finds x, y such that $d_{x,y}$ is maximal. The following process is then repeated until $L \cup R = \{1, \ldots, n\}$:

- Let A be the set of currently available taxa.
- Find $i \in A$ such that $d_{x,i}$ or $d_{y,i}$ is minimal.
- If $d_{x,i}$ is minimal, $L = L \cup \{i\}$, otherwise $R = R \cup \{i\}$.

A new distance matrix is calculated in the same way as for TD: For each $i \in L$, let $r_i = d_{i,y} - d_{x,y}$, and for each $j \in R$, let $r_j = d_{j,x} - d_{x,y}$. Then the entries $d'_{i,j}$ of $bf\ D'$ are calculated by $d'_{i,j} = d_{i,j} - r_i - r_j$. These modified distances $\mathbf{D}'$ are fed into UPGMA.

SL requires $O(n^3)$ operations to construct a tree on n taxa. An algorithmic description of SL is given in Appendix C, Algorithm C.24.

3.5 NJa

Suppose a pair of pendant vertices $\{x, y\}$ of a tree T, are adjacent to internal vertex z. For each other (pendant) vertex k the neighbour-joining method estimates the distance $d_{z,k}$ with $(d_{x,k} + d_{y,k} - d_{x,y})/2$. If distances on the tree are additive, this will correctly find $d_{z,k}$.

Another estimate of $d_{z,k}$ can be made by first obtaining an estimate of the edge lengths $q_{(x,z)}$ and $q_{(y,z)}$. This can be achieved by find the average, say β, of the difference between $d_{x,j}$ and $d_{y,j}$, for all other pendant vertices j. Then

$$q_{(x,z)} = \frac{d_{x,y} - \beta}{2},$$

and

$$q_{(y,z)} = d_{x,y} - q_{(x,z)}.$$

Hence, for all other vertices k, $d_{z,k} = d_{x,k} - q_{(x,z)}$ and $d_{z,k} = d_{y,k} - q_{(y,z)}$.

In the case of additive distances, $d_{x,k} - d_{y,k}$ will be equal for all k, so in this case NJ and NJa will be identical. This proves that NJa is consistent with additive data, as NJ is.

The extra calculations described above are $O(n^2)$, but this does not increase the overall complexity of NJa above $O(n^3)$.

This modification of NJ unfortunately does not perform well.

Chapter 4

Experimental methods

> Computers are useless. They can only give you answers.
>
> [*Pablo Picasso*]

The experiments were carried out by computer simulation on a range of Sun SPARCStations; a 1+, an ELC and two SLC's. Computational time has been a guiding force in the design of the simulation experiments.

4.1 Models of sequence evolution used

In this study I have adopted the simplest models of sequence evolution, to enable the broadest possible survey of the behaviour of phylogenetic methods with the available computing resources. The number of parameters was as small as possible, while still allowing statistically meaningful experiments to be conducted. Hence in the investigation dedicated to "small n", i.e., with $4 \leq n \leq 10$, I have used the Cavender two-state model, in which the probabilities of character state change are symmetric and independently and identically distributed across all sites of the sequences [10]. (Recall that n is the number of pendant vertices of the generating tree T_G, corresponding to taxa.)

For the case of larger n, the sampling method of data generation used for $n \leq 10$ is no longer viable (see Section 4.4.4). We must therefore use the more conventional 'evolutionary-type' data generation. If we are prepared to do this, we are able at little *further* computational cost to use four character states rather than two, and to include a quite general structure of transition matrix. This is discussed more fully in Section 4.5.3.

4.2 Deviations of the data from a model

In this section I describe some of the numerous ways in which a data set can disagree with a hypothesized model, and confound phylogenetic methods. These violations can be grouped into

- Inadequate models;
- Contradictions with the model.
- Sampling error;
- "White noise";
- "Pink noise";

These different types of violation of the ideal case are described in the next sections. The ways in which such violations have been modelled and their effects assessed are discussed in Chapters 5 and 6.

4.2.1 Inadequate and contradictory models

With an inadequate model of sequence evolution, there are over-simplifications made by hypothesizing that model, which cannot account for all aspects of the mechanism of character state change.

One way a model can fall short of a sufficient description of the true mechanism is in its method of accounting for multiple character-state changes. The Jukes-Cantor correction [51] is adequate for this when there are two or four character states with just one parameter, changing according to a symmetric Poisson process. With four character states and more parameters (e.g., Kimura's 2P and 3ST models [73]) more sophisticated formulae must be used [91]. Without adjustment for multiple changes, none of the methods described here is consistent.

If our hypothesized model contradicts the true mechanism, we are making assumptions about the method of character state change which are patently wrong. For example, this could be the assumption that the rates of change are independently and identically distributed (i.i.d.) along the character sequences, where in reality there may be several sites at which no change is possible, or that some

characters may be linked in their behaviour; in fact there are infinite possibilities for 'violations of the ideal' in this sense.

I have not embarked upon a study of how many such violations affect the performance of phylogenetic methods; such a study would be impossible without a greater understanding than that which is current about the analytical behaviour of phylogenetic methods, and a systematic understanding of how these contradictions occur in nature. I have restricted my study to three cases in which the mechanism of data generation contradicts the models with which the methods are consistent. They are sampling error, 'white noise' and 'pink noise', described below.

4.2.2 Sampling error

This is essentially the random error which cannot be avoided with real data, due to the finite data set size. Typically, nucleotide sequences are obtained of up to ≈ 1000 characters. In the simulations, I have often taken a range of sequence lengths, which show the effect of sampling error on the performance of the phylogenetic methods. The set of sequence lengths used is given in Chapter 5: Experimental Methods.

The effect of having zero sampling error can, for small n, be assessed by calculating exactly the expected frequencies of character state patterns across the taxa and using these as input data for the phylogenetic methods (see later) [41].

4.2.3 "White noise" and "Pink noise"

The term *white noise* is taken for its meaning in modern music: it is, in this case, essentially random error in the data, with no bias in any particular direction. For example, white noise is introduced if, when carrying out DNA sequencing, errors are made which randomly misread one character, say x, as another, say y, with probability independent of x and y. It is this form of white noise which I have investigated here.

On the other hand, *pink noise* is random error in the data which introduces a bias in the parameter(s) estimated from that data, *in a specific direction.* If, in the above case of sequencing errors, the probability of mis-reading a character was not independent of the character, the random error introduced would be regarded as *pink noise.*

In another example, suppose we obtain DNA sequences from a set of bird

species, and obtain also a set of sequences which are from lice of these birds. The underlying phylogenies of the birds and their lice may well be different [62]. Attempting to infer the phylogeny of the birds by using the data from both sets of taxa would then be hampered by the *pink noise* introduced by the parasite data. The combined data set would then regarded as "contaminated" by the parasite data, as it could no longer be guaranteed to be all generated from the tree describing the phylogeny of the birds.

The above is the type of pink noise which I have studied here.

4.3 General approach

There have been many studies performed to assess the performance of phylogenetic methods, but all are necessarily limited by computing time [13], [50], [54], [60], [80]. For example, in a 1981 study by Li [54] the sequence length c was 300, and only 20 trials could be carried out, though more recently the number of trials is around 300 [80]. The tree topologies have been limited to only a few, generally rooted, trees [50], [54].

I have therefore tried to exploit the computing power of the Sun SPARCStations available and fill in some of the gaps, testing over a large range of up to 41 different sequence lengths, all 27 topologies for $4 \leq n \leq 10$, and performing up to 1000 trials for any given set of parameters. (For the test of the effect of tree topology, using the two topologies with $n = 6$ pendant vertices, 10^5 trials were carried out.) The sequence lengths used ranged over

$$\{10, 13, 16, 20, 25, 32, 40, 50, 64, 80, 100, 125, 160, ..., 10^5\}.$$

The large number of trials for each case allowed the general trends in the performance of these phylogenetic methods to be seen easily. Such an exhaustive set of experiments as this, with such a wide range of parameters, has not been carried out before, but is now possible with the advances of modern computers.

In each trial I have generated data according to some known generating tree T_G and set of edge lengths $\mathbf{q}$. Thus the same data are used for each method, to enable comparison between methods. Note that this may give a false impression of coincidentally fluctuating performance of the methods, with varying data sets, but this is in general not the case: such fluctuation is purely an artifact of sampling and other random error in the data.

With each trial, the labels of the pendant vertices were permuted at random, to reduce the effect of the order in which data are presented to the clustering algorithms (see Section D.1).

4.4 Small n

The general simulation procedure is as follows (more detail is provided after this list):

1. Specify the topology of the tree to be used (the list of unrooted binary tree topologies is given in Appendix A), which methods to include, and overall parameters describing the properties of the generating tree T_G.
2. Randomly choose edge lengths q_i, equal to expected numbers of character state changes between the end-points of each edge e_i of T_G.
3. Calculate the expected bipartition frequencies s_i of all 2^{n-1} bipartitions from the vector $\mathbf{q}$ of edge lengths.
4. Sample from this expected frequency spectrum to obtain an *observed* bipartition spectrum, $\mathbf{s}'$ ("**obs**" in the program listing).
5. If distance-based methods are being used, obtain the distance matrix $\mathbf{D}$ from $\mathbf{s}'$.
6. Use the distance and/or bipartition data as inputs to the various methods under study.

4.4.1 Choosing the tree topology and other parameters

For small values of n, I have used unrooted trees. This is because the compatibility, closest tree and maximum parsimony methods do not distinguish the root. Also, it should be noted that there is a lack of irrefutable evidence that, for small n, the most likely or biologically interesting cases are those in which the rooted tree is most appropriate. In fact there is positive evidence supporting differing rates of evolution on different lineages of trees in some cases [6], [31], [55], [56], and also [100]. The data generated from these unrooted trees did not satisfy the molecular

clock hypothesis, which is that the expected distance between each extant taxon and the nearest common ancestor of all the taxa is the same.

I have also often operated under the assumption that *all trees are equally likely* (abbreviated 'a.t.e.l.') for simplicity and so as to not introduce any bias toward any particular type of tree [85]. This is more important in the case of large rooted trees, where investigating each topology and then providing a weighted average of the results is not possible due to the exponential growth with n in the number of tree topologies [8], [36].

Under the 'a.t.e.l.' assumption, the number of trees with a given topology X is calculated using Burnside's Theorem [29]. For any tree on n pendant vertices, divide $n!$ by the order of each of the symmetries of X.

For example, the tree UB5, 2_3 on $n = 9$ pendant vertices (see Appendix A) has three symmetries of order 2 (the three pendant neighbouring pairs) and one symmetry of order $3! = 6$ (the central point). Hence the number of trees with this topology is

$$N(\mathrm{UB5}, 2_3) = \frac{9!}{2!2!2!3!} = 7560.$$

When the behaviour of phylogenetic methods was considered with respect to n, ignoring tree topology, the data from simulations with each topology X were amalgamated in proportion to the numbers $N(X)$.

$N(X)$ is shown in Table 4.1 for all the tree topologies with from 4 to 10 pendant vertices. Diagrams showing these tree topologies are in Appendix A.

In this part of the investigation I have had to assign some parameters which have a degree of arbitrariness:

- The upper bound of the maximum path length σ between taxa (in terms of maximum expected number of observed differences in character state, per site);
- The minimum edge length ϵ;
- The type of distribution of the edge lengths (uniform, normal, and log-normal);
- The ratio r of maximum internal edge length to maximum pendant edge length.

Table 4.1: The number of trees with each topology for $4 \leq n \leq 10$.
In the first column are the values of n used. Columns two to five show the tree topology names, using the Tree Topology Description Notation (TTDN) described in Appendix A, and the last column shows the total number of trees for the given n, equal to $F(n) = (2n-5)!!$.

n	**Tree topologies X and number of trees $N(X)$**				$F(n) = (2n-5)!!$
4	UB2				
	3				**3**
5	UB3				
	15				**15**
6	UB4	$UB3, l_2$			
	90	15			**105**
7	UB5	$UB4, l_2$			
	630	315			**945**
8	UB6	$UB5, l_2$	$UB5, l_3$	$UB4, l_2, l_3$	
	5040	2520	2520	315	**10395**
9	UB7	$UB6, l_2$	$UB6, l_3$	$UB5, l_2, l_3$	
	45360	22680	45360	11340	
	$UB5, l_2, l_4$	$UB5, 2_3$			
	2835	7560			**135135**
10	UB8	$UB7, l_2$	$UB7, l_3$	$UB7, l_4$	
	453600	226800	453600	226800	
	$UB6, l_2, l_3$	$UB6, l_2, l_4$	$UB6, l_2, l_5$	$UB6, l_3, l_4$	
	113400	113400	28350	113400	
	$UB6, 2_3$	$UB5, l_2, l_3, l_4$	$UB5, l_2, 2_3$		
	226800	14175	56700		**2027025**

I have endeavoured to use parameters which are in keeping with the oft conflicting ideals of biological experience, phylogenetic interest, and practicality.

In any given simulation experiment, the probability distribution of the pendant edge lengths was fixed, as was the probability distribution of the internal edge lengths, for each generating tree topology. These two distributions were not necessarily the same in range and mean, but were always of the same general type, i.e., uniform, normal or log-normal.

For a given simulation, using a specific tree topology with diameter $d(T)$, I chose the maximum internal edge length a and the maximum pendant edge length b, such that $\sigma = (d(T)-2)a+2b$. Note that a and b are *upper bounds* on the ranges of the internal and pendant edge lengths, respectively: they were not necessarily realised for any edge length.

The ratio $r = a/b$ was sometimes allowed to vary, but was usually set to 0.5. The maximum path length σ was in general equal to 0.35, but this too was varied at times.

The minimum edge length ϵ for both internal and pendant edge length distributions was usually taken to be $a/10$. This was motivated by wanting the phylogenetic methods to succeed sufficiently often to give meaningful results; if the edge lengths are too short, there will not be enough signal in the data for any method to resolve that edge in T_G, i.e., with finite character sequences, the bipartition corresponding to that edge may not be sampled. Hence, the internal edge lengths were chosen from the interval $[\epsilon, a)$, and the pendant edge lengths were chosen from $[\epsilon, b)$.

As an example, given $\sigma = 0.35$, $a/b = 0.5$, and $\epsilon = a/10$, the edge length probability distributions for different diameters of the generating tree T_G are given in Table 4.2.

In most cases then, where $\sigma = 0.35$,

$$\frac{1}{2}(1 - \exp(-0.7)) \approx 0.25$$

was the upper bound on the maximum *observed* number of differences, per site, between any two taxa. This was taken as a reasonable upper limit.

The above constraints usually enabled the phylogenetic methods to resolve the generating tree T_G with a sequence length less than about 2000 characters, which seems to be a common maximum attainable length for a set of aligned nucleotide sequences. Reducing σ and a to very small values would be interesting certainly,

Table 4.2: Edge-length probability distributions for different diameters of generating tree.
In this example the upper bound of the maximum path length is $\sigma = 0.35$, $a/b = 0.5$, and the lower bound on each edge length is $\epsilon = a/10$.

$d(T)$	internal edge length range $[e, a)$	pendant edge length range $[e, b)$
3	[0.007 , 0.07)	[0.007 , 0.14)
4	[0.005833, 0.058333)	[0.005833, 0.116667)
5	[0.005 , 0.05)	[0.005 , 0.10)
6	[0.004375, 0.04375)	[0.004375, 0.0875)
7	[0.003889, 0.038889)	[0.003889, 0.077778)
8	[0.0035 , 0.035)	[0.0035 , 0.07)
9	[0.003182, 0.031818)	[0.003182, 0.063636)

but, with the constraint of available computing time and the requirement of gaining statistically significant samples, infeasible at present.

4.4.2 Choosing the edge lengths

The edge lengths are chosen using pseudo-random numbers, generated with the "`drand48()`" routine available in the 'C' programming language.' This random-number generator produces double-precision (in this case, 48-bit) floating-point numbers, uniformly distributed on the interval $[0, 1)$. The inevitable finite cycle length (approximately 2^{48}) is in excess of the number of times the procedure has been invoked in the course of this study.

Mostly the probability distribution of the edge lengths on each edge was taken to be uniform; this is the simplest possible case, and while not necessarily accurately reflecting the edge length distribution in nature, is at least not able to be completely rejected. The probability distribution of "real" edge lengths is not known.

4.4.3 Calculation of the expected bipartition frequencies

For small n, in this case up to ten, it is possible to calculate the expected frequencies of each of the 2^{n-1} possible bipartitions of the two character states with n taxa, in a reasonable amount of time. (In fact the method is feasible for data sets of up to ≈ 20 taxa, but this takes too long for simulation studies to be practicable.) This is by the scheme invented by Hendy [40]. Given a vector $\mathbf{q}$ of edge lengths q_i, which are taken to be expected numbers of character state change per site on each edge

e_i, the expected numbers of bipartitions of these taxa are the components of

$$\mathbf{s} = \mathbf{H}_n^{-1} \exp \mathbf{H}_n \mathbf{q},$$

where $\mathbf{H}_n$ is the $2^{n-1} \times 2^{n-1}$ matrix of 1's and -1's such that

$$\mathbf{H}_n = \begin{bmatrix} \mathbf{H}_{n-1} & \mathbf{H}_{n-1} \\ \mathbf{H}_{n-1} & -\mathbf{H}_{n-1} \end{bmatrix}, \qquad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

and $m = 2^{n-1}$.

Note that it is also possible to calculate the expected frequencies of each of the 4^{n-1} *quadripartitions* of taxa, when there are four character states available, but this obviously requires much more storage ($O(4^n)$), and the calculation requires $O(n4^n)$ operations.

There is under development a speed-up of the Hadamard conjugation method for four character states, which does not calculate all the probabilities of all possible quadripartitions, but such an approach is not an appropriate technique in this case. The advantage of the Hadamard conjugation method is in its ability to calculate *exactly* the probabilities of each of the partitions. This distribution of partition probabilities can then be sampled with replacement as many times as there are characters in the hypothesized sequences. This gives a spectrum of observed partition frequencies which has exactly the same statistical properties as one which would be obtained from 'growing' character sequences along the edges of a tree (as in "Growing the data", below). To abandon this precision, in the case of four character states, by missing out the calculation of some quadripartitions would be to introduce a new non-random and in all likelihood biassed source of error. Hence, and to reduce computing cost, I have used only two character states in this set of simulations.

4.4.4 Sampling from the expected bipartition spectrum

The spectrum s is first sorted in descending order, and a vector v of cumulative probabilities is calculated from it. In other words, $v_0 = 0$, and for $i = 1, \ldots, (2^{n-1} - 1), v_i = v_{i-1} + s_i$, so $v_{m-1} = 1$.

For $j = 1, \ldots, c$, pseudo-random numbers r_j are generated in the range $[0, 1)$, and each is sequentially compared with v_i, for $i = 1, \ldots, m-1$ until k such that

$v_{k-1} \leq r_j < v_k$ is found. The number s'_k of observed bipartitions of type k is then incremented.

After c such samples have been taken, $\mathbf{s}'$ is normalised by dividing each of its components by c.

Calculating the bipartition probabilities and then sampling from this vector affords a substantial saving in computing time for small n [13], [41]. The Hadamard conjugation requires approximately $(2n-1)2^{n-1}$ mathematical operations; the $(n-1)2^{n-1}$ for each multiplication by $\mathbf{H}_n$, and the term-wise exponentiation of the intermediate 2^{n-1}-component vector. This vector is sorted, which usually can be achieved in $O(n2^n)$ operations using a Shell sorting procedure [70], and certainly requires no more than $(2^{n-1}-1)(2^{n-1}-2)/2$ operations. Sampling from this c times (to give a spectrum corresponding to a set of sequences c characters long) requires more operations, but exactly how many is largely dependent on the numbers in the sorted vector. In Table 4.3 there are some example numbers of operations required for this whole process, for different edge lengths and numbers of taxa. Also shown are the numbers of operations required for the more traditional method of "growing" the data from a single ancestral sequence.

Table 4.3: Number of operations required to generate a bipartition frequency spectrum.

The edge lengths shown in the first row are the internal edge lengths a; the pendant edge lengths b were taken to be twice these values. In the penultimate column is the mean length of the internal edges of the 'caterpillar' tree UB$(n-2)$, with maximum path length 0.35. In the last column are shown estimates of the number of operations required to "grow" the sequences and then calculate the bipartition frequencies from these. The sequence length c is 1000. With ten or more taxa, there is no saving for the range of edge lengths used in this thesis. (Note that these figures are indicative only; they should not be taken as exact.)

	max. internal edge lengths					number of operations	
n	0.001	0.003	0.010	0.030	0.100	$\frac{0.1925}{n+1}$	evol. type
4	2149	2206	2402	2916	4277	3116	7000
5	2273	2371	2708	3632	6263	3724	9000
6	2639	2789	3323	4898	9970	4700	11000
7	3449	3664	4471	7101	16918	6283	13000

4.4.5 Distance Calculation

The observed bipartition spectrum s′ is used to calculate the pairwise distances between taxa as follows: For each bipartition i and each pair $\{j, k\}$ of taxa, the distance $D_{i,k}$ is

$$D_{j,k} = \sum_{i=1}^{m-1} \delta_{i,j,k} s'_i,$$

where $\delta_{i,j,k} = 0$ if the j-th and k-th least significant bits in the binary expansion of i are equal, and 1 if they are different.

This can be represented in algorithmic form as follows:

Algorithm 4.1 : get_distances

$m \leftarrow 2^{n-1}$
for $j = 1$ **to** $n-1$ **do**
  for $k = i+1$ **to** n **do** $d'_{i,j} \leftarrow 0$
for $i = 1$ **to** $m-1$ **do**
  for $j = 1$ **to** $n-1$ **do**
    for $k = i+1$ **to** n **do**
      if $((i$ **AND** $2^{j-1}) \neq (i$ **AND** $2^{k-1}))$ **then**
        { Compare the j-th and k-th bits of i. }
        $d'_{j,k} \leftarrow d'_{j,k} + s'_i$
for $j = 1$ **to** $n-1$ **do**
  for $k = j+1$ **to** n **do** $d'_{j,k} \leftarrow d'_{j,k}/c$
end.

These distances are adjusted to account for multiple changes, according to the Cavender model [10], so given observed distances $d'_{i,j}$, the inferred distances are $d_{j,k} = -(1 - \ln(2d'_{j,k}))/2$.

After the distances have been obtained, the data are used for each of the methods under scrutiny.

4.4.6 Inferring edge lengths

Edge lengths can be inferred from the data in several ways. The performance of the phylogenetic methods will of course be affected by the treatment of the original

(observed) data. The following are the ways in which edge lengths $\mathbf{q}'$ are inferred from observed data in this study:

- For each $i \in \{1, \ldots, 2^{n-1} - 1\}$, q'_i is taken as the observed relative frequency of bipartition i, as indexed in the scheme described in Section 1.4.1;
- The vector $\mathbf{q}'$ of inferred edge lengths is obtained from the observed bipartition spectrum $\mathbf{s}'$ with the Hadamard conjugation;
- A distance spectrum $\mathbf{g}'$ is obtained from the observed distance matrix $\mathbf{D}'$ after correction with the Jukes-Cantor formula [51], and another vector $\mathbf{w}'$ of inferred edge lengths obtained from $\mathbf{g}'$ with the Hadamard conjugation.

Note that the search methods in themselves are independent of the method of inferring the edge lengths of T_G. The search methods used here are all consistent with the model of evolution used in this study, *provided* the calculation of $\mathbf{q}'$ is such that $\mathbf{q}' \to \mathbf{q}$ as sampling error tends to zero (see [45], [88], and Appendix B, Theorem 3). When the observed bipartition frequencies are used directly for $\mathbf{q}'$, none of the search methods is consistent ([88]).

Observed bipartition frequencies as inferred edge lengths

If the expected number of bipartitions occurring on every edge is small, the expected number of such changes will be close to the number observed, as there will be very few (tending to zero) multiple changes of character state for a given character on each edge. Also the number of parallel changes on different edges will be low (tending to zero) as the probabilities of character state change on different edges are independent in this model (see Section 4.1). Hence the observed bipartition frequencies can be used to estimate the expected numbers of character state change, for the edges of the generating tree T_G.

The search methods (Co, CT, MP), when they take this kind of input data as inferred edge lengths, are denoted by the suffix 'SO', for 'Sequences, Observed'. Hence the 'Sequence Observed' methods are CoSO, CTSO and MPSO.

Hadamard conjugation for inferring edge lengths

The Hadamard conjugation can also be used to infer the edge lengths of T_G. The relation given in Section 4.4.3 is reversible, so that given the observed bipartition spectrum $\mathbf{s}'$, we can estimate $\mathbf{q}$.

Since $\mathbf{s}'$ will estimate the expected bipartition spectrum $\mathbf{s}$, we use this to estimate $\mathbf{q}$ by the relation

$$\mathbf{q}' = \frac{1}{m}\mathbf{H}_n \ln \mathbf{H}_n\mathbf{s}'.$$

If any of the components of $\mathbf{H}_n\mathbf{s}'$ are negative, the ln operation will be undefined. I have chosen to reject such data in the simulation studies.

When the search methods use $\mathbf{q}'$ as calculated above as their input data, are denoted by the suffix 'SH', for 'Sequences, Hadamard conjugated'. Hence the 'Sequence Hadamard' methods are CoSH, CTSH and MPSH.

Distance spectrum for inferring edge lengths

As previously noted (see Section 3.2), the interim vector $\rho = \mathbf{H}_n\mathbf{s}'$ corresponds to the minimum pathset distances of the generating tree T_G. These minimum pathset distances can also be estimated from the observed distance matrix $\mathbf{D}'$, and used to infer edge lengths using the rest of the Hadamard conjugation.

When such data are used for the search methods, they are denoted by the suffix 'DH', for 'Distances, Hadamard conjugated'. Hence the 'Distance Hadamard' methods are CoDH, CTDH and MPDH.

4.4.7 Example

I discuss here a typical case with $n = 6$ pendant vertices, describing the generating tree, the true and inferred edge lengths and distances, and the operation of the phylogenetic methods used. The sequence length used was $c = 250$.

The generating tree T_G is as shown in Figure 4.1. In Table 4.4 are shown the true distances between the pendant vertices, those observed as the Hamming distance between sequences, and the distances inferred using the Jukes-Cantor formula [51].

Table 4.5 shows the expected, observed and inferred edge lengths for the generating tree shown in Figure 4.1.

With the data shown in Tables 4.4 and 4.5, the correct tree was inferred only by CoSO, CTSO, MPSO and NJa. In this example CoDH, CoSH, CTDH, CTSH, MPDH, MPSH, NJ, SL, ST, TD and UPGMA all inferred the incorrect tree shown in Figure 4.2.

The inferred edge length spectra, and their relationship to the inferred tree T, can be represented on a column chart, and the data from this example are shown

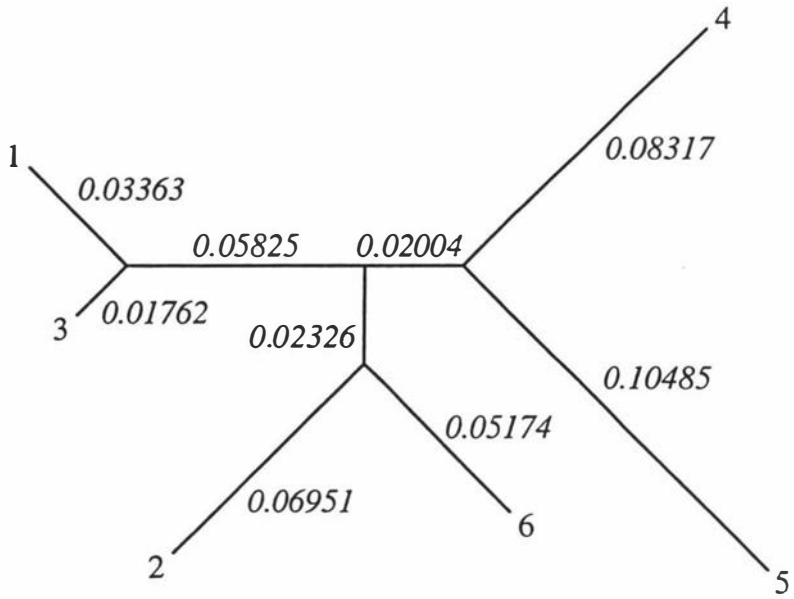


Figure 4.1: Typical generating tree used in simulations

The above tree shows the actual edge lengths, in expected numbers of character state change per site, in italics for each edge. The edges are drawn approximately to scale with their lengths.

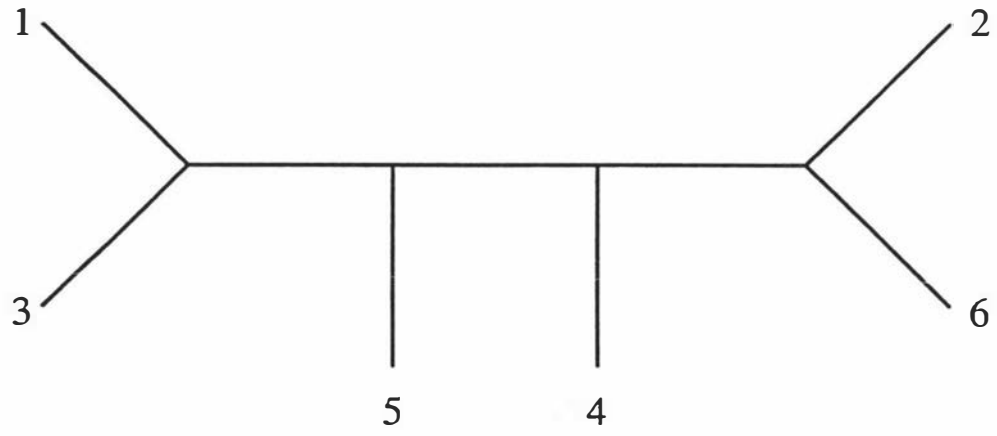


Figure 4.2: Example incorrect tree inferred by some methods

The above tree was incorrectly inferred by CoDH, CoSH, CTDH, CTSH, MPDH, MPSH, NJ, SL, ST, TD and UPGMA, with the experimental data shown in Tables 4.4 and 4.5. This tree is one edge different from the generating tree T_G. The edges of the tree are not to scale as different methods infer different edge lengths.

Table 4.4: Typical distance matrices, true, observed and inferred.
This table shows the distance matrices which were obtained in one trial, with $n = 6$ taxa and sequence length $c = 250$. The 'True distances' were obtained from the edge lengths assigned to the generating tree T_G. The 'Observed distances' were calculated from the observed bipartition spectrum $\mathbf{s}'$, and the 'Inferred distances' were calculated using the Jukes-Cantor correction formula [51], [91]. In the absence of sampling error (i.e., as $c \to \infty$) we would find the Inferred distances $\to$ the True distances.

True distances (from $\mathbf{q}$):

pendant vertices	2	3	4	5	6
1	0.18465	0.05125	0.19509	0.21677	0.16688
2		0.16864	0.19598	0.21766	0.12125
3			0.17908	0.20076	0.15087
4				0.18802	0.17821
5					0.19989

Observed distances (from $\mathbf{s}'$):

pendant vertices	2	3	4	5	6
1	0.14800	0.06000	0.17200	0.18800	0.13200
2		0.17600	0.17600	0.20000	0.10400
3			0.18400	0.20000	0.15200
4				0.20800	0.16000
5					0.20000

Inferred distances (from $\mathbf{q}'$):

pendant vertices	2	3	4	5	6
1	0.17549	0.06392	0.21080	0.23580	0.15326
2		0.21693	0.21693	0.25541	0.11660
3			0.22943	0.25541	0.18120
4				0.26893	0.19283
5					0.25541

Table 4.5: Typical expected, observed and inferred edge lengths

The edge lengths shown below are in terms of numbers of character state change per site for each edge of T_G. The first column shows the edge and bipartition labels for this number of pendant vertices. In the second column the true edge lengths **q** are shown (remember that edges which are not in T_G are assigned zero length). In the third column are the probabilities **s** of each bipartition occurring, and in the fourth are the relative frequencies **s′** at which each of the bipartitions occurred in the sample. The fifth and sixth columns show the edge lengths (**q′** and **w′**) which were inferred from the sequence spectrum and from the distance spectrum, respectively. Those edges which were in T_G are labelled in bold type. Edge 21, which was incorrectly included in the inferred tree T'_G by some of the methods, is labelled in bold type, as are its inferred lengths q'_{21} and w'_{21}.

edge/bipartition label	q	s	s′	q′	w′
1	**0.03363**	0.02218	0.01600	0.02433	0.01947
2	**0.06951**	0.04524	0.03200	0.04813	0.06783
3	0.00000	0.00158	0.00400	0.00545	0.00096
4	**0.01762**	0.01255	0.03600	0.06259	0.04445
5	**0.05825**	0.03811	0.03200	0.05233	0.05930
6	0.00000	0.00092	0.00000	-0.00309	-0.00096
7	0.00000	0.00363	0.00400	0.00310	-0.00351
8	**0.08317**	0.05455	0.07200	0.12283	0.11450
9	0.00000	0.00192	0.00000	-0.00163	-0.00108
10	0.00000	0.00406	0.00800	0.00656	-0.00229
11	0.00000	0.00020	0.00000	-0.00027	0.00073
12	0.00000	0.00113	0.00000	-0.00809	0.00108
13	0.00000	0.00502	0.00000	-0.01465	-0.00530
14	0.00000	0.00020	0.00000	0.00026	-0.00073
15	0.00000	0.00385	0.00400	-0.00184	-0.00374
16	**0.10485**	0.06800	0.09200	0.15678	0.15251
17	0.00000	0.00238	0.00000	-0.00382	-0.00084
18	0.00000	0.00497	0.01600	0.01969	0.00374
19	0.00000	0.00022	0.00000	-0.00088	0.00073
20	0.00000	0.00138	0.00000	-0.01053	0.00084
21	0.00000	0.00545	0.01200	**0.00656**	**0.00722**
22	0.00000	0.00019	0.00000	-0.00060	-0.00073
23	0.00000	0.00318	0.00400	-0.00146	0.00229
24	**0.02004**	0.01938	0.01200	-0.00179	0.00714
25	0.00000	0.00097	0.00000	-0.00265	-0.00293
26	0.00000	0.00330	0.00400	0.00003	0.00351
27	0.00000	0.00071	0.00000	-0.00255	0.00096
28	0.00000	0.00095	0.00400	0.00696	0.00293
29	**0.02326**	0.01826	0.02800	0.04561	0.03024
30	0.00000	0.00121	0.00000	-0.00156	-0.00096
31	**0.05174**	0.03419	0.02800	0.04563	0.04877

in such a chart in Figure 4.3.

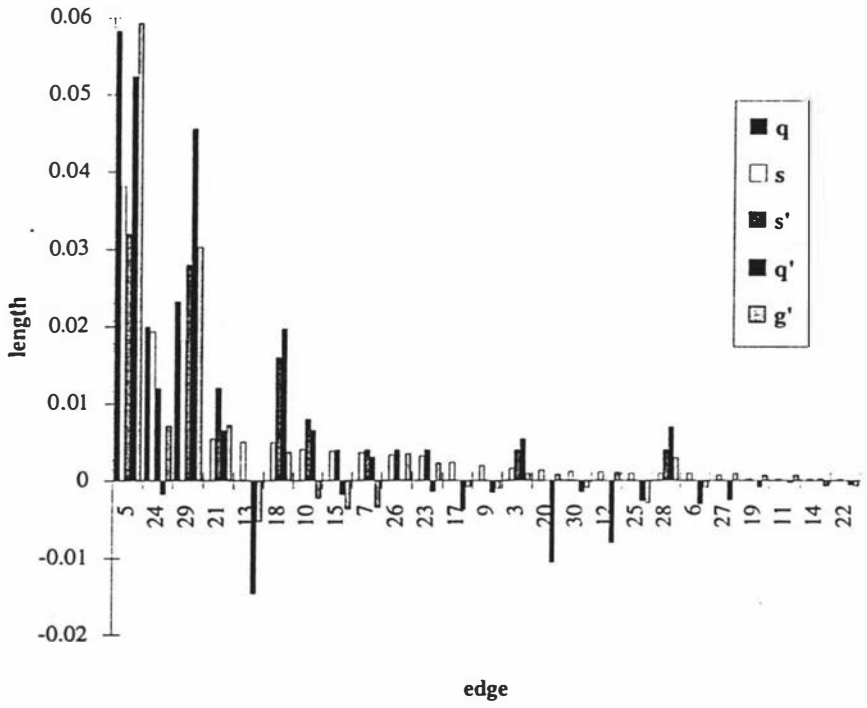


Figure 4.3: Spectra of edge lengths for an example tree

In the above chart are shown the true edge lengths (**q**, in black) of the generating tree shown in Figure 4.1, the expected bipartition frequencies (**s**, in white), the observed bipartition frequencies (**s′**, mid grey), and the edge lengths inferred from **s′** and from **D** (**q′**, dark grey, and **g′**, light grey, respectively). The pendant edges are omitted, and the edges are sorted in decreasing size of s_i. Note the close values of g'_{24} and g'_{21} (in fact g'_{21} is slightly larger) and that q'_{21} is substantially larger than q'_{24}. These cause some of the methods to choose the tree shown in Figure 4.2.

4.5 Large n

With larger values of n, the above approach is not feasible, so we have to go back to the procedure in which the sequences are "grown" from an ancestral sequence. The general procedure is as follows:

1. Specify the number n of taxa (pendant vertices) to use, the methods to include, and overall parameters describing the properties of the generating tree T_G. The structure of the transition matrix is *fixed*, that is, one matrix

$\mathbf{M}$ gives the *instantaneous relative rates of change* for each of the character states, for every point on the tree. $\mathbf{M}$ is taken to be a symmetric matrix for these investigations, so the probability of a character state change from state i to state j is the same as from state j to state i.

2. Randomly choose a generating tree T_G for the given n.
3. Assign random edge lengths for the tree, in terms of times between bifurcation events.
4. Initialize the ancestral sequence of length c.
5. Generate the descendent sequences.
6. After all sequences have been generated, use the data as inputs to the various NPMs under study.

The details of each of these main parts are described in the next few sections. The descriptions below are for four character states, corresponding to the four nucleotides adenosine (A), cytosine (C), guanine (G) and thymine (T), of DNA sequences.

4.5.1 Choosing a random tree

With large values of n, as has been mentioned, the number of distinct tree topologies is too large to enable study of each separately. I have therefore adopted the practise of choosing a random tree topology at the start, and include the assumption that all trees are equally likely in the choice mechanism for these trees. We find later (see Section 5.5) that there is only slight dependence of most methods on the topology of the generating tree. This also means that choosing trees with random topologies is an acceptible practice.

The following technique of deriving a tree from a (random) permutation is described by Harding [36], and has been made into an algorithm here.

The tree T_G will eventually be described by an array $\mathbf{t} = (t_1, \ldots, t_{n-1})$ of pointers, where $t_i = j \iff$ vertex i is an immediate descendent of vertex j. This implicitly imposes a direction on all the edges of the tree, so that all edges are directed towards the root vertex.

We first form a permutation of the numbers $1, 2, \ldots, n-1$; call this permutation $\mathbf{p} = (p_1, \ldots, p_{n-1})$.

We form an array $\mathbf{e} = (e_1, \ldots, e_{2n-2})$, so that for each i, e_i is the label of the 'earliest' vertex which has been chosen as an ancestor of vertex i. Earliest in this sense means closest to the root. Initially $e_i = i$ for $1 \leq i \leq n$ and $e_i = 0$ otherwise, but eventually we will have $e_i = 2n - 1$ for all i ($2n - 1$ is the label of the root).

Beginning with $i = 1$, we let vertices p_i and p_{i+1} be descendent vertices of vertex e_{n+i}. Let $x = e_{p_i}$ and $y = e_{p_{i+1}}$, i.e., x and y are the earliest ancestors chosen thus far of vertices p_i and p_{i+1}, respectively.

Each vertex j whose earliest ancestor was x or y now has earliest ancestor $n+i$.

The vertices x and y are immediate descendants of vertex $n + i$; hence we set t_x and t_y to $n + i$.

The above procedure is repeated until all of $\mathbf{t}$ is defined. It finds a random rooted tree on n pendant vertices, with all such trees equally likely.

An example of this technique is shown in 4.4.

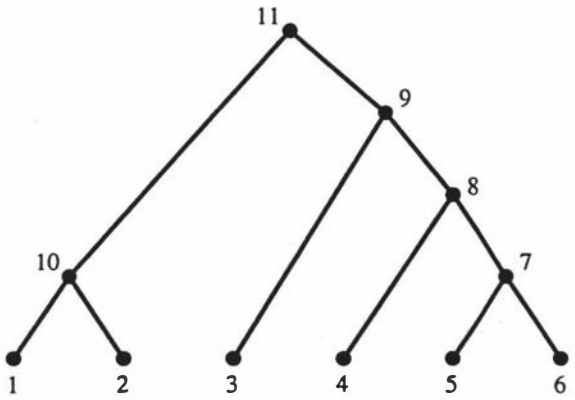


Figure 4.4: Choosing a rooted binary tree from a given permutation.
Given $n = 6$ and the permutation (5,4,3,1,2) we find that pendant vertices 5 and 6 have a common ancestor, which we label 7. Then $\mathbf{e} \leftarrow (1,2,3,4,7,7)$ and $\mathbf{t} \leftarrow (0,0,0,0,7,7,0,0,0,0)$. Next we find that pendant vertices 4 and 5 have a common ancestor, so $\mathbf{e} \leftarrow (1,2,3,8,8,8)$ and $\mathbf{t} \leftarrow (0,0,0,8,7,7,8,0,0,0)$. Proceeding the same way we find pendant vertices 3 and 4 have a common ancestor, so $\mathbf{e} \leftarrow (1,2,9,9,9,9)$ and $\mathbf{t} \leftarrow (0,0,9,8,7,7,8,9,0,0)$; then 1 and 2 have a common ancestor, so $\mathbf{e} \leftarrow (10,10,9,9,9,9)$ and $\mathbf{t} \leftarrow (10,10,9,8,7,7,8,9,0,0)$; finally we have $\mathbf{e} \leftarrow (11,11,11,11,11,11)$ and $\mathbf{t} \leftarrow (10,10,9,8,7,7,8,9,11,11)$.

Note that in this procedure the branching order within trees is important: the two trees shown in Figure 4.5, which correspond to the permutations (1,3,2) and (2,3,1), are *distinct* under this scheme, as in the first tree the taxa labelled 1 and 2

bifurcate before taxa 3 and 4, while in the second tree they bifurcate after taxa 3 and 4. The two trees in Figure 4.5 do have the same topology and would correspond to the same phylogeny, when regarded as unrooted trees.

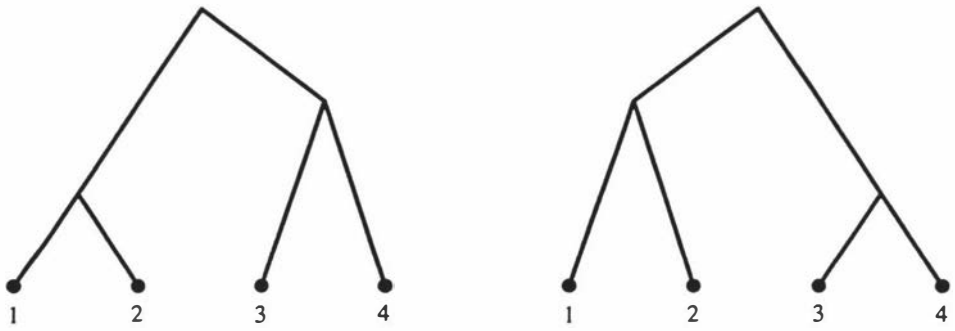


Figure 4.5: Two rooted binary trees on three pendant vertices
The above trees, corresponding to permutations (1,3,2) and (2,3,1), have the same topology, but are considered to be different by the scheme deriving rooted binary trees from permutations.

(Harding [36] provided exact probabilities of each unrooted binary tree topology, derived from the method described above, for up to 20 pendant vertices, and an asymptotic formula for larger n.)

4.5.2 Deriving the ancestral sequence

Recall that in all of these studies I have adopted the common practise of having the probabilities of character state change independently and identically distributed on the sites of the sequences [10]. Hence, given the distribution of frequencies of character states at the root (the *root distribution* π_r) we can arbitrarily set the positions in the sequence at which these states occur. Hence, for a root distribution $\pi_r = \{f_A, f_C, f_G, f_T\}$ where $f_A + f_C + f_G + f_T = 1$ and sequence length c we put the first $c \times f_A$ root characters in state A, the next $c \times f_C$ root characters in state C, and so on. (We may have to round these numbers to the nearest integer.) The sequence at the root is then of the form $(A, \ldots, A, C, \ldots, C, G, \ldots, G, T, \ldots, T)$, though in the computer these states are stored as the numbers 1, 2, 4 and 8, respectively.

4.5.3 Growing the data

When "growing" sequences along the edges of a tree, it is necessary to check each character, at each edge, for possible character state change. Whether a given character changes state is determined by comparison with a pseudo-random number, generated for each character and edge of the tree. With two character states, there are just two possibilities for state change: either the character remains in the same state, or it changes to the other state. Therefore only one comparison need be made, to see if each character changes. With four character states, there are four possibilities for state change, and up to three comparisons may be made.

This does not imply that using four character states requires three times as many operations as using two. In these studies, each character on each sequence has only a small probability of change. Hence in most cases only one comparison must be made, to determine whether the character changes state or not, and *if* a change occurs, either one or two further comparisons must be made to determine the next state of the character. Thus using four character states is not, in this case, substantially slower than using two.

For each internal vertex v whose sequence is known, we find the vertices of T_G which are immediate descendants of v. Let these vertices be x and y. We must determine the transition matrix which will give each of the 16 probabilities of character state change for the two edges (x, v) and (y, v). Since we have chosen $\mathbf{M}$ for the whole tree, we use the lengths of these edges, say l_x and l_y respectively, to get the transition matrices $\mathbf{M}^{l_x}$ and $\mathbf{M}^{l_y}$, which contain the probabilities of character state change for the length of (relative) time given by the edge lengths.

In order to obtain $\mathbf{M}^l$ we diagonalise $\mathbf{M}$ by

$$\mathbf{M} = \Lambda^{-1}\mathbf{D}\Lambda,$$

where

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-k_1} & 0 & 0 \\ 0 & 0 & e^{-k_2} & 0 \\ 0 & 0 & 0 & e^{-k_3} \end{bmatrix},$$

e^{-k_1}, e^{-k_2} and e^{-k_3} are eigenvalues of $\mathbf{M}$ and the columns of Λ are the eigenvectors of $\mathbf{M}$.

By the above relation,

$$\mathbf{M}^{l_x} = \Lambda^{-1}\mathbf{D}^{l_x}\Lambda = \Lambda^{-1}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-k_1 l_x} & 0 & 0 \\ 0 & 0 & e^{-k_2 l_x} & 0 \\ 0 & 0 & 0 & e^{-k_3 l_x} \end{bmatrix}\Lambda.$$

The diagonalization of M is possible if M is symmetric: hence symmetric transition matrices have always been used for this "large n" case.

For each initial character state, the probabilities of the new state are arranged so that the most likely new character state is examined first, to save time. (For further detail, see Appendix C, Algorithms C.37 - C.40.)

Thus far the investigation into the performance of phylogenetic methods with large n has used only the simple 1-parameter model of Jukes and Cantor [51], where the probability of character state change is independent of the states.

The matrix used for the above diagonalization in my experiments was

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0.95 & 0 \\ 0 & 0 & 0 & 0.95 \end{bmatrix}.$$

This gives the Jukes-Cantor type transition matrices for the edges of the generating tree T_G, with the steady-state relative frequencies of each character state being 0.25, 0.25, 0.25, 0.25. The distribution of character states at the root was also uniform.

4.5.4 Parameters

As with the "small n" case, I have chosen operating parameters to maintain a sensible balance between biological experience, phylogenetic interest, and computational practicality.

The maximum path length is once again restricted, but not explicitly as it was before. In a further effort to mimic biological experience, the mean, over all lineages in the tree, of the expected time between bifurcation (speciation) events on a single lineage, is constant, so that there is the same probability *per lineage* of a bifurcation event occurring, per unit time.

This means that the mean number of bifurcation events per unit time increases as time progresses, so there are more bifurcations per unit time near the pendant vertices of the tree than there are near the root.

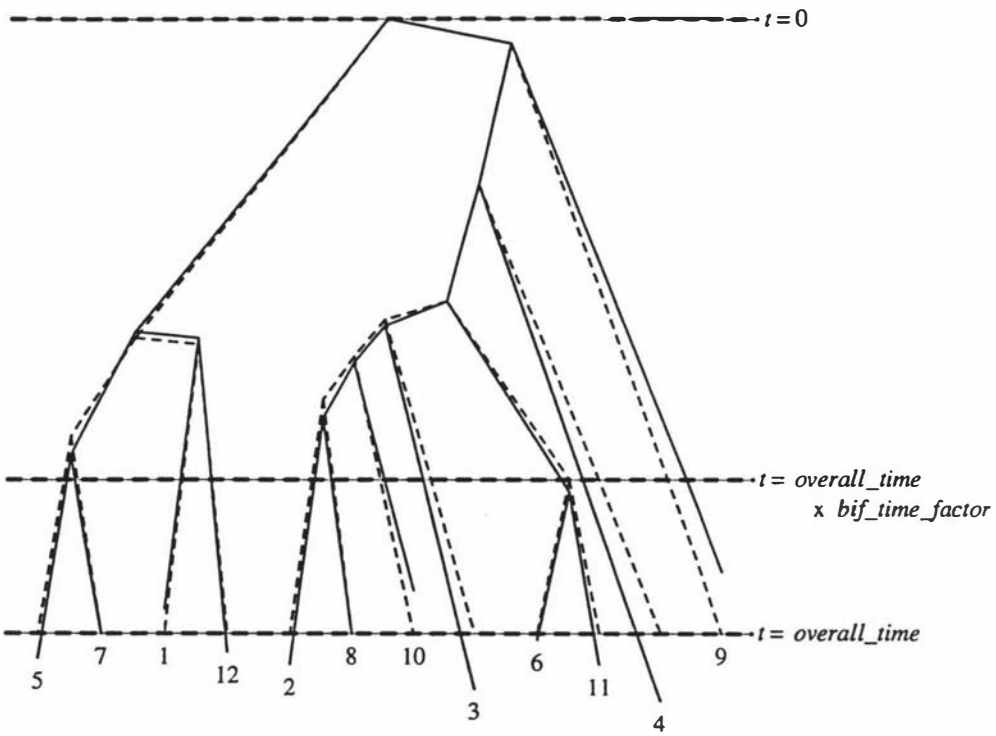


Figure 4.6: An example of a rooted tree used in `big.c`.
In this figure the mean edge lengths, from which the actual edge lengths were sampled, are shown by dotted lines, and the actual edge lengths are shown by solid lines. The lengths are proportional to the vertical displacement of the ends of the edges; they are not proportional to the lengths of the lines themselves. The horizontal dotted lines represent the time at the "origin" ($t = 0$), the time at which the last bifurcation event is expected to happen ($t = overall\_time \times bif\_time\_factor$), and the "present" time ($t = overall\_time$), from top to bottom.

The normalisation of the edge lengths, and the sampling by which pseudo-random edge lengths were chosen, are described in Appendix C, Algorithms C.35 and C.37.

Chapter 5

Results 1: "Small n"

5.1 Introduction

In this chapter I shall describe several experiments carried out with unrooted generating trees with from four to ten pendant vertices, corresponding to extant taxa. The model of evolution used is as described in Chapter 4, Section 4.1.

Section 5.2 describes how the results from these experiments are represented. Section 5.3 describes the agreement between methods which was observed for a particular set of input data. This agreement was observed for all the experiments.

The subsequent sections describe the effects of the following parameters:

- Sampling error;
- Tree topology;
- Edge length probability distribution;
- Number of taxa;
- Use of the distance spectrum;
- White noise;
- Pink noise.

Recall that in all the experiments comparing the performance of phylogenetic methods, the same observed data were used for each method. This imposes a correlation between the methods, which is most obvious for the tests with short sequences and small numbers of trials.

5.2 Representation of findings

I have sought to represent my findings in an easily interpretable way, and also so the data can, with reasonable ease, be analysed to establish statistical properties of the performance of the methods.

For this reason I have chosen most often to represent the accuracy of the methods under investigation in two ways. The method most frequently used is the *mean number of edges wrongly inferred.* This is equivalent to *half the partition distance* between the inferred trees T_i and the generating tree T_G [82]. The expected distribution of the partition distance, which is a *metric*, is known for $n \leq 16$ [8], [82].

The other predominant method of representing the accuracy of tree-building and tree-finding methods is the frequency at which the correct tree is inferred, i.e., where no edges are wrongly inferred.

Both ignore the matter of the accuracy of inferred edge lengths, which is not being studied here, for two reasons. Firstly, not all methods provide comparable estimates of edge lengths (e.g., MP usually infers minimum numbers of character state changes — a discrete measure over the whole tree, whereas NJ returns estimates of edge lengths which are continuous variables, and ST in its basic form does not provide estimates of edge lengths at all). Secondly, there is the pragmatic reason of keeping the size of the study to a reasonable level.

Other methods of representing the performance of the phylogenetic methods are occasionally used in this study, and they are explained in the context in which they are used.

5.3 Agreement between methods

The agreement between methods can be measured by recording the mean partition distance between the inferred trees, in a matrix.

This was carried out for the all the tree topologies with 8 pendant vertices, over a range of sequence lengths c. An indication of the agreement between all the methods is shown in Figure 5.1, with $c = 100$ and 250 trials. In this figure the reciprocal of half the average partition distance between the inferred trees is shown; thus those methods which consistently infer trees which are close to each other show a high value on the figure.

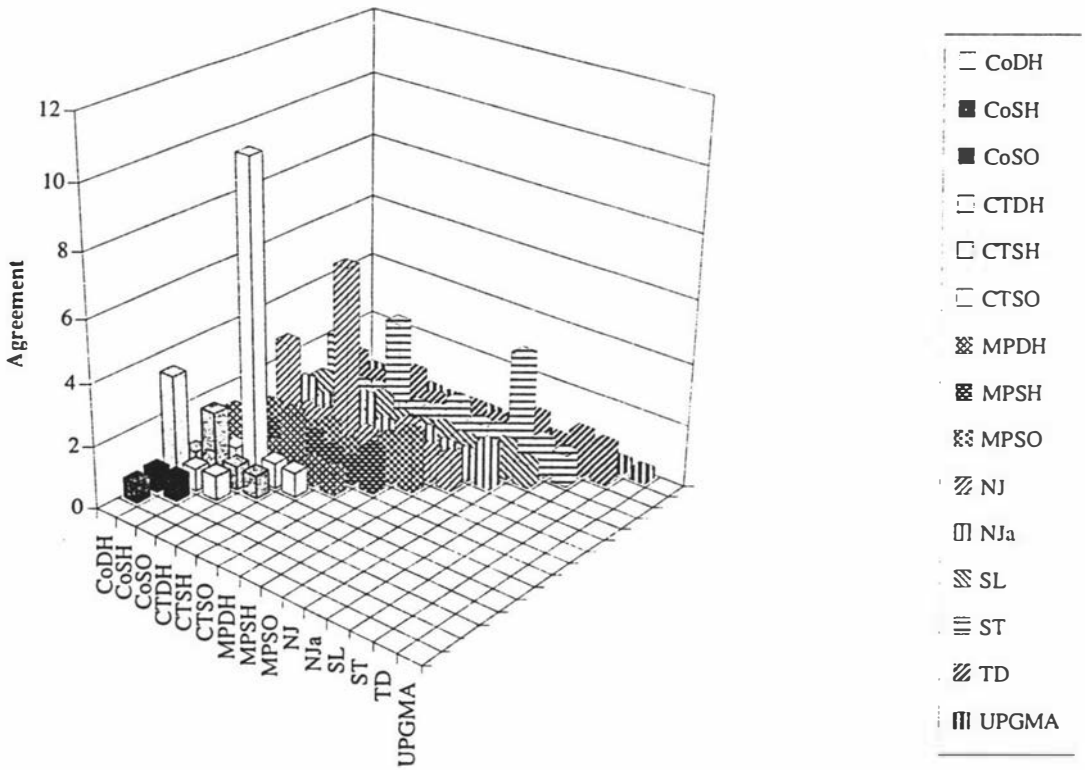


Figure 5.1: The agreement of phylogenetic methods

In this figure is shown the reciprocal of half the average partition distance between the trees inferred by all the phylogenetic methods under study. The generating trees, on 8 pendant vertices, had topology UB6, and the sequence length c was 100. 250 trials were carried out. The highest value on this graph is $\approx$ 10, between CoSO and CTSO, indicating that on average the distance between the trees inferred by these two methods was ≈ 0.1.

In this figure we see that CoSO and CTSO are the most highly correlated. The second most highly correlated methods are CTDH and NJ.

Also highly correlated are the pairs of methods CoDH and CTDH, CoDH and NJ, CoDH and ST, CoSH and CTSH, CTDH and MPDH, CTDH and ST, MPSH and NJ, NJ and ST.

The mean number of edges different between the inferred trees for the more highly correlated methods mentioned above are shown in Table 5.1, below.

Table 5.1: Half the mean partition distance between inferred trees of different methods

Note that while in the Figure 5.1 a high agreement is shown by tall columns, high agreement is shown here by numbers close to zero. Numbers less than 0.5 are in bold type, and those between 0.5 and 1.0 are underlined.

	CoSH	CoSO	CTDH	CTSH	CTSO	MPDH	MPSH	MPSO	NJ	ST
CoDH	1.5062	1.3951	**0.2881**	1.4938	1.3786	0.7202	1.1564	1.0823	**0.3621**	**0.4691**
CoSH	0	1.3210	1.3704	**0.4486**	1.3251	1.4033	1.3292	1.4403	1.3951	1.4403
CoSO		0	1.2675	1.3210	**0.0988**	1.3868	1.4444	1.4074	1.2840	1.2881
CTDH			0	1.3621	1.2510	**0.4815**	0.9794	0.9300	**0.1687**	**0.3045**
CTSH				0	1.2757	1.3951	1.2798	1.4280	1.3704	1.4321
CTSO					0	1.3786	1.4321	1.3992	1.2593	1.2716
MPDH						0	0.7160	0.7078	0.6255	0.7037
MPSH							0	0.5309	0.9959	1.0700
MPSO								0	0.9588	1.0041
NJ									0	**0.2675**

Let the *net disagreement* of a phylogenetic method M with the other phylogenetic methods be the sum of the mean distances between the tree inferred by M and those inferred by the other methods. Hence for example the net disagreement of CoDH with the other methods shown in Table 5.1 is the sum of the numbers in the first row of that table.

Table 5.2 shows the net disagreements of all the methods shown in Table 5.1. Note that CTDH, which in this case has the lowest net disagreement, is not the most accurate method in this trial.

5.4 Sampling error

The effect of sampling error on the performance of the phylogenetic methods is investigated by varying the sequence length c used. I have used a range of sequence lengths both in the "small n" and the "large n" cases. For the "small n" case I have

Table 5.2: The net disagreement of some phylogenetic methods
The second column of this table shows the net disagreement of several methods with each other. In the third column is shown the mean number of edges wrongly chosen by each method, which is equal to half the partition distance between the inferred tree and the generating tree. The generating tree had topology UB6, the sequence length was $c = 100$ and 250 trials were carried out.

	net disagreement	mean number of edges wrongly chosen
CoDH	9.8519	1.387
CoSH	12.9795	1.679
CoSO	12.2141	1.601
CTDH	8.4032	1.267
CTSH	12.8066	1.642
CTSO	12.0700	1.609
MPDH	9.5185	1.082
MPSH	10.9341	0.963
MPSO	10.8888	0.753
NJ	8.6873	1.267
ST	9.2510	1.300

used a minimum of 10 characters, and a maximum of 10^5 characters: in practical terms, from the ridiculous to the sublime. The number of trials carried out for each sequence length and each tree topology was 1000. The amount of data this generated is too large to be included in entirety here: rather, I have selected those results which best illustrate the overall behaviour of the phylogenetic methods used.

Also, in the "small n" case, it is possible to generate data with zero sampling error, something impossible to achieve when using the "evolutionary" method of data generation, as used in the "large n" case. This data generation is uses the *expected* bipartition spectrum $\mathbf{s}$ as the "observed" spectrum $\mathbf{s}'$, since, when $c \to \infty$, $\mathbf{s}' \to \mathbf{s}$ [41]. This is useful in determining whether phylogenetic methods are consistent with a model, for in the absence of sampling error, consistent methods will correctly deduce the generating tree T_G with probability 1. This technique demonstrates the inconsistency of UPGMA with the additive model used (see Figure 5.2).

In Figures 5.2 to 5.4 are shown the proportions of 1000 trials in which the estimated tree T differs from the generating tree T_G by $0, 1, \ldots, 7$ edges, for sequence lengths c ranging from 100 to 10^4. These figures show the strong dependence of the performance of UPGMA, TD and NJ (which is also seen for the other methods; data not shown) on sequence length.

Also worthy of note is the clear indication of the inconsistency of UPGMA with

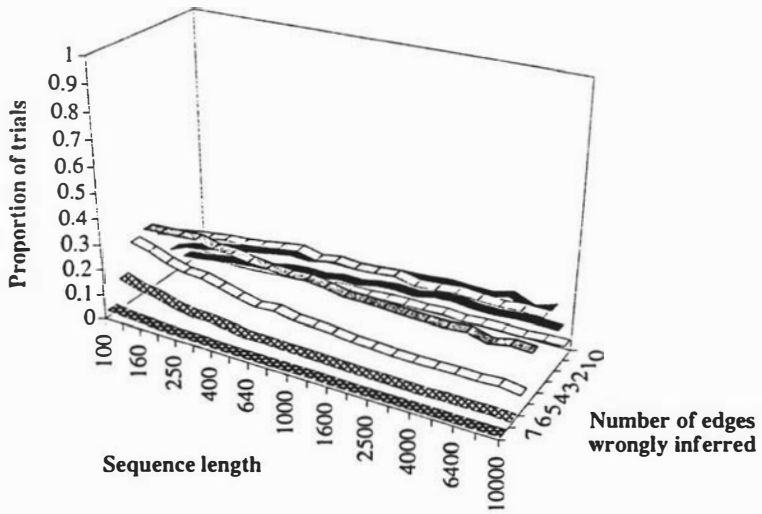


Figure 5.2: UPGMA vs. c with $n = 10$ and all trees equally likely

The mean proportion of trials in which UPGMA infers $0, 1, \ldots, n-3 = 7$ edges incorrectly is shown. Sequence length c ranges from 100 to 10^4. Maximum path length σ is 0.35 and the ratio r between maximum internal and maximum pendant edge lengths is 0.5.

the model used: as c gets large, the most likely number of edges wrongly inferred tends to 4.

In Figure 5.3 we see the improvement in accuracy of TD over UPGMA, but note that the probability of inferring T_G does not tend to unity. This demonstrates the importance of accuracy and consistency: UPGMA is both inaccurate and inconsistent, and TD is much more accurate, though still inconsistent, with the model used.

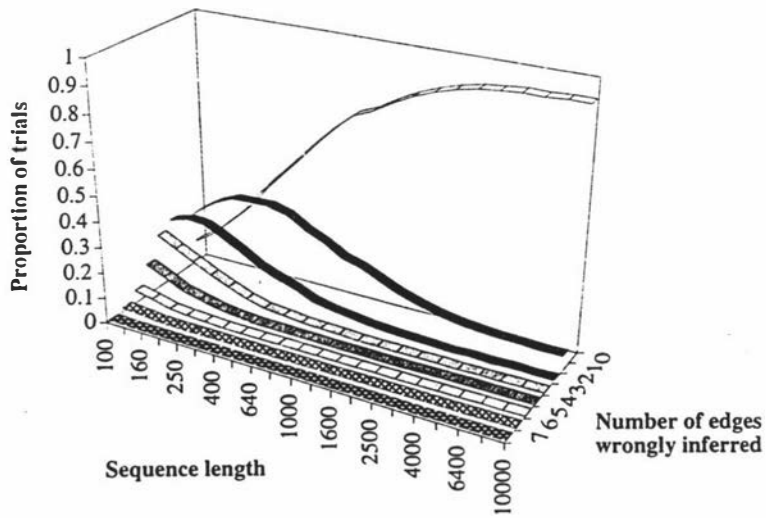


Figure 5.3: TD vs. c with $n = 10$ and all trees equally likely

The mean proportion of trials in which TD infers $0, 1, \ldots, n-3 = 7$ edges incorrectly is shown. Sequence length c ranges from 10 to 10^5. Maximum path length σ is 0.35 and the ratio r between maximum internal and maximum pendant edge lengths is 0.5.

Figure 5.4 shows the mean proportion of trials in which NJ infers $0, 1, \ldots, 7$ edges wrongly. The probability that more than zero edges will wrongly be inferred tends to zero when the sequence length c reaches ≈ 5000 for this experiment (this will not necessarily be true in other cases). This figure shows the importance of consistency with the model: NJ is both accurate and consistent in this case.

The inconsistency of MPSO does not become apparent in this part of the investigation (data not shown).

Unfortunately, the distance methods which attempt to "correct" for multiple

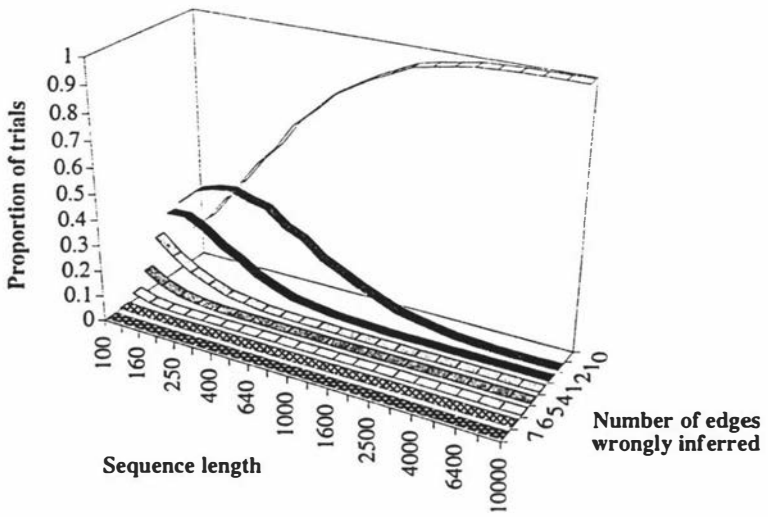


Figure 5.4: NJ vs. c with $n = 10$ and all trees equally likely

The mean proportion of trials in which NJ infers $0, 1, \ldots, n-3 = 7$ edges incorrectly is shown. Sequence length c ranges from 10 to 10^5. Maximum path length σ is 0.35 and the ratio r between maximum internal and maximum pendant edge lengths is 0.5.

changes, and the Hadamard conjugation process (the $\mathbf{H} \ln \mathbf{Hs}$ calculation), can give undefined values for the inferred distances when some of the components of $\mathbf{Hs}$ are non-positive. This problem occurs most often with small sequence lengths, to wit, $10 \leq c \leq 100$, and with very long path lengths between taxa, i.e., greater than about 1.2 character-state changes expected per site. If this happened for a particular observed $\mathbf{s}'$, I chose to reject such data sets for all methods, as it was not possible to compare all methods under these circumstances.

In order to reduce the overall running time of the simulations, I have not attempted to obtain the same number of trials with these short sequences. Rather, I have borne in mind that the statistical significance of such trials is markedly reduced: there is still useful information to obtained from such trials, but it is imprecise.

In Figure 5.5 below, the mean number of trials in which 0 edges were incorrectly inferred (i.e., the generating tree T_G was inferred) is shown for CoDH, CoSH, CTDH, CTSH, MPSH, MPSO, NJ, SL, ST and TD. Data for UPGMA is not shown, as in this experiment the proportion of trials for which T_G was found by UPGMA never exceeded 0.05. This shows the inconsistency of TD in that as $c \to \infty$, the probability of inferring $T_G \not\to 1$.

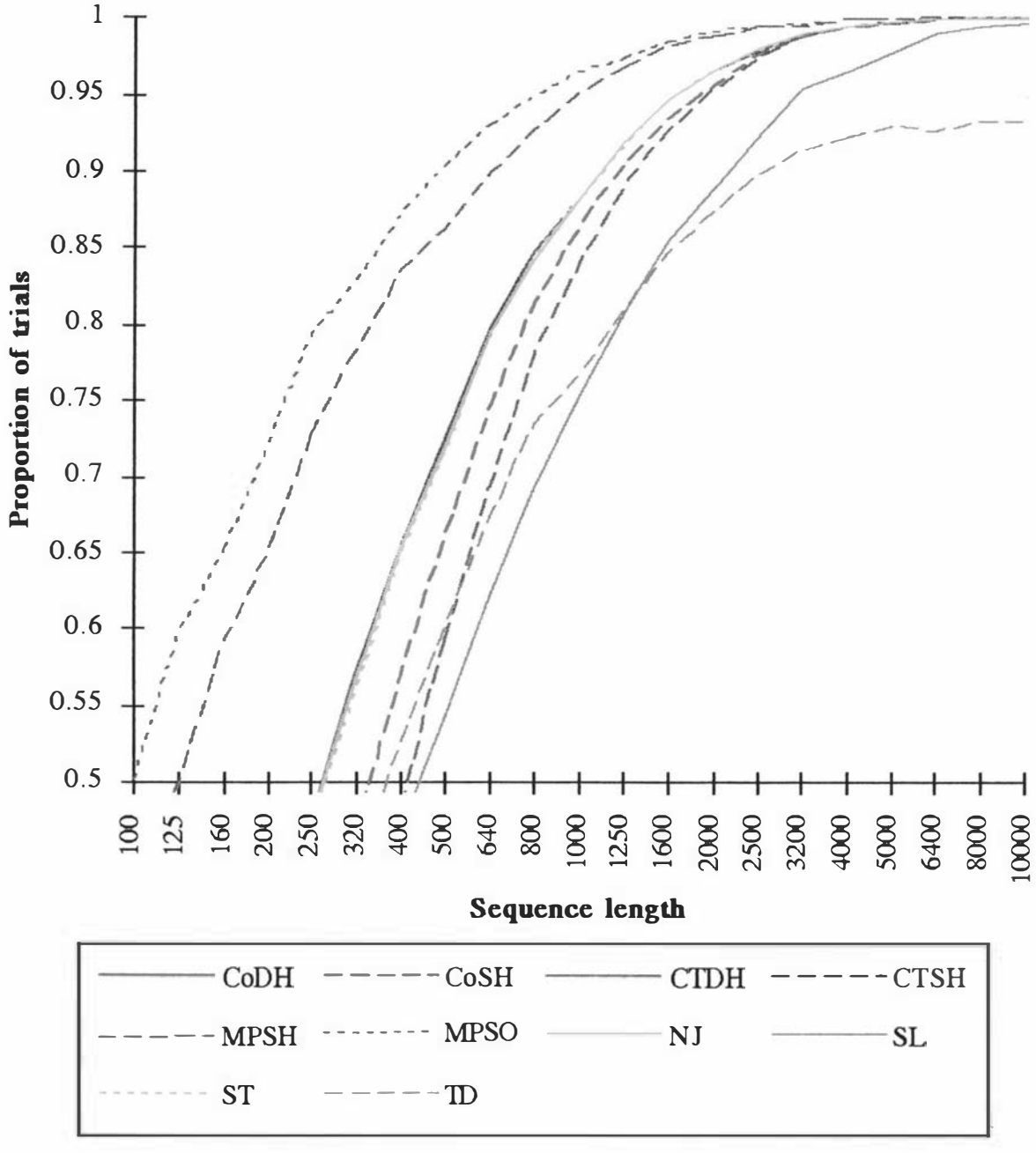


Figure 5.5: Mean proportion of trials in which the correct tree is inferred

The proportion of trials averaged with all trees considered equally likely in which several methods correctly infer the generating tree T_G is shown. Maximum path length σ is 0.35 and the ratio r between maximum internal and maximum pendant edge lengths is 0.5. The number of taxa n is 10.

The good performance of MPSH and MPSO is clear in Figure 5.5, and despite MPSO being inconsistent with the model, it is the most accurate in this case. The inconsistency of MPSO is not evident, probably due to the small edge lengths making multiple changes of character state unlikely on each edge: the overall maximum path length is bounded above by $\sigma = 0.35$, but the expected maximum path length is approximately half this value, and the expected internal edge lengths are 0.023215, calculated using the a.t.e.l. assumption. Hence the expected number of multiple changes of character state on an internal edge is small, $0.000263 + 0.000002 + ... = 0.000265$ (this is calculated using the Poisson model, as detailed in [39]). The expected number of multiple changes of character state on pendant edges is 0.001045, calculated as above.

5.5 Tree topology

There have been many simulation studies of phylogenetic methods, which have been conducted for specific tree topologies [92], [93], etc. These have not generally addressed the question of whether the tree topology *in itself* is a major contributing factor to the performance of the phylogenetic methods concerned, but rather, have usually taken certain fixed combinations of edge lengths in conjunction with a given tree topology.

I have tried therefore to eradicate all other possible influencing factors and establish whether topology is intrinsically important. Though it is not possible to completely counteract all influences other than tree topology from these simulation tests, it is possible to reduce them to a large extent.

It is not obvious that the tree topology really affects the performance of the phylogenetic methods, as closely linked with tree topology is the edge-length probability distribution. For example, trees with larger diameters will have shorter internal edges, so the expected number of multiple character-state changes will be lower. The joint effect of different tree topologies and edge-length probability distributions for the generating tree T_G is shown in Figure 5.6, below. (The 11 topologies of unrooted binary trees on 10 pendant vertices are listed in Appendix A. Of these, one topology has diameter $d(T) = 9$, three have $d(T) = 8$, five have $d(T) = 7$, and two have $d(T) = 6$.)

Figure 5.6 shows the slightly different accuracy NJ for those generating trees

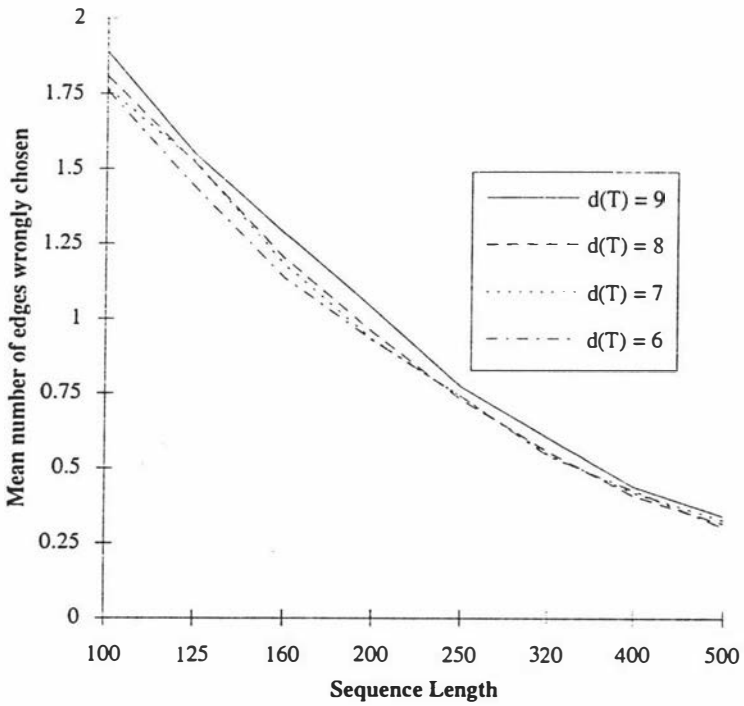


Figure 5.6: Performance of NJ with all tree topologies of the same diameter, with 10 taxa

In the above graph the mean number of edges wrongly inferred is shown, for sequence length c in the range $[100, 500]$. For each topology, 1000 trials were carried out, and the mean accuracy was obtained for each diameter using the a.t.e.l. assumption. Those tree topologies with the same diameter had the same probability distribution of edge lengths.

with different diameters. One explanation for this might be that the effect of different tree topologies may be mainly due to the different edge length probability distributions.

If the edge-length probability distributions were made independent of tree topology in these simulations, the "caterpillar" topologies UB($n-2$) would have much larger expected distance between the most distant pendant vertices than would the more "star-like" topologies, like UB5, 2_3 (see Appendix A).

A compromise can be reached by using a value of n which is small enough that the range of diameters of trees over all topologies is small (so the same edge-length probability distributions can be used for all trees without the maximum expected distance between pendant vertices being markedly different for different topologies), but with at least two different topologies for that n.

I have used $n = 6$ with the two topologies UB4 and UB3, 1_2, and have used the same edge-length probability distributions for all the trees, in an experiment to estimate the dependence of accuracy of phylogenetic inference on tree topology. Results from the experiment, in which 10^5 trials were conducted, are given in Table 5.3.

From this table the following trends are apparent:

- The sequence-spectrum based methods all show small but highly statistically significant effects, of tree topology on their performance. The most significant difference in performance for the two topologies UB4 and UB3, 1_2 is shown by CoSO. In each case the methods are more accurate when the generating tree had topology UB3, 1_2, and the level of significance was less than 0.01%.

- The distance-spectrum based methods all show very low dependence of tree topology on their performance, with CoDH and CTDH significant at approximately 4.5% and MPDH significant at approximately 18%.

- The clustering methods show much more variability in the effect of generating tree topology, with UPGMA the most affected, followed by NJa and SL, then ST, TD, and NJ the least affected. These methods do not show the same uniformly better performance when the generating tree T_G had topology $UB3, 1_2$ as do the sequence-spectrum based methods. However it is noteworthy that UPGMA shows a much higher accuracy when T_G has topology $UB3, 1_2$.

In summary, though the effect of tree topology under this model is statistically

Table 5.3: Effect of tree topology on performance of phylogenetic methods

$p(X)$ is the proportion of the 10^5 trials in which the topology of the generating tree was X and the correct tree was found. The two tree topologies used were UB4 and UB3, 1_2. $\bar{p} = (p(\text{UB4}) + p(\text{UB3}, 1_2))/2$. Under the null hypothesis that $p(X)$ follows a binomial distribution, independent of tree topology, z gives the standard error of $p(\text{UB4}) - p(\text{UB3}, 1_2)$, equal to

$$z = \frac{p(\text{UB4}) - p(\text{UB3}, 1_2)}{\sqrt{(\bar{p}(1-\bar{p})/(5 \times 10^4)}}.$$

The percentage probabilities of these values of $p(\text{UB4}) - p(\text{UB3}, 1_2)$ occurring by chance, and hence the level of significance at which we can reject the null hypothesis, are shown in the last row of the table. Note that the edge length probability distribution was the precisely the same for both topologies.

Sequence-spectrum based methods:

	CoSH	CoSO	CTSH	CTSO	MPSH	MPSO
$p(\text{UB4})$	0.95436	0.93357	0.95614	0.94901	0.98620	0.97157
$p(\text{UB3}, 1_2)$	0.95893	0.96337	0.95934	0.96777	0.98787	0.98343
$\bar{p}$	0.95664	0.94847	0.95774	0.95839	0.98703	0.97750
z	-5.0177	-30.1411	-3.5567	-21.0062	-3.3010	-17.8822
% prob.	$\approx 0\%$	$\approx 0\%$	$\approx 0\ \%$	$\approx 0\ \%$	$\approx 0\%$	$\approx 0\%$

Distance-spectrum based methods:

	CoDH	CTDH	MPDH
$p(\text{UB4})$	0.95452	0.95471	0.97477
$p(\text{UB3}, 1_2)$	0.95293	0.95312	0.97541
$\bar{p}$	0.95372	0.95391	0.97509
z	1.6924	1.6957	-0.9182
% prob.	4.53%	4.49%	17.94%

Clustering methods:

	NJ	NJa	SL	ST	TD	UPGMA
$p(\text{UB4})$	0.95475	0.93908	0.92305	0.95442	0.89563	0.28586
$p(\text{UB3}, 1_2)$	0.95316	0.92374	0.92995	0.95201	0.89881	0.49809
$\bar{p}$	0.95395	0.93141	0.92650	0.95321	0.89722	0.39197
z	1.6964	13.5709	-5.9125	2.5518	-2.3416	-97.2080
% prob.	4.49 %	$\approx 0\%$	$\approx 0\%$	0.96 %	0.54 %	$\approx 0\ \%$

significant, except for the distance-spectrum methods and NJ, it is in general small. The performance of any of these methods is not greatly affected by tree topology except for UPGMA.

Hence for most studies it is reasonable to use just one tree topology or randomly chosen topologies for T_G, to get an impression of the performance of a method for all topologies with that number of pendant vertices. In subsequent studies in this thesis I have therefore either used a single fixed topology or have randomly chosen tree topologies for T_G.

5.6 Edge length distribution

In this part of the study, the following parameters were allowed to vary:

- ratio r between maximum internal and maximum pendant edge length;
- overall maximum path length σ.

The generating tree had topology UB5,2_3 ($n = 9$) for all these trials; using $n > 9$ would have been infeasible with the large number of trials required for a statistically meaningful experiment.

In this section of the study the overall maximum path length σ ranged over the set $\{0.112, 0.14, 0.175, 0.224, 0.28, 0.35, 0.4375, 0.56, 0.7, 0.875, 1.12\}$. For each of these values of σ the ratio r between maximum internal and maximum pendant edge lengths ranged over the set $\{0.16, 0.2, 0.25, 0.32, 0.4, 0.5, 0.64, 0.8, 1, 1.25, 1.6\}$ Hence these ranges include the values of σ and r usually used, i.e., 0.35 and 0.5 respectively.

The sequence lengths c used ranged from 40 to 2500, in the usual exponential expression described previously, though only data using $c = 1000$ is shown here.

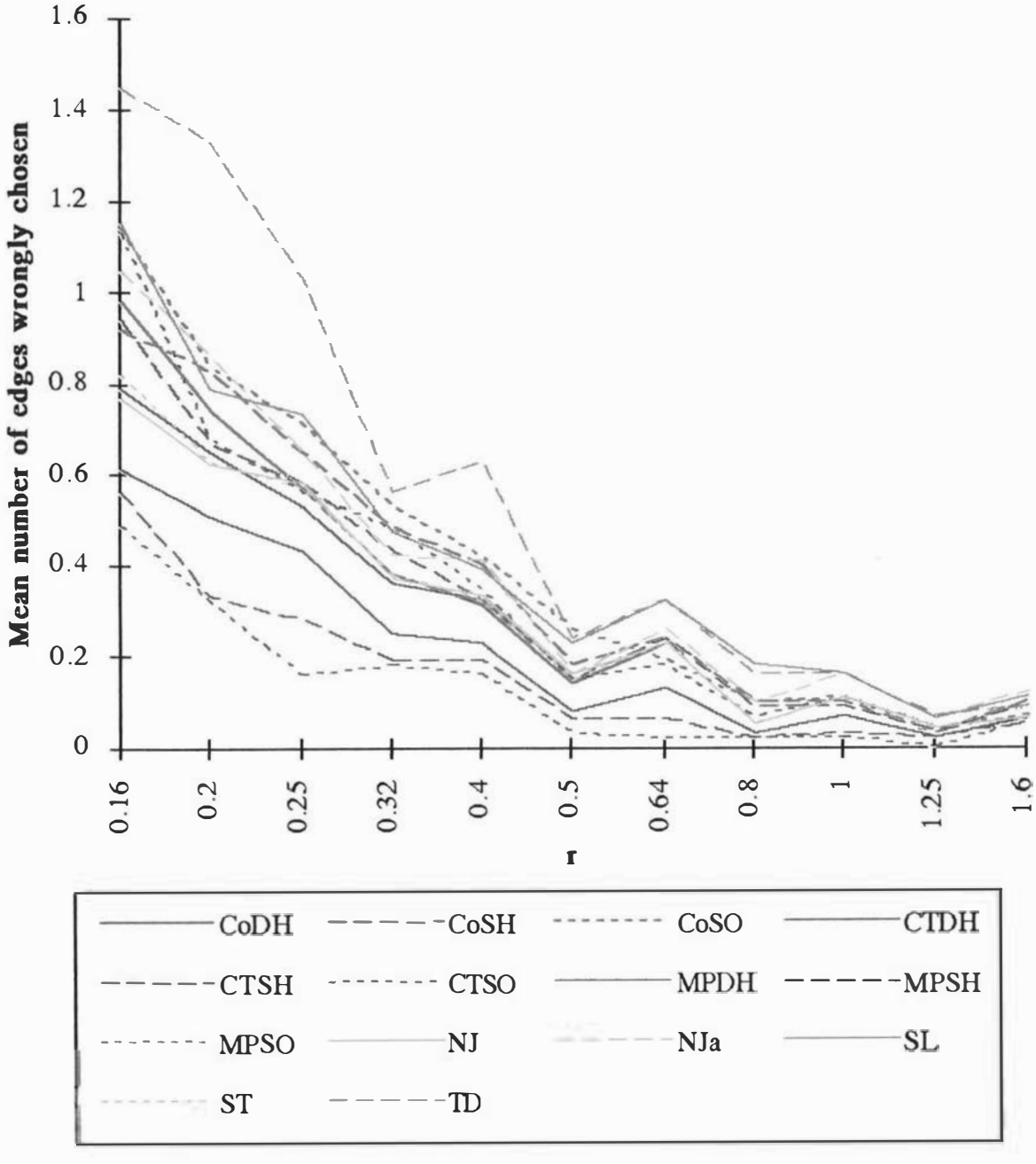


Figure 5.7: Effect of varying r with maximum path length $\sigma = 0.112$
The mean number of edges wrongly chosen by all methods with $\sigma = 0.112$ and r allowed to vary. The sequence length c is 1000.

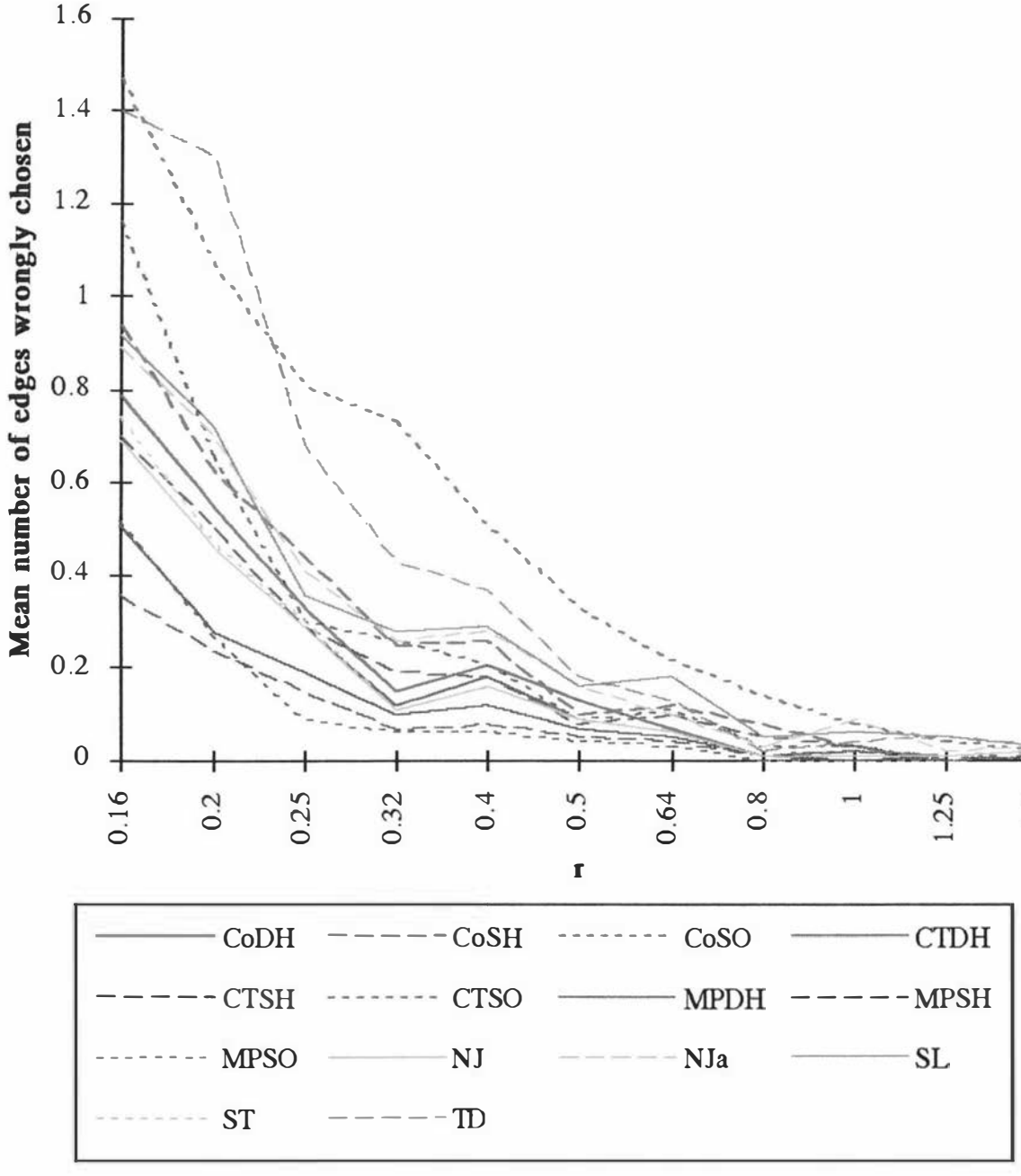


Figure 5.8: Effect of varying r with maximum path length $\sigma = 0.35$
The mean number of edges wrongly chosen by all methods with $\sigma = 0.35$ and r allowed to vary. The sequence length c is 1000.

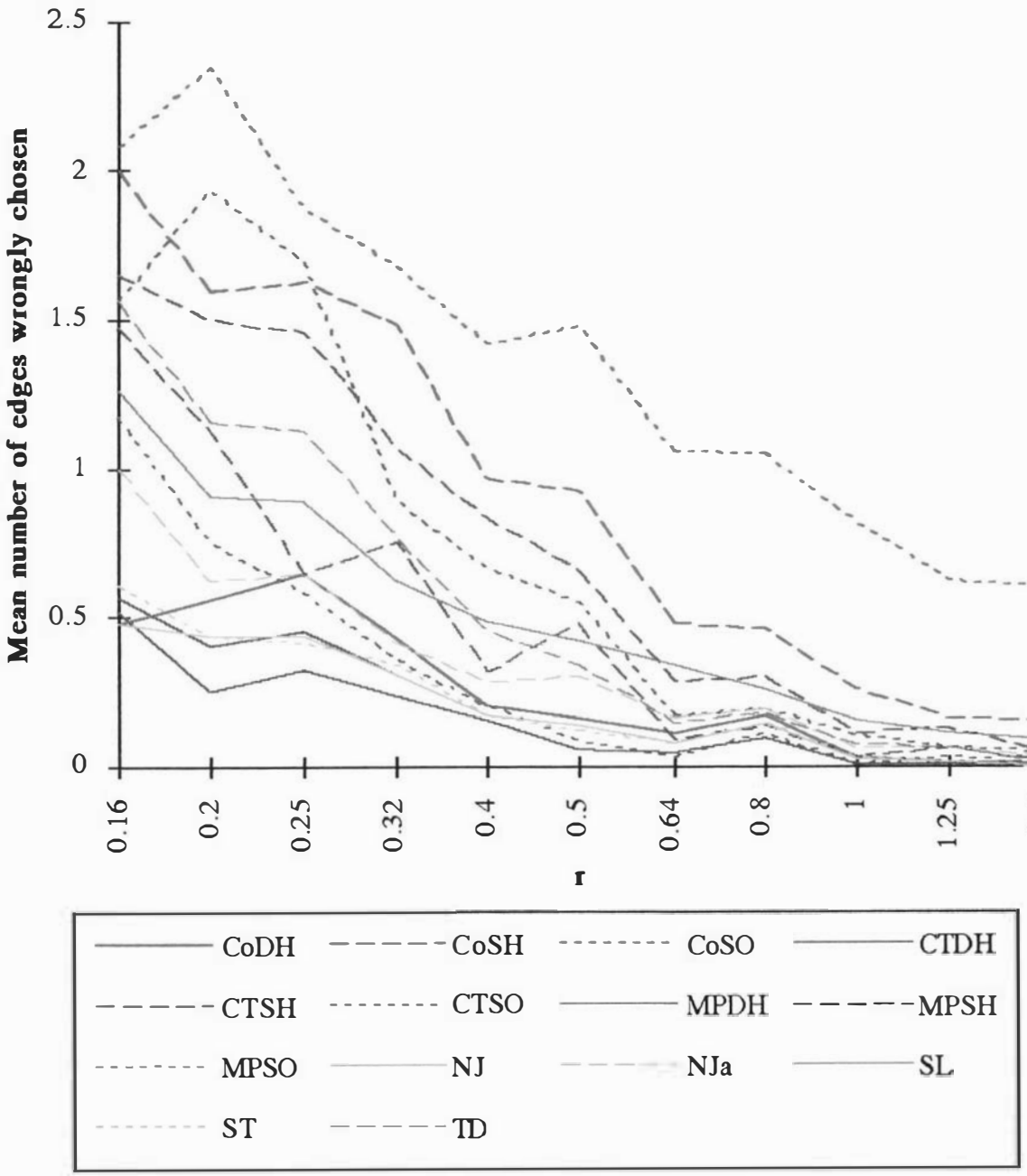


Figure 5.9: Effect of varying r with maximum path length $\sigma = 1.12$
The mean number of edges wrongly chosen by all methods with $\sigma = 1.12$ and r allowed to vary. The sequence length c is 1000.

In Figures 5.7, 5.8 and 5.9 we see the high dependence of the performance of the phylogenetic methods on r for fixed values of σ. Also, the effect on the different methods of varying σ is evident.

The following points are noteworthy:

- The apparent parallel variation between phylogenetic methods is caused by using the same data sets for each method, and would disappear with larger sample sizes.

- All the methods become more accurate as r increases. This is to be expected because the internal edge lengths become longer with increasing r and fixed σ, and are therefore more likely to have a character state change than shorter edges.

- MPSO, which does not use any information about the pendant edges (i.e., the "singletons" in the set of bipartitions, in which only one taxon has a different character state), does not appear to show a strong dependence on r above $r = 0.64$. This is, however, not a strong indication as MPSO is accurate at this level anyway.

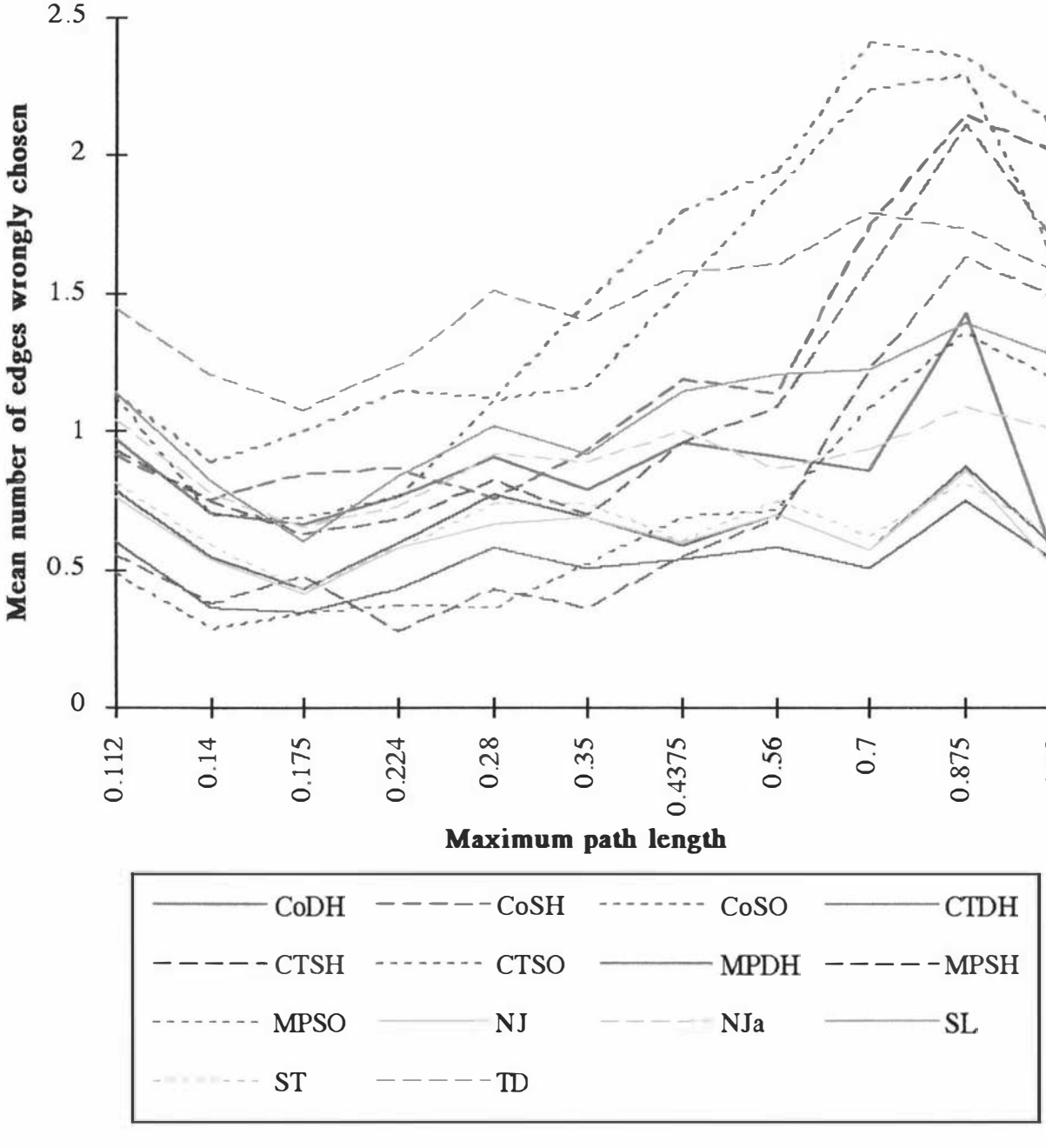


Figure 5.10: Effect of varying maximum path length σ with $r = 0.16$
The mean number of edges wrongly chosen by all methods with $r = 0.16$, $c = 1000$, $T_G = \mathrm{UB5}, 2_3$ and varying σ.

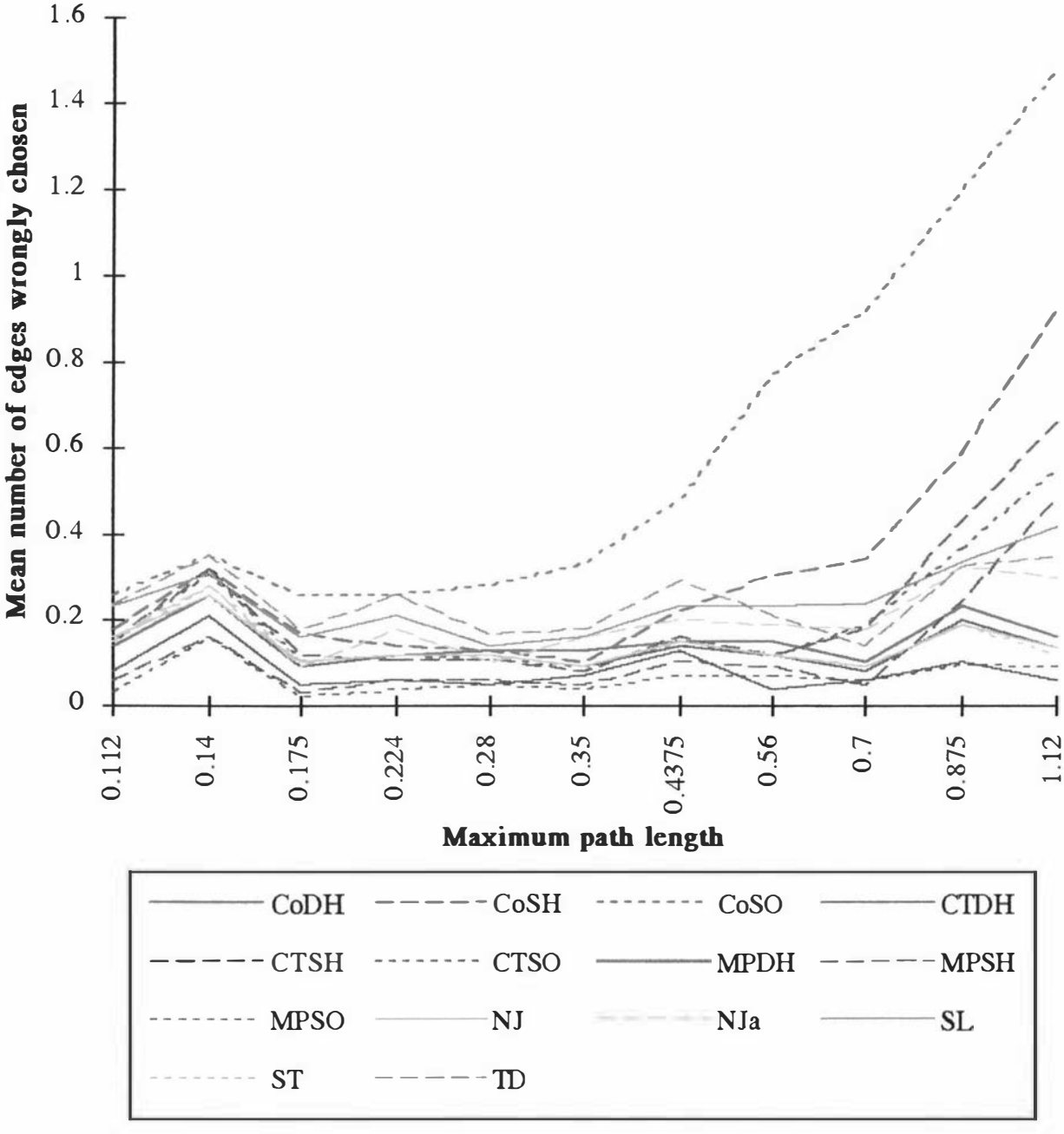


Figure 5.11: Effect of varying maximum path length σ with $r = 0.5$
The mean number of edges wrongly chosen by all methods with $r = 0.5$, $c = 1000$, $T_G = \mathrm{UB}5, 2_3$ and varying σ.

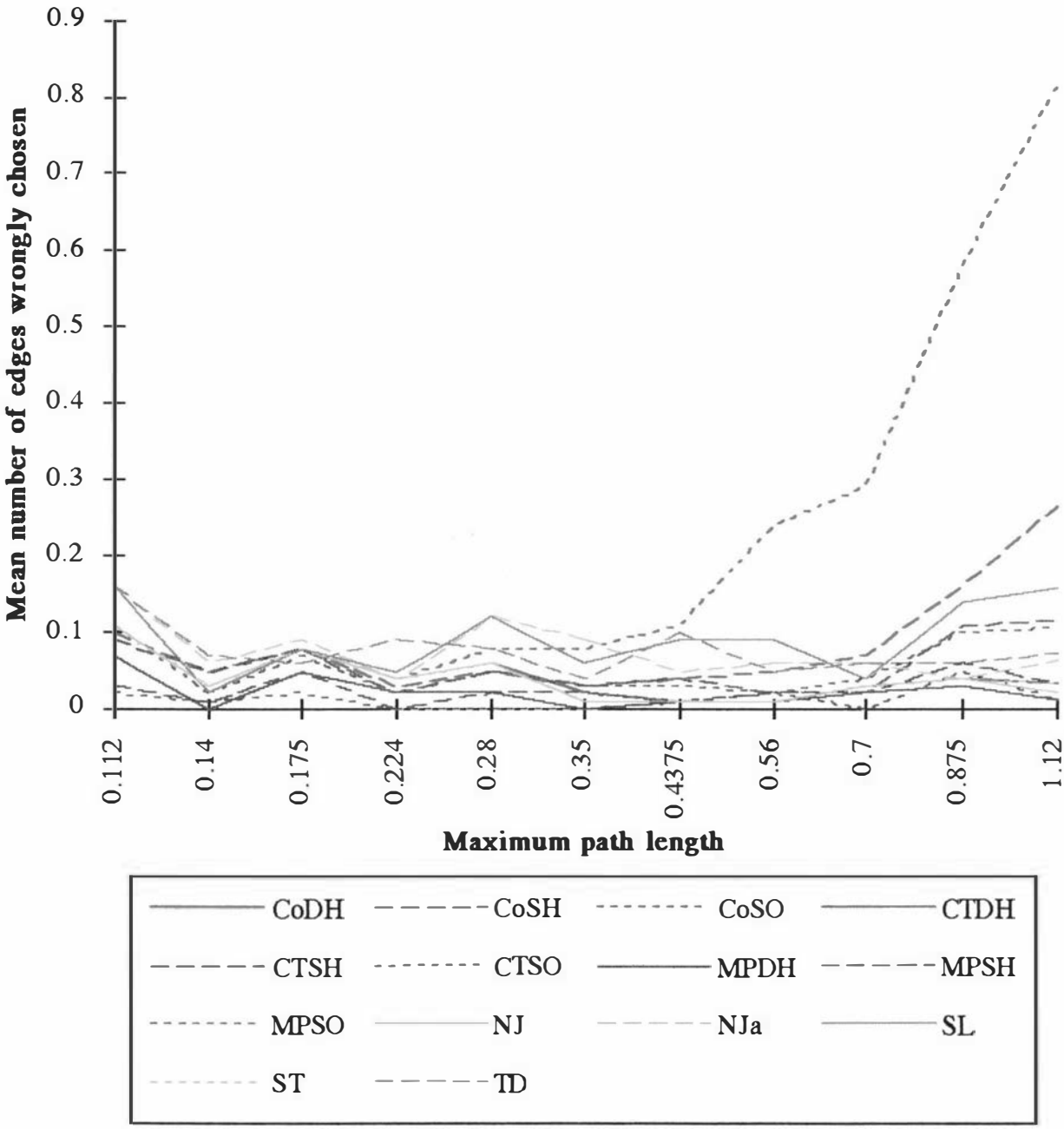


Figure 5.12: Effect of varying maximum path length σ with $r = 1$

The mean number of edges wrongly chosen by all methods with $r = 1$, $c = 1000$, $T_G = \mathrm{UB}5, 2_3$ and varying σ.

Varying σ while keeping r fixed (see Figures 5.10, 5.11 and 5.12) allows more conclusions about the effect of overall maximum path length:

- Maximum parsimony performs the most reliably, though the MPSO variant becomes less accurate than MPSH when $\sigma = 0.35$, and remains so for larger σ. This is to be expected because with large σ the internal edges are longer, so allowing for more multiple character state changes, which are not corrected for in MPSO (recall that MPSO is not consistent with the model used to generate the data here).

- All methods initially either improve their accuracy or remain at the same accuracy as path lengths start to increase. This is to be expected due to the higher probability that character state changes will occur on each edge. However when σ increases further, we find that the accuracies of these methods decrease, as the observed bipartition spectra approach randomization (data not shown).

- The three variants MPDH, MPSH and MPSO do not show any great effect on their performance with varying σ, for $r \in \{0.16, 0.5, 1\}$, and neither do CoDH or CTDH.

- The accuracies of CoSH, CoSO, CTSH and CTSO improve more slowly than do those of the other methods, with CoSO the slowest, followed by CTSO. The slow improvement in CoSO and CTSO is to be expected because they are non consistent with the model used to generate the data.

With $\sigma \geq 1.4$ the number of changes of character state between taxa is so large that the sequences approach randomness with respect to each other. This means that without impractically long sequences, there is not enough signal in the data to infer any internal edges of the generating tree, hence the accuracy of all methods decreases. Also with these large values of σ the distance 'correction' formula for inferring inter-taxon distances is frequently undefined, so these data sets are rejected, with the consequence that statistically significant experiments rapidly become infeasible.

For example, in one experiment with $\sigma = 3.5$, $c = 2000$ and the generating tree with topology UB6, every observed data set in 1000 trials had to be rejected due to the inferred edge lengths being undefined. (Data not shown.)

5.7 Number of taxa

This section details two questions which have been addressed using results from a broad set of experiments, in which all tree topologies with from four to ten pendant vertices were used for the generating trees, and the sequence length c ranged from 10 to 10^5. For each generating tree topology and each sequence length, 1000 trials were carried out.

5.7.1 Required growth in sequence length with number of taxa

Suppose we wish to have a certain confidence level in the inferred tree, with data generated according to our idealised model, and we have access to sequences of unlimited length. It would be useful to know how long the sequences must be to have the desired confidence level in the inferred tree. Though the number of possible trees increases exponentially with n, it has been shown that the rate at which the sequence length c must grow with n to retain a fixed probability P that a given method will reconstruct T_G need not be exponential. In fact, the minimal rate of growth in c with n for fixed P is no more than $O(n^2 \ln(n))$ [84].

This growth rate has been estimated for the evolutionary model used in this study, by interpolation from data like those in Figure 5.5. A graph showing the required growth rate in c with n for an 85% confidence in the inferred tree is shown in Figure 5.13, below.

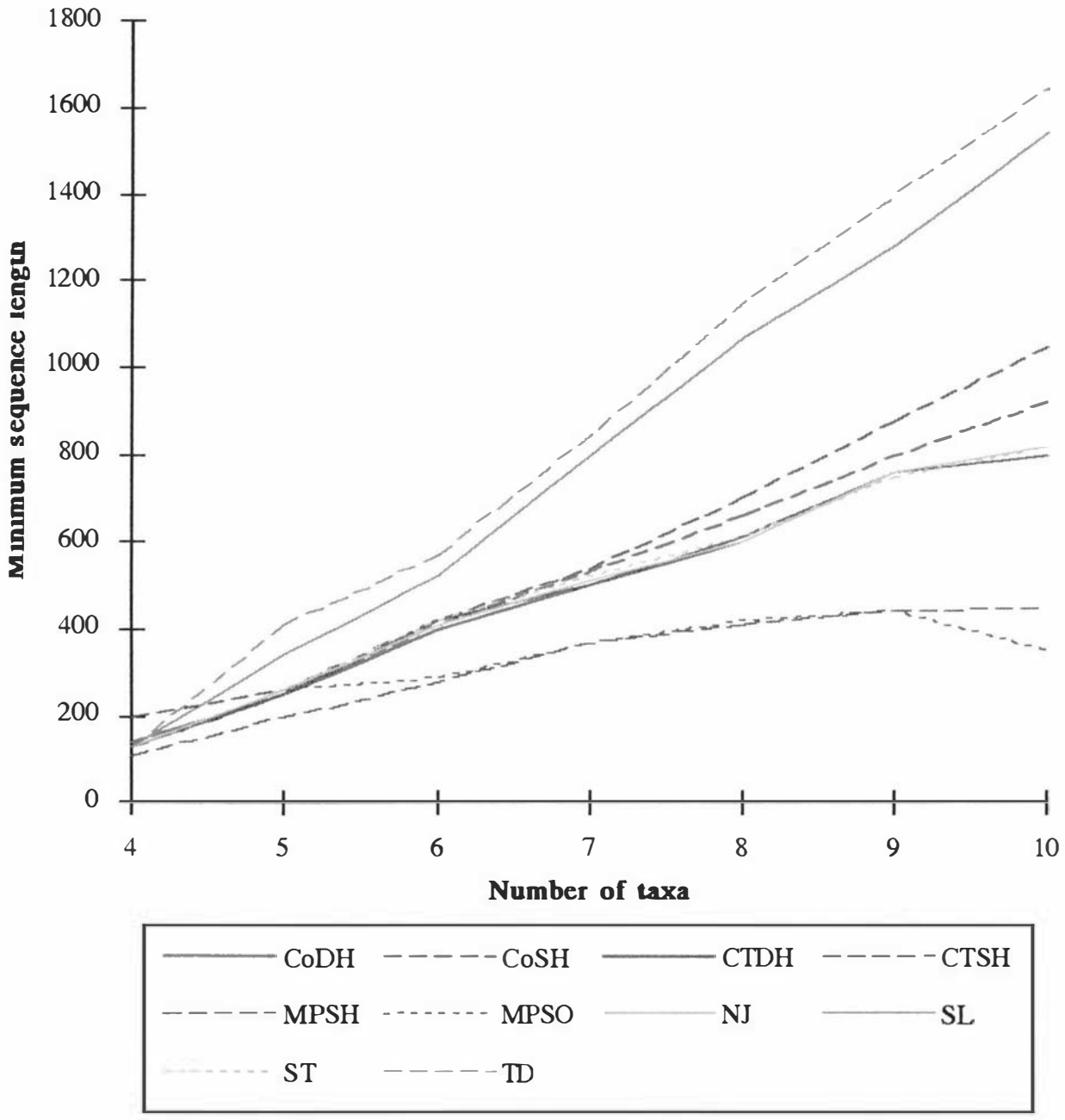


Figure 5.13: Minimum required growth rate in c with n for 85% confidence in inferred tree

This graph shows the minimum required growth rate in sequence length c with $4 \leq n \leq 10$, to retain a 0.85 probability that the inferred tree will be T_G. In this experiment the number of trials was 1000.

Figure 5.13 shows that in this case, the required minimum growth in c is approximately *linear* with n, which is less than the $O(n^2 \ln(n))$ which has been proved to be an upper bound on the growth rate. This may be counterintuitive, as the number of trees between which a choice must be made rises exponentially with n. However, it must be remembered that the number of parameters which are being estimated (e.g., clusters, internal edges) rises linearly with n, so finding an approximately linear increase in required sequence length is perhaps not so surprising.

This indicates that to obtain reliable trees for large numbers of taxa we must expect to need proportionally long sequences. Note that with real data the growth in c may well be more than linear with n.

5.7.2 Optimal number of taxa for inferring a given edge

Another question related to the number of taxa used is the following: 'What is the optimal number of taxa to include from a set of n taxa, to obtain the best estimate of the existence or non-existence of a specific internal edge of the phylogeny relating the n taxa?' Putting this another way, it can be regarded as the problem of how many taxa to add to an inferred tree to obtain the highest confidence in an edge of T_G.

As the simulations carried out in this study used different data sets for each value of n, the above question cannot be answered directly. However, by considering the mean number of edges wrongly inferred by an NPM when increasing n and keeping c fixed, we can get some idea of whether we should add more taxa or not. If the increase in the mean number of edges wrongly inferred rises proportionally less than the number of internal edges, then the probability of each internal edge begin correctly inferred is increasing with added taxa. This would mean that the reliability of the existence of a particular edge is improved by adding taxa. If on the other hand the mean number of edges wrongly inferred rises proportionally faster than the number of internal edges, adding taxa is positively unfavourable.

Considering Figure 5.14, we can see that the increase in mean number of edges wrongly inferred increases *at least* linearly with n, for all methods other than MPSH and MPSO. (Data for CoSO, CTSO and MPDH are not available for this figure.)

From the results in this section we can conclude that for these methods it is not helpful to include more taxa, in attempting to increase the confidence with which a given internal edge is inferred. The curves for MPSH and MPSO are too

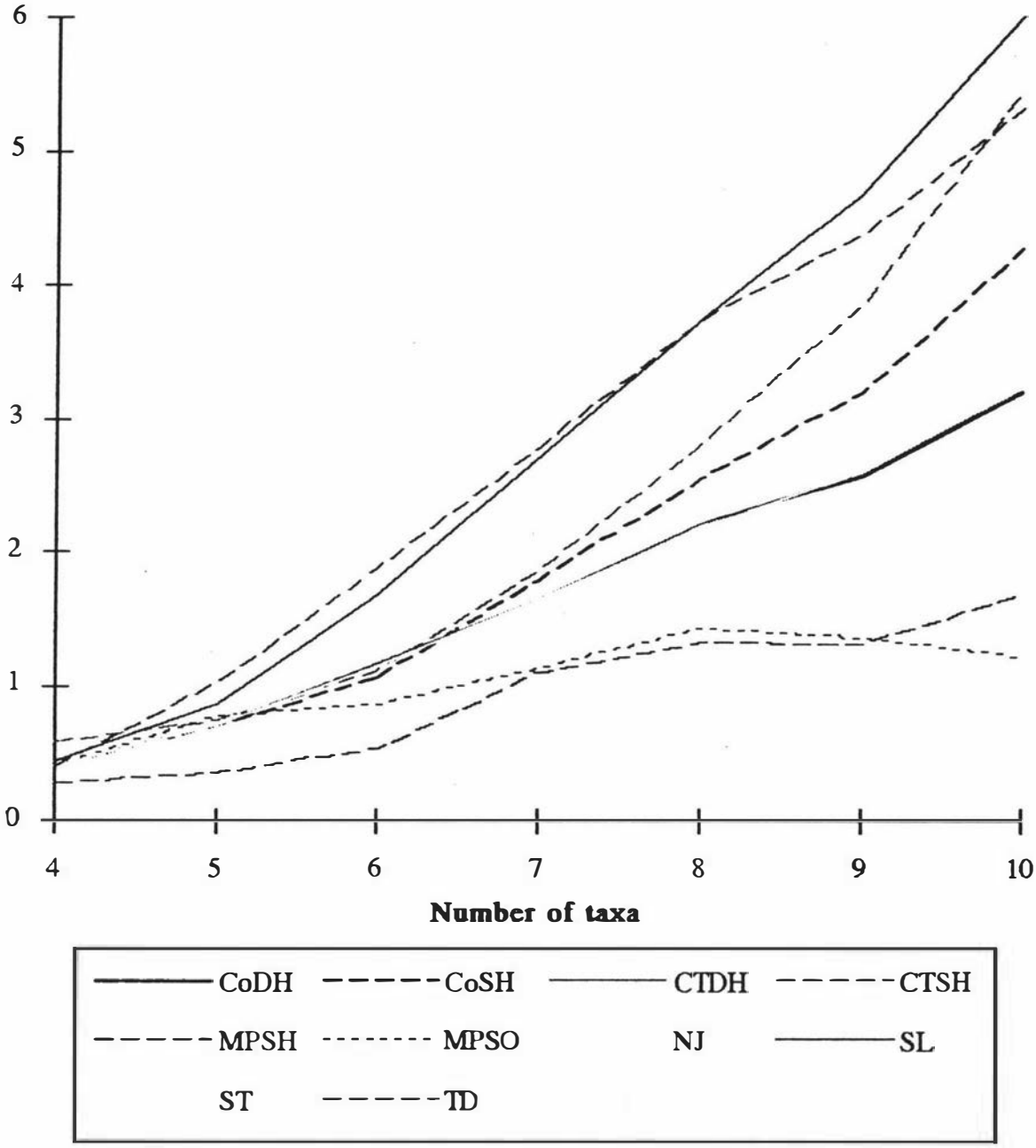


Figure 5.14: The mean number of edges wrongly inferred with increasing n
The above graph shows the mean number of edges wrongly inferred by several methods. The sequence length used was 500, and 1000 trials were carried out for each value of n. The means were calculated using the a.t.e.l. assumption.

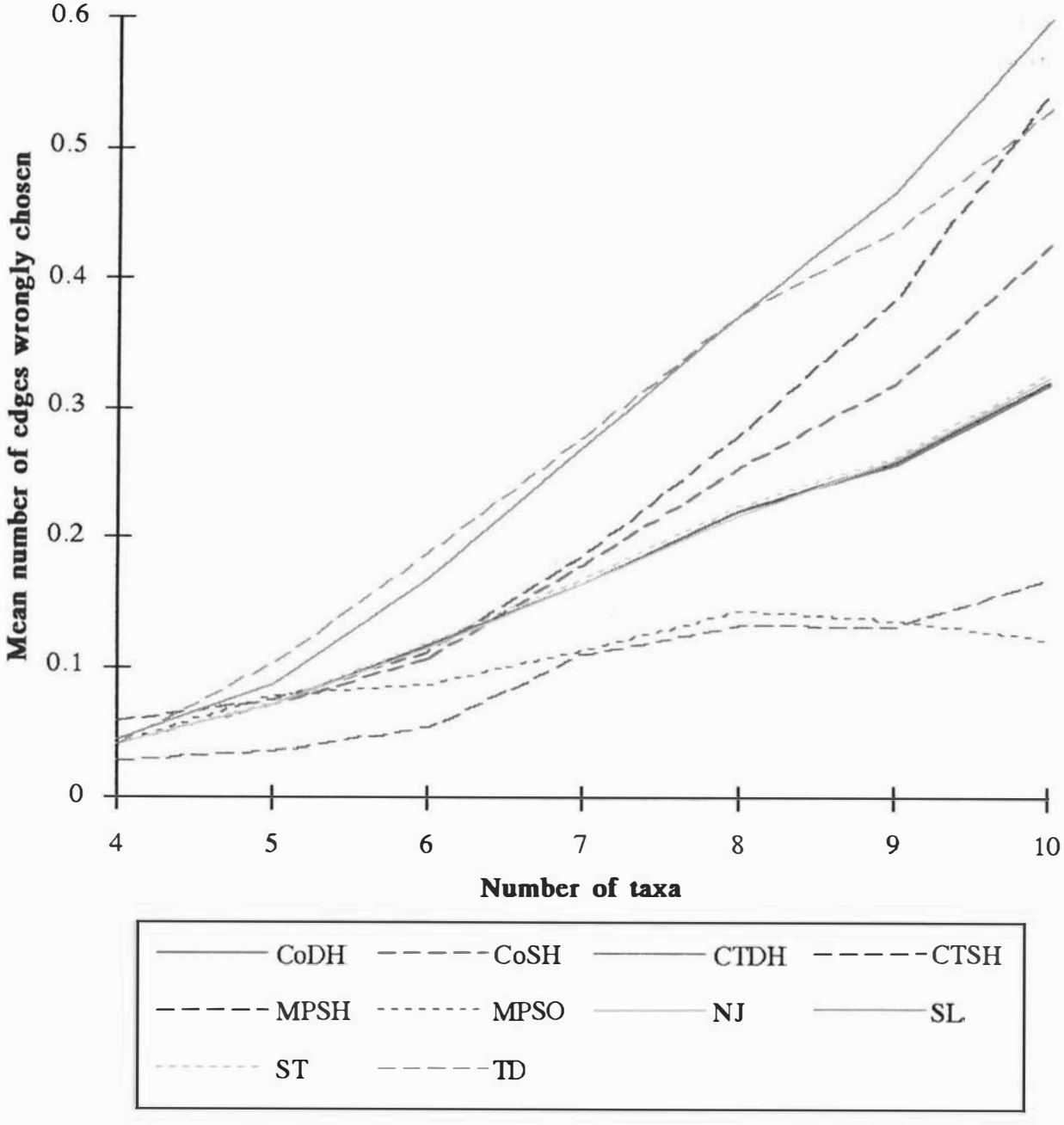


Figure 5.14: The mean number of edges wrongly inferred with increasing n
The above graph shows the mean number of edges wrongly inferred by several methods. The sequence length used was 500, and 1000 trials were carried out for each value of n. The means were calculated using the a.t.e.l. assumption.

irregular to infer that including more taxa is helpful or not. (Astolfi *et al* in 1981 showed that the number of errors — in a well-defined sense — in reconstructing trees, using several methods, increases approximately linearly with n for a much simpler case, with $n \in \{6, 10, 15\}$ [3].)

5.8 Use of the Distance Spectrum

In the results seen so far, and in subsequent results, it is apparent that using the *distance spectrum* $\mathbf{g}'$ as input to the spectrum-based methods is more effective than using the *bipartition spectrum* $\mathbf{s}'$. This is counterintuitive in that there is a huge loss of information in using the $2^{n-1}-1$ observed bipartition frequencies as opposed to the $\binom{n}{2}$ distance measures, on a set of n taxa [87].

This effect may be due to the relative sizes of the variances of the edge lengths inferred from $\mathbf{g}'$ and the variances of the edge lengths inferred from $\mathbf{s}'$.

To estimate these variances the program `sim.c` was modified as outlined below:

Algorithm 5.1 : variance.c

{ The length of edge j as inferred from the distance spectrum is here written g'_j;
that inferred from the sequence spectrum is q'_j. }
for each tree topology X
choose a vector **q** of edge lengths
calculate the vector **s** of expected bipartition frequencies
for each sequence length c
for $i = 1$ **to** 100 **do**
sample from **s** to get observed bipartition vector $\mathbf{s}'$
calculate matrix **D** of pair-wise distances between taxa
calculate observed distance spectrum $\mathbf{g}'$ from **D**
calculate inferred edge lengths $\mathbf{q}'$ and $\mathbf{w}'$ from $\mathbf{s}'$ and $\mathbf{g}'$, respectively
for each edge j calculate $var(q'_j)$ and $var(w'_j)$
print $L_{X,c} = \frac{1}{m-1} \sum_{j=1}^{m-1} \ln \frac{var(q'_j)}{var(w'_j)}$
end.

If on average the ratio of $var(q'_j)$ to $var(w'_j)$ were greater than one, the quantity $L_{X,c}$ would be positive, whereas if the ratio were on average less than one, $L_{X,c}$

would be negative. If the two variances were on average approximately the same, $L_{X,c}$ would be close to zero. (The geometric mean of the ratios $var(q'_j)$ to $var(w'_j)$ is equal to $\exp(L_{X,c})$, for tree topology X and sequence length c.)

Some results from an investigation into the variance of the inferred edge lengths are shown in Table 5.4, below. In Table 5.4 are shown values of $L_{X,c}$ for the tree topology X = UB3 and for sequence lengths $c = 100, 160, 250, 400, 640, 1000$.

The most obvious feature of the data in Table 5.4 is that they are all positive. This means that on average the q'_i values inferred from s′ have a higher variance than the w'_i values inferred from g′.

~~Table 5.4 shows little dependence of $L_{X,c}$ on sampling error except when $n = 9$, where $L_{X,c}$ increases with c.~~

Table 5.4: Variance of edge lengths as inferred from the distance and bipartition spectra

In this table are shown the results of the experiment described above. The number of trials was 100, the sequence length $c \in \{100, 160, 250, 400, 640, 1000\}$, and the upper bound σ on the maximum path length was 0.35. The generating tree had topology X = UB3. Note that the same set of edge lengths was used for all the trials, but data sampled from the expected bipartition frequencies for each trial and each value of c.

	sequence length c					
	100	160	250	400	640	1000
$L_{X,c}$	0.588	0.557	0.573	0.548	0.519	0.515

The values of $L_{X,c}$ were obtained for several other topology of generating tree, with $4 \leq n \leq 9$ and the same set of sequence lengths. In these experiments too $L_{X,c}$ was always positive (data not shown).

My conclusion from these results is that the distance-spectrum based methods perform more accurately than the sequence-spectrum based methods because the variances of the inferred edge lengths in the former group of methods are in general less than those in the latter group. This is in agreement with the results of Waddell *et al* [97].

The results from this section help explain the unexpected result that, despite the large information loss with conversion from bipartition frequencies to distances, methods using edge lengths inferred from distances frequently perform more accurately than those using edge lengths inferred from bipartition frequencies.

5.9 White noise

The technique of DNA sequencing is very accurate, but is not absolutely so [15]. There is a finite (non-zero) probability that each character on a sequencing gel will be *mis-read*. I have modelled this possibility by modifying the sampling process, so that at each site in the aligned 2-state character sequences, there is the same probability e that a character selected at random from the set of taxa will be "flipped" from one state to the other. This probability ranged exponentially from 0.001 to 0.1.

The model I have used to mimic sequencing errors can also be regarded in the more general setting of white noise: the random flipping of one character state to the other is unbiassed, essentially random, noise which may have other sources than sequencing errors. The simulations were carried out on the 'caterpillar' tree with $n = 9$ (UB7), and 100 trials were conducted for each value of e and c.

The values of e used were $\{0.001, 0.0016, 0.0025, 0.004, ..., 0.1\}$ and the values of c used were $\{25, 40, 64, 100, ..., 1000\}$. UPGMA was not used here; its behaviour is so poor for this model that it is not of interest.

The simulations show the expected gradual decrease in accuracy when the error rate e is small (less than about 0.01). The overall effect on the accuracy of the phylogenetic methods is not great in this experiment for $e \leq 0.01$, though it is detectable: for example, the mean number of edges wrongly inferred by CoDH with $c = 400$ increased from ≈ 0.35 to ≈ 0.5 when e increased from 0.001 to 0.01 (data not shown). The accuracy of the other methods behaved similarly to the above example with $e \leq 0.01$.

However, when e increases beyond 0.01, several features become apparent:

1. Above $e = 0.016$, CoDH, previously the most accurate of the Co family, loses accuracy faster than do CoSH and CoSO. At $c \geq 400$ and $e = 0.025$, the accuracy of CoDH is not significantly different from that of CoSH, with CoSO still the least accurate (recall that CoSO is not consistent with the model) (see Figure 5.15). When e reaches 0.1, CoDH is much less accurate than CoSH and CoSH; with $c = 10000$, the mean number of edges wrongly inferred by CoDH is ≈ 1, whereas with CoSH and CoSO this number is ≈ 0.75 (see Figure 5.16).

2. For $e \leq 0.04$ and $c < 640$ the relative accuracies of CTDH, CTSH and CTSO

remain the same; CTDH being more accurate than CTSH and CTSO, these latter two having approximately the same accuracy (see Figure 5.17). With $e \leq 0.04$ and $c \geq 640$, CTDH performs similarly to CTSH and CTSO. With $e = 0.064$ CTDH begins to become significantly less accurate than CTSH and CTSO when c reaches 640, and this continues with $e = 0.1$, the disagreement beginning at $c = 400$ (see Figure 5.18).

3. Maximum parsimony exhibits similar behaviour for all three variants (MPDH, MPSH and MPSO) over $0.001 \leq e \leq 0.1$, as far as can be determined from the relatively small number of trials. It appears however that MPDH gains accuracy, relative to MPSH, when the error rate e becomes very large (see Figure 5.19).

4. In general, the relative accuracies of the Distance-Hadamard methods (CoDH, CTDH and MPDH) are maintained over $0.001 \leq e \leq 0.1$. In this case MPDH is more accurate than CTDH, in turn more accurate than CoDH (see Figure 5.20).

5. The Sequence Hadamard methods (CoSH, CTSH and MPSH) also preserve this ranking (MPSH the most accurate, then CTSH, then CoSH), though when e exceeds 0.025 and for c less than about 250, the difference between CTSH and MPSH is small (see Figure 5.21). For $e = 0.064$ and $e = 0.1$ CTSH becomes the most accurate method for $c \geq 1000$ as the accuracy of MPSH in this range of c decreases more rapidly than does that of CoSH and CTSH (see Figure 5.22).

6. The Sequence Observed methods (CoSO, CTSO, MPSO) exhibit the same relative behaviour as the Sequence Hadamard methods, outlined above (figures not shown).

7. The constructive methods (NJ, NJa, SL, ST and TD) have similar accuracies to each other throughout, but NJ and ST are slightly more accurate than the others, whereas SL and NJa are slightly less accurate, and TD lies somewhere in between. This distinction between the relative accuracies becomes more apparent as e increases (see Figure 5.23).

To summarise, it appears that white noise of this type affects the performance of the phylogenetic methods gradually, as the error rate increases. Above $e = 0.025$,

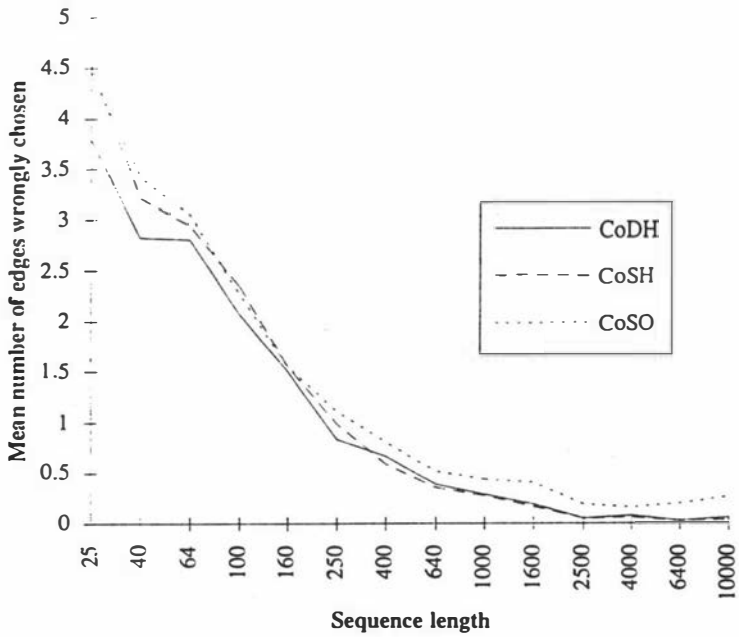


Figure 5.15: Compatibility methods with sequencing error rate $e = 0.025$

This figure shows the accuracy of CoDH, CoSH and CoSO with sequencing error rate $e = 0.025$.

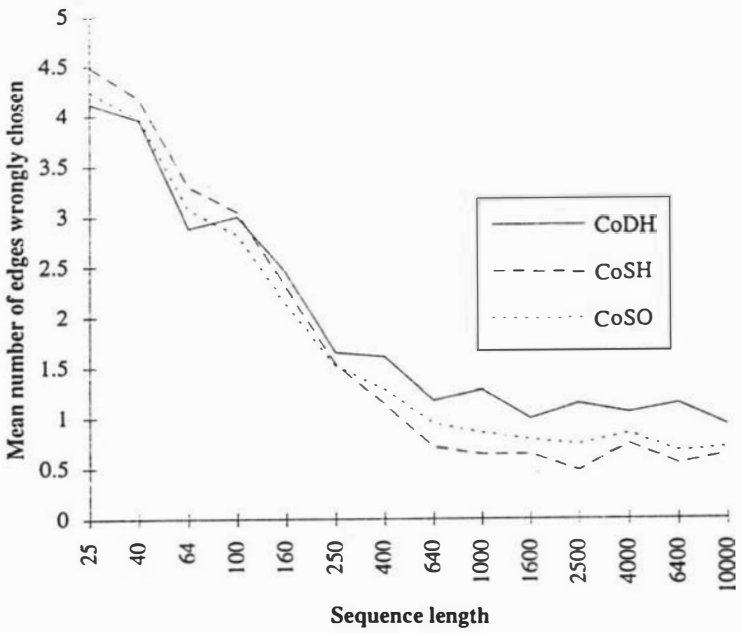


Figure 5.16: Compatibility methods with sequencing error rate $e = 0.1$

This figure shows the accuracy of CoDH, CoSH and CoSO with sequencing error rate $e = 0.1$.

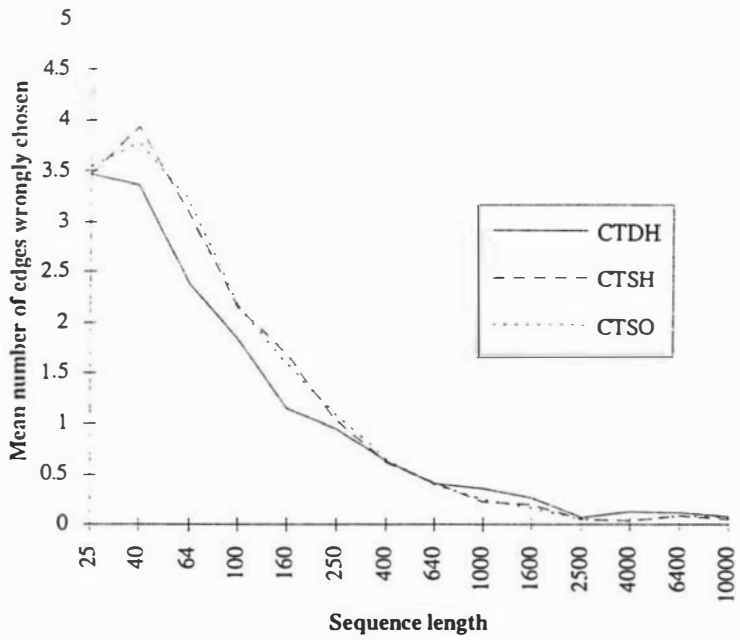


Figure 5.17: Closest tree methods with sequencing error rate $e = 0.04$

This figure shows the accuracy of CTDH, CTSH and CTSO with sequencing error rate $e = 0.04$.

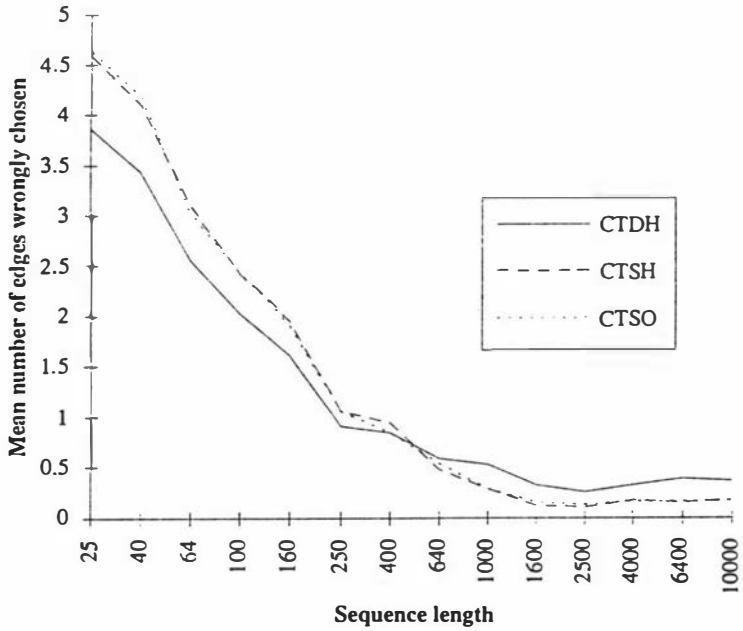


Figure 5.18: Closest tree methods with sequencing error rate $e = 0.064$

This figure shows the accuracy of CTDH, CTSH and CTSO with sequencing error rate $e = 0.064$.

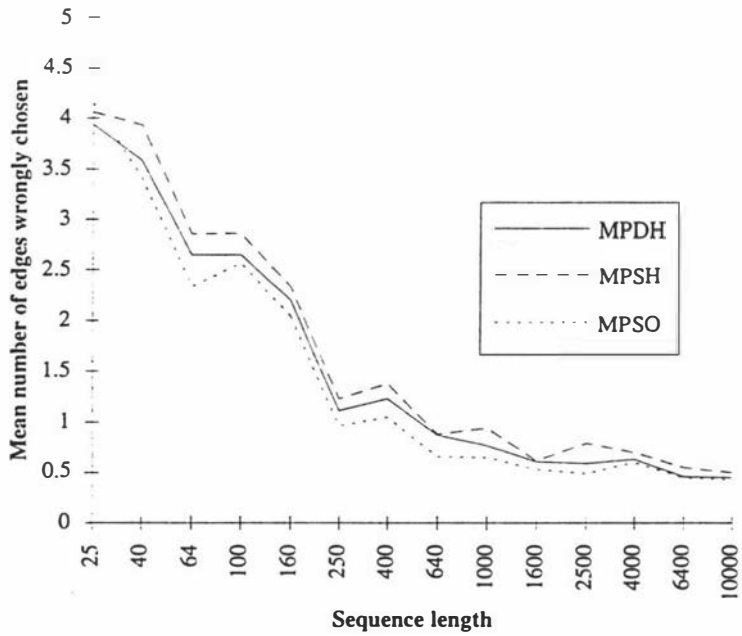


Figure 5.19: Maximum parsimony methods with sequencing error rate $e = 0.1$

This figure shows the accuracy of MPDH, MPSH and MPSO with sequencing error rate $e = 0.1$.

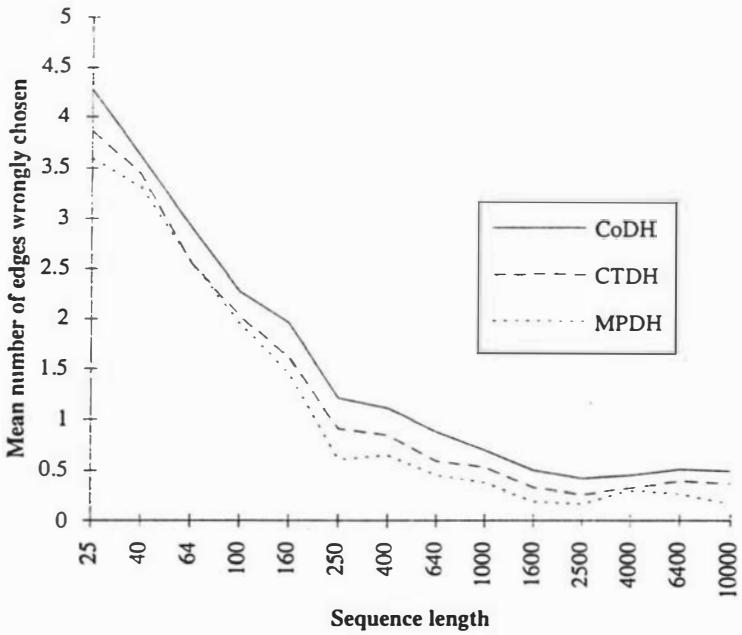


Figure 5.20: Distance Hadamard methods with sequencing error rate $e = 0.064$

This figure shows the accuracy of CoDH, CTDH and MPDH with sequencing error rate $e = 0.064$.

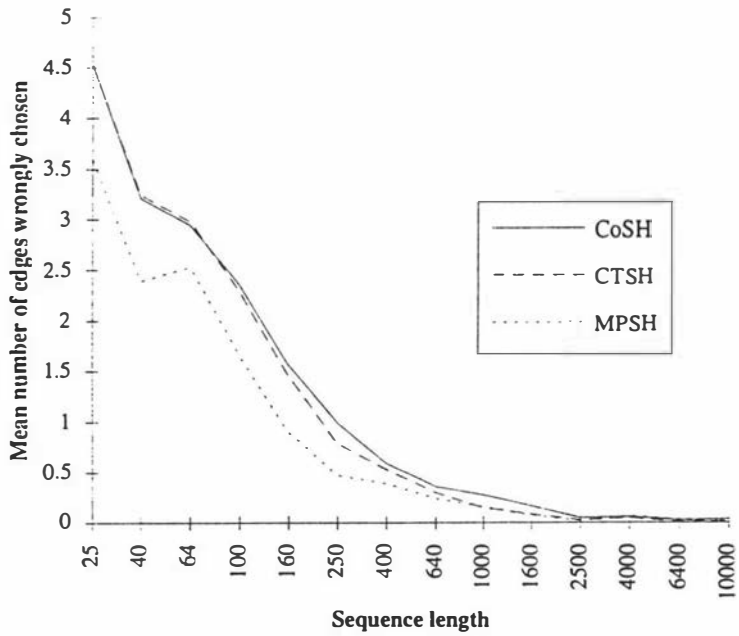


Figure 5.21: Sequence Hadamard methods with sequencing error rate $e = 0.025$

This figure shows the accuracy of CoSH, CTSH and MPSH with sequencing error rate $e = 0.025$.

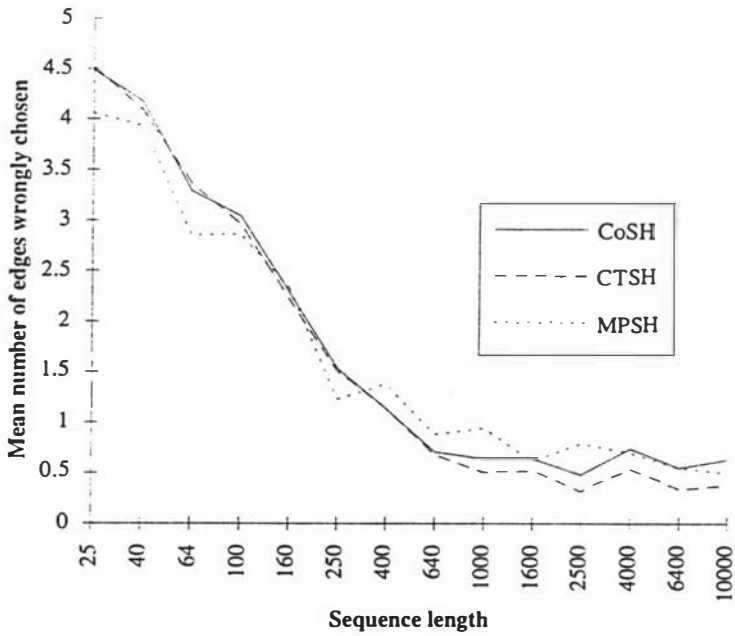


Figure 5.22: Sequence Hadamard methods with sequencing error rate $e = 0.1$

This figure shows the accuracy of CoSH, CTSH and MPSH with sequencing error rate $e = 0.1$.

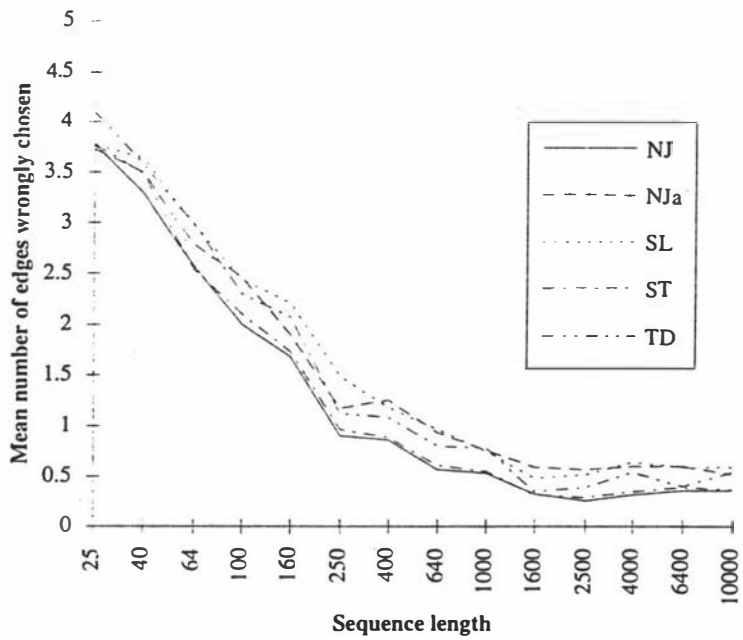


Figure 5.23: Constructive methods with sequencing error rate $e = 0.064$

This figure shows the accuracy of NJ, NJa, SL, ST and TD with sequencing error rate $e = 0.064$.

which is very high in real terms [15], the accuracy of all the methods is such that the inferred tree is unreliable, particularly for the Distance Hadamard methods, followed by Sequence Observed methods and Sequence Hadamard methods. This indicates that the Sequence Hadamard methods are *more robust* than the Distance Hadamard and Sequence Observed methods.

The constructive methods also fared badly, with the mean number of edges wrongly inferred tending towards ≈ 1 with $e \geq 0.064$ and high c. The inconsistency of CoSO had a significant effect unless e was very high (0.064 or 0.1), though the inconsistency of TD was in sense eclipsed by the sequencing errors when e rose to 0.064.

5.10 Pink Noise

The introduction of positive bias in the data towards a specific tree, different from the "true" tree, is introducing pink noise into the data. The effect of this can be investigated by amalgamating data generated from two different phylogenies and testing the methods to see how well they perform.

It is important to determine then, (1) if data from two distinct phylogenies are amalgamated, whether our method will give some indication of this (which is a measure of falsifiability — see Section 2.2.4), and (2) how such amalgamation affects the confidence we can place in the inferred tree.

In order to answer these questions, I have chosen to use two randomly chosen (but distinct) trees.

The relevant portion of the simulation program was modified as follows:

1. Choose sequence length c, number of taxa n, and proportion $\alpha \in [0.05, 1]$ of data to come from each tree T_{G1} and T_{G2}.
2. Randomly choose a tree T_{G1} with topology X_1 (in practise, X_1 = UB8).
3. Randomly choose a tree topology X_2 on n pendant vertices and a tree T_{G2} with this topology, which has at least one edge different from T_{G1}.
4. Generate observed sequence data $\mathbf{s}'$ with T_{G1} and sequence length $c_1 = \alpha \times c$.
5. Generate observed sequence data $\mathbf{s}''$ with T_{G2} and sequence length $c_2 = c - c_1$.
6. For each $i \in \{0, \ldots, 2^{n-1} - 1\}, s_i \longleftarrow \alpha s'_i + (1 - \alpha) s''_i$.

7. Use the amalgamated data as input to the NPMs, comparing the inferred tree with T_{G1}.

The simulations were carried out for all methods and the topology of the generating tree T_{G1} was UBS. α ranged from 1.0 down to 0.05, in steps of 0.05. The maximum path length was bounded by $\sigma = 0.35$ and r, the ratio of maximum internal edge length to maximum pendant edge length, was 0.5. 100 trials were carried out for each sequence length c used, and c ranged from 100 to 2000 characters. Only the data from using $c = 2000$ are shown: the effect is most clear with this value of c.

Ideally, we would like methods to correctly infer T_{G1} with $\alpha < 0.5$ and to infer the 'contaminant' tree T_{G2} when $\alpha > 0.5$: this would look like a step-function on the graph of mean number of edges wrongly inferred.

Note that the expected number of edges different between these trees is approximately 6.756 [42]. (Hendy *et al* provide mean partition distances between any two randomly chosen unrooted trees on $n = 10$, but in this case I have fixed one tree with topology UBS.)

In Figure 5.24 we see that CoSH and CTSH are much more 'step-like' than the other methods, and that of the other methods, all but CoDH, NJa, SL and UPGMA group together. This effect is also visible for lower values of c, down to $c = 500$, but is not as obvious.

This shows the higher robustness of CoSH and CTSH to interference in the sequence data in the form of pink noise.

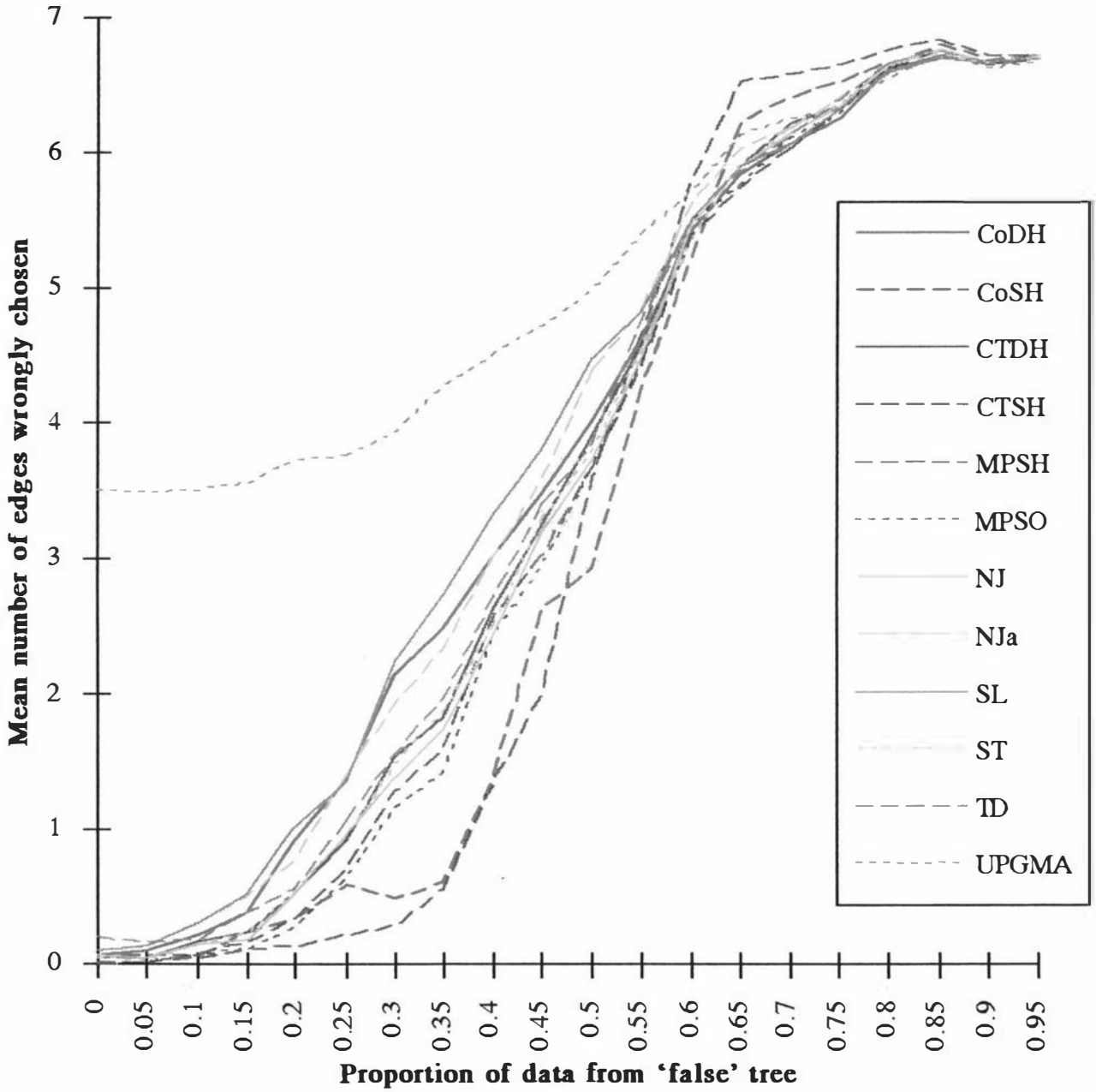


Figure 5.24: Effect of amalgamating data from two trees

In this figure is shown the effect of amalgamating data from two different trees, T_{G1} and T_{G2}. The topology of T_{G1} was UB8, and T_{G2} was chosen randomly, to be at least one edge different from T_{G1}. The sequence length used was $c = 2000$ and the number of trials was 100.

The original questions can now be answered:

1 If data from two distinct phylogenies are accidentally amalgamated, will our method will give some indication of this?
This is a question of falsifiability, in that we wish to know whether the phylogenetic method can reject the assumed model of data generation (in this case, the model of a single tree generating the data). The answer to this question has to be "no, not as yet".

2 How does such amalgamation affect the confidence we can place in the inferred tree?
If we know the proportion α of "true" data and $(1-\alpha)$ of "false" data (from T_{G1} and T_{G2} respectively), we can estimate the effect this has on the reliability of the inferred tree T. For example, with $c = 500$, the effect of introducing 35% "noise" in the form of data from an unknown tree is to increase the expected number of edges wrongly inferred by MP methods by about 1.5, whereas the expected number of edges wrongly inferred by CTSH increases by about 0.7 (data not shown). With $c = 2000$ and the same α the difference is more marked: MP is expected to get around 1.5 more edges wrong than if the data were pure, while CTSH is expected to get around 0.5 more edges wrong (see Figure 5.24).

In general the results show that, though CT does not perform as well as MP with pure data, it is considerably more *robust* with regard to contamination of the data.

It is an open question as to the conditions of amalgamating data as described above under which CTSH and CoSH remain consistent.

5.11 Summary

The results from this chapter show several basic trends, itemised below:

- Table 5.1 shows that CoDH, CTDH, MPDH, NJ and ST infer similar trees in the examples used, though the agreements between CoDH and MPDH and between MPDH and NJ are not as high as the others. Thus as NJ is reasonably accurate in the above tests, and is of a lower complexity than the others in this group, it seems a good method to use in this case.

- The performances of the methods in this chapter are predominantly affected by sampling error, which is dependent on the sequence length. We see this as, whatever the model used to generate the data (with no white noise or pink noise), when sufficiently large sequences were used (above about 2000 characters), the consistent methods all performed well, and when very short sequences were used (below about 100 characters), none performed well.

- The effect of tree topology was not significant over 100,000 trials in NJ, TD, and the Distance Hadamard methods (CoDH, CTDH and MPDH). MPDH exhibited the least dependence on tree topology, and UPGMA showed the highest dependence.

- Up to a point, as the edge lengths increase, the accuracies of the phylogenetic methods also increase. (Recall that the accuracy of a phylogenetic method is the probability that it will infer the generating tree.) This is due to the increased probability that sufficient numbers of each of the most likely bipartitions will be sampled, so that the observed bipartition spectrum will accurately reflect the expected bipartition spectrum. Conversely, when the edge lengths are very small the accuracy of the methods decreases, for the same reason.

  However, when the edge lengths become very large, the sequences approach randomness with respect to each other, and the informative signals of the most likely bipartitions become swamped in the multiple character-state changes along paths of the generating tree T_G. Hence the accuracies of the phylogenetic methods decrease once more when the path lengths exceed $\approx$ 1.25 expected character state changes per site, and tend to zero as the path lengths increase further.

- With each of the methods, the sequence length c must grow at least in proportion to n, to retain a certain probability of correctly inferring the generating tree. This is a promising result when considering that the growth in the number of potential trees is exponential, and the best known theoretical upper bound for this growth in c is $O(n^2 \ln(n))$ [84]. Also the correct inference of a specific internal edge is not helped by the addition of more taxa, for this model. Kidd and Cavalli-Svorza in 1971 [53] studied related problems, and found that the probability of correctly inferring T_G increased monotonically

with n, but they did not discuss the shape of the growth function.

- The use of the observed distance spectrum $\mathbf{g}'$ as input data reduces the variance in the inferred edge lengths of the tree. This gives the unexpected result that although much information is lost in converting from a bipartition spectrum $\mathbf{s}'$ to $\mathbf{g}'$, evidently more *misinformation* is lost, causing the edge lengths inferred from the $\mathbf{g}'$ to be more reliable than those estimated from $\mathbf{s}'$, and rendering more accurate inference of the generating tree.

- The effect of sequencing errors and similar sources of white noise is not large when the error rate e is less than about 0.01, but becomes important for $e \geq 0.04$.

- Amalgamation of data sets from two trees is less of a hindrance to the performance of CTSH and CoSH than it is to the other methods. CTSH and CoSH are therefore more robust to this form of pink noise than are the other methods.

  If confidence levels can be found for trees inferred by CoSH and CTSH and compared with confidence levels obtained for other phylogenetic methods, then the relatively high accuracy of CoSH and CTSH can indicate that the data used were not generated from a single tree. Thus with *a priori* knowledge of the reliability of the inferred trees, CoSH and CTSH have the property of falsifiability (see Section 2.2.4). Such confidence levels could be obtained using the bootstrap resampling technique [20], but this has not been tested yet.

Chapter 6

Results 2: "Large n"

> Throw big enough things at anything, and it will fall over.
>
> [*Shane Dye, 1993*]

Determining large phylogenies is certainly of interest. Hughes and Nei investigated sets of 29 and 12 taxa using histocompatibility data [49]; Hedges *et al* looked for MP trees on 136 human mitochondrial DNA sequences [38]; Chase sought MP trees with 499 land plant taxa [14].

Establishing how well it is possible to estimate the true phylogeny of big data sets like these can in part be achieved by computer simulation, as I have endeavoured to do in this part of the study.

The next section outlines some restrictions placed by the available computing power. The subsequent sections describe the effects of the following, on the performance of the clustering methods:

- Sampling error;
- Number of taxa;
- Overall time;
- Proportional time to last bifurcation event;
- Edge length probability distribution (type and spread);
- Depth of edges in the tree.

The last item in the above list, depth of edges, is considered with regard to the probability of correctly inferring individual edges, rather than with regard to correctly inferring the complete tree.

6.1 Computational considerations

Due to the large number of operations which are required to conduct a simulation trial with $n > 10$, the number of trials per set of parameters is limited. In some of the experiments in this part of the study no more than 100 trials for each parameter set were performed. However, such studies with large numbers of taxa have been limited in the past, so even this relatively small sample size provides new information. (The performance of UPGMA and some other methods not discussed here were studied by Tateno *et al* with 32 pendant vertices [93]).

The number of possible tree topologies is too large to admit testing each independently; rather, they must be chosen at random for a given value of n. The method of choice is as outlined in Chapter 4, Section 4.5.1. This is acceptable because the effect of topology on the performance is now known to be small (see Chapter 5, Section 5.5).

It should be noted that the ST method, requiring $O(n^5)$ operations to construct a tree, is not feasible for these values of n. The exhaustive search methods are not feasible either, due to the exponential number of potential trees to search and the large computational overhead of the Hadamard conjugation for inferring edge lengths. As has been mentioned (in Section 2.5), this study is not concerned with the performance of heuristic search methods, which consider only a subset of the potential trees, and therefore cannot guarantee to find a globally optimal tree.

Also NJa has been shown to perform badly, hence for $n > 10$ the only methods I have used are NJ, SL, TD and UPGMA.

6.2 Sampling error

As before, the effect of sampling error has been investigated using differing sequence lengths c. The maximum value of c is restricted for large n to no more than 1000.

Sampling error remains a strong influence on the performance of these constructive methods. The behaviour of the phylogenetic methods follows the same trends

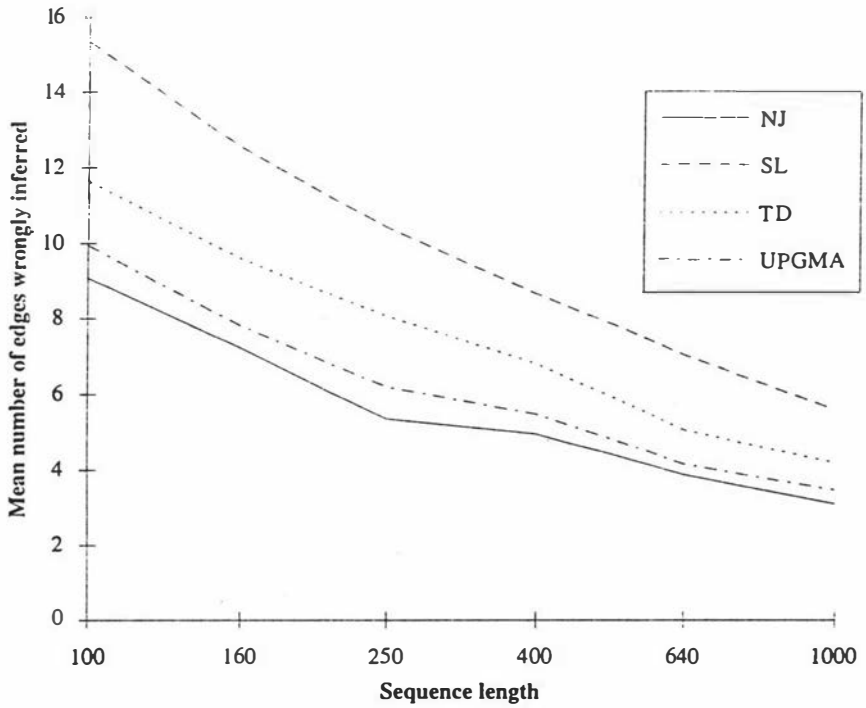


Figure 6.1: Effect of sequence length with $n = 30$

This graph shows the mean number of edges wrongly inferred by NJ, SL, TD and UPGMA with 30 pendant vertices and $100 \leq c \leq 1000$.

.pparent for small n, though the their relative behaviour is different: NJ is the nost accurate, followed by UPGMA, TD and SL. The good behaviour of UPGMA nay be accounted for by the model of data generation used, i.e., the rooted tree, vith edge lengths satisfying the molecular clock model. The molecular clock model s equivalent to having equal expected amounts of change between the root and the endant vertices, for all lineages.

With more variable rates of evolution on different lineages, I show later that JPGMA behaves less well, as expected by its inconsistency with data which are ıot generated according to a molecular clock.

3.3 Number of taxa

)ne obvious problem in phylogenetic inference lies in having a large number of axa; when n increases we must expect the accuracy of phylogenetic methods to lecrease, and this is shown in Figure 6.2 below. There is a strong dependence on ı of the accuracy of constructive methods.

In Figure 6.3 is shown the number of trials in which $0, 1, \ldots, 15$ internal edges ıre wrongly inferred, for a sequence length of $c = 1000$ and for $n = 30$. In this experiment no more than 15 edges were incorrectly inferred.

It has been suggested that generating an initial tree with a clustering method such as NJ, and then searching nearby trees (either one or two edges different from :he initial tree) will be a useful heuristic search of tree-space to find "significantly better" trees according to some optimality criterion [59], [74]. Figure 6.3 shows that the distribution of numbers of edges wrongly inferred for large values of n is potentially more than two edges removed from the generating tree: the most likely number of edges wrongly inferred ranges from two with NJ, through four with TD and UPGMA, to five with SL, even with this simple model.

Hence, I contend that the use of most constructive methods to obtain a starting point for search methods based on hill-climbing is not necessarily sufficient, though of the methods tested here, the tree inferred by NJ is the most likely to be within two edges of the true tree.

The other obvious feature of Figure 6.3 is that for each of the clustering methods the probability of reconstructing T_G is very small with even this moderate value of n. When considering trees built by clustering methods with 100 or more taxa, the

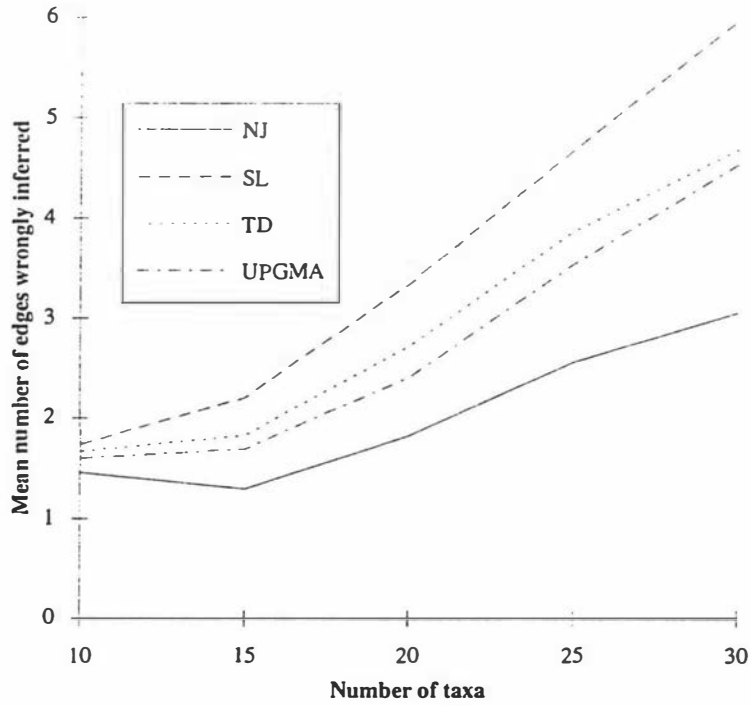


Figure 6.2: Effect of number of taxa on the accuracy of constructive methods
The mean number of edges wrongly chosen is shown, with varying n and $c = 1000$.

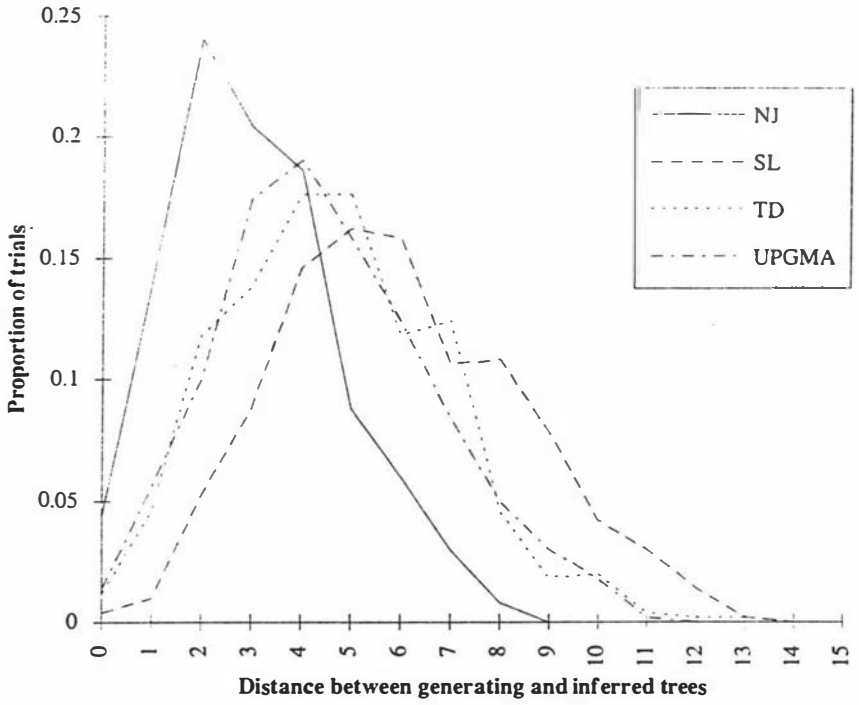


Figure 6.3: Distance from T_G of inferred trees

This graph shows the proportion of trials in which each of the constructive methods infer $0, 1, \ldots, 15$ edges incorrectly, with $n = 30$ and $c = 1000$. In this experiment 500 trials were carried out, and no more than 14 edges were incorrectly inferred by any of the methods used.

probability of inferring T_G must therefore be vanishingly small: extrapolating from Figure 6.2 would suggest that for a generating tree with 100 taxa, the expected number of edges wrongly inferred by, for example, NJ, would be at least 10.

6.4 Overall evolutionary time

The principle of varying the overall evolutionary time for these large rooted trees is similar to that of varying the maximum path length of the small unrooted trees (see Section 5.6. Note that the overall time is now expressed in terms of the mean number of character state changes to have taken place between the root and the pendant vertices.

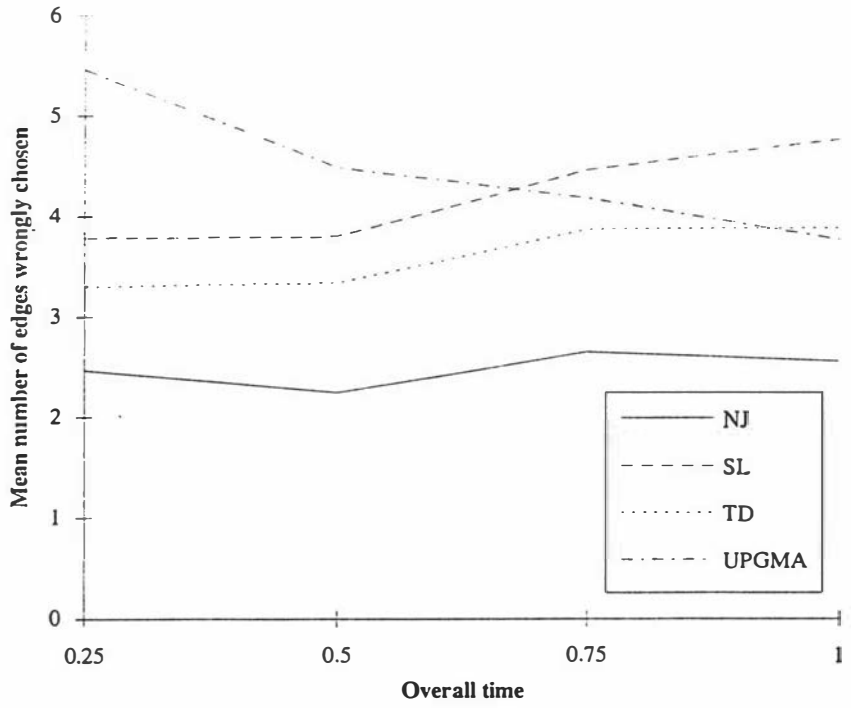


Figure 6.4: Effect of time from the first bifurcation to the present
The mean number of internal edges wrongly inferred by NJ, SL, TD and UPGMA is shown. The number of taxa is 26, and the sequence length is 1000.

Figure 6.4 shows the mean number of edges wrongly inferred by NJ, SL, TD and UPGMA with sequence length $c = 1000$ and overall time $ot \in \{0.25, 0.5, 0.75, 1.00\}$. (These times are of course relative; the overall amount of change expected between the root and the present is also a function of the transition matrix M.) The figure

shows that the accuracies of NJ, SL and TD all decrease slightly with increasing overall time, but UPGMA becomes more accurate. We shall see in the following sections that UPGMA is highly dependent on the "molecular-clockness" of the tree.

The effect of overall time can be further investigated by considering the divergence of just two sequences through time. (This experiment was discussed by Penny *et al* [63].) In the following experiment, two initially identical sequences of four character states were allowed to "evolve" for the same amount of time t, and their dissimilarity used to infer the overall time, using the Jukes-Cantor formula. The sequence length c was 1000 characters. For each character the probability that it changed state was set to 0.01 (1 %) in 1 million years, and the transition matrix governing the probabilities of character state change over t calculated as described in Section 4.5.3.

The overall time t ranged exponentially from 0.01 to 1000 million years. For each time t, 100 trials were carried out, so the probability distribution of the inferred time t' could be estimated.

With small values of t ($\leq$ 0.1 million years) the probabilities of change were so low that often there was no difference between the two sequences, so t' would be 0. For very large values of t ($\geq$ 100 million years) there was often so much change that infinite distances would be inferred with the Jukes-Cantor formula. If the sequences were identical, in which case $t' = 0$, or so dissimilar that $t' \rightarrow \infty$, the sequences were abandoned.

In Figure 6.5 below, the mean standard error of t' is shown. Also shown are the proportion of trials which provided meaningful results (where $t' \notin \{0, \infty\}$), and the mean value of t'/t, for each value of t.

Note that the mean standard error of t' is lowest for 10 million years $\leq t \leq$ 100 million years, and that when t gets larger or smaller than these values, the inferred overall time rapidly becomes more unreliable. Hence the most useful range of t is such that the expected number of character state changes between pairs of taxa is between approximately 0.1 and 1.0, per character.

6.5 Time to last bifurcation event

This section examines the effect of varying the relative time until the last bifurcation event. Such variation is equivalent to varying the ratio r between maximum internal

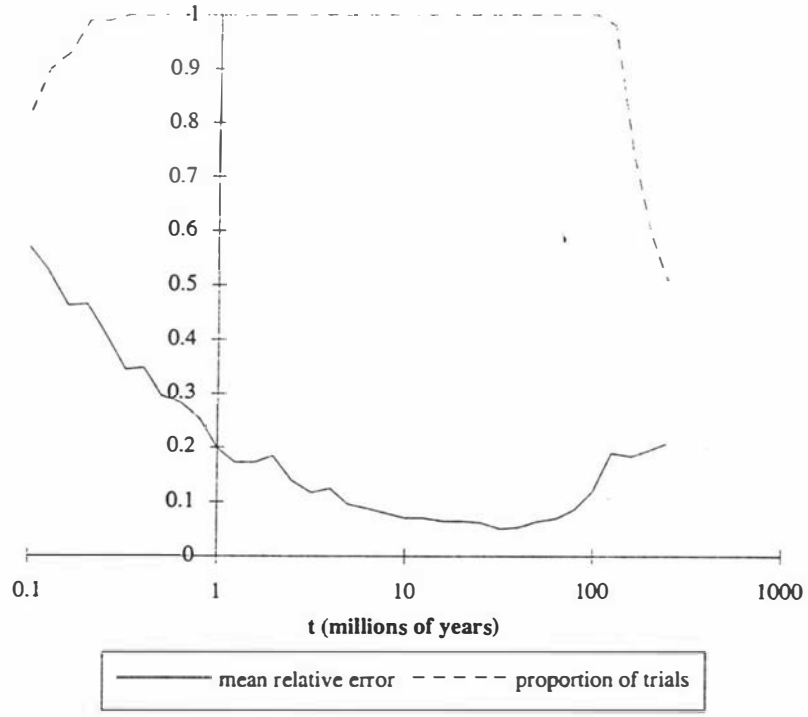


Figure 6.5: Inferred time of divergence of two sequences

The above figure shows the behaviour of estimated divergence times of two sequences which were allowed to changed stochastically from being originally identical. Given the actual time t, the inferred time for any given trial is t'. The horizontal scale is logarithmic. The proportion of trials in which $t' \notin \{0, \infty\}$ is shown, ~~as are the mean values of t'/t~~ and the mean standard error $\sigma(t')/t$ when for $t' \notin \{0, \infty\}$.

and maximum pendant edge lengths. This helps explain the effect of the 'depth' of edges within T_G has upon the probability that they will be correctly inferred, but this is discussed more fully later.

When assigning edge lengths to the generating tree T_G, the proportion of time in which bifurcation events can take place is the *divergence time factor* f (see Section 4.5.4). In this part of the investigation f was allowed to range over $\{0.2, 0.325, 0.45, 0.575, 0.7, 0.825, 0.95\}$. The number n of taxa ranged over $\{10, 15, 20, 25, 30\}$. With f small, the internal edge lengths are shorter than the pendant edge lengths, and with $f \to 1$, the internal edge lengths are longer than the pendant edge lengths.

The results shown in Figure 6.6, below, show trends which were common to all the values of n used (data not shown).

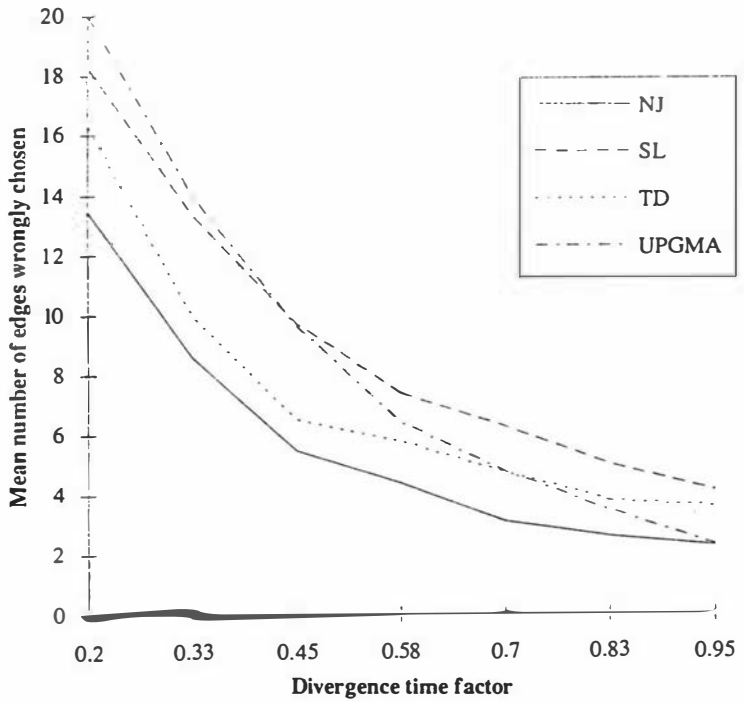


Figure 6.6: Effect of divergence time factor f

The graph shows the mean number of edges wrongly inferred by NJ, SL, TD and UPGMA, for $n = 30$, $c = 1000$, and varying divergence time factor f. 100 trials were carried out for each value of f.

Figure 6.6 and similar figures for $n \in \{10, 15, 20, 25\}$ show that

- All of the methods improve their accuracy with increasing f (and therefore increasing internal edge lengths). This is similar behaviour to the "small n"

case.

- NJ performs the most accurately of the methods used, always more accurate than TD, in turn more accurate than SL.
- UPGMA, while the least accurate method with $f \leq 0.33$, becomes at least as accurate as SL when $f = 0.45$, and becomes more accurate than TD when $f = 0.83$.
- UPGMA is approximately as accurate as NJ when $f = 0.95$.

The proportional time until the last bifurcation event is clearly a major factor affecting the performance of these constructive methods. When f is small, none of the methods perform well, getting from approximately ***half to two-thirds of the internal edges wrong.*** When f becomes close to the overall time from the root to the pendant vertices, all the methods perform well. This effect is almost certainly due to the relatively large size of the internal edges when f is larger.

Once again, the main factor here is the length of the internal edges, and hence, the sample size: with small internal edges, there is lower probability that any character state change will occur on the edges, so longer sequences are necessary to obtain enough 'signal' in the data to infer a tree.

6.6 Edge length distribution

The effect of the edge length probability distribution has been indirectly observed in previous sections (6.4 and 6.5). Now the question of how the type and spread of the edge length probability distribution affects the performance of the clusterinf methods is addressed explicitly.

6.6.1 Type of distribution

The three types of probability distribution I have used are uniform, normal and log-normal. In one experiment with varying n, each of these three distributions was used, with the other parameters constant.

The behaviour of the clustering methods was not greatly affected by using different forms of probability distribution (see Table 6.1).

Table 6.1: Effect of edge length probability distribution
The mean number of edges wrongly inferred is shown, for $n \in \{10, 20, 30\}$ and the clustering methods NJ, SL, ST, TD and UPGMA. The sequence length c was 640 and 500 trials were carried out for each value of n.

$n = 10$	NJ	SL	ST	TD	UPGMA
uniform	1.918	2.086	1.920	2.046	1.966
normal	1.836	2.070	1.846	2.020	1.916
log-normal	1.830	1.992	1.818	1.950	1.850

$n = 20$	NJ	SL	TD	UPGMA
uniform	2.204	3.870	3.130	2.320
normal	2.274	3.916	3.106	2.824
log-normal	2.226	3.930	3.082	2.404

$n = 30$	NJ	SL	TD	UPGMA
uniform	3.558	6.730	5.158	3.970
normal	3.714	6.888	5.436	4.954
log-normal	3.716	6.778	5.348	4.236

I conclude that the *type* of distribution is not as important as its spread, as outlined in the next section. With the small number of trials, only 100 for each experiment, the differences in performance are usually not significant with spread of the distribution, the one obvious exception being UPGMA with $n = 30$, this method being significantly less accurate when the spread of the edge length probability distribution increased.

6.6.2 Variance of the distribution

Making the variance of the probability distribution of the lengths of the edges of these rooted trees large will force the trees away from the molecular clock model. By letting this variance range from zero (corresponding to a perfect molecular clock) to the mean edge length from which each edge length is sampled, we are able to investigate the robustness to violation of the molecular clock hypothesis of each of the clustering methods used.

In Figure 6.7 we see the much higher dependence of UPGMA on the variance of the edge length probability distribution. This is of course to be expected as UPGMA is not consistent with distance data which do not satisfy the molecular clock model. Note that the other methods (NJ, SL and TD) also become less accurate as this variance increases.

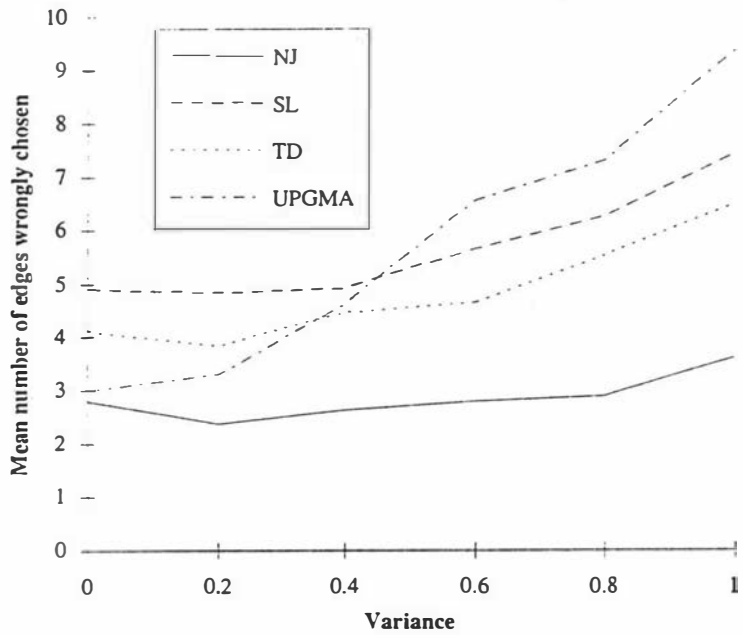


Figure 6.7: Effect of variance of edge length probability distribution
The mean number of edges wrongly chosen is shown, with $n = 26$, $c = 1000$, and overall time = 1.0.

6.7 Depth of edges

Define the *depth* of an internal edge e to be the minimum of the two numbers of pendant vertices which are partitioned by that edge. Then $2 \leq depth(e) \leq n/2$. The question of how the depth of an edge in T_G affects the probability that it will be inferred correctly can be addressed by measuring the depth of each correctly inferred edge when comparing the inferred tree T with T_G.

The number of possible edges with a given depth $k \neq n/2$ is $h(n,k) = \binom{n}{k} b_k b_{n-k}$, where b_k is the number of rooted binary trees on k pendant vertices, equal to $(2k-3)!! = (2k-3)(2k-5)\ldots(3)(1)$. This is because there are $\binom{n}{k}$ ways to choose k pendant vertices from $\{1,\ldots,n\}$, b_k possible rooted binary trees with those k pendant vertices, and b_{n-k} possible rooted binary trees with the remaining $(n-k)$ pendant vertices.

For $k = n/2$, $h(n,k) = \frac{1}{2}\binom{n}{k} b_k^2$. The division by 2 is because $\binom{n}{k} b_k^2$ counts the rooted trees with k pendant vertices twice.

The numbers of edges with depth k for $n = 26$ are shown in Table 6.2, below.

Hence the count of number of trials in which edges of each possible depth k are correctly inferred must be normalised by dividing by $h(n,k)$.

Table 6.2: Number of edges with depth k over all binary trees on 26 pendant vertices

depth k of edge	$h(26,k)/10^{30}$
2	8.2464821840
3	4.3981238315
4	2.9406060501
5	2.2090406425
6	1.7842251344
7	1.5155580292
8	1.3369386900
9	1.2153988091
10	1.1330653414
11	1.0797864069
12	1.0497923400
13	0.5200509746

The normalised values are depicted in Figure 6.8, below.

In Figure 6.8 we see the influence of edge $depth(e)$ on the probability of four clustering methods correctly choosing edge e as being in the generating tree T_G. (Note that the numbers in the figure are *relative* frequencies at which edges of each

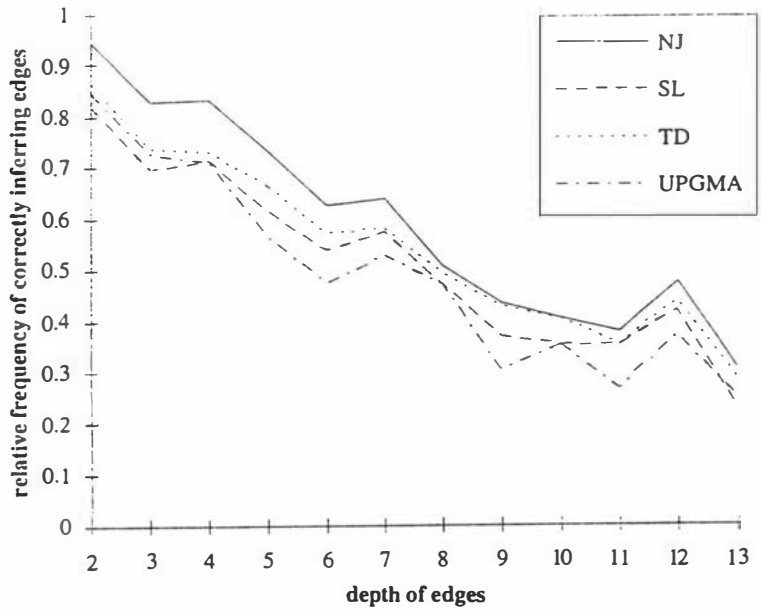


Figure 6.8: Effect of edge depth on the probability of its correct inference
The graph shows the relative frequency at which edges of depth k, for $k = 2, \ldots, 13$, were correctly inferred by NJ, SL, TD and UPGMA. These values have been adjusted to take into account the relative frequency of edges of each depth k.

depth k were correctly inferred. These relative frequencies estimate the actual probabilities up to multiplication by a constant.)

These probabilities decay approximately exponentially, which indicates that the probability of inferring edge e may follow a Poisson distribution in the depth of e.

If the probability of correctly inferring an internal edge at each clustering stage is, say, some constant w, then the probability of inferring an edge of depth 3 will be approximately w^2, and the probability of inferring an edge of depth 4 will be approximately w^3. In general the probability of inferring an edge of depth p would be approximately w^{p-1} for $n > 2p$. (Note that the depth of the edge is the minimum of the numbers of taxa which are bipartitioned by that edge. Each internal edge could also be inferred by correctly deducing the monophyly of the larger of the two sets, which introduces some higher order terms in the probability.)

If then the above were true, we would expect that the graph in Figure 6.8 would be an exponential decay. In Table 6.3 are shown the correlation coefficients of the natural log of the relative frequency of inferring edge e correctly, against the depth of e.

Table 6.3: Effect of depth of edges on their correct inference

This table shows the correlation coefficient of ln(relative frequency of correctly inferring edge e) with $depth(e)$. Here $n = 26$ and $c = 1000$.

	NJ	SL	TD	UPGMA
r	−0.9581	−0.9403	−0.9648	−0.9427

From this table it is clear that there is a strong negative correlation between the log of these relative frequencies and the depth of the edges, but this is not strong enough to imply the process closely follows a Poisson distribution.

6.8 Summary

From the work described in this chapter the following general results are obtained:

- It has been demonstrated that with larger numbers of taxa all the clustering methods perform less accurately. This is to be expected for two reasons: that as n increases, (1) there are more clustering steps which must be correctly made to infer the generating tree T_G, and (2) the internal edge lengths are shorter, with consequently lower expected numbers of character state change on them.

- The overall time from the root to the pendant vertices has a large effect on the performance of these clustering methods. As it increases, the internal edge lengths increase and the accuracy of all the methods increases. This trend reverses when the lineages are all so long that they approach randomization with respect to the root. The same conclusion was reached in the "small n" case.

- Varying the relative time to the last bifurcation event shows that with internal edges which are relatively short with respect to the pendant edges, the accuracy of the clustering methods is again reduced. This is in accordance with the result obtained in the "Small n" case.

- The general type of the edge length probability distribution does not in this case have a significant effect on the performance of the clustering methods. However, the *spread* of the distribution, which is a measure of the "molecular-clockness" of the generating tree, does have a strong effect. The spread of this distribution is highly negatively correlated with the accuracy of the clustering methods.

- The ability of these methods to correctly infer the internal edges of T_G is reduced approximately exponentially with the depth of these edges. This indicates that for large n the probability that the whole of T_G will be reconstructed must tend to zero, while there is still a reasonable probability that the 'shallower' edges will be correctly inferred.

  This suggests that an appropriate strategy when dealing with large data sets may be to use only a subset of the data set to infer the deeper edges of the tree.

Chapter 7

Discussion

> Once a method proves to be superior to others, we can use it for all data sets.
>
> [*Tateno et al* [93]]

7.1 Introduction

This thesis was not intended as a comparison of the performance of phylogenetic methods. Rather, it was to study the effects of several factors on the performance of some phylogenetic methods.

Comparisons of the performance of methods are abundant in the literature already; for example, see [48], [50], [54], [56], [58], [75], [80], and [93]. These studies have used specific models of evolution to generate the data: the type of transition matrix, the tree topology, the sequence length, and the edge lengths have all been taken from a small set of possibilities, and the performance of the methods assessed. One danger in such studies is of assuming that because in some set of circumstances, method 'A' works more accurately than method 'B', it will be the better method to use in some other circumstances. This may indeed be true, but is not definite, and the results described in this thesis do not support a single 'best' method. I therefore have to disagree with the statement of Tateno *et al* quoted above.

Hence throughout Chapters 5 and 6 ("Small n" and "Large n", respectively) I have avoided making any overall judgement of the superiority of any one method over the others. In differing sets of circumstances, each method has its own merits.

But such an investigation as this is bound to reveal trends in the relative performances of these methods, and of course these trends must be considered with regard to other simulation and theoretical studies which have been carried out. Hence in this chapter an overall view is provided of the results presented so far, and how they relate to the results of some other experiments.

This study has been inspired in part by previous work, because of necessity such work was limited by computational costs and theoretical and empirical understanding. It would be folly to suppose that, just because this study has used relatively large simulations with variation of many parameters, it is in some way conclusive, and that no more need be done. For this reason I include an indication of the direction in which further research into this problem may be headed.

7.2 Summary of results

The investigations I have carried out investigating factors affecting the performance of phylogenetic methods have involved a large amount of computer processing. The results from the two programs "`sim.c`" and "`big.c`" have been summarized, to find those which are the most informative and interesting for inclusion in this thesis. Hence some conclusions have been made with reference to data not shown. This omission of data has been made purely in the interests of saving space and not interrupting the flow of the discussion with countless figures and tables.

The investigation has provided several conclusions about the effects on the performance of phylogenetic methods of sampling error, tree topology, number of taxa, edge length probability distribution, treatment of observed data, white noise and pink noise. It has also enabled the desirable properties (accuracy, consistency, efficiency, falsifiability, and robustness) of the phylogenetic methods used to be tested both empirically and theoretically.

7.2.1 Sampling error

My results support the expected conclusion that sampling error has a strong effect on the accuracy of the phylogenetic methods used: all perform poorly when the sequence length c is less than about 100 characters, and all the methods consistent with the model by which the data were generated perform well when c is large, in the order of 1000 or 2000 characters. This is of course to be expected, but the rate

at which the accuracy changes with varying c is now easily observed, with the large set of sequence lengths used.

With the "large n" case, the same effect of sampling error was seen with respect to the clustering methods NJ, SL, TD and UPGMA.

When the internal edges were very short, there was often no change of state of any characters across that edge (in the "small n" case, this would mean the bipartition corresponding to that edge was not sampled). In such cases there was not enough information in the data set for phylogenetic methods to reconstruct the generating tree T_G, unless the sequences were very long.

When the internal edge lengths, or indeed the pendant edge lengths, grew very large, the sequences effectively grew more and more dissimilar, until once again there was not enough information in the observed data to reconstruct T_G unless the sequences were very long. In the "small n" case, this took place when the upper bound σ on the maximum path length exceeded 1.4. With the "large n" case an experiment was carried out to find the variance of the inferred divergence time between two sequences, given a known rate of evolutionary change. This highlighted the difficulty of inferring edge lengths from sequences in which there is not enough change to be reliable, or when the data become too dissimilar.

By using the Hadamard conjugation technique to calculate the expected bipartition frequencies and effectively eradicate sampling error [41], and by using very long sequences, it was possible to see easily which of the methods is consistent with the model of evolution used to generate the data.

It was also possible to prove theoretically that the neighbour-joining method (NJ) uses the only possible weighting scheme for net divergence which is consistent with the model (see Appendix B, Theorem 5).

7.2.2 Tree topology

It has been found that, with the models used, tree topology has little effect on the performance of the tree reconstruction methods used, other than UPGMA. The other methods which are the most affected are NJa, SL, UPGMA, and the sequence-spectrum based methods. Those least affected are the distance-spectrum based methods (CoDH, CTDH, MPDH), NJ, ST and TD. I have shown the accuracy of UPGMA to be strongly dependent on tree topology. This means simulation studies on UPGMA need to explicitly take tree topology into account.

Knowing that, apart from UPGMA, tree topology for small n makes little difference to the performance of these phylogenetic methods, it is then reasonable to use one tree topology for the generating tree T_G, or choose the topology of T_G at random, in simulation studies such as this. Therefore with the "large n" case T_G could be selected at random, without unnecessary concern about the affect of the topology of T_G on the accuracy of NJ, SL, ST or TD, but bearing in mind that tree topology may have a large effect on the accuracy of UPGMA in this case also. This allowed the extension of the simulation experiments to 30 taxa.

7.2.3 Number of taxa

The number of taxa (n) greatly affects performance of phylogenetic methods. The number of unrooted binary trees with n pendant vertices is $(2n - 5)!!$, and that of rooted binary trees is $(2n - 3)!!$: these numbers grow exponentially with n, so with increasing n, we predict that the accuracy of the methods must decrease.

Also, as the maximum distance between any two sequences in a given trial was bounded above (usually by 0.25 — see Section 4.4.1), when n was increased, the edge lengths decreased. Hence, with the problems known to be caused by short internal edges, this was another reason to expect the accuracy of the reconstruction methods to decrease markedly with increasing n.

Sections 5.7 and 6.3 discuss this problem, and the degree to which n affects the mean number of edges wrongly inferred and the probability of reconstructing T_G.

It was possible to estimate the rate at which the sequence length must increase (which is related to the rate at which sampling error must decrease) with increasing number of taxa, for a particular phylogenetic method to retain a given confidence in the inferred tree. This rate has been shown to be close to linear for the cases studied, which is a promising result, but it still suggests that for 100 or more taxa, proportionally long sequences must be obtained (in the order of 10^4 characters), which may be infeasible.

The model used for data generation was 'ideal', in the sense that it was known, and the methods used for tree reconstruction were (in this experiment) consistent with that model. 'Real' data cannot be assumed to be so well behaved, and it may well be that with real DNA sequences the required growth rate in c is much faster than a linear function of n, if a fixed confidence level is sought.

Adding taxa to increase the confidence in a particular internal edge was not

supported by these experiments: the mean number of edges wrongly inferred by each method increased at least linearly with n, indicating that the probability of wrongly inferring a particular edge is at least increasing. This leaves aside the position of the edge in the tree: edges which are 'deeper' than others (see Section 6.7) may have a lower probability of being inferred than those which are 'shallower'.

A study of the above effect would be rather involved, and complicated by the relative lengths of the internal edges and the position of the edge(s) in question: this may be a fruitful avenue for further study.

7.2.4 Edge length probability distribution

The edge length probability distribution experiments highlighted once again the influence of sampling error: with short internal edges, there was frequently not enough information in the data to recover the generating tree, while with very long edges (whether internal or pendant) there was so much character state change between sequences that the relevant information was hidden.

In fact all the experiments in which the lengths of the internal edges were varied revealed the dependence of accuracy on edges which are neither 'too long' nor 'too short'. Section 6.4 highlighted this problem explicitly with Figure 6.5, which demonstrated the limits to the amount of information which can be gleaned from sequence data: 'too short' was in this experiment less than about 0.1 expected change of state per character, per site, between sequences, and 'too long' was more than about 1.0 expected changes of character state.

7.2.5 Treatment of observed data

It is known that no tree selection criterion using the observed data directly, without some compensation for multiple changes, can be consistent [88]. Hence it was expected that CoSO, CTSO and MPSO would behave relatively poorly with long sequences and when multiple character state changes were likely. This was indeed the case, as described in Sections 5.6 and 6.4.

However, the result that using the distance spectrum $\mathbf{g}'$ for Co, CT and MP would actually *improve* their performance over using the sequence spectrum $\mathbf{s}'$ was unexpected. This led to the independent discovery in my experiments and by Waddell *et al* [97] that edge lengths inferred from $\mathbf{g}'$ had a smaller variance than

those inferred from s′, and hence were often more reliable in this study. Further investigation into this area is certainly warranted by these new results.

The order in which data are presented to the clustering algorithms can affect their performance, as was discovered in early simulations when the same labels were used for the pendant vertices of a given T_G for all trials. It was decided to permute the labels of the pendant vertices for each trial, to remove this effect.

7.2.6 Relationship between phylogenetic methods

Section 5.3 showed some of the relationships between the phylogenetic methods used in this study. Some of these relationships are in the treatment of the input data as mentioned above, and some were theoretical.

The close relationships between CTDH and NJ, and between NJ and ST, were first seen experimentally, and then investigated mathematically: this is one of the great advantages of computer simulation, that answers can be obtained to questions not even asked [11], and can give indications of the directions in which theroetical study can be fruitful.

The close agreement between CTDH and NJ continued for many of the experiments, and led to the proof that CTDH and NJ are equivalent when $n = 4$ (see Appendix B, Theorem 6).

In attempting to improve the performance of ST, to be more sensitive, it was discovered that the modified method ST+ (see Section 2.4.3) was identical to NJ (up to rounding error). This led to the proof that ST+ was mathematically equivalent to NJ (see Appendix B, Theorem 7).

Other close agreements were between the performances of CoDH and CTDH, CoDH and ST, CoSH and CTSH, CTDH and MPDH, CTDH and ST, MPSH and finally NJ. This should not be overemphasized though, as the phylogenetic methods were of course estimating the generating tree T_G, and when accurately determining T_G they will naturally agree with each other.

7.2.7 White noise

With the introduction of white noise, in the form of random sequencing error, the robustness of the phylogenetic methods was investigated. It appears that while using the distance spectrum g′ from which to infer edge lengths of T_G is preferable in

the earlier experiments, these inferred edge lengths are less robust to sequencing error than are edge lengths inferred from the sequence spectrum s′. The constructive methods were also shown not to be robust to this kind of error.

If the sequencing error rate is below about 2.5%, the relative accuracies of the methods is not affected, though the accuracies of all the methods decrease with increasing error rate.

7.2.8 Pink noise

Pink noise, which is biassed random error, was introduced into the data by amalgamating observed bipartition frequencies from two independent generating trees.

Several tests were carried out, for sequence lengths ranging from 100 to 2000 characters, but the effect of this amalgamation of data was most obvious with the longer sequences.

With large amounts of 'contaminant' data, i.e., from 10% to 45%, it was discovered that CoSH and CTSH are much more accurate than the other methods. This *may* provide some falsifiability for these methods: If the confidence in the inferred tree can be estimated, perhaps by the bootstrap resampling technique, the relatively high accuracy of CoDH and CoSH could indicate that the data come from two or more independent phylogenies.

This may also be useful in determining whether two data sets, known to be from different sources, come from independent generating phylogenies. Further investigation into this question is warranted.

7.3 Comparison with some other studies

This section discusses some of the many previous simulation studies referred to above, in connection with the investigations in this thesis. It is not possible to cover all such studies, and a good overall impression of the current work being carried out in this area can be obtained from a selection of these studies.

All the experiments carried out by other researchers and discussed here have 'grown' the data, rather than sampled it (see Chapter 4). This is not, in the case of two character states, effectively different from using the sampling process as described in Chapter 4. However, in the case of four character states growing the data does allow more general transition matrices to be used. This was the case in

most previous studies. In general, different models of evolution were used, with few tree topologies and sets of edge lengths. Often, the method of judging the performance of the methods was based on the frequency at which T_G was inferred, whereas in my study the accuracy was based on the mean number of edges wrongly inferred. Many of the implementations of MPSO — the only of MPDH, MPSH and MPSO previously studied — were in earlier experiments not exhaustive searches, but were restricted to trees "close to" T_G. For all these reasons, the results obtained here are not readily comparable with those obtained elsewhere, though they do require consideration as part of the whole area of research.

The following discusses four previous studies which are similar in approach to this one, and their relationships to this thesis.

In 1981 Li [54] described his method for dealing with unequal rates of molecular evolution on different lineages of a generating tree. In fact, he described two methods, which he labelled versions 1 and 2 (version 1 shall here be labelled Li1, and version 2 is the transformed distance method, TD). In his paper he showed that TD performed better than Li1, and it is TD that I have used in this study. He used two rooted tree topologies for his T_G's: RB3 and RB7, on 4 and 8 pendant vertices (i.e., taxa) respectively. His study compared several other methods also: the Modified Farris method (MF), Farris' method (Fr), Fitch-Margoliash's method (F-M) and UPGMA. The study was limited to 20 trials on each run, with sequence length $c = 300$.

Li found that with the model used, the accuracy of MF was the highest, followed by TD, then Li1 (though these two behaved very similarly), then Fr, F-M and UPGMA, in decreasing order of accuracy.

Sourdis and Nei's 1988 study [80] was more extensive: it compared MPSO, NJ, the Distance Wagner method (DW), MF, Faith's method (Ft) and TD. The number of trials was increased to 300, and c was 300, 600 or 1200. Four character states were used. The tree topologies used for T_G were RB5 and RB3, 2_1 on 6 pendant vertices; RB7 and RB3, $(2, 1_1)_1, 1_2$ on 8 pendant vertices (RB3, $(2, 1_1)_1, 1_2$ is the rooted "star tree" on 8 pendant vertices). The study concluded that when the evolutionary rates are low and c is small, MPSO is not as good as the other methods. With larger c, MPSO did improve relatively, but was still not as accurate as DW and NJ. This result was independent of constancy of substitution rate, and any transition rate bias.

This result may seem to conflict with the general behaviour exhibited in this thesis by MP, which is that MP is usually the most accurate method. Of course the models used are different: Sourdis and Nei used a more general model of molecular evolution, in that they took four character states and allowed different rates of character state change, depending on the initial state. Also, their study used rooted trees for T_G (though the inferred trees were unrooted), whereas I have used unrooted trees. Another important reason may well be in the method used to find most parsimonious trees.

I have glibly been speaking of "the maximum parsimony tree", as in my simulations I have used a branch and bound method to effectively search all possible trees. But Sourdis and Nei, to save computing time, used a *local* search to find the "maximum parsimony" trees: with statistical justification, they only considered trees which were at most one edge different from T_G, so could not guarantee that the tree found in this way would be the *most* parsimonious. Hence this type of search gives a slight underestimate of the accuracy of MPSO. (Note that the assessment of the accuracy of methods was, in the above study, based on the probability of inferring T_G, rather than the mean number of edges wrongly inferred.)

They also noted that when the comparing the *set*, say S, of most parsimonious trees with the trees inferred by other methods, MPSO performed much better, in that it had a high probability that S contained T_G.

My simulations have not kept track of this set S, an omission made consciously to reduce computing time and memory requirements, so the above conclusion is not directly comparable with my results. The MP implementation used here found a single minimal-length tree for each data set.

Another example of this misleading use of the term "maximum parsimony tree" is in Czelusniak *et al* [16], where such a tree is proposed for a set of 116 taxa. For such a large data set it is impossible to test all trees; therefore a more appropriate term might be "highly parsimonious".

In 1991 Jin and Nei [50] investigated the relative accuracies (though the term used was "efficiencies" — see Section 2.2.3) of NJ, MPSO and UPGMA, with restriction site data. The generating tree T_G had topology RB3, 2_1, being the rooted "star tree" on 6 pendant vertices. The sequence lengths used were 500, 1000 and 16000, and 300 trials were carried out for each. Once again four character states were used.

They found that when the substitution rate was constant (i.e., obeying the molecular clock hypothesis), the probability that NJ correctly inferred T_G was greater than that of MPSO "generally", but the average topological distance [72] between the inferred trees and T_G was about equal for these two methods. When the evolutionary rates varied with lineage, NJ was more accurate than MPSO, whichever measure was used.

The above conclusions do not necessarily conflict with those made in this thesis: as with the study of Li [54] and Sourdis and Nei [80], the model used had four character states, and the data were generated (for the most part) according to the molecular clock hypothesis. Also the comparison between NJ and MPSO was only made, in this thesis, with *unequal* rates of evolution (the "Small n" case), so no conclusions are available as to the relative performance of NJ and MPSO when the data satisfy the molecular clock hypothesis. However, the higher accuracy of NJ with respect to MPSO does invite further investigation.

They also found that UPGMA only performed well when T_G satisfied the molecular clock hypothesis and when the edge lengths were large. This is a similar conclusion to that which has been reached in this thesis: UPGMA does not perform well with unequal rates of evolution, and no methods perform well with small edge lengths.

Nei, in 1991 [58], conducted an extensive comparison of DW, NJ, MF, MPSO, ST, TD and UPGMA, and also Evolutionary Parsimony (EP) and Minimum Evolution (ME). The generating trees used had the same topologies as Sourdis and Nei's 1988 study, discussed above. The sequence length c took several different values, ranging from 300 to 2000 characters.

With T_G satisfying a molecular clock, he found, as I have, that NJ and ST performed very similarly, the reason for which I believe is the close relationship these two methods have (see Section 2.4.3, and Appendix B, Theorem 7). NJ and ST were found to be more accurate than DW, TD, MF and UPGMA, for several different values of overall time (see Chapter 6).

MPSO was found to be slightly less accurate than DW, TD, NJ and ST, though with low overall time and high c, MPSO was slightly more accurate than these.

With varying rates of evolution (i.e., not satisfying a molecular clock) Nei found that UPGMA performed the least accurately of all the methods. He also found that NJ, ST and ME were more accurate than the other distance methods (DW,

TD and MF). With varying rates the same sorts of conclusions were reached as for Jin and Nei's 1991 study.

Nei made several useful recommendations, regarding phylogenetic inference with real data and simulation studies. With regard to the latter, he made the point that further simulation work should be done, and made particular reference to the need for simulations with large n.

7.4 That which may be

This section describes some possible directions in which research in this area could continue. These directions are (1) toward a greater statistical understanding of the sources of error in finite data sets, (2) investigation of the properties of the selection criteria of phylogenetic methods, and (3) toward improved strategies for the search of potential trees (the "tree space").

7.4.1 Sources of error in finite data sets

As was discussed in Section 6.4, the true amount of evolutionary change since the divergence of two sequences has a great effect on the inferred amount of evolutionary change between the sequences. Note that the inference of evolutionary change between two sequences proceeds in two steps: the first is the calculation of the number of observed differences (or some other measure of distance between the sequences), and the second is the adjustment for multiple changes of character state on the lineages separating the two sequences, to estimate the number of character state changes which took place.

When the amount of evolutionary change is small, the distance between the two sequences is highly susceptible to sampling error, as the expected number of differences between the sequences is low. Though the correction for multiple changes of character state has very little effect when the number of observed differences is low, the sampling error just mentioned means that the inference of evolutionary change between these two sequences is made unreliable.

With high evolutionary change, the distance between the two sequences is less susceptible to sampling error, as the expected number of differences is larger. However the correction for multiple changes of character state is in this case very sensitive to any sampling error in the observed differences, once again leading to

unreliablility in the estimate amount of evolutionary change between the sequences.

This concept can be visualised as shown in Figure 7.1.

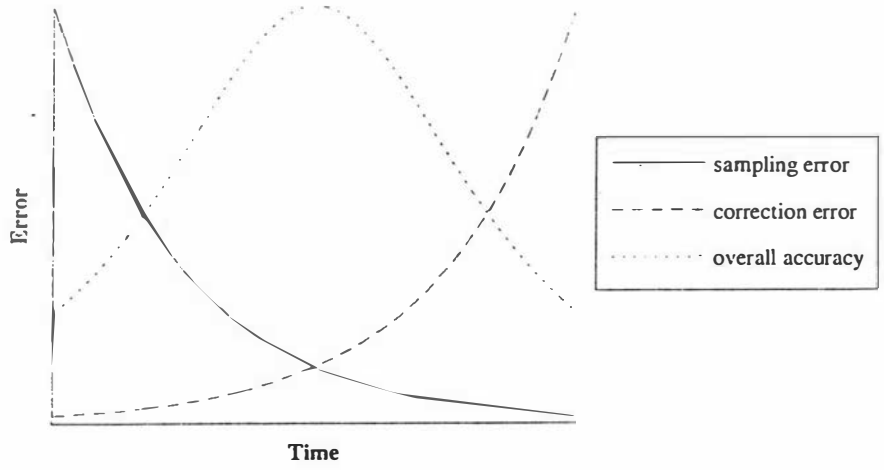


Figure 7.1: Possible visualisation of the sources of error in inference of amount of evolutionary change

The figure above shows the two sources of sampling error which can arise in the inference of amount of evolutionary change, in terms of number of character state changes, between two character sequences. This is plotted against the expected number of character state changes. Also shown is an indication of how the accuracy of inferred number of character state changes might behave.

The problem of determining how much information can be obtained from a given finite data set comprising two or more character sequences is clearly of importance. Simulation studies and theoretical analysis with two or more taxa and more complex models of evolutionary change should throw some light on this area, and lead to better estimates of the reliability of this aspect of phylogenetic inference.

7.4.2 Properties of selection criteria

The performance of the compatibility (Co), closest tree (CT) and maximum parsimony (MP) methods has been discussed at length in this thesis, but little has been made of the fundamental properties of the tree selection criteria used by these methods, and in what way selection criteria affect performance.

Consider the way in which each tree selection method assigns a 'score' to inferred edge lengths. These edge lengths can be simply the observed relative frequencies of bipartitions of the taxon set into different character states, as with the 'Sequence Observed' variants, or could be inferred with some adjustment made for multiple character state changes, from either distances or sequences (the 'Distance Hadamard' and the 'Sequence Hadamard' variants, respectively). Once these edge lengths have been estimated, potential trees, consisting of sets of compatible edges, are assessed for the optimality criteria of the methods.

As in Section 2.5 let the function to be optimized be $M(T')$, for each potential tree T'. With Co, $M(T')$ is just the sum of the inferred edge lengths $\mathbf{q}$, while CT maximises a quadratic function of the $\mathbf{q}$ values. Hence, Co can be considered to penalise each edge *not* included in T', in proportion to its inferred length, and CT can be considered to penalise each edge not in T' according to the *square* of its inferred length. This penalty assigned by CT to edges not in T' *is dependent on the edges in* T'.

MP methods, on the other hand, penalise each edge excluded from T' by the number of additional character state changes that must occur to account for the observed or inferred frequency of bipartitions corresponding to that edge. This number can vary from 1 up to $\lfloor n/2 \rfloor$ with 2 character states and n taxa ("$\lfloor n/2 \rfloor$" means the integral part of $n/2$), and is *also* dependent on the other edges in the tree.

We therefore might suppose that since the penalty for non-inclusion of an edge in a potential tree T' is dependent on those edges in T' for CT and for MP that thse two tree selection methods would be more accurate than Co, but this has not always been the case.

To understand the reasons these methods perform as they do, we must therefore seek deeper understanding of the theoretical properties of the tree selection criteria, and this is a promising avenue for future research. These theoretical properties are often counter-intuitive: some surprising properties of the maximum parsimony method are discussed in [81].

7.4.3 Search strategies

The problem remains of inference of very large generating trees, with in the order of hundreds of taxa. While the advisability of attempting reconstruction of such

large trees is questionable, the performance of clustering methods with such data sets is not well understood. This can be investigated with computer simulation.

As the number of taxa increases the number of replications must be limited (the generation of distance data must require $O(n^2c)$ operations), but with massively parallel supercomputers available, this generation could take only $O(nc)$ time with a sufficient number of processors.

Future work will almost certainly be directed towards such large simulations.

Unfortunately, this leaves aside the search methods. The exponential number of potential trees means that with large n exhaustive search is impossible, and heuristic search strategies must be used.

One method which has been used with some success is the Great Deluge algorithm [18], [68], which was able to find several significantly shorter parsimony trees on a subset of the "Out of Africa" mitochondrial DNA data set [38], [96], than were originally found using the program PAUP [90]. The best parsimony tree found by the Great Deluge method was 20 steps shorter than the best found previously.

The ability of this method to search the tree-space for parsimonious trees should be further investigated, and also for the other search methods (CT, Co). This too can be accomplished using massively parallel computers, by running several solution trees at once.

A new strategy for conducting heuristic searches of tree space [68] I am calling "Hitch-hiking". The basic premise under which Hitch-hiking works is that good solutions (for example, highly parsimonious trees) have common characteristics.

Under Hitch-hiking there are a number of *driver* solutions, with each of which is associated a number of *hitcher* solutions, which are its *passenger solutions.* At each search step, each driver solution will be perturbed, according to some other search strategy, for example Great Deluge [68], Hill-Climbing, Tabu Search [32], Simulated Annealing [95]. Each passenger solution for which this perturbation is possible is also perturbed in the same way.

Properties for perturbation of the solution trees must be common to as many trees as possible for Hitch-hiking to work. Thus a particular non-trivial bipartition of the taxon set (i.e., an internal edge of the tree) is not a good choice for alteration, since the probability that two randomly chosen trees have one or more internal edges in common approaches zero with increasing n and hence in almost all cases the perturbation of the driver solution would not be possible for the passenger

solutions.

One property might be to consider the paths between three taxa: every three pendant vertices, say i, j, k, have a point z at which the three paths between them ($\pi_{i,j}, \pi_{i,k}$ and $\pi_{j,k}$) intersect. Changing the point at which the path $\pi_{i,z}$ intersects $\pi_{j,k}$ is a change which can be carried out for all trees (see Figure 7.2, below).

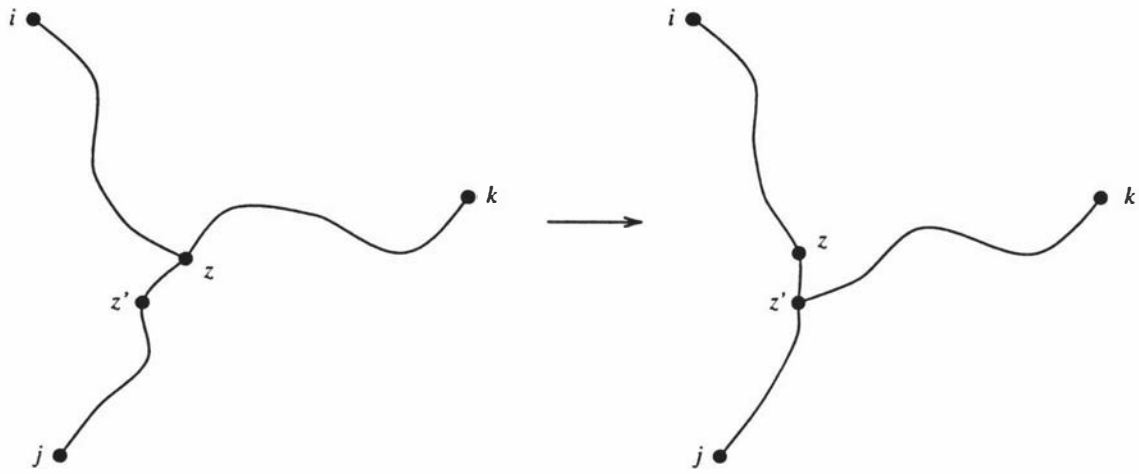


Figure 7.2: Possible move in the Hitch-hiking heuristic search strategy
In this figure is shown a hypothetical move which could be undertaken by a driver or passenger solution in a Hitch-hiking search. In the left-hand graph is shown only the pendant vertices i, j, k, the paths $\pi_{i,j}, \pi_{i,k}$ and $\pi_{j,k}$, and the point z at which these paths intersect. The path $\pi_{i,z}$ is then moved along $\pi_{j,k}$, to now intersect it at vertex z'. This is shown in the right-hand graph.

Hitch-hiking is a quite general strategy, and many details can be varied, for example the ratio of drivers to hitchers, the capacity of each driver (i.e., the number of passengers it can take), and the method of choosing the driver solutions and of assignment of their passenger solutions.

Hitch-hiking has not as yet been tested, but it is ideally suited to parallel computers, as each solution can be stored and processed on one processor.

I believe that this sort of investigation will be a fruitful direction for further investigation.

Algorithm 7.1 : Example of Hitch-hiking

{ This description is using the Great Deluge method with maximum parsimony }

Choose running parameters:

n	{ number of taxa }
s	{ number of solutions, $\leq S$ }
d	{ number of drivers, $\leq D$ }
c	{ capacity of each driver solution, $\leq C$ }
w	{ increment in waterlevel, ≈ 0.01 }

Generate data set

Generate root sequence

Grow other sequences

Pick s random start solutions $tree_x$

Find parsimony lengths of these $tree_x$'s with Fitch's algorithm

Fix parsimony bounds b_i for each candidate solution x (these are upper bounds)

Repeat

Choose d driver solutions e.g.,

from the top d current solutions,

from the bottom d current solutions, or

at random from the whole set of solutions, etc. (depending on implementation)

For each driver solution x

Choose three pendant vertices i, j, k from $tree_x$

Find the point z at the intersection of $\pi_{i,j}$, $\pi_{i,k}$ and $\pi_{j,k}$

Repeat

Remove the subtree rooted at z containing i but not j or k

Choose r from a random probability distribution with spread parameter v and mean m

Insert internal vertex z', r places along the path $\pi_{j,k}$ from z

Add the subtree removed above to vertex z', to get $tree'_x$

Find parsimony length l'_x of $tree'_x$

until $(l'_x < b_x)$ or (can't improve after several tries)

if $(l'_x < b_x)$ **then**

$tree_x \leftarrow tree'_x$

For each passenger solution p associated with driver solution x

Perform the same perturbation as above if the parsimony length l'_p of the resultant tree is shorter than b_p

$b_p \leftarrow b_p - (b_p - l'_p) \times w$

$l_p \leftarrow l'_p$

else if $(v < v_{max})$ **then**

increase the probability distribution spread parameter v to give larger possible moves and get out of local optimum

abandon current iteration

if $(v = v_{max})$ **then break**

end.

Appendix A

Tree Topology Description Notation

[*UB40*]

A notation is described here which can be used to characterise the distinct topologies of trees. The notation for a tree (called its 'TTDN' for brevity) can be applied to any tree, but it is most succinct when binary trees only are used. The notation follows the spirit of the IUPAC naming convention for the naming of organic molecules [2].

A notation has been proposed by Harding [36], but I believe my notation to be more intuitive.

Note that the *topology* of a given tree is used throughout this thesis to indicate the underlying structure of the tree, that is, *with no edge or vertex labels.* Thus, the three unrooted binary trees on four pendant vertices all have the same topology (see Figure A.1).

The mapping from tree topology to description is *onto*, but not *one-to-one*, i.e., there may be more than one notation to describe the topology of a given tree, but a given notation describes just one topology.

Rooted Binary trees

To characterise the topology of a rooted binary tree the following rules are used:

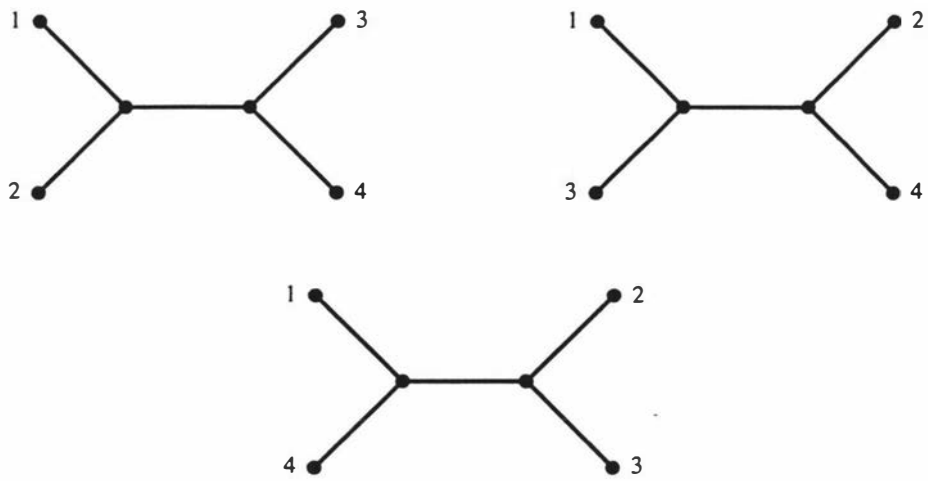


Figure A.1: The three binary unrooted trees on 4 pendant vertices.

1. The first letters in the notation tell us what general type the tree is: we write 'UB' for 'Unrooted Binary', and 'RB' for 'Rooted Binary'. If the tree is non-binary we write 'UN' and 'RN' to indicate unrooted non-binary and rooted non-binary trees, respectively.

2. Except for the root, which we label 1, delete all the pendant vertices and edges, leaving a *skeleton* of the original tree.

3. In this skeleton, find a path π of maximal length which includes the root, and count the number of vertices this path contains; this is the first number in the notation of the topology. The choice of π may be arbitrary if there is more than one path of maximal length which includes the root, but in general π is chosen to minimise the number of subtrees which branch off it. This serves to simplify the notation as much as possible.

4. Proceeding from the root towards the other end of π, we treat each subtree in turn, and determine its notation by considering it to be a rooted tree, rooted at the vertex *adjacent to* π.

5. The notation for a single path, with no subtrees branching from it, is simply the number of vertices which are contained in the path.

6. The position of each subtree branching off a particular path is given by the subscript which follows the notation describing that subtree.

For example, consider the binary tree shown in Figure A.2, rooted at R. In this tree we can take as our maximal path the one along the ten vertices in the horizontal path.

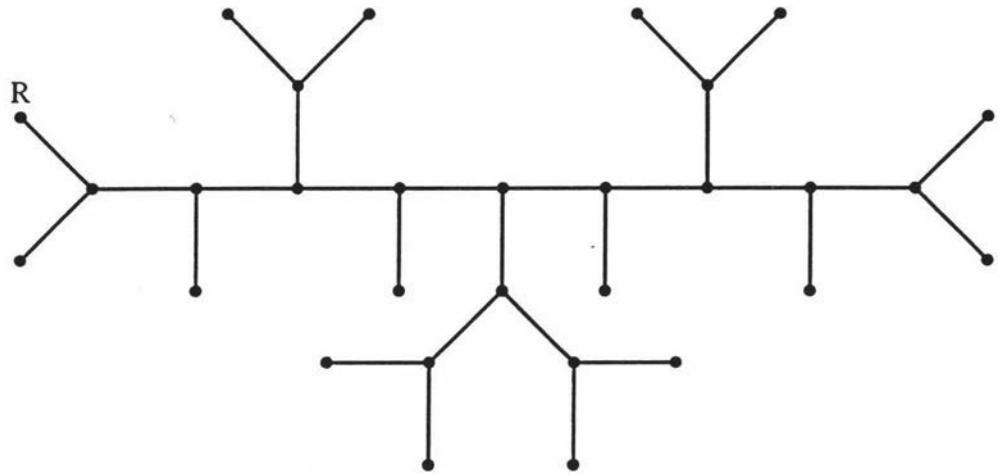


Figure A.2: Example tree T for which the TTDN is sought

We form the skeleton by deleting pendant vertices and edges, other than vertex R and the edge incident on it (Figure A.3).

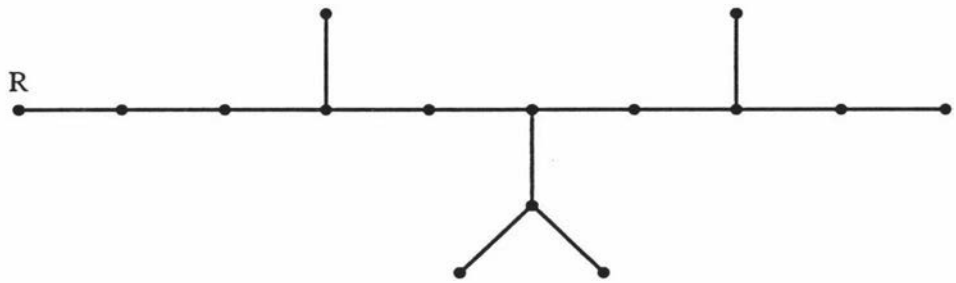


Figure A.3: Skeleton of tree T in the previous figure.

The next term in the notation is 10, as there are 10 vertices remaining in π. Starting at the root, we find the first subtree branching off π has only one vertex and is adjacent to the fourth vertex of π. Hence the next term in the notation is 1_4. The next term is found by first finding the notation for the subtree branching off π at vertex 6. It has a path π' of length 2, and another vertex branching off π' at position 1. The description of this subtree is then $2, 1_1$, and we write $(2, 1_1)_6$ to indicate that it branches off π at position 6. The last term in the notation is 1_8. Hence for this topology the TTDN characterisation is RB$10, 1_4, (2, 1_1)_6, 1_8$.

Unrooted Binary trees

For unrooted binary trees, we first delete *all* the pendant vertices and the edges incident on them. We then choose a path π of maximal length, if possible to minimise the number of subtrees branching from it. The first part of the notation is then 'UB' followed by the number of vertices in π.

We choose one end of π, to minimise the sum of the indices of the subtrees branching off π (once again, to simplify as much as possible the characterisation of the tree). We then proceed in the same way as we did with the rooted tree.

Reconstructing a tree from its TTDN

To construct an unlabelled tree from its TTDN, we use the reverse procedure. Consider a rooted tree first, as before.

Suppose we are presented with the following TTDN characterisation:

$$\mathrm{RB}k, (X_2)_2, (X_3)_3, \ldots, (X_{k-1})_{k-1},$$

where each $(X_i)_i$ may or may not be present.

We first construct the path π with k vertices, and beginning at one end, label the vertices $1, 2, \ldots, k$. We determine the rooted topologies described by each of $X_2, X_3, \ldots, X_{k-1}$ in turn, and join them to π at vertices $2, 3, \ldots, k-1$ respectively. Then to each vertex of degree 2 we attach another edge and pendant vertex, and to the others we attach two edges and pendant vertices.

For example, suppose we have the notation $\mathrm{RB}7, 1_2, (2, 1_1)_3, 2_4$. We first construct the path π with seven vertices, one end vertex of which is the root (Figure A.4). Adding subtrees in turn we get the tree shown in Figure A.5. Finally, by adding pendant vertices to this skeleton we get the complete rooted binary tree shown in Figure A.6.

Figure A.4: The first stage in reconstructing a tree from its TTDN.
In this figure only the initial maximal path π is shown.

For unrooted trees we proceed as above, but in the last stage we add new pendant vertices to every vertex of degree one or two.

We note some useful properties of this notation. For an unrooted binary tree T whose TTDN description is $\mathrm{UB}k, (X_2)_2, (X_3)_3, \ldots, (X_{k-1})_{k-1}$, the diameter $d(T)$

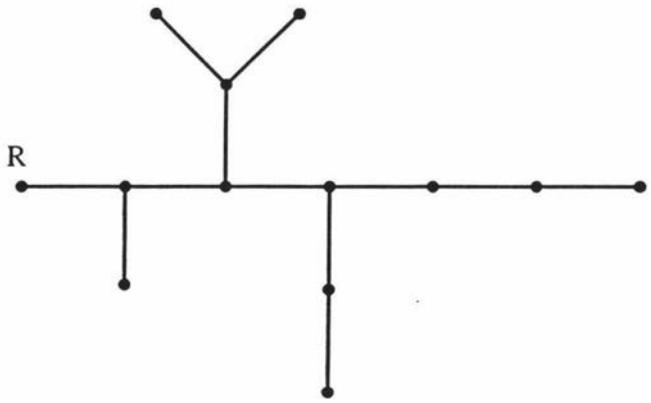


Figure A.5: The second stage in reconstructing a tree from its TTDN.
In this figure the subtrees branching off the initial maximal path π have been added, to give the skeleton shown.

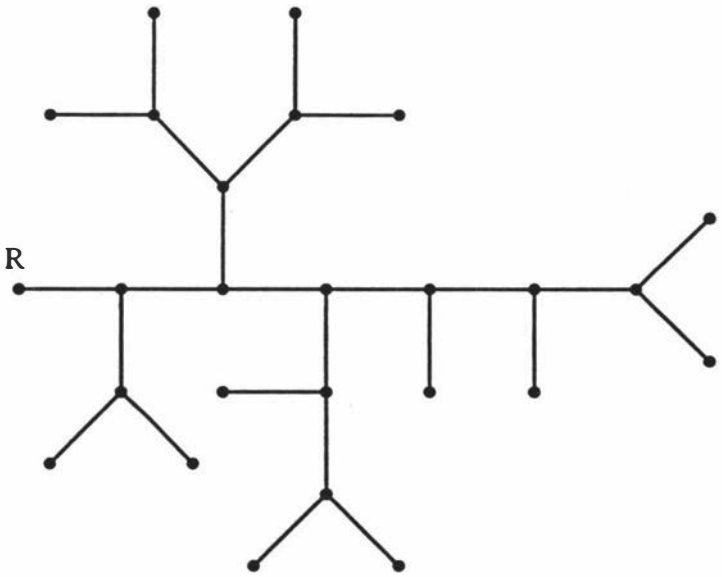


Figure A.6: The third stage in reconstructing a tree from its TTDN.
In this figure the pendant edges and vertices have been added to the skeleton in Figure A.5 above, to give the complete tree.

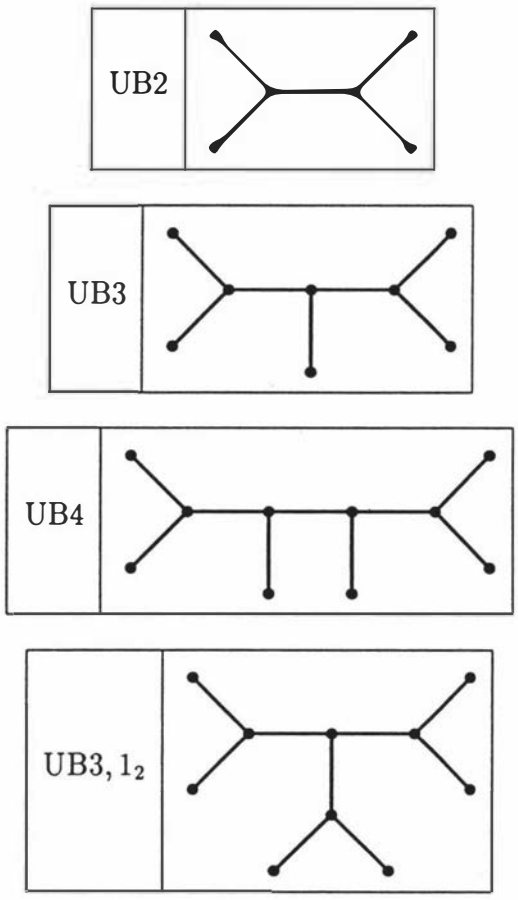
of T is $k + 1$. Also, the number, say a, of internal vertices of T is the sum of the in-subscripted numbers, so the number of pendant vertices is $a + 2$.

Non-binary trees

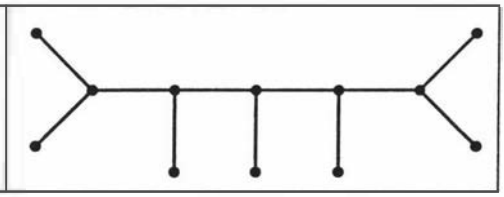
The notation can be generalised to include non-binary trees, but by doing so much of its conciseness is lost: since in binary trees all vertices have either degree 1 or 3 (apart from the root, which may have degree 2), it is possible to simplify the characterisation of the topology of such a tree by forming its skeleton, as outlined above. However, with non-binary trees we can make no such simplification, and hence when describing the topology of a non-binary tree, rule number 2 is omitted.

List of topologies

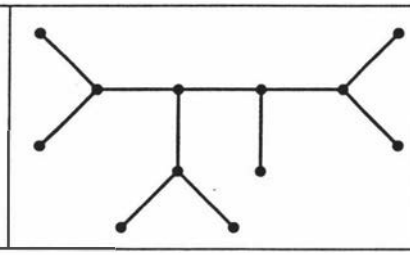
A list of all the tree topologies of unrooted binary trees with from 4 to 10 pendant vertices is provided below.



UB5



UB4, l_2

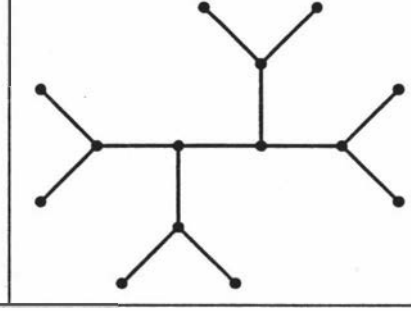


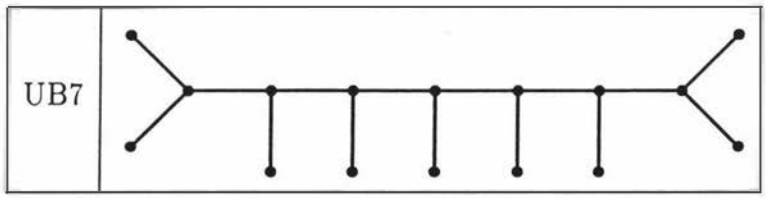
UB6

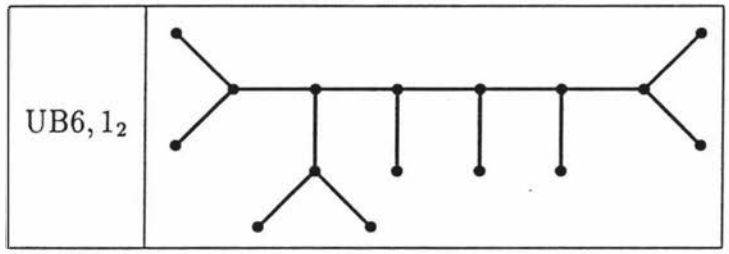
UB5, l_2

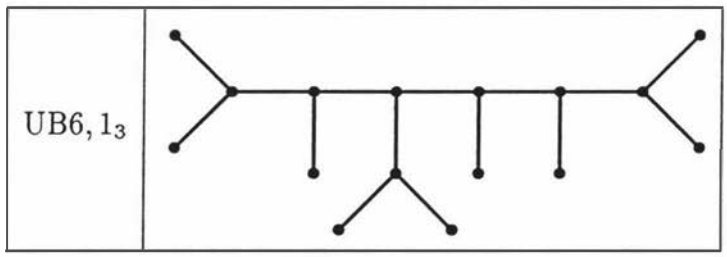
UB5, l_3

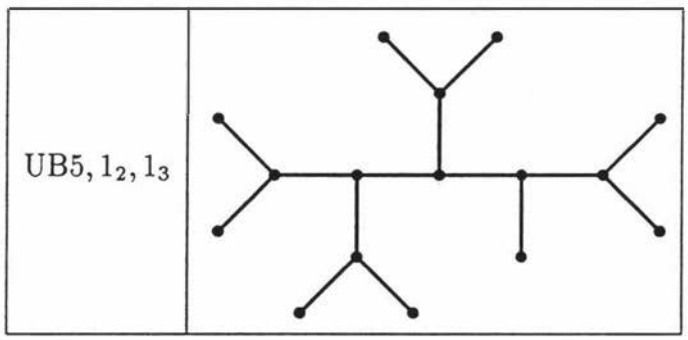
UB4, l_2, l_3

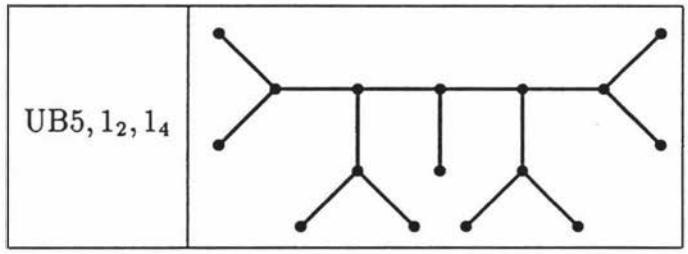


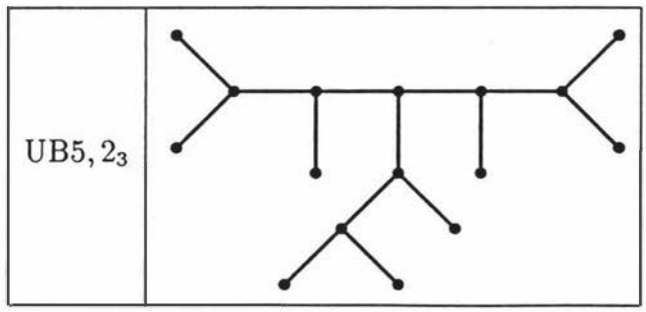

UB7

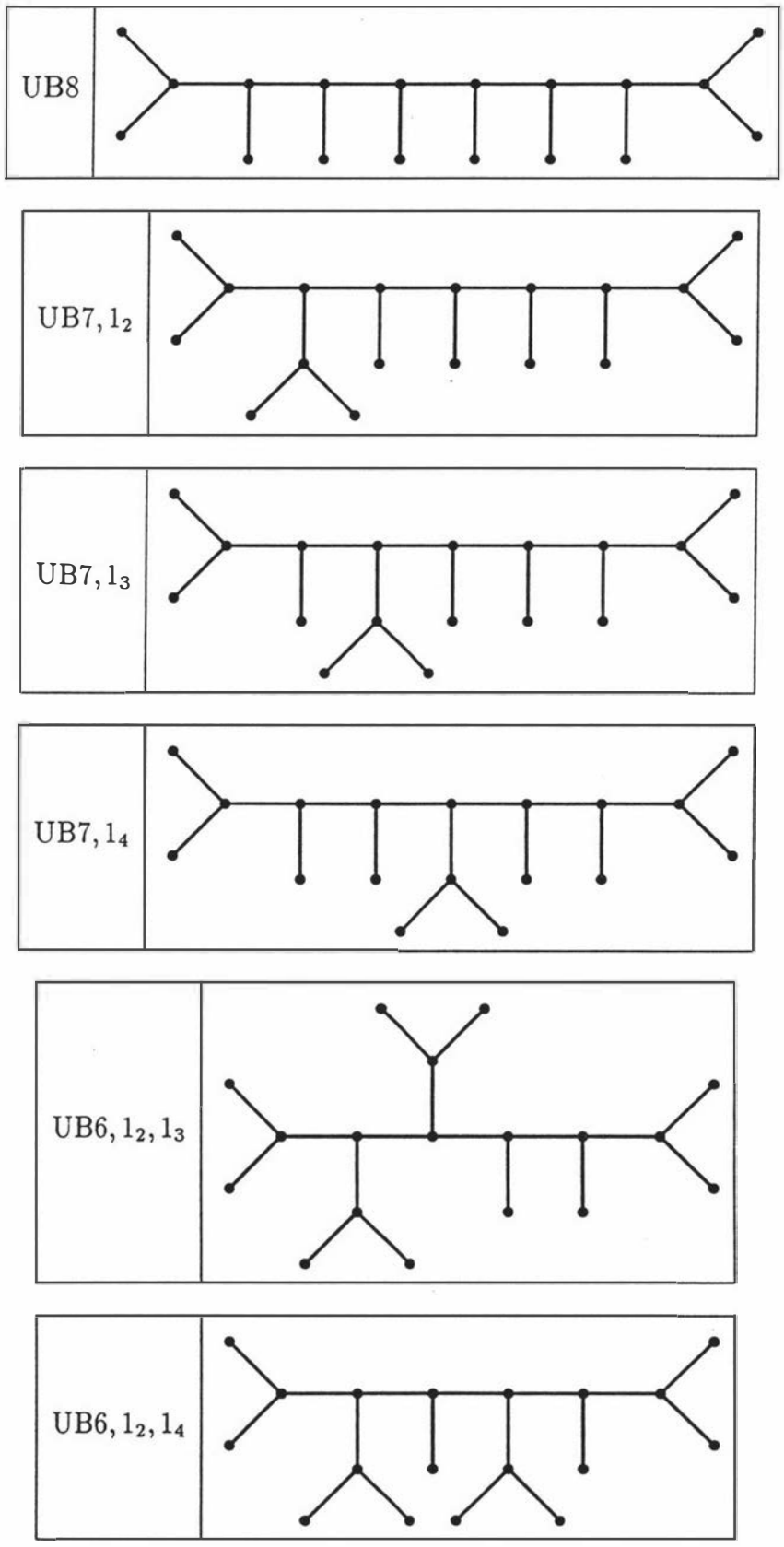

UB6, 1_2

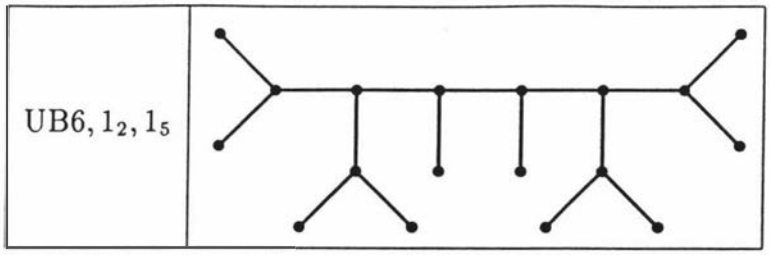

UB6, 1_3

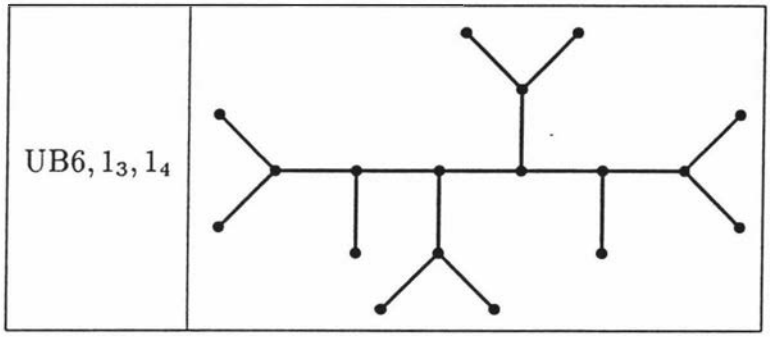

UB5, 1_2, 1_3

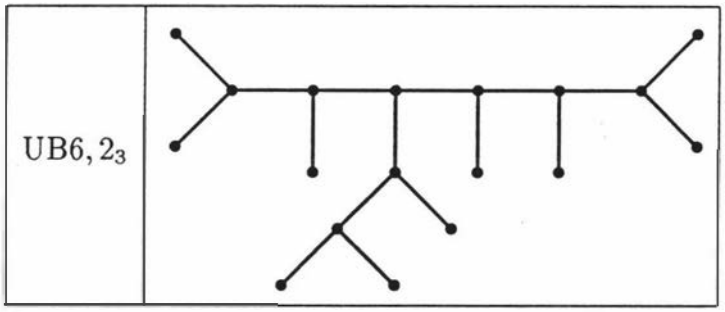

UB5, 1_2, 1_4

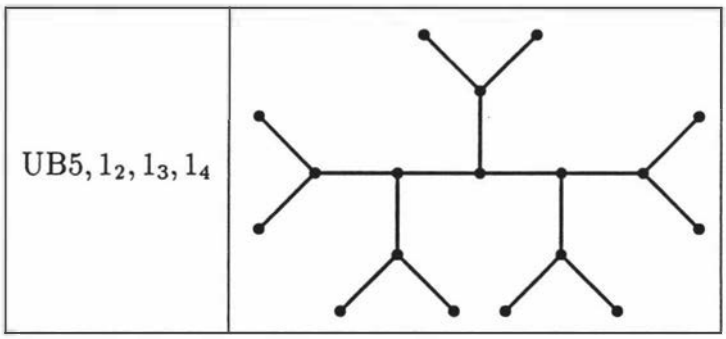

UB5, 2_3

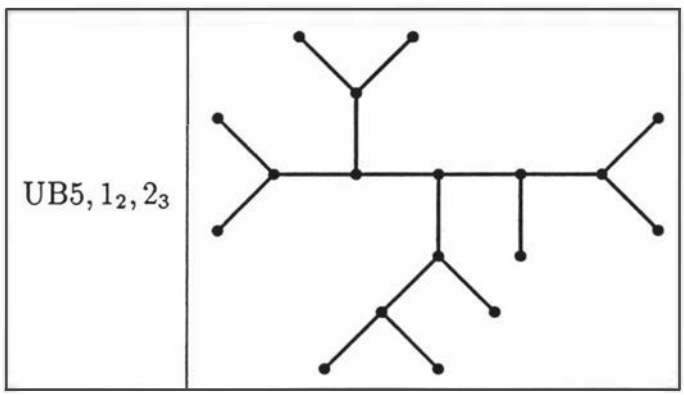

UB8
UB7, l_2
UB7, l_3
UB7, l_4
UB6, l_2, l_3
UB6, l_2, l_4


UB6, $1_2, 1_5$


UB6, $1_3, 1_4$


UB6, 2_3


UB5, $1_2, 1_3, 1_4$


UB5, $1_2, 2_3$

Appendix B

Proofs of theorems

Theorem 1 *UPGMA is consistent with distances which are additive and which satisfy the "molecular clock" model.*

Proof: The theorem is proved by induction on the clustering process. Consider the base case first. Suppose we have a set S of taxa labelled $\{1,\ldots,n\}$, corresponding to the pendant vertices of a given rooted phylogenetic tree and satisfying the above hypotheses. Let the root of the phylogenetic tree on S be r.

Then by the molecular clock hypothesis, $d_{i,r} = d_{j,r}$ for all $i,j \in S$. For each $i,j \in S$, let the nearest common relative of i and j be $ncr(i,j)$. Then $d_{i,j} = d_{i,r} + d_{j,r} - 2d_{ncr(i,j),r}$.

Since $d_{i,r}$ is constant for $i \in S$, this means the minimum value of $d_{i,j}$ occurs for the pair $\{i,j\}$ of taxa which have most recently diverged.

Hence UPGMA, which at the first stage amalgamates taxa i and j if and only if $d_{i,j}$ is minimal, will correctly deduce the first cluster if the distances satisfy the above hypotheses.

Now suppose the clustering process has reached step k, and correctly deduced the first k clusters. If we can show that UPGMA correctly deduces the next pair of clusters to amalgamate, then the proof is complete.

Recall that the distance matrix $\mathbf{D} = [d_{i,j}]$ is replaced by $\mathbf{D}' = [d'_{i,j}]$ at each clustering stage as follows: If clusters x and y are amalgamated into a new cluster, say z, then for each other cluster a,

$$d'_{z,a} = \frac{\alpha_x d_{x,a} + \alpha_y d_{y,a}}{\alpha_x + \alpha_y},$$

where α_i is the number of taxa in cluster i.

Now since the distances satisfy the molecular clock hypothesis, we have

$$d_{x,a} = d_{y,a},$$

so

$$d_{z,a} = d_{x,a} = d_{y,a}.$$

Hence the only difference between the distance matrices $\mathbf{D}$ and $\mathbf{D}'$ is that $\mathbf{D}'$ has one row and one column removed, corresponding either to x or to y.

The minimal $d_{x,y}$ entry in $\mathbf{D}$ having been removed, the next most closely related pair of clusters will have the smallest entry in $\mathbf{D}'$.

Hence UPGMA will correctly determine the next clusters to amalgamate, which completes the proof.

Theorem 2 *TD is inconsistent with additive data*

Proof: The proof is by counterexample.

Consider the tree T shown in Figure B.1, with edge lengths shown in italics. Under the additive distances hypothesis, the distance matrix $\mathbf{D}$ is

$$\mathbf{D} = \begin{bmatrix} 0 & 8 & 14 & 22 & 15 \\ 8 & 0 & 10 & 18 & 15 \\ 14 & 10 & 0 & 14 & 21 \\ 22 & 18 & 14 & 0 & 29 \\ 15 & 15 & 21 & 29 & 0 \end{bmatrix}.$$

We proceed with TD:

Step 1: Put the two most distant taxa, here 4 and 5, into L and R respectively.

Step 2: Find $d_{L,i}$ for all $i \notin L$ and $d_{R,j}$ for all $j \notin R$ (recall that $d_{L,i} = \sum_{j \in L} d_{j,i}/|L|$ and $d_{R,j} = \sum_{j \in R} d_{j,i}/|R|$):

$$\begin{aligned} d_{L,1} &= d_{4,1} = 22; \quad d_{R,1} = d_{5,1} = 15; \\ d_{L,2} &= d_{4,2} = 18; \quad d_{R,2} = d_{5,2} = 15; \\ d_{L,3} &= d_{4,3} = 14; \quad d_{R,3} = d_{5,3} = 21. \end{aligned}$$

Step 3: $d_{L,3}$ is minimal in the above, so $L \longleftarrow \{3,4\}$.

Step 4:

$$\begin{aligned} d_{L,1} &\longleftarrow \tfrac{d_{1,3}+d_{1,4}}{2} = 18; \quad d_{R,1} = 15; \\ d_{L,2} &\longleftarrow \tfrac{d_{2,3}+d_{2,4}}{2} = 14; \quad d_{R,2} = 15. \end{aligned}$$

Step 5: $d_{L,2}$ is minimal in the above, so put $L = \{2,3,4\}$.

Step 6: $d_{L,1} = \frac{d_{1,2}+d_{1,3}+d_{1,4}}{3} = 14\frac{2}{3}; \quad d_{R,1} = 15.$

Step 7: $d_{L,1}$ is minimal in the above, so put $L = \{1,2,3,4\}$.

Step 8:

$$\begin{aligned} r_1 &= d_{1,5} - d_{4,5}; \\ r_2 &= d_{2,5} - d_{4,5}; \\ r_3 &= d_{3,5} - d_{4,5}; \\ r_4 &= 0; \\ r_5 &= 0. \end{aligned}$$

Step 9: This gives the adjusted distance matrix

$$\mathbf{D}' = \left[d'_{i,j}\right] = [d_{i,j} - r_i - r_j] = \begin{matrix} 1: \\ 2: \\ 3: \\ 4: \\ 5: \end{matrix} \begin{bmatrix} 0 & 36 & 36 & 36 & 29 \\ 36 & 0 & 32 & 32 & 29 \\ 36 & 32 & 0 & 22 & 29 \\ 36 & 32 & 22 & 0 & 29 \\ 29 & 29 & 29 & 29 & 0 \end{bmatrix}.$$

Step 10: Now using UPGMA with $\mathbf{D}'$ as its input data, the first pair of taxa to amalgamate to form a new cluster is $\{3,4\}$.

Step 11: The distance matrix is reduced to

$$\mathbf{D}'_1 = \begin{matrix} 1: \\ 2: \\ \{3,4\}: \\ 5: \end{matrix} \begin{bmatrix} 0 & 36 & 36 & 29 \\ 36 & 0 & 32 & 29 \\ 36 & 32 & 0 & 29 \\ 29 & 29 & 29 & 0 \end{bmatrix}.$$

Step 12: There are now three possible pairings to make, $\{1,5\}$, $\{2,5\}$, and $\{\{3,4\},5\}$, all of which have the same entry in $\mathbf{D}'_1$.

Since UPGMA must now choose arbitrarily between three possible pairings, each of which give different final trees, it is not possible to guarantee to pick the correct tree. Hence TD is inconsistent with additive data.

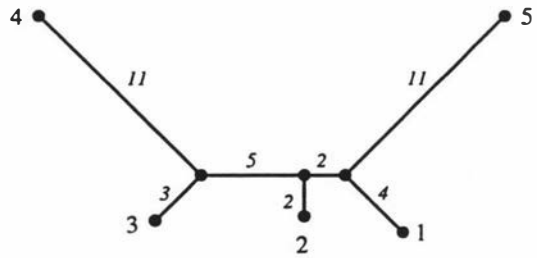


Figure B.1: Tree T used in the proof that TD is inconsistent.

Theorem 3 *Co is consistent with the same models of evolution with which CT is consistent.*

Proof: Suppose we have a data set which was generated according to a model with which CT is consistent. As has been shown by Hendy [45], the edge length spectrum $\mathbf{q}'$ inferred from observed bipartition frequencies will converge to the true edge lengths (in expected numbers of change of character state, for a Poisson process) as sampling error tends to zero.

As the entries of $\mathbf{q}'$ which do not correspond to the true tree (barring q'_0) approach zero in this case, the objective function

$$\sum_{e_i \in T} q'_i$$

is maximised only when T is the generating tree.

Hence Co is consistent.

Note that Co being consistent with a given model does not imply that CT is consistent with the model: we have shown the converse.

Theorem 4 *SL is consistent with additive data.*

Proof: Consider a tree T_G on n pendant vertices labelled $\{1,\ldots,n\}$, with distances $\mathbf{D} = d_{i,j}$ which are additive, i.e., the distance between any two vertices i and j in T_G is equal to the sum of the lengths q_k of the edges e_k which are in the path $\pi_{i,j}$. Suppose also that $q_k > 0\ \forall\ k$ and that there are no vertices of degree 2.

Because UPGMA is known to be consistent with additive distances which, in addition, satisfy the molecular clock hypothesis, it is sufficient to prove that the modification to $\mathbf{D}$ made by SL yields distances which satisfy the molecular clock hypothesis, and do not conflict with the topology of T_G.

The first step in SL finds the two most distant taxa, say x and y, and puts them into sets L and R, respectively. (The order is not important in the implementation, but we will use it for convenience here.)

Let

$$L = \{i \in \{1,\ldots,n\} : d_{i,x} \leq d_{i,y}\}$$

and

$$R = \{i \in \{1,\ldots,n\} : d_{i,x} > d_{i,y}\}.$$

(This automatically includes x and y in L and R, respectively.)

Choose an internal vertex r such that $d_{x,r} = d_{r,y}$, and if there isn't such a vertex already in T, insert one (it will be deleted later).

For each $i \in L$, find v_i, the *residue* of vertex i, given by $v_i = d_{x,y} - d_{i,y}$. Similarly, find $v_i = d_{x,y} - d_{x,i}$ for each $i \in R$.

Now for each $i \in L$ and each $j \in R$, put $d'_{i,j} = d_{i,j} + v_i + v_j = d_{x,y}$. This is equivalent to adding v_i to the length of the pendant edge incident on vertex i, for each i. Call the tree with these new edge lengths T'. Clearly T' has exactly the same branching pattern as T: the branching order has not been changed by the above process.

For all $i,j \in L$ let $d'_{i,j} = d_{i,j}$ and for all $i,j \in R$ let $d'_{i,j} = d_{i,j}$. The new distances $\mathbf{D}' = [d_{i,j}]$ define an additive tree metric, as did the original distances $\mathbf{D} = [d_{i,j}]$, but with the added constraint that all pendant vertices are equidistant from one point, which we have labelled r (the molecular clock hypothesis). We now remove the vertex r from T'.

Due to a theorem of Buneman [7], which states that an additive tree metric specifies a unique tree, the distances $\mathbf{D}'$ can only correspond to T', which has the

same branching pattern as T.

UPGMA is known to correctly construct T' from $\mathbf{D}'$, hence SL constructs T. (Note that the *edge lengths* are *not* correctly inferred by the above method.)

This completes the proof.

Theorem 5 *NJ uses the optimal weight for net divergence.*

Proof: Consider a tree T on $n \geq 4$ pendant vertices, with distances between all pairs of vertices of T satisfying the constraints

1. $d_{i,j} > 0 \iff i \neq j$;
2. $d_{i,i} = 0$;
3. $d_{i,j} = \sum_{e \in \pi_{i,j}} \mu(e)$, where $\mu(e)$ is the length of edge e and $\pi_{i,j}$ is the path between vertices i and j.

Note that this requires the length of each edge to be positive.

Recall that at each clustering stage, NJ chooses clusters x and y to amalgamate into cluster z if and only if

$$d_{x,y} - \frac{1}{n'-2}(v_x + v_y)$$

is minimal, where v_x, the *net divergence* of cluster x, is the sum over all other clusters k of $d_{x,k}$, and n' is the number of clusters currently available for clustering.

We show that the weighting $\frac{1}{n'-2}$ for the net divergence is the only one which yields a method consistent with additive data.

Let $m_{x,y} = d_{x,y} - w(v_x + v_y)$, and suppose that $\{x, y\}$ is a neighbouring pair. Let u be another vertex of T, which is not adjacent to either of x or y. Let the edge adjacent to x be e_1, and that adjacent to y be e_2. Let e_z be some edge on the path $\pi_{x,u}$, other than e_1.

Since a proof of the consistency of NJ is known [89], it is sufficient to show that $m_{x,y} - m_{x,u} < 0$ for all possible u.

Now

$$\begin{aligned} m_{x,y} - m_{x,u} &= d_{x,y} - w(v_x + v_y) - d_{x,u} + w(v_x + v_u) \\ &= \sum_{e \in \pi_{x,y}} \mu(e) - \sum_{e \in \pi_{x,u}} \mu(e) - w \sum_{i \neq y} \left(\sum_{e \in \pi_{i,y}} \mu(e) \right) + w \sum_{i \neq u} \left(\sum_{e \in \pi_{i,u}} \mu(e) \right). \end{aligned}$$

Let $[\mu(e)]$ be the coefficient of $\mu(e)$ in $(m_{x,y} - m_{x,u})$. Then

$$\begin{aligned} [\mu(e_1)] &= 1 - 1 - w + w = 0, \\ [\mu(e_2)] &= 1 - w(n'-1) + w = 1 - w(n'-2), \text{ and} \\ [\mu(e_z)] &= -1 - w(g_z - h_z), \end{aligned}$$

where g_z is the number of vertices i such that e_z is in the path $\pi_{i,y}$, and h_z is the number of vertices i such that e_z is in the path $\pi_{i,u}$.

Note that $1 \leq g_z \leq n' - 2$, $2 \leq h_z \leq n' - 1$, and $g_z + h_z = n'$.

Since we require $m_{x,y} - m_{x,u} < 0$ for all positive edge lengths μ_i, we must have $[\mu(e)] \leq 0$ for all edges, and $[\mu(e)] < 0$ for some edge e. We already have $[\mu(e_1)] = 0$, so consider $[\mu(e_2)]$.

$$[\mu(e_2)] \leq 0 \iff w \geq \frac{1}{n' - 2}.$$

Now

$$\begin{aligned} [\mu(e_z)] \leq 0 &\iff -w(g_z - h_z) \leq 1 \\ &\iff \frac{-1}{n' - 4} \leq w \leq \frac{1}{n' - 2}, \end{aligned}$$

so $w = \frac{1}{n'-2}$ is the only weight function that will guarantee NJ does not choose $\{x, u\}$ as a neighbouring pair.

We also require that $[\mu(e)] < 0$ for some edge e. Let the edge adjacent to e_1 and e_2 be e_3. Then $g_{e_3} = n' - 2$ and $h_{e_3} = 2$, so $|\mu(e_3)| = -1 - \frac{n'-4}{n'-2} < 0$ and therefore we have $m_{x,y} - m_{x,u} < 0$.

Since u was chosen arbitrarily, NJ will only choose a neighbouring pair of vertices of T to amalgamate into a new cluster.

This completes the proof.

Theorem 6 *NJ and CTDH are identical for the case $n = 4$.*

Proof: Let $\alpha_3 = d_{1,2} + d_{3,4}$, $\alpha_5 = d_{1,3} + d_{2,4}$, and $\alpha_6 = d_{1,4} + d_{2,3}$, corresponding to the minimal path sets of trees T_3, T_5, and T_6 respectively, as shown in Figure B.2. (The internal edges of these trees are 3, 5, and 6 respectively.)

Suppose the generating tree is T_3, as in Figure B.2. Suppose also that $\alpha_3 < \alpha_5$ and $\alpha_3 < \alpha_6$, so in the distance spectrum entry $g_7 = d_{1,2,3,4} = \alpha_3$. (Recall that in the calculation of the distance spectrum, d_X is the minimum sum of distances between all pairs of elements in X.)

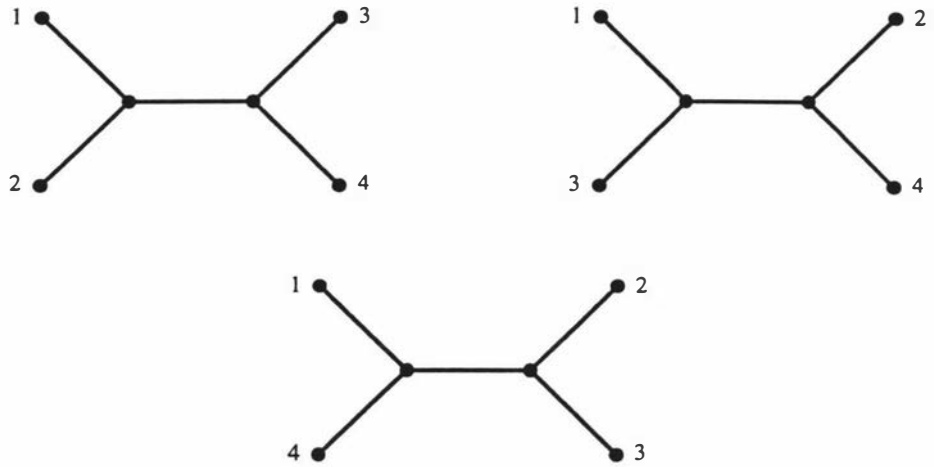


Figure B.2: The three unrooted binary trees on four pendant vertices

Let the square of the Euclidean distance between the observed distance spectrum $\mathbf{g}$ and the expected distance spectrum if the generating tree were T_k be $\Delta^2(T_k)$.

Then we have

$$\begin{aligned}\Delta^2(T_3) &= \frac{1}{16}(\alpha_5 - \alpha_6)^2,\\ \Delta^2(T_5) &= \frac{1}{16}(\alpha_5 - \alpha_6)^2 + \frac{1}{16}(-2\alpha_3 + \alpha_5 + \alpha_6)^2 + \frac{1}{12}(\alpha_5 - \alpha_3)^2.\end{aligned}$$

Therefore

$$\begin{aligned}\Delta^2(T_5) - \Delta^2(T_3) &= \frac{1}{48}\left(3(\alpha_5 + \alpha_6 - 2\alpha_3)^2 - 3(\alpha_5 - \alpha_6)^2 + 4(\alpha_5 - \alpha_3)^2\right)\\ &= \frac{1}{48}(\alpha_5 - \alpha_3)\left(12(\alpha_6 - \alpha_3) + 4(\alpha_5 - \alpha_3)\right),\end{aligned}$$

and similarly,

$$\Delta^2(T_6) - \Delta^2(T_3) = \frac{1}{48}(\alpha_6 - \alpha_3)\left(12(\alpha_5 - \alpha_3) + 4(\alpha_6 - \alpha_3)\right).$$

The quantities $\Delta^2(T_5) - \Delta^2(T_3)$ and $\Delta^2(T_6) - \Delta^2(T_3)$ are both strictly positive by the hypothesis, so CTDH selects T_3.

Likewise if $d_{1,2,3,4} = \alpha_5$ (respectively α_6), then CTDH selects T_5 (respectively T_6) by the same argument.

We have supposed that $d_{1,2,3,4} = \alpha_3$, i.e., that $\alpha_3 < \alpha_5$ and $\alpha_3 < \alpha_6$. Recall that in NJ, the initial distance matrix $\mathbf{D} = [d_{i,j}]$ is replaced by a modified matrix $\mathbf{M} = [m_{i,j}]$, where $m_{i,j} = d_{i,j} - \frac{1}{n-2}(v_i + v_j)$. The smallest value, say $m_{x,y}$, of $\mathbf{M}$ is found and the pair $\{x, y\}$ amalgamated to form cluster. With $n = 4$, only one clustering decision is necessary, and $m_{1,2} = -(d_{1,3} + d_{1,4} + d_{2,3} + d_{2,4})/2 = m_{3,4}$, $m_{1,3} = -(d_{1,2} + d_{1,4} + d_{2,3} + d_{3,4})/2 = m_{2,4}$, and $m_{1,4} = -(d_{1,2} + d_{1,3} + d_{2,4} + d_{3,4})/2 = m_{2,3}$.

Hence NJ selects $T_3 \iff m_{1,2} < m_{1,3}$ and $m_{1,2} < m_{1,4}$.

$$\begin{aligned} m_{1,2} < m_{1,3} &\iff d_{1,3} + d_{1,4} + d_{2,3} + d_{2,4} > d_{1,2} + d_{1,4} + d_{2,3} + d_{3,4} \\ &\iff d_{1,3} + d_{2,4} > d_{1,2} + d_{3,4} \\ &\iff \alpha_5 > \alpha_3, \end{aligned}$$

and

$$\begin{aligned} m_{1.2} < m_{1,4} &\iff d_{1,3} + d_{1,4} + d_{2,3} + d_{2,4} > d_{1,2} + d_{1,3} + d_{2,4} + d_{3,4} \\ &\iff d_{1,4} + d_{2,3} > d_{1,2} + d_{3,4} \\ &\iff \alpha_6 > \alpha_3. \end{aligned}$$

If the minimum of $\{\alpha_3, \alpha_5, \alpha_6\}$ is not unique, then neither CTDH nor NJ can resolve the tree.

Hence NJ and CTDH select T_3 if and only if α_3 is minimal in $\{\alpha_3, \alpha_5, \alpha_6\}$, so the two methods are identical in this case.

Theorem 7 *A simple modification of ST yields NJ.*

Proof: Let $\delta_{i,j,k,l} = 1$ if $d_{i,j} + d_{k,l} < d_{i,k} + d_{j,l}$ and $d_{i,j} + d_{k,l} < d_{i,l} + d_{j,k}$, and 0 otherwise. Let

$$S_{\{a,b\}} = \sum_{i,j \neq a,b} \delta_{i,j,a,b}$$

where the sum is taken over all quartets $\{a, b, i, j\}$.

The ST algorithm chooses $\{a,b\}$ as a neighbouring pair $\iff$ $\{a,b\}$ maximises $S_{\{a,b\}}$.

Now let $p_{a,b,i,j} = d_{a,i} + d_{b,j} + d_{a,j} + d_{b,i} - 2(d_{a,b} + d_{i,j})$.

Let

$$v_a = \sum_{i=1}^{n} d_{a,i}.$$

and

$$Q_{\{a,b\}} = \sum_{i,j \in A} p_{a,b,i,j}.$$

We will choose $\{x, y\}$ as a neighbouring pair $\iff$ $Q_{\{x,y\}}$ is maximal.

Let $A = \{1, 2, \ldots, n\} - \{a, b\}$. Note that this is particular to a given $\{a, b\}$.

Then

$$S_{(a,b)} = \sum_{i,j \in A} \delta_{i,j,a,b}.$$

For some $p_{i,j}$,

$$\sum_{i,j \in A} p_{a,b,i,j} = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{a,b,i,j} - \sum_{i=1}^{n} p_{a,b,i,i} - 2 \sum_{i \in A} p_{a,b,a,i} - 2 \sum_{i \in A} p_{a,b,b,i} - 2p_{a,b,a,b}.$$

So

$$Q_{\{a,b\}} = 2n(v_a + v_b) - 2n^2 d_{a,b} - 2 \sum_{i=1}^{n} v_i - 2 \sum_{i=1}^{n} (2d_{a,i} + 2d_{b,i} - 2(d_{a,b} + d_{i,i}))$$

$$-2 \sum_{i \in A} (d_{a,a} + d_{b,i} + d_{a,i} + d_{b,a} - 2(d_{a,b} + d_{a,i}))$$

$$-2 \sum_{i \in A} (d_{a,b} + d_{b,i} + d_{a,i} + d_{b,b} - 2(d_{a,b} + d_{b,i})) + 4d_{a,b}.$$

This is

$$2n(v_a + v_b) - 2n^2 d_{a,b} - 2 \sum_{i=1}^{n} v_i - 2(v_a + v_b) + 2nd_{a,b}$$

$$-2(v_b - d_{a,b} + v_a - d_{a,b} - (n-2)d_{b,a} - 2(v_a - d_{a,b}))$$

$$-2(v_b - d_{a,b} + v_a - d_{a,b} - (n-2)d_{b,a} - 2(v_b - d_{a,b})) + 4d_{a,b}.$$

Further manipulation yields

$$2n(v_a + v_b) - 2n^2 d_{a,b} - 2\sum_{i=1}^{n} v_i - 2(v_a + v_b) + 2nd_{a,b}$$

$$-2(v_b - v_a) + 2(n-2)d_{a,b} - 2(v_a - v_b) + 2(n-2)d_{a,b} + 4d_{a,b}$$

$$= 2(n-1)(v_a + v_b) - 2\sum_{i=1}^{n} v_i + d_{a,b}(-2n^2 + 2n + 4(n-2) + 4)$$

$$= 2(n-1)(v_a + v_b) - 2\sum_{i=1}^{n} v_i - 2(n-1)(n-2)d_{a,b}.$$

This is maximal $\iff$ $2(n-1)(n-2)(v_a + v_b - (n-2)d_{a,b})$ is maximal $\iff$ $v_a + v_b - (n-2)d_{a,b}$ is maximal $\iff$ $d_{a,b} - \frac{1}{n-2}(v_a + v_b)$ is minimal.

The second part of the proof shows that the modification of the distance matrix **D** used by ST could equally well be made in the same way as it is in NJ.

We have just shown that at the beginning of the clustering process, maximising $Q_{\{x,y\}}$ is equivalent to minimising $d_{x,y} - \frac{1}{n-2}(v_x + v_y)$, as with NJ.

Suppose we choose some $\{x, y\}$ as a neighbouring pair, and amalgamate clusters x and y to form the new cluster z.

Let **D′** be defined by

- $d'_{i,j} = d_{i,j}$ if $i, j \neq x, y$;
- $d'_{z,j} = \frac{d_{x,j} + d_{y,j} - d_{x,y}}{2}$.

Let **D″** be defined by

- $d''_{i,j} = d_{i,j}$ if $i, j \neq x, y$;
- $d''_{z,j} = \frac{d_{x,j} + d_{y,j}}{2}$.

If

$$p'_{a,b,i,j} = d'_{a,i} + d'_{b,j} + d'_{a,j} + d'_{b,i} - 2(d'_{a,b} + d'_{i,j})$$

and

$$p''_{a,b,i,j} = d''_{a,i} + d''_{b,j} + d''_{a,j} + d''_{b,i} - 2(d''_{a,b} + d''_{i,j}),$$

we need to prove that $p'_{a,b,i,j} = p''_{a,b,i,j}$.

Clearly this is true for all $a, b, i, j \neq z$, as the relevant distances have not been changed.

If one of a, b, i, j is z, we can without loss of generality suppose that $a = z$, which gives

$$p'_{z,b,i,j} = d'_{z,i} + d'_{b,j} + d'_{z,j} + d'_{b,i} - 2(d'_{z,b} + d'_{i,j})$$

$$= \frac{d_{x,i} + d_{y,i} - d_{x,y}}{2} + d_{b,j} + \frac{d_{x,j} + d_{y,j} - d_{x,y}}{2} + d_{b,i} - 2(\frac{d_{x,b} + d_{y,b} - d_{x,y}}{2} + d_{i,j})$$

$$= \frac{d_{x,i} + d_{y,i}}{2} + d_{b,j} + \frac{d_{x,j} + d_{y,j}}{2} + d_{b,i} - 2(\frac{d_{x,b} + d_{y,b}}{2} + d_{i,j})$$

$$= p''_{z,b,i,j}.$$

Hence we could equally well calculate the new distances from **D** in the manner used in NJ.

This concludes the proof.

Appendix C

Pseudocode

C.1 Introduction

This appendix lists and describes the fundamental components of the simulations programs used in this study. Such a description, free of the requirement of prior computer programming knowledge, and accessible to the intelligent layman, is necessary when comparing algorithms and heuristics in a systematic way, and is useful for those who wish to implement such methods in a programming language of their choice.

There are a few conventions used here, which require explanation:

- The hierarchy of instructions is indicated by their indentation, so loops and other repeated structures (“**for**”, “**do** (...) **while** (...)”, “**while** (...) **do** (...)” and “**case** (...) **of** (...)”) are indented.
- The *assignment operator* “$\leftarrow$” is used so that $a \leftarrow b$ means the value of a is set to the value of b.
- Swapping the values of two variables, say a and b, is indicated by “$a \leftrightarrow b$”.
- A variable quantity inside a bracket, e.g. “(x)” serves as a logical operator, *true* if $x \neq 0$ and *false* if $x = 0$.
- A “**break**” indicates that the program leaves the current repeat structure (as listed above).
- “random” is a short-hand for the “`drand48()`” pseudo-random number generator, which returns a value from a uniform distribution on the interval

[0,1).

- The "**AND**" operator returns the logical AND of two numbers, say x and y, so that each bit of $z \leftarrow x$ **AND** y is '1' if and only if the corresponding bits of x and y are both '1'.
- Arrays of variables are indicated by square brackets after their name, enclosing the number of elements in the array. A vector **v** with 4 components is thus indicated by "**v**[4]". k-dimensional arrays have k such pairs of brackets.
- Sets are stored as arrays, so for a set $A = [a_1, ..., a_k]$, $x \in A$ if $a_x \neq 0$. The union of two sets $A = [a_1, ...a_k]$ and $B = [b_1, ..., b_k]$ is $C = [c_1, ...c_k]$, where $c_i = 1$ if $a_i \neq 0$ or $b_i \neq 0$.

The two main programs used are `sim.c` and `big.c`.

C.2 General functions

Algorithm C.1 : compare_sets(A, B, max)

{ Returns the number of elements in A which are not in B. A and B are assumed to be the same size, max. }

```
local variables:
  differences,          { number of elements of A which are different
                          from all the elements in B }
  i, j,                 { counters }
  same;                 { flag }
differences ← 0
for i = 1 to max do
  same ← 0
  for j = 1 to max do
    if (A_i = B_j) then
      same ← 1
      break
  if (same = 0) then
    differences ← differences + 1
return (differences)
end.
```

Algorithm C.2 : Hadamard($\mathbf{v}, \mathbf{w}$)

{ This function multiplies the vector $\mathbf{v}$ by the Hadamard matrix $\mathbf{H}_n$ and puts the result into vector $\mathbf{w}$. This uses the "fast-Hadamard method" devised by Hendy [45]. }

local variables:

$i, j, k, l, a, jump, steps,$ { counters }

x, y

for $i = 0$ **to** m **do** $w_i \leftarrow v_i$ { $m = 2^{n-1}$ }

$jump \leftarrow 1$

$steps \leftarrow m/2$

while $(jump < m)$ **do**

$a \leftarrow 0$

for $i = 1$ **to** $steps$ **do**

for $j = 0$ **to** $jump - 1$ **do**

$k \leftarrow a + j$

$l \leftarrow k + jump$

$x \leftarrow w_k + w_l$

$y \leftarrow w_k - w_l$

$w_k \leftarrow x$

$w_l \leftarrow y$

$a \leftarrow a + 2 \times jump$

$jump \leftarrow 2 \times jump$

$steps \leftarrow steps/2$

end.

Algorithm C.3 : HexpH(inv,outv)

{ This function calculates $\frac{1}{m}$**H**(exp(**H**(**inv**))) and puts the result into **outv.** }

local variables:
  i, { counter }
  $\rho[M]$; { intermediate vector }
$inv_0 \leftarrow 0$
for $i = 1$ **to** $m-1$ **do** $inv_0 \leftarrow inv_0 - inv_i$
Hadamard(**inv**, ρ)
for $i = 0$ **to** $m-1$ **do** $\rho_i \leftarrow \exp(\rho_i)$
Hadamard(ρ, **outv**)
for $i = 0$ **to** $m-1$ **do** $outv_i \leftarrow outv_i/m$
end.

Algorithm C.4 : HlnH

{ This function calculates $\frac{1}{m}$**H**(ln(**H**(**inv**))) and puts the result into **outv.** }

local variables:
  i, { counter }
  err, { $err = 1 \Leftrightarrow$ a non-positive edge length is inferred }
  ρ; { intermediate vector }
Hadamard(**inv**,ρ)
$err \leftarrow 0$
for $i = 0$ **to** $m-1$ **do**
  if $(\rho_i > 0)$ **then** $\rho_i \leftarrow \ln(\rho_i)$
  else
    $err \leftarrow 1$
    break
if $(err = 0)$ **then**
  Hadamard(ρ,**outv**)
  for $i = 0$ **to** $m-1$ **do** $outv_i \leftarrow outv_i/m$
return (err)
end.

Algorithm C.5 : permute(x, $perm$)

{ This returns a pseudo-random permutation of $\{1, \ldots, x\}$. }

local variables: i, j; { counters }
for $i = 1$ **to** x **do** $perm_i \leftarrow i$
for $i = 1$ **to** x **do**
  $j \leftarrow random \times (x - i) + i + 1$
  { when converting type **double** to type **int**, the number is truncated }
  $perm_i \longleftrightarrow perm_j$
end.

Algorithm C.6 : rough_exp(i)

{ This function returns $\approx 100 \times 10^{i/10}$. }

local variable: j
$j \leftarrow 0$
if ($i \geq -10$) **then**
  case i **of**
    -10: $j \leftarrow 10$
    -9: $j \leftarrow 13$
    -8: $j \leftarrow 16$
    -7: $j \leftarrow 20$
    -6: $j \leftarrow 25$
    -5: $j \leftarrow 32$
    -4: $j \leftarrow 40$
    -3: $j \leftarrow 50$
    -2: $j \leftarrow 64$
    -1: $j \leftarrow 80$
    0: $j \leftarrow 100$
    1: $j \leftarrow 125$
    default: $j \leftarrow 10 \times$ rough_exp($i - 10$)
  return (j)
end.

Algorithm C.7 : permutation_to_tree($perm, output\_tree$)

{ $perm$ is an array, a permutation of the integers $1, ..., n-1$. }

local variables:
  $ea[N]$, { $ea[N]$ is the array of earliest ancestors chosen thus far }
  x, y, new, i, j; { counters }

for $i = 1$ **to** n **do**
  $ea_i \leftarrow i$
  $output\_tree_i \leftarrow 0$
for $i = n+1$ **to** $2n$ **do**
  $ea_i \leftarrow 0$
  $output\_tree_i \leftarrow 0$
for $i = 1$ **to** $n-1$ **do**
  $new \leftarrow n+i$
  $x \leftarrow ea_{perm_i}$
  $y \leftarrow ea_{perm_i+1}$
  $output\_tree_x \leftarrow new$
  $output\_tree_y \leftarrow new$
  $ea_x \leftarrow new$
  $ea_y \leftarrow new$
  $ea_{new} \leftarrow new$
  for $j = 1$ **to** $new - 1$ **do**
    if $((ea_j = x)$ **or** $(ea_j = y))$ **then** $ea_j \leftarrow new$

end.

Algorithm C.8 : sample_uniform($mean$, $range$)

{ Here, $range$ is $max\_value - min\_value$ }

local variable: temp; { interim floating-point number }

$temp \leftarrow ((random - 0.5) \times range) + mean$

return ($temp$)

end.

Algorithm C.9 : sample_normal($mean$, std_dev)

{ This uses the Box-Muller method of generating normally distributed random numbers from uniformly distributed random numbers [70]. }

local variables:

$x_1, x_2, y, fac, rsq, bm$;

$iset \leftarrow 0$; { $iset$ is initially set to 0. }

$gset$

if ($iset = 0$) **then**

do

$x_1 \leftarrow 2 \times random - 1$

$x_2 \leftarrow 2 \times random - 1$

$rsq \leftarrow x_1^2 + x_2^2$

while ($rsq \geq 1$) **or** ($rsq \leftarrow 0$)

$fac \leftarrow \sqrt{-2\ln(rsq)/rsq}$

$gset \leftarrow x_1 fac$

$iset \leftarrow 1$

$bm \leftarrow x_2 fac$

else

$iset \leftarrow 0$

$bm \leftarrow gset$

$bm \leftarrow bm \times std\_dev + mean$

return (bm)

end.

Algorithm C.10 : sample_log_normal($mean, std\_dev$)

local variable: *temp*

temp ← sample_normal(ln(*mean*), *std_dev*)

temp ← exp(*temp*)

return (*temp*)

end.

C.3 Functions used in `sim.c`

Algorithm C.11 : compat(x, f, A)

{ Checks to see if the number x is compatible with the numbers which are indexed by the first f elements of the set A. }

local variables:

i, *com*, *tempi*; { counters }

{ First assume j is compatible with the first f edges indexed by A. }

com ← 1

if ($f > 0$) **then**

  if ($x = 0$) **then** *com* ← 0

  else

    for $i = 1$ **to** f **do**

      tempi ← x **AND** *opos*(A_i)

      if (*tempi* $\notin \{0, x, opos(A_i)\}$) **then**

        com ← 0

        break

      if (*opos*(A_i) = x) **then**

        com ← 0 { A bipartition is not compatible with itself }

        break

return (*com*)

end.

Algorithm C.12 : choose_topology(*topnumber*, *edge_set*)

local variables:

```
  i, j, x,                    { counters }
  C[N][N],                    { array of clusters, each a subset of {1,...,n}. }
  r[N];                       { array of permuted integers, {r_1,...,r_n}. }
for i = 1 to n do  C_i ← ∅
permute(n, r)
case topnumber of
  1:
    C_1 ← {r_1, r_2}          { UB2 }
  2:
    C_1 ← {r_1, r_2}          { UB3 }
    C_2 ← {r_3, r_4}
  3:
    C_1 ← {r_1, r_2}          { UB4 }
    C_2 ← {r_1, r_2, r_3}
    C_3 ← {r_4, r_5}
  .....                       { The other cases, 4 to 26, have been omitted for
                                brevity. }
  27:
    C_1 ← {r_1, r_2}          { UB5, 1_2, 2_3 }
    C_2 ← {r_3, r_4}
    C_3 ← {r_1, r_2, r_3, r_4}
    C_4 ← {r_5, r_6}
    C_5 ← {r_5, r_6, r_7}
    C_6 ← {r_8, r_9}
    C_7 ← {r_8, r_9, r_10}
  default: break
edge_set ← ∅
for i = 1 to n − 3 do
  if (n ∈ C_i) then
    x ← m − 1
    for j = 1 to n − 1 do
      if (j ∈ C_i) then  x ← x − 2^(j−1)
  else
```

```
        x ← 0
        for j = 1 to n − 1 do
          if (j ∈ C_i) then   x ← x + 2^(j−1)
      edge_set_i ← x
end.
```

Algorithm C.13 : bipartitions_to_distances(**v**)

{ This calculates the distance matrix **D** from **v**. }

local variables:

$i, j, k,$	{ counters }
$ok,$	{ flag }
$t[N],$	{ the numbers $1, 2, 4, ..., 2^n$ }
$d0;$	{ = 1 if a zero pair-wise distance is observed. }

for $i = 1$ **to** n **do**

  for $j = 1$ **to** n **do** $D_{i,j} \leftarrow 0$

$t_1 \leftarrow 1$

for $i = 2$ **to** n **do** $t_i \leftarrow 2 \times t_{i-1}$

{ $D_{i,j}$ is the sum of the observed proportions of the bipartitions for which the characters in the i-th and j-th places differ. }

for $k = 1$ **to** $m - 1$ **do**

  for $i = 1$ **to** $n - 1$ **do**

    for $j = i + 1$ **to** n **do**

      $ok \leftarrow 0$

      if $(((k$ **AND** $t_i) = t_i)$ **and** $((k$ **AND** $t_j) = 0)))$ **then** $ok \leftarrow 1$

      if $(((k$ **AND** $t_i) = 0)$ **and** $((k$ **AND** $t_j) = t_j))$ **then** $ok \leftarrow 1$

      if (ok) **then** $D_{i,j} \leftarrow D_{i,j} + v_k$

$d0 \leftarrow 0$

for $i = 1$ **to** $n - 1$ **do**

  for $j = i + 1$ **to** n **do**

    $D_{j,i} \leftarrow D_{i,j}$

    if $(D_{i,j} \leftarrow 0)$ **then** $d0 \leftarrow 1$

return $(d0)$

end.

Algorithm C.14 : choose_tree($x, edge\_set$)

{ Chooses a tree with x pendant vertices, from the uniform distribution on all trees equally likely, and puts the edge set describing that tree into *edge_set*. }

```
local variables: k, t;                { counters }
case x of
  4:  t ← 1
  5:  t ← 2
  6:
    if (random ≤ 3/4) then  t ← 3
    else t ← 4
  7:
    if (random ≤ 2/3) then  t ← 5
    else t ← 6
  8:
    k ← random × 33
    if (k ≤ 16) then  t ← 7
    else if (k ≤ 24) then  t ← 8
    else if (k ≤ 32) then  t ← 9
    else t ← 10
  9:
    k ← random × 143
    if (k ≤ 48) then  t ← 11
    else if (k ≤ 96) then  t ← 13
    else if (k ≤ 120) then  t ← 12
    else if (k ≤ 132) then  t ← 14
    else if (k ≤ 140) then  t ← 16
    else t ← 15
 10:
    k ← random × 143
    if (k ≤ 32) then  t ← 17
    else if (k ≤ 64) then  t ← 19
```

else if ($k \leq 80$) **then** $t \leftarrow 18$
else if ($k \leq 96$) **then** $t \leftarrow 20$
else if ($k \leq 112$) **then** $t \leftarrow 26$
else if ($k \leq 120$) **then** $t \leftarrow 21$
else if ($k \leq 128$) **then** $t \leftarrow 22$
else if ($k \leq 136$) **then** $t \leftarrow 24$
else if ($k \leq 140$) **then** $t \leftarrow 27$
else if ($k \leq 142$) **then** $t \leftarrow 23$
else $t \leftarrow 25$
default: **break**
choose_topology(t, *edge_set*)
end.

Algorithm C.15 : correct_distances

{ This function uses the Jukes-Cantor distance adjustment to account for multiple changes of character state along lineages. The input distances are in matrix **D** and the adjusted values are returned to **D**. The model is the simplest possible, with 2 states and symmetric i.i.d. rates of change. }

local variables:
i, j, { counters }
d_inf; { returns 1 $\Longleftrightarrow$ the inferred distance is infinite }
$d\_inf \leftarrow 0$
for $i = 1$ **to** $n - 1$ **do**
for $j = i + 1$ **to** n **do**
if ($D_{i,j} \geq 0.5$) **then**
$D_{i,j} \leftarrow 1000000$
$d\_inf \leftarrow d\_inf + 1$
else $D_{i,j} \leftarrow -log(1 - 2 \times D_{i,j})/2$
$D_{j,i} \leftarrow D_{i,j}$
return (*d_inf*)
end.

Algorithm C.16 : get_pathsets(**D**)

local variables:

```
  i, j, k, l, b,                 { counters }
  i2, j2,                        { i2 = 2^(i-1), j2 = 2^(j-1) }
  parity,
  dji,
  sum;                           { total of all pairwise distances }
sum ← 0
for i = 1 to n − 1 do
  for j = i + 1 to n do  sum ← sum + D_{i,j}
for i = 1 to m − 1 do  g_i ← sum
g_0 ← 0
for i = 1 to n − 1 do
  parity ← 1
  i2 ← 2^(i-1)
  for j = 2^(i-1) to 2^i do
    if (parity) then
      temp ← D_{i,n} + g_{j−2^(i-1)}
      if (temp < g_j) then  g_j ← temp
    b ← j
    while (b ≡ 1(mod2)) do
      b ← b/2
      parity ← 1 − parity
    parity ← 1 − parity
  for j = 1 to i − 1 do
    k ← 2^(i-1)
    j2 ← 2^(j-1)
    dji ← D_{i,j}
    while (k < 2^i) do
      k ← k + 2^(i-1)
      for l = k to k + 2^(j-1) do
      temp ← dji + g_{l−2^(i-1)−2^(j-1)}
      if (temp < g_l) then  g_l ← temp
    k ← k + 2^(j-1)
end.
```

Algorithm C.17 : number_of_bipartitions

{ This function returns the number of parsimony sites (i.e., those which are used by MPSO) of the data set. }

local variables:

$i, j, num,$ { counters }

$f[M];$ { temporary vector }

$num \leftarrow 0$

for $i = 1$ **to** $m - 1$ **do** $f_i \leftarrow freq_i$

for $i = 1$ **to** $n - 1$ **do**

$j \leftarrow 2^{i-1}$

$f_j \leftarrow 0$

$f_{m-1} \leftarrow 0$

for $i = 1$ **to** $m - 1$ **do** **if** $(f_i > 0)$ **then** $num \leftarrow num + 1$

return (num)

end.

Algorithm C.18 : random_edge_lengths(**edge_set**, **outv**)

{ Assigns random edge lengths to the edges in **edge_set**, and puts them in the vector **outv**. }

local variable: i; { counter }

for $i = 1$ **to** $m - 1$ **do** $outv_i \leftarrow 0$

{ assign lengths to internal edges: }

for $i = 1$ **to** $n - 3$ **do**

$outv_{edge\_set_i} \leftarrow random \times (max\_int\_length - min\_edge\_length) + min\_edge\_length$

$outv_0 \leftarrow outv_0 - outv_{edge\_set_i}$

{ assign lengths to pendant edges: }

$i \leftarrow 1$

while $(i < m)$ **do**

$outv_i \leftarrow random \times (max\_pen\_length - min\_edge\_length) + min\_edge\_length$

$outv_0 \leftarrow outv_0 - outv_i$

$i \leftarrow 2 \times i$

$outv_{m-1} \leftarrow random \times (max\_pen\_length - min\_edge\_length) + min\_edge\_length$

$outv_0 \leftarrow outv_0 - outv_{m-1}$

end.

Algorithm C.19 : sample_bipartitions(*length*, *error_rate*, **inv**, **outv**)

{ samples *length* times from the expected bi-partition spectrum **inv**, with possible reading errors given by *error_rate*, and puts the result into **outv** }

local variables:

y$[M]$,	{ vector of cumulative probabilities }
$i, j, gap, tempi, k, taxon, bit, temp$,	{ counters }
$ranked[M]$,	{ indices of the components of y in decreasing order of size }
r,	{ interim floating-point number }
v$[M]$;	{ interim vector of floating-point numbers }

for $i = 0$ **to** $m - 1$ **do** $freq_i \leftarrow 0$

for $i = 0$ **to** $m - 1$ **do**

$v_i \leftarrow inv_i$

$ranked_i \leftarrow i$

{ Perform a ShellSort first for speed of sampling: }

$gap \leftarrow m/2$

while $(gap > 0)$ **do**

for $i = gap$ **to** $m - 1$ **do**

$j \leftarrow i - gap$

while $((j \geq 0)$ **and** $(v_j < v_{j+gap}))$ **do**

$k \leftarrow j + gap$

$v_j \longleftrightarrow v_k$

$ranked_j \longleftrightarrow ranked_k$

$j \leftarrow j - gap$

$gap \leftarrow gap/2$

$y_0 \leftarrow inv_{ranked_0}$

for $i = 1$ **to** $m - 1$ **do** $y_i \leftarrow y_{i-1} + inv_{ranked_i}$

if $(error\_rate > 0)$ **then**

for $i = 1$ **to** *length* **do**

$r \leftarrow random$

$j \leftarrow 0$

while $((r > y_j)$ **and** $(j < m))$ **do** $j \leftarrow j + 1$

if $(random < error\_rate)$ **then**

```
            taxon ← random × n + 1
            if (taxon = n) then  k ← m − j − 1
            else
              temp ← 2^(taxon−1)
              bit ← ((ranked_j AND temp) > 0)
              k ← ranked_j + (1 − 2 * bit) × temp
            freq_ranked_k ← freq_ranked_k + 1
          else  freq_ranked_j ← freq_ranked_j + 1
        else
          for i = 1 to length do
            r ← random
            j ← 0
            while ((r > y_j) and (j < m)) do  j ← j + 1
            freq_ranked_j ← freq_ranked_j + 1
        for i = 0 to m − 1 do  outv_i ← freq_i/c
                                   { this normalises the frequencies }
    end.
```

Algorithm C.20 : sort_vector_descending(**inv**, **outv**)

{ Uses Shell's method of sorting an array **inv** of real numbers. The *ranks* of the components of **inv** are put into **outv**, which is a vector of integers [70]. }

local variables:

$gap, i, j, k,$ { counters }

h$[M]$; { temporary vector }

for $i = 0$ **to** $m - 1$ **do**

$h_i \leftarrow inv_i$

$out_i \leftarrow i$

$gap \leftarrow m/2$

while $(gap > 0)$ **do**

for $i = gap$ **to** $m - 1$ **do**

$j \leftarrow i - gap$

while $((j > 0)$ **and** $(h_j < h_{j+gap}))$ **do**

$k \leftarrow j + gap$

$h_j \longleftrightarrow h_k$

$outv_j \longleftrightarrow outv_k$

$j \leftarrow j - gap$

$gap \leftarrow gap/2$

end.

C.3.1 Clustering methods

Algorithm C.21 : NJ(*averaging*)

{ If *averaging* = 1, do the NJa method, otherwise do NJ. }

local variables:

Active[*N*],	{ set of clusters which are available }
edge,	{ name of the new edge }
i, *j*,	{ counters }
internals,	{ number of internal edges currently chosen }
λ,	{ mean over all $k \notin \{x, y\}$ of $D_{x,k} - D_{y,k}$ }
minr,	{ handy temporary real number }
netdiv[*N*],	{ the vector of net divergences }
new,	{ label of the new cluster }
ok,	{ logical flag used in comparing sets of inferred edges }
sub[*N*];	{ $sub_i = 1 \iff$ the cluster i contains pendant vertex n. }
tn,	{ current number of unattached clusters }
wrong,	{ the number of edges in NJ_intedges and not in intedges }
x, *y*,	{ clusters chosen to form pair }
A[*N*][*N*];	{ $A_{i,j} = D_{i,j} - (netdiv_i + netdiv_j)/(tn - 2)$ }

$Active \leftarrow \{1, 2, \ldots, n\}$
$new \leftarrow n + 1$
for $i = 0$ **to** $2n - 1$ **do** $sub_i \leftarrow 0$
$sub_n \leftarrow 1$
$e_1 \leftarrow 1$
for $i = 2$ **to** $n - 1$ **do** $e_i \leftarrow 2 \times e_{i-1}$
$e_n \leftarrow 2 \times e_{n-1} - 1$
$internals \leftarrow 1$
while ($internals < n - 2$) **do**
  for $i = 1$ **to** $new - 1$ **do**
    $netdiv_i \leftarrow 0$
    for $j = 1$ **to** $new - 1$ **do**
      if ($i \in Active$ **and** $j \in Active$ **and** $(i \neq j)$) **then**
        $netdiv_i \leftarrow netdiv_i + D_{i,j}$

```
for j = new to N − 1 do  netdiv_j ← 0
for i = 1 to new − 1 do
  for j = 1 to new − 1 do
    if (i ∈ Active and j ∈ Active) then
      A_{i,j} ← D_{i,j} − (netdiv_i + netdiv_j)/(tn − 2)
  i ← 1
  while ((i < 2n) and (i ∉ Active)) do  i ← i + 1
  x ← i
  i ← i + 1
  while ((i < 2n) and (i ∉ Active)) do  i ← i + 1
  y ← i
  minr ← A_{x,y}
  arbitrary ← 0
  for i = 1 to new − 2 do
    if (i ∈ Active) then
      for j = i + 1 to new − 1 do
        if ((j ∈ Active) and (A_{i,j} ≤ minr)) then
          if (A_{i,j} = minr) then  arbitrary ← arbitrary + 1
          else arbitrary ← 0
          minr ← A_{i,j}
          x ← i
          y ← j
  Active ← Active − {x, y}
  Active ← Active ∪ {new}
  if ((x = n) or (y = n) or (sub_x = 1) or (sub_y = 1)) then  sub_new ← 1
  if (sub_new) then  edge ← |e_x − e_y|
  else edge ← e_x + e_y
  e_new ← edge
  if (averaging) then  NJa_intedges_internals ← edge
  else NJ_intedges_internals ← edge
  if (averaging) then
    λ ← 0
    for i = 1 to new − 1 do
      if (Active_i) then  λ ← λ + D_{x,i} − D_{y,i}
    λ ← λ/(tn − 2)
    for i = 1 to new do
```

```
            if (i ∈ Active) then  D_{i,new} ← D_{i,x} − (D_{x,y} + λ)/2
            else D_{i,new} ← 0
        else
          for i = 1 to new do
            if (i ∈ Active) then  D_{i,new} ← (D_{i,x} + D_{i,y} − D_{x,y})/2
            else D_{i,new} ← 0
        for i = 1 to new − 1 do
          for j = i + 1 to new do  D_{j,i} ← D_{i,j}
        for i = 1 to new do  D_{i,i} ← 0
        tn ← tn − 1
        internals ← internals + 1
        new ← new + 1
    wrong ← 0
    if (averaging) then
      for i = 1 to n − 3 do
        ok ← 0
        for j = 1 to n − 3 do
          if (intedges_i = NJa_intedges_j) then
            ok ← 1
            break
        if (ok = 0) then  wrong ← wrong + 1
      NJa_score_wrong ← NJa_score_wrong + 1
    else
      for i = 1 to n − 3 do
        ok ← 0
        for j = 1 to n − 3 do
          if (intedges_i = NJ_intedges_j) then
            ok ← 1
            break
        if (ok = 0) then  wrong ← wrong + 1
      NJ_score_wrong ← NJ_score_wrong + 1
end.
```

Algorithm C.22 : ST

local variables:

$Active[N]$,	{ set of clusters which are available }
$edge$,	{ name of the new edge }
i, j, k, l,	{ counters }
$internals$,	{ number of internal edges currently chosen }
$maxSup$;	{ maximum support found for a pair $\{i, j\}$ }
new,	{ label of the new cluster }
ok,	
$sub[M]$,	{ $sub_i = 1$ if cluster i includes pendant vertex n, 0 otherwise. }
$Sup[N][N]$,	{ support for neighbouring pair $\{i, j\}$ }
$wrong$,	{ number of edges wrongly chosen by ST }
x, y,	{ clusters chosen to form pair }

$wrong \leftarrow 0$
$internals \leftarrow 1$
$Active \leftarrow \{1, 2, \ldots, n\}$
$new \leftarrow n + 1$
for $i = 0$ **to** $N - 1$ **do** $sub_i \leftarrow 0$
$sub_n \leftarrow 1$
$e_1 \leftarrow 1$
for $i = 2$ **to** $n - 1$ **do** $e_i \leftarrow 2 \times e_{i-1}$
$e_n \leftarrow 2 \times e_{n-1} - 1$
$internals \leftarrow 1$
while $(internals < n - 2)$ **do**
  $i \leftarrow 1$
  while $((i < 2n)$ **and** $(i \notin Active))$ **do** $i \leftarrow i + 1$
  $x \leftarrow i$
  $i \leftarrow i + 1$
  while $((i < 2n)$ **and** $(i \notin Active))$ **do** $i \leftarrow i + 1$
  $y \leftarrow i$
  $maxSup \leftarrow 0$
  for $i = 1$ **to** $2n - 2$ **do**
    $Sup_{i,i} \leftarrow 0$
    for $j = i + 1$ **to** $2n - 1$ **do**

```
      Sup_{i,j} ← 0
      Sup_{j,i} ← 0
  arbitrary ← 0
  for i = 1 to new − 2 do
    if (i ∈ Active) then
      for j = i + 1 to new − 1 do
        if (j ∈ Active) then
          { Get support Sup_{i,j} for the pair {i,j}: }
          for k = 1 to new − 2 do
            if ((k ≠ i) and (k ≠ j) and (k ∈ Active)) then
              for l = k + 1 to new − 1 do
                if ((l ≠ i) and (l ≠ j) and (l ∈ Active)) then
                  if ((D_{i,j} + D_{k,l} < D_{i,k} + D_{j,l}) and (D_{i,j} + D_{k,l} < D_{i,l} + D_{j,k}))
                  then
                    Sup_{i,j} ← Sup_{i,j} + 1
          if (Sup_{i,j} ≥ maxSup) then
            if (Sup_{i,j} = maxSup) then  arbitrary ← arbitrary + 1
            else arbitrary ← 0
            x ← i
            y ← j
            maxSup ← Sup_{i,j}
  Active ← Active − {x, y}
  Active ← Active ∪ {new}
  { if x or y ← n, or if sub_x or sub_y ← 1, then sub_new ← 1: }
  if ((x = n) or (y = n) or (sub_x = 1) or (sub_y = 1)) then
    sub_new ← 1
  if (sub_new = 1) then  edge ← |e_x − e_y|
  else edge ← e_x + e_y
  e_new ← edge
  ST_intedges_internals ← edge
  for i = 1 to new do
    if (i ∈ Active) then  D_{i,new} ← (D_{i,x} + D_{i,y})/2
    else D_{i,new} ← 0
  for i = 1 to new − 1 do
    for j = i + 1 to new do  D_{j,i} ← D_{i,j}
```

```
    for i = 1 to new do D_{i,i} ← 0
    internals ← internals + 1
    new ← new + 1
  wrong ← 0
  for i = 1 to n − 3 do
    ok ← 0
    for j = 1 to n − 3 do
      if (intedges_i = ST_intedges_j) then
        ok ← 1
        break
    if (ok = 0) then  wrong ← wrong + 1
  ST_score_wrong ← ST_score_wrong + 1
end.
```

Algorithm C.23 : UPGMA(*version*)

{ *version* = 0 for UPGMA, 1 for SL and 2 for TD. }

local variables:

$A[N][N]$,	{ the distance matrix used as input }
$Active[N]$,	{ set of clusters which are available }
$cs[N]$,	{ the number of pendant vertices in each cluster }
edge,	{ name of the new edge }
i, j,	{ counters }
internals,	{ number of internal edges currently chosen }
minr,	{ handy temporary real number }
new,	{ label of the new cluster }
ok,	{ temporary flag }
$sub[M]$;	{ $sub_i = 1$ if cluster i includes pendant vertex n, 0 otherwise. }
wrong,	{ the number of edges UPGMA gets wrong }
x, y;	{ clusters chosen to form pair }

if (*version* > 0) **then**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do** $A_{i,j} \leftarrow tranD_{i,j}$
else
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do** $A_{i,j} \leftarrow D_{i,j}$
wrong $\leftarrow 0$
if (*version* = 1) **then**
  { SL }
  for $i = 0$ **to** $N - 1$ **do** $SL\_intedges_i \leftarrow 0$
else if (*version* = 2) **then**
  { TD }
  for $i = 0$ **to** $N - 1$ **do** $TD\_intedges_i \leftarrow 0$
else
  { UPGMA }
  for $i = 0$ **to** $N - 1$ **do** $UPGMA\_intedges_i \leftarrow 0$
Active $\leftarrow \{1, 2, \ldots, n\}$
for $i = 1$ **to** n **do** $cs_i \leftarrow 1$

$new \leftarrow n + 1$
for $i = 1$ **to** $N - 1$ **do** $sub_i \leftarrow 0$
$sub_n \leftarrow 1$
$e_1 \leftarrow 1$
for $i = 1$ **to** $n - 1$ **do** $e_i \leftarrow 2^{i-1}$
$e_n \leftarrow 2^{n-1} - 1$
$internals \leftarrow 1$
while $(internals < n - 2)$ **do**
  $i \leftarrow 1$
  while $((i < N)$ **and** $(i \notin Active))$ **do** $i \leftarrow i + 1$
  $x \leftarrow i$
  $i \leftarrow i + 1$
  while $((i < N)$ **and** $(i \notin Active))$ **do** $i \leftarrow i + 1$
  $y \leftarrow i$
  $arbitrary \leftarrow 0$
  $minr \leftarrow A_{x,y}$
  for $i = 1$ **to** $new - 2$ **do**
    if $(i \in Active)$ **then**
      for $j = i + 1$ **to** $new - 1$ **do**
        if $((j \in Active)$ **and** $(A_{i,j} \leq minr))$ **then**
          if $(A_{i,j} = minr)$ **then** $arbitrary \leftarrow arbitrary + 1$
          else $arbitrary \leftarrow 0$
          $minr \leftarrow A_{i,j}$
          $x \leftarrow i$
          $y \leftarrow j$
  $Active \leftarrow Active - \{x, y\}$
  $Active \leftarrow Active \cup \{new\}$
  if $((x = n)$ **or** $(y = n)$ **or** $(sub_x = 1)$ **or** $(sub_y = 1))$ **then**
    $sub_{new} \leftarrow 1$
  if $(sub_{new} = 1)$ **then** $edge \leftarrow |e_x - e_y|$
  else $edge \leftarrow e_x + e_y$
  $e_{new} \leftarrow edge$
  $cs_{new} \leftarrow cs_x + cs_y$
  if $(version = 1)$ **then**
    $SL\_intedges_{internals} \leftarrow edge$

```
    else if (version = 2) then
      TD_intedges_internals ← edge
    else UPGMA_intedges_internals ← edge
    for i = 1 to new do
      if (Active_i) then
        A_i,new ← (cs_x × A_i,x + cs_y × A_i,y)/(cs_x + cs_y)
      else A_i,new ← 0
      for i = 1 to new − 1 do
        for j = i + 1 to new do  A_j,i ← A_i,j
      for i = 1 to new do  A_i,i ← 0
      internals ← internals + 1
      new ← new + 1
  wrong ← 0
  if (version = 1) then
    for i = 1 to n − 3 do
      ok ← 0
      for j = 1 to n − 3 do
        if (intedges_i = SL_intedges_j) then
          ok ← 1
          break
      if (ok = 0) then  wrong ← wrong + 1
    SL_score_wrong ← SL_score_wrong + 1
  else if (version = 2) then
    for i = 1 to n − 3 do
      ok ← 0
      for j = 1 to n − 3 do
        if (intedges_i = TD_intedges_j) then
          ok ← 1
          break
      if (ok = 0) then  wrong ← wrong + 1
      TD_score_wrong ← TD_score_wrong + 1
  else
    for i = 1 to n − 3 do
      ok ← 0
      for j = 1 to n − 3 do
        if (intedges_i = UPGMA_intedges_j) then
```

$ok \leftarrow 1$

break

if $(ok = 0)$ **then** $wrong \leftarrow wrong + 1$

$UPGMA\_score_{wrong} \leftarrow UPGMA\_score_{wrong} + 1$

end.

Algorithm C.24 : TD(*version*)

{ SL (*version* = 1) and TD (*version* = 2). }

local variables:

$d_L[N], d_R[N]$,	{ average distances between elements in L (or R) and the other taxa }
$dmax$,	{ $D_{x,y}$ }
$dmin$,	{ current shortest distance between R (or L) and any other taxon }
i, j, k,	{ counters }
$L[N], R[N]$,	{ sets of taxa to the left and right of the root (the cardinality of each is in the zero-th place of the array) }
$tempsum$,	
x, y,	{ the two most distant taxa }
z,	{ the taxon to be added to R or L }
$r[N]$;	{ the residue of extra changes picked up by each taxon, as estimated by differences in $d_{R,i}$ and $d_{L,i}$ }

{ set up R and L: }

$L \leftarrow \emptyset$
$R \leftarrow \emptyset$
{ find most distant taxa: }

$dmax \leftarrow -1$
for $i = 1$ **to** $n - 1$ **do**
  for $j = i + 1$ **to** n **do**
    if ($D_{i,j} > dmax$) **then**
      $x \leftarrow i$
      $y \leftarrow j$
      $dmax \leftarrow D_{i,j}$
$L \leftarrow \{y\}$
$R \leftarrow \{x\}$
$L_0 \leftarrow 1$
$R_0 \leftarrow 1$
for $i = 1$ **to** n **do**

$d_{L,i} \leftarrow D_{i,y}$
$d_{R,i} \leftarrow D_{i,x}$
if $(version = 2)$ **then**
while $((R_0 + L_0) < n)$ **do**
$dmin \leftarrow dmax + 1$
for $i = 1$ **to** n **do**
if $((i \notin R)$ **and** $(i \notin L))$ **then**
if $(d_{L,i} \leq dmin)$ **then**
$z \leftarrow i$
$dmin \leftarrow d_{L,i}$
$k \leftarrow 0$
if $(d_{R,i} \leq dmin)$ **then**
$z \leftarrow i$
$dmin \leftarrow d_{R,i}$
$k \leftarrow 1$
if $(k > 0)$ **then**
$R \leftarrow R \cup \{z\}$
$R_0 \leftarrow R_0 + 1$
for $i = 1$ **to** n **do**
if (R_i) **then** $d_{R,i} \leftarrow 0$
else
$tempsum \leftarrow 0$
for $j = 1$ **to** n **do**
if (R_j) **then** $tempsum \leftarrow tempsum + D_{i,j}$
$d_{R,i} \leftarrow tempsum/R_0$
else
$L \leftarrow L \cup \{z\}$
$L_0 \leftarrow L_0 + 1$
for $i = 1$ **to** n **do**
if $(i \in L)$ **then** $d_{L,i} \leftarrow 0$
else
$tempsum \leftarrow 0$
for $j = 1$ **to** n **do**
if $(j \in L)$ **then** $tempsum \leftarrow tempsum + D_{i,j}$
$d_{L,i} \leftarrow tempsum/L_0$
else
for $i = 1$ **to** n **do**

```
        if (D_{x,i} < D_{y,i}) then
          R ← R ∪ {i}
          R_0 ← R_0 + 1
        else
          L ← L ∪ {i}
          L_0 ← L_0 + 1
    for i = 1 to n do
      if (i ∈ L) then  r_i ← d_{R,i} − d_{R,y}
      else r_i ← d_{L,i} − d_{L,x}
    for i = 1 to n do
      for j = 1 to n do  tranD_{i,j} ← D_{i,j} − r_i − r_j
    for i = 1 to n do  tranD_{i,i} ← 0
    UPGMA(version)
  end.
```

C.3.2 Search methods

Algorithm C.25 : CT(v, *distances*, *use_Hadamard*)

{ *distances* = 1 if the input vector is inferred from the distance matrix D, 0 otherwise.

use_Hadamard = 1 if the Hadamard conjugation method is used, 0 otherwise. }

local variables:

b,	{ the best possible value for Δ^2 with the current edge set S }
$bestS[N]$,	{ the set of edges which gives the best Δ^2 value }
bound,	{ the best value found for Δ^2 with a fully resolved tree }
i, j, x, y,	{ counters }
essentials,	{ the sum of the inferred edge lengths of the pendant vertices }
fixed,	{ number of "good" edges in S }
gen_bound,	{ the Δ^2 value for the generating tree }
ok,	{ flag for edge label comparison }
optimal,	{ flag }
$sum\_gs\_in\_S[N]$,	
$sum\_g2s\_in\_S[N]$,	
wrong;	{ number of edges CT gets wrong }

if (*distances*) **then**

  for $i = 1$ **to** $n - 3$ **do** $CTDH\_intedges_i \leftarrow 0$

else if (*use_Hadamard*) **then**

  for $i = 1$ **to** $n - 3$ **do** $CTSH\_intedges_i \leftarrow 0$

else

  for $i = 1$ **to** $n - 3$ **do** $CTSO\_intedges_i \leftarrow 0$

$v_0 \leftarrow 0$

for $i = 1$ **to** $m - 1$ **do**

  $h_i \leftarrow v_i$

  $v_0 \leftarrow v_0 - v_i$

$h_0 \leftarrow -1$

for $i = 1$ **to** $m - 1$ **do** $h_i \leftarrow v_i$

$i \leftarrow 1$

while $(i < m)$ **do**

$h_i \leftarrow -1$
$i \leftarrow 2 \times i$
$h_{m-1} \leftarrow -1$
sort_vector_descending(**h**, **opos**)
$S_1 \leftarrow 1$
$bestS_1 \leftarrow 1$
$sum\_gs\_in\_S_0 \leftarrow 0$
$sum\_g2s\_in\_S_0 \leftarrow 0$
for $i = 1$ **to** $n - 3$ **do**
  $sum\_gs\_in\_S_i \leftarrow sum\_gs\_in\_S_{i-1} + v_{intedges_i}$
  $sum\_g2s\_in\_S_i \leftarrow sum\_g2s\_in\_S_{i-1} + (v_{intedges_i})^2$
$essentials \leftarrow 0$
$i \leftarrow 1$
while $(i < m)$ **do**
  $essentials \leftarrow essentials + v_i$
  $i \leftarrow 2 \times i$
$essentials \leftarrow essentials + v_{m-1}$
$essentials \leftarrow essentials + v_0$
$gen\_bound \leftarrow sum\_g2s\_in\_S_{n-3}$
  $- ((sum\_gs\_in\_S_{n-3} + essentials)^2)/(2n-2)$
{ get greedy tree: }
for $i = 2$ **to** $n - 3$ **do**
  $j \leftarrow i$
  while (compat$(opos(j), i-1, S) = 0$) **do** $j \leftarrow j + 1$
  $S_i \leftarrow j$
  $bestS_i \leftarrow j$
$sum\_gs\_in\_S_0 \leftarrow 0$
$sum\_g2s\_in\_S_0 \leftarrow 0$
for $i = 1$ **to** $n - 3$ **do**
  $sum\_gs\_in\_S_i \leftarrow sum\_gs\_in\_S_{i-1} + v_{opos(S_i)}$
  $sum\_g2s\_in\_S_i \leftarrow sum\_g2s\_in\_S_{i-1} + (v_{opos(S_i)})^2$
$bound \leftarrow sum\_g2s\_in\_S_{n-3} - ((sum\_gs\_in\_S_{n-3} + essentials)^2)/(2n-2)$
$bound \leftarrow gen\_bound$
$optimal \leftarrow 0$
$fixed \leftarrow n - 4$
while $(optimal = 0)$ **do**

$k \leftarrow fixed + 1$
while $((k \leq n-3)$ **and** $(k > 0))$ **do**
$j \leftarrow k$
do
$j \leftarrow j + 1$
if $(k = 0)$ **then** $b \leftarrow -(essentials^2)/(2n-2)$
else
$b \leftarrow sum\_g2s\_in\_S_{k-1} + (n-k-2) \times v^2_{opos(j)}$
$-(sum\_gs\_in\_S_{k-1} + (n-k-2) \times v_{opos(j)} + essentials)^2/(2n-2)$
if $(b \geq bound)$ **then** $y \leftarrow$ compat$(opos(j), k-1, S)$
else $y \leftarrow 0$
while $(b \geq bound)$ **and** $(j < m)$ **and** $(y = 0)$ **and** $(v_{opos(j)} \geq 0)$
if $((b \geq bound)$ **and** $(j < m))$ **then**
$S_k \leftarrow j$
$sum\_gs\_in\_S_k \leftarrow sum\_gs\_in\_S_{k-1} + v_{opos(j)}$
$sum\_g2s\_in\_S_k \leftarrow sum\_g2s\_in\_S_{k-1} + v^2_{opos(j)}$
if $(k \geq n-3)$ **then**
$bound \leftarrow b$
for $x = 1$ **to** $n-3$ **do** $bestS_x \leftarrow S_x$
$k \leftarrow k + 1$
if $((b \leq bound)$ **or** $(k > n-3))$ **then**
$fixed \leftarrow fixed - 1$
for $x = fixed + 1$ **to** $N-1$ **do** $S_x \leftarrow 0$
break
if $(fixed < 0)$ **then** $optimal \leftarrow 1$
if $(distances)$ **then**
for $i = 1$ **to** $n-3$ **do** $CTDH\_intedges_i \leftarrow opos(bestS_i)$
else if $(use\_Hadamard)$ **then**
for $i = 1$ **to** $n-3$ **do** $CTSH\_intedges_i \leftarrow opos(bestS_i)$
else
for $i = 1$ **to** $n-3$ **do** $CTSO\_intedges_i \leftarrow opos(bestS_i)$
$wrong \leftarrow 0$
for $i = 1$ **to** $n-3$ **do**
$ok \leftarrow 0$
for $j = 1$ **to** $n-3$ **do**
if $(intedges_i = opos(bestS_j))$ **then**

```
            ok ← 1
            break
        if (ok = 0) then   wrong ← wrong + 1
    if (distances) then
        CTDH_score_wrong ← CTDH_score_wrong + 1
    else if (use_Hadamard) then
        CTSH_score_wrong ← CTSH_score_wrong + 1
    else
        CTSO_score_wrong ← CTSO_score_wrong + 1
end.
```

Algorithm C.26 : Co(**v**, *distances*, *use_Hadamard*)

{ *distances* = 1 if the input vector is inferred from the distance matrix D, 0 otherwise.
use_Hadamard = 1 if the Hadamard conjugation method is used, 0 otherwise. }

local variables:

b,	{ the best possible value for Ω with the current edge set S }
$bestS[N]$	{ the set of edges which gives the best sum of estimated edge lengths }
bound,	{ the best value found for Ω with a fully resolved tree }
i, j, k, x, y,	{ counters }
$fixed$,	{ number of "good" edges in S }
gen_bound,	{ the Ω value for the generating tree }
ok,	{ flag for edge label comparison }
optimal,	{ flag }
$sum\_gs\_in\_S[N]$,	
wrong,	{ the number of edges Co gets wrong }

$h_0 \leftarrow -v_0$
for $i = 1$ **to** $m-1$ **do** $h_i \leftarrow v_i$
$i \leftarrow 1$
while $(i < m)$ **do**
  $h_i \leftarrow -1$
  $i \leftarrow 2 \times i$
$h_{m-1} \leftarrow -1$
sort_vector_descending(**h**,**opos**)
gen_bound $\leftarrow 0$
for $i = 1$ **to** $n-3$ **do** *gen_bound* $\leftarrow$ *gen_bound* $+ v_{intedges_i}$
if (*distances*) **then**
  for $i = 1$ **to** $n-3$ **do** $CoDH\_intedges_i \leftarrow intedges_i$
else if (*use_Hadamard*) **then**
  for $i = 1$ **to** $n-3$ **do** $CoSH\_intedges_i \leftarrow intedges_i$
else
  for $i = 1$ **to** $n-3$ **do** $CoSO\_intedges_i \leftarrow intedges_i$

```
{ Get greedy tree in terms of positions in opos: }
S_1 ← 1
j ← 2
for i = 2 to n − 3 do
  while (compat(opos(j), i − 1, S) = 0) do  j ← j + 1
  S_i ← j
  bestS_i ← j
sum_gs_in_S_0 ← 0
for i = 1 to n − 3 do  sum_gs_in_S_i ← sum_gs_in_S_{i−1} + v_{opos(S_i)}
bound ← sum_gs_in_S_{n−3}
if (gen_bound < bound) then  bound ← gen_bound
optimal ← 0
fixed ← n − 4
while (optimal = 0) do
  k ← fixed
  arbitrary ← 0
  while (k ≥ 0) do
    j ← k + 1
    do
      j ← j + 1
      if (k = 0) then  b ← v_{opos(j)}
      else b ← sum_gs_in_S_{k−1} + (n − 3 − fixed) × v_{opos(j)}
      if (b ≥ bound) then  y ← compat(opos(j), k − 1, S)
      else y ← 0
    while (b ≥ bound) and (j < m) and (y = 0)
    if ((b ≥ bound) and (j < m)) then
      S_k ← j
      sum_gs_in_S_k ← sum_gs_in_S_{k−1} + v_{opos(j)}
      if ((k = n − 3) and (compare_sets(lastS, S, n − 3) > 0)) then
        for i = 1 to n − 3 do  lastS_i ← S_i
        b ← sum_gs_in_S_k
        bound ← b
        for x = 1 to n − 3 do  bestS_x ← S_x
      k ← k + 1
    if ((b ≤ bound) or (k > n − 3)) then
```

```
            fixed ← fixed − 1
            break
        if (fixed < 0) then optimal ← 1
    if (distances) then
        for i = 1 to n − 3 do CoDH_intedges_i ← opos(bestS_i)
    else if (use_Hadamard) then
        for i = 1 to n − 3 do CoSH_intedges_i ← opos(bestS_i)
    else
        for i = 1 to n − 3 do CoSO_intedges_i ← opos(bestS_i)
    wrong ← 0
    for i = 1 to n − 3 do
        ok ← 0
        for j = 1 to n − 3 do
            if (intedges_i = opos(bestS_j)) then
                ok ← 1
                break
        if (ok = 0) then wrong ← wrong + 1
    if (distances) then
        CoDH_score_wrong ← CoDH_score_wrong + 1
    else if (use_Hadamard) then
        CoSH_score_wrong ← CoSH_score_wrong + 1
    else
        CoSO_score_wrong ← CoSO_score_wrong + 1
end.
```

Algorithm C.27 : set_up_first_tree(A)

{ A is the array into which we shall put the pointers describing the first tree. The first tree is the tree on the three pendant vertices $n-2$, $n-1$ and n. }

local variable: i; { counter }

for $i = 1$ **to** $2n$ **do** $A_i \leftarrow 0$

$A_{n-2} \leftarrow n+1$

$A_{n-1} \leftarrow n+1$

$A_{n-3} \leftarrow n+2$

$A_{n+1} \leftarrow n+2$

$A_{n+2} \leftarrow n$

end.

Algorithm C.28 : add_taxon($taxon, position, A, new\_node$)

{ *taxon* is the label of the pendant vertex which we are adding to the tree described by the array A. *position* is the insertion point of *taxon*, and the label of the new internal vertex thus created is *new_node*. }

$A_{taxon} \leftarrow new\_node$

$A_{new\_node} \leftarrow A_{position}$

$A_{position} \leftarrow new\_node$

end.

Algorithm C.29 : remove_taxon($taxon, position, A, node$)

{ *taxon* is the label of the pendant vertex to be removed, and *node* is the internal vertex which must be removed also. Vertex *position* is directed towards *node* also. }

$A_{position} \leftarrow A_{node}$

$A_{taxon} \leftarrow 0$

$A_{node} \leftarrow 0$

end.

Algorithm C.30 : convert_edges_to_tree($edge\_set, tree\_array$)

{ $edge\_set$ is the set of internal edge labels of a tree T, and $tree\_array$ is the array of pointers which will describe T. }

local variables: i, j; { counters }

for $i = 1$ **to** $n - 1$ **do** $S_i \leftarrow 2^{i-1}$

for $i = 1$ **to** $n - 3$ **do** $S_{n-1+i} \leftarrow edge\_set_i$

{ Sort the edge labels in S in ascending order: }

for $i = n$ **to** $2n - 5$ **do**
  for $j = i + 1$ **to** $2n - 4$ **do**
    if $(S_i > S_j)$ **then** $S_i \longleftrightarrow S_j$

$S_{2n-3} \leftarrow m - 1$

for $i = 1$ **to** $2n - 4$ **do** $tree\_array_i \leftarrow 2n - 2$

for $j = 2n - 3$ **down to** n **do**
  for $i = j - 1$ **down to** 1 **do**
    if $((S_j$ **AND** $S_i) = S_i)$ **then** $tree\_array_i \leftarrow j + 1$

$tree\_array_{2n-2} \leftarrow n$

for $i = 2n - 3$ **down to** $n + 1$ **do** $tree\_array_i \leftarrow tree\_array_{i-1}$

$tree\_array_n \leftarrow 0$

end.

Algorithm C.31 : convert_tree_to_edges(*tree_array*, *edge_set*)

{ *tree_array* is the array of pointers which describe as tree T, and *edge_set* is the set of internal edge labels of T. }

local variables:

$A[N][2]$,	{ matrix of pointers and edge labels }
edges_found,	{ number found so far }
i, j;	{ counters }

for $i = 1$ **to** $N - 1$ **do**

$edge\_set_i \leftarrow 0$

$A_{i,1} \leftarrow 0$

$A_{i,0} \leftarrow 0$

for $i = 1$ **to** $2n - 1$ **do** $A_{i,1} \leftarrow tree\_array_i$

for $i = 1$ **to** n **do** $A_{i,0} \leftarrow 2^{i-1}$

$i \leftarrow 1$

for $edges\_found = 1$ **to** $n - 3$ **do**

while $((A_{i,0} = 0)$ **and** $(i < 2n - 1))$ **do** $i \leftarrow i + 1$

if $(i < 2n)$ **then**

$j \leftarrow i + 1$

while $((A_{j,1} \neq A_{i,1})$ **and** $(j < 2n))$ **do** $j \leftarrow j + 1$

if $(A_{j,0} = 0)$ **then** $i \leftarrow i + 1$

else

$A_{A_{i,1},0} \leftarrow A_{i,0} + A_{j,0}$

$edge\_set_{edges\_found} \leftarrow A_{i,0} + A_{j,0}$

$A_{i,0} \leftarrow A_{j,0} \leftarrow 0$

$edges\_found \leftarrow edges\_found + 1$

$i \leftarrow 1$

end.

Algorithm C.32 : Fitch(A, *nodes*, **v**, *bound*)

{ This function returns the parsimony length of the input tree described by A, up to the size of *bound*. As soon as the length exceeds *bound* then this length is returned.

A is the array of pointers to vertices which describes the tree, and A_0 is the cardinality of A. All edges are directed to the root at taxon n.

nodes is the number of vertices in the tree described by A. **v** is the vector of bipartition frequencies used. *bound* is the maximum desired parsimony length of the tree described by A: if the length is larger than *bound*, we are not interested in the tree anyway. }

local variables:

$i, j, a, b, nodes\_assigned, k$, { counters }

$assigned[N][M]$, { the characters assigned to the internal nodes of the tree }

$Ftree[N]$, { internal storage of the tree described by A }

$length$; { the overall parsimony length of the tree }

for $i = 0$ **to** $2n - 2$ **do**

  for $j = 1$ **to** *num_bips* **do** $assigned_{i,j} \leftarrow 0$

  $Ftree_i \leftarrow A_i$

$k \leftarrow 0$

for $i = 1$ **to** $n - 1$ **do**

  if $(Ftree_i > 0)$ **then**

    for $j = 1$ **to** *num_bips* **do**

      if $(opos(j)$ **AND** $2^{(i-1)})$ **then** $assigned_{i,j} \leftarrow 2$

      else $assigned_{i,j} \leftarrow 1$

    $k \leftarrow k + 1$

for $j = 1$ **to** *num_bips* **do** $assigned_{n,j} \leftarrow 1$

$length \leftarrow 0$

$nodes\_assigned \leftarrow 0$

while $((nodes\_assigned < k - 1)$ **and** $(length \leq bound))$ **do**

  $i \leftarrow n + 1$

  while $((i \leq n + 1)$ **and** $(length \leq bound))$ **do**

```
        a ← 1
        while ((a < nodes) and (Ftree_a ≠ i)) do  a ← a + 1
        b ← a + 1
        while ((b ≤ nodes) and (Ftree_b ≠ i)) do  b ← b + 1
        if ((b ≤ nodes) and (Ftree_a = i) and (Ftree_b = i)
              and (assigned_a,1) and (assigned_b,1)) then
          j ← 1
          while ((j ≤ num_bips) and (length ≤ bound)) do
            assigned_i,j ← assigned_a,j AND assigned_b,j
            if (assigned_i,j = 0) then
              assigned_i,j ← 3
              length ← length + v_opos(j)
            j ← j + 1
          Ftree_a ← 0
          Ftree_b ← 0
          nodes_assigned ← nodes_assigned + 1
      i ← i + 1
    i ← 1
    while ((i ≤ nodes) and (Ftree_i ≠ n)) do  i ← i + 1
    j ← 1
    while ((length ≤ bound) and (j ≤ num_bips)) do
      if ((assigned_i,j AND assigned_n,j = 0)) then
        length ← length + v_opos(j)
    return (length)
  end.
```

Algorithm C.33 : MP($\mathbf{v}, distances, use\_Hadamard$)

{ $\mathbf{v}$ is the input vector of bipartition frequencies, with or without the Hadamard conjugation, determined by *use_Hadamard*.
Another branch and bound method for MP has been given by Penny and Hendy [67]. }

local variables:

$w[M]$,	{ temporary vector, used when sorting }
bound,	{ current shortest length of a tree }
length,	{ current Fitch (minimum) length of the tree }
gen_length,	{ length of generating tree }
taxon,	{ number of the taxon we're adding, initially $n-4$ }
new_node,	{ label of the new internal vertex we create by inserting a taxon }
nodes,	{ number of nodes in the current tree }
optimal,	{ flag to tell when to stop }
insert_pos$[N]$,	{ current position of insertion of each taxon }
i, j,	{ counters }
tree$[N]$,	{ array of pointers to nodes }
num_tied_trees,	{ number found so far with same smallest length: redundant, should be removed }
best_tree$[N]$;	{ best tree found so far, up to ties }

$bound \leftarrow n \times c$
$num\_tied\_trees \leftarrow 1$
convert_edges_to_tree($intedges, tree$)
for $i = 1$ **to** $2n-2$ **do** $best\_tree_i \leftarrow tree_i$
for $i = 2n-1$ **to** $N-1$ **do**
  $tree_i \leftarrow 0$
  $best\_tree_i \leftarrow 0$
for $i = 0$ **to** $m-1$ **do** $w_i \leftarrow v_i$
for $i = 1$ **to** $n-1$ **do**
  $j \leftarrow 2^{i-1}$
  $w_j \leftarrow -1$

```
w_{m-1} ← -1
sort_vector_descending(w, opos)
num_bips ← number_of_bipartitions
nodes ← 2n - 2
new_node ← 2n - 2                { = number of nodes in complete binary tree }
bound ← Fitch(tree, nodes, v, bound)
gen_length ← bound
set_up_first_tree(tree)
taxon ← n - 3                    { begin by including this taxon }
insert_pos_taxon ← n + 1         { insert in positions in decreasing order }
nodes ← n + 2
new_node ← n + 2                 { the name of the new node we'll create by the
                                   insertion }
for i = 1 to n - 4 do  insert_pos_i ← 2n - 2 - i
optimal ← 0
if (gen_length < bound) then  bound ← gen_length
while (optimal = 0) do
  length ← Fitch(tree, nodes, v, bound)
  if (length ≤ bound) then
    if ((taxon = 1) and (length < bound)) then
      for i = 1 to 2n - 2 do  best_tree_i ← tree_i
      bound ← length
    { Now we need to get the next tree: }
    if ((length > bound) or (taxon = 1)) then
      while ((insert_pos_taxon = taxon + 1) and (taxon < n - 2)) do
        remove_taxon(taxon, insert_pos_taxon, tree, tree_taxon)
        taxon ← taxon + 1
      if (taxon = n - 2) then  optimal ← 1
      else
        new_node ← 2n - 1 - taxon
        if (insert_pos_taxon > taxon + 1) then
          remove_taxon(taxon, insert_pos_taxon, tree, tree_taxon)
          do
            insert_pos_taxon ← insert_pos_taxon - 1
          while (tree_{insert_pos_taxon} = 0) and (insert_pos_taxon > taxon + 1)
          add_taxon(taxon, insert_pos_taxon, tree, new_node)
    else
```

```
            taxon ← taxon − 1
            nodes ← new_node ← 2n − 1 − taxon
            insert_pos_taxon ← 2n − 2 − taxon
            do
              insert_pos_taxon ← insert_pos_taxon − 1
            while (tree_insert_pos_taxon = taxon + 1) and (insert_pos_taxon > taxon)
            add_taxon(taxon, insert_pos_taxon, tree, new_node)
          if (taxon = n − 2) then optimal ← 1
      if (distances) then
        convert_tree_to_edges(best_tree, MPDH_intedges)
        i ← compare_sets(MPDH_intedges, intedges, n − 3)
        MPDH_score_i ← MPDH_score_i + 1
      else if (use_Hadamard) then
        convert_tree_to_edges(best_tree, MPSH_intedges)
        i ← compare_sets(MPSH_intedges, intedges, n − 3)
        MPSH_score_i ← MPSH_score_i + 1
      else
        convert_tree_to_edges(best_tree, MPSO_intedges)
        i ← compare_sets(MPSO_intedges, intedges, n − 3)
        MPSO_score_i ← MPSO_score_i + 1
    end.
```

C.4 Main program structure of `sim.c`

Algorithm C.34 : sim.c

local variables:

topnum,	{ denotes the topology number }
mpl,	{ maximum path length descriptor }
mpl_{min}, mpl_{max},	{ range of *mpl* }
max_path_length,	{ upper bound on maximum number of expected character state changes between any two taxa }
ratio,	{ descriptor for (max. internal edge length)/(max. pendant edge length) }
$ratio_{min}$, $ratio_{max}$,	{ range of *ratio* }
er,	{ sequencing error rate descriptor }
er_{min}, er_{max},	{ range of *er* }
amal,	{ proportion of data to come from a second generating tree T_2 }
$amal_{min}$, $amal_{max}$,	{ range of *amal* }
len,	{ sequence length descriptor }
len_{min}, len_{max},	{ range of *len* }
*c*1, *c*2,	{ sequence lengths for generating trees T_1 and T_2 }
run, *i*,	{ counters }
a, *b*,	{ logical flags }
ok_trials;	{ number of trials in which all methods could be used }

read in run-time arguments
for $topnum = topnum_{min}$ **to** $topnum_{max}$ **do**
  for $mpl = mpl_{min}$ **to** mpl_{max} **do**
    max_path_length ← rough_exp(*mpl*) × 0.0035
    for $ratio = ratio_{min}$ **to** $ratio_{max}$ **do**
      α ← rough_exp(*ratio*) × 0.05
      for $er = er_{min}$ **to** er_{max} **do**
        if (*reading_errors*) **then**
          reading_error_rate ← rough_exp(*er*)/10000
        else *reading_error_rate* ← 0
        for $amal = amal_{min}$ **to** $amal_{max}$ **do**

```
for len = len_min to len_max do
  c2 ← amal × c
  c1 ← c − c2
  for run = 1 to tests do
    choose_topology(topnum, intedges)
    random_edge_lengths(intedges, q1)
    HexpH(q1,s)
    sample_bipartitions(c1, reading_error_rate, s, obs)
    if (amal > 0) then
      do
        choose_tree(n, intedges2)
      while compare_sets(intedges, intedges2, n − 3) = 0
      random_edge_lengths(intedges2, q2)
      HexpH(q2, s)
      sample_bipartitions(c2, reading_error_rate, s, obs2)
      for i = 0 to m − 1 do  obs_i ← obs_i + obs2_i
    num_bips ← number_of_bipartitions()
    a ← HlnH(obs, g)
    b ← bipartitions_to_distances(obs)
    if ((a > 0) or (b > 0) or (num_bips = 0)) then
      if (a > 0) then  d_negative ← d_negative + 1
      if (b > 0) then  d_zero ← d_zero + 1
    else
      if (Correct_distances() > 0) then  d_infinite ← d_infinite + 1
      if ( using any Distance Hadamard methods ) then
        get_distancespectrum
      ok_trials ← ok_trials + 1
      Do methods
  print scores
  zero scores
end.
```

C.5 Functions used in `big.c`

Algorithm C.35 : get_bif_times(**output_bif_time**)

{ **output_bif_time** is an array of the bifurcation times of each of the internal vertices of T_G. }

local variables:

i, j, { counters }

$random[N]$, { temporary array of random numbers }

for $i = 1$ **to** $n - 1$ **do** $random_i \leftarrow random$

for $i = 1$ **to** $n - 1$ **do**

  for $j = i + 1$ **to** n **do**

    if $(random_i > random_j)$ **then** $random_i \longleftrightarrow random_j$

$output\_bif\_time_{2n-1} \leftarrow 0$

$random_0 \leftarrow 0$

for $i = 1$ **to** $n - 2$ **do**

  $output\_bif\_time_{2n-1-i} \leftarrow output\_bif\_time_{2n-i}$

    $+ (random_i - random_{i-1})/(i + 1)$

for $i = 2n$ **down to** $n + 1$ **do**

  $output\_bif\_time_i \leftarrow output\_bif\_time_i \times dtf / output\_bif\_time_{n+1}$

for $i = 1$ **to** n **do** $output\_bif\_time_i \leftarrow 1$

end.

Algorithm C.36 : ones_count(z)

local variables: $i, ones$ { counters }

ones $\leftarrow$ 0

for $i = 1$ **to** n **do**

  if $(z[i]$ **AND** $(2^{i-1}))$ **then** $ones \leftarrow ones + 1$

if $(2 \times ones > n)$ **then** $ones \leftarrow n - ones$

return $(ones)$

end.

Algorithm C.37 : get_whole_tree

local variables:

```
  i, j,                          { counters }
  perm[N],                       { temporary array of (1,...,n-1), permuted }
  labels[N],                     { array of permuted taxon labels }
  temp_tree[N2],                 { in pointer form }
  bif_time[N2];                  { bifurcation times of nodes 2n-1,...,n+1 }
get_bif_times(bif_time)
permute(n-1, perm)
permutation_to_tree(perm, tree)
permute(n, labels)
for i = 1 to n do  temp_tree_i ← tree_labels(i)
for i = 1 to n do  tree_i ← temp_tree_i
for i = 1 to 2n-2 do
  for j = i+1 to 2n-1 do
    if (tree_i = j) then
      mean_edge_length_i,j ← (bif_time_i - bif_time_j) × ot
case sampling_method of
  1:                             { uniform distribution }
    for i = 1 to 2n-2 do
      for j = i+1 to 2n-1 do
        if (tree_i = j) then
          do
            edge_length_i,j ← sample_uniform(mean_edge_length_i,j,
                                  var × mean_edge_length_i,j)
          while edge_length_i,j ≤ 0
    break
  2:                             { normal distribution }
    for i = 1 to 2n-2 do
      for j = i+1 to 2n-1 do
        if (tree_i = j) then
          do
            edge_length_i,j ← sample_normal(mean_edge_length_i,j,
                                  var × mean_edge_length_i,j)
```

```
            while edge_length_{i,j} ≤ 0
        break
    3:                              { log-normal distribution }
        for i = 1 to 2n − 2 do
          for j = i + 1 to 2n − 1 do
            if (tree_i = j) then
              do
                edge_length_{i,j} ← sample_log-normal(mean_edge_length_{i,j},
                                    var × mean_edge_length_{i,j})
              while edge_length_{i,j} ≤ 0
        break
    default: break
end.
```

Algorithm C.38 : grow_data

{ **EV** is the first matrix in the diagonalization of **TM**, and **diag** is the set of four eigenvalues of **TM**. $diag_1$ is always 1. }

local variables:

i, j, k,	{ counters }
r,	{ temporary real number }
edge_prob[$N2$][$N2$],	{ matrix of edge lengths }
Diag,	{ 4×4 matrix whose diagonal entries are the entries of **diag** and the off-diagonal entries of which are zero. }
tempM,	{ temporary 4×4 matrix }
M,	{ transition matrix for an edge }
C;	{ matrix of sorted components of **M**, to speed up sampling. }

for $i = 1$ **to** 4 **do**

  for $j = 1$ **to** 4 **do** $Diag_{i,j} \leftarrow 0$

$Diag_{1,1} \leftarrow 1$

for $i = 2$ **to** 4 **do** $Diag_{i,i} \leftarrow \exp(-diag_i)$

{ fix the root distribution: }

$j \leftarrow 2n - 1$

for $i = 1$ **to** $c \times \pi_A$ **do**

  $character_{j,i} \leftarrow 1$

for $i = c \times \pi_A + 1$ **to** $c \times (\pi_A + \pi_C)$ **do**

  $character_{j,i} \leftarrow 2$

for $i = c \times (\pi_A + \pi_C) + 1$ **to** $c \times (\pi_A + \pi_C + \pi_G)$ **do**

  $character_{j,i} \leftarrow 4$

for $i = c \times (\pi_A + \pi_C + \pi_G) + 1$ **to** c **do**

  $character_{j,i} \leftarrow 8$

for $k = 2n - 1$ **down to** $n + 1$ **do**

  for $i = k - 1$ **down to** 1 **do**

    if $(tree_i = k)$ **then** { i.e., there exists the edge $\{i, k\}$ }

      { Find the product **EV**(**Diag**^{t})(**invEV**): this gives the transition matrix for the edge. }

      for $j = 2$ **to** 4 **do** { note that $Diag_{1,1} = 1$ }

$Diag_{j,j} \leftarrow exp(-diag_j \times edge\_length_{i,tree(i)})$

tempM $\leftarrow$ **Diag** $\times$ **invEV**

M $\leftarrow$ **EV** $\times$ **tempM**

{ Rank the components of **M** into **C** to speed up the sampling process: }

$C_{1,1} \leftarrow M_{1,1}$; $C_{1,2} \leftarrow C_{1,1} + M_{1,2}$; $C_{1,3} \leftarrow C_{1,2} + M_{1,3}$

$C_{2,2} \leftarrow M_{2,2}$; $C_{2,1} \leftarrow C_{2,2} + M_{2,1}$; $C_{2,3} \leftarrow C_{2,1} + M_{2,3}$

$C_{3,3} \leftarrow M_{3,3}$; $C_{3,1} \leftarrow C_{3,3} + M_{3,1}$; $C_{3,2} \leftarrow C_{3,1} + M_{3,2}$

$C_{4,4} \leftarrow M_{4,4}$; $C_{4,1} \leftarrow C_{4,4} + M_{4,1}$; $C_{4,2} \leftarrow C_{4,1} + M_{4,2}$

$C_{1,4} \leftarrow C_{2,4} \leftarrow C_{3,4} \leftarrow C_{4,3} \leftarrow 1$

for $j = 1$ **to** c **do**

  $r \leftarrow random$

  case $character_{k,j}$ **of**

    1:

      if $(r < C_{1,1})$ **then** $character_{i,j} \leftarrow 1$

      else if $(r < C_{1,2})$ **then** $character_{i,j} \leftarrow 2$

      else if $(r < C_{1,3})$ **then** $character_{i,j} \leftarrow 4$

      else $character_{i,j} \leftarrow 8$

      break

    2:

      if $(r < C_{2,2})$ **then** $character_{i,j} \leftarrow 2$

      else if $(r < C_{2,1})$ **then** $character_{i,j} \leftarrow 1$

      else if $(r < C_{2,3})$ **then** $character_{i,j} \leftarrow 4$

      else $character_{i,j} \leftarrow 8$

      break

    4:

      if $(r < C_{3,3})$ **then** $character_{i,j} \leftarrow 4$

      else if $(r < C_{3,1})$ **then** $character_{i,j} \leftarrow 1$

      else if $(r < C_{3,2})$ **then** $character_{i,j} \leftarrow 2$

      else $character_{i,j} \leftarrow 8$

      break

    8:

      if $(r < C_{4,4})$ **then** $character_{i,j} \leftarrow 8$

      else if $(r < C_{4,1})$ **then** $character_{i,j} \leftarrow 1$

      else if $(r < C_{4,2})$ **then** $character_{i,j} \leftarrow 2$

      else $character_{i,j} \leftarrow 4$

      break

default: **break**

end.

Algorithm C.39 : generalJC

{ This is a generalised version of the Jukes-Cantor distance correction, allowing for different character state frequencies between two taxa [91]. }

local variables:

$i, j, k,$	{ counters }
$b,$	{ temporary real number }
$f[N][4],$	{ observed frequency of each of the character states in the sequences }
$td[4];$	{ temporary array of real numbers }

{ Obtain observed frequencies: }

for $i = 1$ **to** $n - 1$ **do**
  for $j = 1$ **to** 4 **do** $f_{i,j} \leftarrow 0$
  for $j = 1$ **to** c **do**
    if $(character_{i,j} = 1)$ **then** $f_{i,1} \leftarrow f_{i,1} + 1$
    else if $(character_{i,j} = 2)$ **then** $f_{i,2} \leftarrow f_{i,2} + 1$
    else if $(character_{i,j} = 4)$ **then** $f_{i,3} \leftarrow f_{i,3} + 1$
    else $f_{i,4} \leftarrow f_{i,4} + 1$
  for $j = 1$ **to** 4 **do** $f_{i,j} \leftarrow f_{i,j}/c$
for $i = 1$ **to** $n - 1$ **do**
  for $j = i + 1$ **to** n **do**
    for $k = 1$ **to** 4 **do** $td_k \leftarrow (f_{i,k} + f_{j,k})/2$
    $b \leftarrow 1 - td_1^2 - td_2^2 - td_3^2 - td_4^2$
    if $(D_{i,j} < b)$ **then**
      $D_{i,j} \leftarrow -b \times \log(1 - D_{i,j}/b)$
      $D_{j,i} \leftarrow D_{i,j}$
    else
      $D_{i,j} \leftarrow BIG$
      $D_{j,i} \leftarrow D_{i,j}$

end.

C.6 Main program structure of `big.c`

Algorithm C.40 : big.c

local variables:

n,	{ number of taxa }
n_{min}, n_{max},	{ range of n }
ot,	{ expected overall 'time' from root of tree to pendant vertices }
ot_{min}, ot_{max},	{ range of ot }
dtf	{ 'divergence time factor', the proportal time at which the last bifurcation event is expected to occur }
dtf_{min}, dtf_{max},	{ range of dtf }
var,	{ descriptor for variance of edge length probability distribution }
var_{min}, var_{max},	{ range of var }
len,	{ sequence length descriptor }
len_{min}, len_{max},	{ range of len }
c,	{ sequence length }
run;	{ counter }

read in run-time arguments
invEV $\leftarrow$ **EV**$^{-1}$
$detEV \leftarrow det(EV)$
for $n = n_{min}$ **to** n_{max} in steps of n_{incr} **do**
  for $ot = ot_{min}$ **to** ot_{max} in steps of ot_{incr} **do**
    for $dtf = dtf_{min}$ **to** dtf_{max} in steps of dtf_{incr} **do**
      for $var = var_{min}$ **to** var_{max} in steps of var_{incr} **do**
        for $len = len_{min}$ **to** len_{max} in steps of len_{incr} **do**
          $c \leftarrow$ rough_exp(len)
          for $run = 1$ **to** $test$ **do**

```
            get_whole_tree
            convert_tree_to_edges(tree, intedges)
            grow_data
            sequences_to_distances
            if (stdJC) then JukesCantor
            else if (genJC) then generalJC
            Do methods
          print scores
          zero scores
end.
```

Appendix D

Dangers of Computer Simulation

> Never worry about theory as long as the machinery does what it's supposed to do.
>
> [*Robert A. Heinlein*]

It seems appropriate that some of the problems and caveats that I have encountered while conducting these experiments should be described. There are a great many simulation studies in the literature, which have already been cited in the body of this thesis, but little is said about the computations themselves; for example, how ties were resolved, the effect of rounding error, the amount of time required, and so on.

Hence in this section I outline some of these potential problems, and some of their remedies.

D.1 Tied Decisions

Problem: Consider a clustering method 'A', which works by clustering together groups of taxa until there is just one cluster. It checks, at each clustering stage, all the potential pairs of clusters to see which is the "best" in some sense. This equates to evaluating some function on the pairs and maximising it.

However, if there is more than one pair that maximises the function, some choice has to be made between them.

Most program loops to do this will take the form shown in Algorithm D.1, below.

Algorithm D.1 : Typical loop to find $\{i,j\}$ to maximise $f(i,j)$

```
for i = 1 to n − 1 do          { n is the number of clusters }
for j = i + 1 to n do
  evaluate f(i,j)
  if (f(i,j) is better than anything previously found) then
    best_pair ← {i,j}
    best_function_value ← f(i,j)
end.
```

So if we had decided to accept only those pairs $\{i,j\}$ for which $f(i,j)$ was strictly greater than our previous best, in the event of a tie we would keep the first pair we found.

If on the other hand we accepted as our best option a pair $\{i,j\}$ for which $f(i,j)$ *equals* our previous best, in the event of a tie we would end up with the *last* pair.

Another alternative might be that we choose arbitrarily between them using a random number generator, but this has drawbacks too: it introduces indeterminacy into the method.

One way around the problem for clustering methods may be to build up trees in parallel, being abandoned if they are found to give 'worse' solutions than others. This has not been implemented in this study.

The order in which the taxa are labelled obviously has a strong effect on the resulting tree, particularly for those input data sets which can take only a small number of values, and can give spurious results:

Consider the generating tree T in Figure D.1, in which the pendant vertices are labelled $1, 2, \ldots, 8$. Suppose the loop used for the clustering methods was of the form given above, taking the first pair $\{i,j\}$ of taxa if subsequent pairs do not give strictly greater values for f. Suppose also that the sequence lengths were small, so that the number of possible distance values between any two taxa was also small. In this case tied decisions would occur comparatively often. The first pair upon which f would be evaluated is $\{1,2\}$, which would have a finite probability of giving

at least as high a value of f as any other subsequently considered pair, so there would be a definite bias toward producing the correct initial cluster $\{1, 2\}$.

This effect did give biassed results, as in the first implementation of the simulation program `sim.c` the generating trees had fixed labelling schemes.

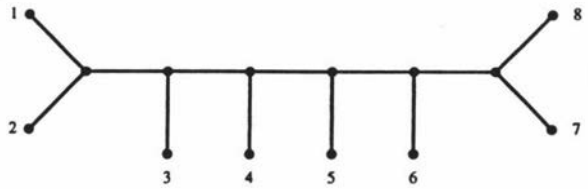


Figure D.1: An example labelled tree

Remedy: The effect was countered by pseudo-randomly permuting the labels of the pendant vertices before constructing the generating tree.

D.2 Rounding error

Problem: Computers do not know that two numbers are theoretically equal: if the computational paths to two *theoretically identical* floating-point numbers are different, rounding error is almost certain to render them different.

Suppose method M works by counting up the number of times $d_{i,j} + d_{k,l}$ is strictly less than $d_{i,k} + d_{j,l}$ and clusters together $\{i, j\}$ if it maximises this count. Whereas $d_{w,x} + d_{y,z}$ may equal $d_{w,y} + d_{x,z}$, in all likelihood they will be different, thus spuriously affecting the count.

For example, having proved the two methods NJ and CTDH were identical when $n = 4$, it was surprising to discover that they performed differently. This was because of rounding error; the actual calculations were different (though theoretically equivalent), so came up with different numbers.

Remedy: One suggested stratagem for dealing with this problem is to use a somewhat looser definition of equality, in the context of computer calculations. This would have to take into account the maximum size, say δ of the rounding errors, and call two numbers 'equal' if their difference is less than δ.

In this study however the technique was simply to use the highest precision available: 48-bit floating point numbers. Though memory requirements are concommitantly larger, this is not a major problem in most cases. (Note that using

variables of this type, "`double`", actually requires *less* computing time for basic mathematical operations than is required for the 24-bit floating-point numbers of type "`float`". This is because the variables of type `float` are converted anyway to type `double` before the operations are carried out.)

D.3 Programming errors

Problem: It is not always easy to detect program errors. One reason is that it is easily possible for a program to perform "as expected", and thus close inspection of the operation of the program is not seen as necessary. This close inspection is not easy either: when the programs are of the order of 100 kbytes long, detecting 'bugs' is difficult, and when the output data sets are of the order of 100 kbytes to 1000 kbytes in size, closely investigating and checking them is in many cases impossible.

Also, the results of simulation experiments like this one are published without explicit descriptions of their implementation, which makes checking of published results by other researchers impossible.

Remedy: One obvious remedy is to take ever more care when using computer programs, checking their operation is correct, whether or not they are operating 'as we expect'. This is of course time-consuming, but must be done, even when the results seem 'normal'.

Another way in which simulation experiments can be made more reliable is for the programs to be explicitly described, in an algorithmic form, rather than, as I have seen in more than one case, where the 'algorithm' is 'described' by a single example [93]. For this reason the pseudocode for the algorithms used in this study has been included.

Bibliography

[1] D. Adams. The Hitch-hikers' Guide to the Galaxy. Original radio series, 1978.

[2] D. Applequist, C. Depuy, and K. L. Rinehart. *Introduction to Organic Chemistry.* John Wiley and Sons, 3rd edition, 1982.

[3] P. Astolfi, K. K. Kidd, and L. L. Cavalli-Svorza. A comparison of methods for reconstructing evolutionary trees. *Systematic Zoology*, 30(2):156–169, 1981.

[4] M. J. Bishop and E. A. Thompson. Maximum likelihood alignment of DNA sequences. *Journal of Molecular Biology*, 190:159–165, 1986.

[5] E. J. Borowski and J. M. Borwein. *Dictionary of Mathematics.* Collins, Glasgow, 1989.

[6] R. J. Britten. Rates of DNA sequence evolution differ between taxonomic groups. *Science*, 231:1393–1398, 1986.

[7] P. Buneman. The recovery of trees from measures of dissimilarity. In F. R. Hodson, D. G. Kendall, and P. Tăutu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1971.

[8] M. Carter, M. Hendy, D. Penny, L. A. Székely, and N. C. Wormald. On the distribution of lengths of evolutionary trees. *SIAM Journal of Discrete Mathematics*, 3(1):38–47, 1990.

[9] L. L. Cavalli-Svorza and A. W. F. Edwards. Phylogenetic analysis: models and estimation procedures. *American Journal of Human Genetics*, 19:233–257, 1967.

[10] J. A. Cavender. Taxonomy with confidence. *Mathematical Biosciences*, 40:271–280, 1978.

[11] M. A. Charleston. A beginner's guide to computer simulation of phylogenetic methods. *Frontiers in Molecular Evolution, Thredbo Alpine Hotel, NSW, Australia, 1-3 February*, 1993.

[12] M. A. Charleston, M. D. Hendy, and D. Penny. Neighbor-joining uses the optimal weight for net divergence. *Molecular Phylogenetics and Evolution*, 2(1):6–12, 1993.

[13] M. A. Charleston, M. D. Hendy, and D. Penny. Effects of sequence length, tree topology and number of taxa on the performance of phylogenetic methods. *Computational Biology*, 1994 (in press).

[14] M. Chase. The molecular systematics of seed plants. *Robertson Symposium: Frontiers of Molecular Evolution, ANU, Canberra, 27-29 January*, 1993.

[15] A. G. Clark and T. S. Whittam. Sequencing errors and molecular evolutionary analysis. *Molecular Biology and Evolution*, 9(4):744–752, 1992.

[16] J. Czelusniak, M. Goodman, N. D. Moncrieff, and S. M. Kehoe. Maximum parsimony approach to construction of evolutionary trees from aligned homologous sequences. *Methods in Enzymology*, 183:601–615, 1990.

[17] G. De Soete. Tree representations of proximity data by least squares methods. In H. H. Bock, editor, *Classification and Related Methods of Data Analysis*, pages 147–156. Elsevier Science Publications B. V. (North-Holland), 1988.

[18] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Technical Report 89.06.011, IBM Heidelberg Scientific Centre, IBM, Heidelberg, 1991.

[19] A. W. F. Edwards. Mathematical approaches to the study of human evolution. In F. R. Hodson, D. G. Kendall, and P. Tăutu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 347–355. Edinburgh University Press, 1971.

[20] B. Efron. The jacknife, the bootstrap and other resampling plans. *Society for Industrial and Applied Mathematics, Philadelphia*, 1982.

[21] J. S. Farris. Estimating phylogenetic trees from distance matrices. *The American Naturalist*, 106(951):645–668, 1972.

[22] J. Felsenstein. Cases in which parsimony and compatibility methods will be positively misleading. *Systematic Zoology*, 27:401–410, 1978.

[23] J. Felsenstein. Alternative methods of phylogenetic inference and their interrelationship. *Systematic Zoology*, 28(1):49–62, 1979.

[24] J. Felsenstein. Distance methods for inferring phylogenies: a justification. *Evolution*, 38(1):16–24, 1984.

[25] J. Felsenstein. Phylogenies from molecular sequences: Inference and reliability. *Annual Review of Genetics*, 22:521–565, 1988.

[26] W. M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

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

[28] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, 2nd edition, 1987.

[29] J. B. Fraleigh. *A First Course in Abstract Algebra.* Addison-Wesley Publishing Company, 3rd edition, 1982. (page 160).

[30] K. Fukami and Y. Tateno. On the maximum likelihood methods for estimating molecular trees: Uniqueness of the likelihood point. *Journal of Molecular Evolution*, 28:460–464, 1989.

[31] J. H. Gillespie. The molecular clock may be an episodic clock. *Proceedings of the National Academy of Sciences, USA*, 81:8009–8013, December 1984.

[32] F. Glover. Tabu search - part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[33] N. Goldman. Maximum likelihood inference of phylogenetic trees, with special reference to a Poisson process model of DNA substitution and to parsimony analysis. *Systematic Zoology*, 39(4):345–361, 1990.

[34] R. L. Graham and L. R. Foulds. Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Mathematical Biosciences*, 60:133–142, 1982.

[35] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993.

[36] E. F. Harding. The probabilities of rooted tree-shapes generated by random bifurcation. *Advances in Applied Probability*, 3:44–77, 1971.

[37] J. Hartigan. Representation of similarity matrices by trees. *Journal of the American Statistical Association*, 62:1140–1158, 1967.

[38] S. B. Hedges, S. Kumar, K. Tamura, and M. Stoneking. Human origins and analysis of mitochondrial DNA sequences. *Science*, 255:737–739, feb 1992.

[39] M. D. Hendy. The relationship between simple evolutionary tree models and observable sequence data. *Systematic Zoology*, 38(4):310–321, 1989.

[40] M. D. Hendy. A combinatorial description of the closest tree algorithm for finding evolutionary trees. *Discrete Mathematics*, 96:51–58, 1991.

[41] M. D. Hendy and M. A. Charleston. Hadamard conjugation: A versatile tool for modelling sequence evolution. *New Zealand Journal of Botany*, 31:231–237, 1993.

[42] M. D. Hendy, C. H. C. Little, and D. Penny. Comparing trees with pendant vertices labelled. *SIAM Journal of Applied Mathematics*, 44(5):1054–1065, oct 1984.

[43] M. D. Hendy and D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, pages 277–290, 1982.

[44] M. D. Hendy and D. Penny. Cladograms should be called trees. *Systematic Zoology*, 33(2):245–247, 1984.

[45] M. D. Hendy and D. Penny. Spectral analysis of phylogenetic data. *Journal of Classification*, 10:5–23, 1993.

[46] M. D. Hendy, D. Penny, and M. A. Steel. A discrete Fourier analysis for evolutionary trees. *Proceedings of the National Academy of Sciences*, 1994 (in press).

[47] M. D. Hendy, M. A. Steel, and D. Penny. Invertible models of sequence evolution. April, 1993 (unpublished).

[48] J. P. Huelsenbeck and D. M. Hillis. Success of phylogenetic methods in the four-taxon case. *Systematic Biology*, 42(3):247–264, 1993.

[49] A. L. Hughes and M. Nei. Evolutionary relationships of the classes of major histocompatibility complex genes. *Immunogenetics*, 37:337–346, 1993.

[50] L. Jin and M. Nei. Relative efficiencies of the maximum-parsimony and distance-matrix methods of phylogeny construction for restriction data. *Molecular Biology and Evolution*, 8(3):356–365, 1991.

[51] T. H. Jukes and C. R. Cantor. *Mammalian protein metabolism*, chapter 1: Evolution of protein molecules, pages 21–132. Academic Press, New York, 1969.

[52] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith. Solving the graphical Steiner tree problem using genetic algorithms. *Journal of the Operational Research Society*, 44(4):397–406, 1993.

[53] K. K. Kidd and L. L. Cavalli-Svorza. Number of characters examined and error in reconstructing evolutionary trees. In F. R. Hodson, D. G. Kendall, and P. Tăutu, editors, *Mathematics in the Archaelogical and Historical Sciences*, pages 335–346. Edinburgh University Press, 1971.

[54] W.-H. Li. Simple method for constructing phylogenetic trees from distance matrices. *Proceedings of the National Academy of Sciences of the U.S.A.*, 78(2):1085–1089, feb 1981.

[55] W.-H. Li and M. Tanimura. The molecular clock runs more slowly in man than in apes and monkeys. 326:93–96, 1987. Nature, 5th March 1987.

[56] W.-H. Li, K. H. Wolfe, J. Sourdis, and P. M. Sharp. Reconstruction of phylogenetic trees and estimation of divergence times under nonconstant rates of

evolution. *Cold Spring Harbor Symposia on Quantitative Biology*, 52:847–856, 1987.

[57] R. Mott. Maximum-likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores. *Bulletin of Mathematical Biology*, 1:59–75, 1992.

[58] M. Nei. Relative efficiencies of different tree-making methods for molecular data. In M.M. Miyamoto and J. Cracraft, editors, *Phylogenetic Analysis of DNA Sequences*, pages 90–128. Oxford University Press, 1991.

[59] M. Nei. Personal communication. Canberra, Australia, January 1993.

[60] M. Nei and A. Rzhetsky. *Evolution of MHC genes*, chapter 2: Reconstruction of phylogenetic trees and evolution of major histocompatibility complex genes, pages 13–27. Springer-Verlag, Heidelberg, 1991.

[61] S. Pääbo. Ancient DNA, human origins, molecular systematics: A conference in memory of the scientific work of allan c. wilson. Wellington, New Zealand, August 1992.

[62] A. M. Paterson, R. D. Gray, and G. P. Wallis. Parasites, petrels and penguins: Does louse presence reflect seabird phylogeny? *International Journal for Parasitology*, 23(4):515–526, 1993.

[63] D. Penny, M. A. Charleston, P. L. Lockhart, and P. J. Waddell. Phylogeny stretched to its limits. *US/NZ Workshop on Thermophiles, University of Waikato, 8th-11th December*, 1993.

[64] D. Penny, L. R. Foulds, and M. D. Hendy. Testing the theory of evolution by comparing phylogenetic trees constructed from five different protein sequences. *Nature*, 297(5863):197–200, may 1982.

[65] D. Penny, M. D. Hendy, and M. A. Steel. Progress with methods for constructing evolutionary trees. *Trends in Ecology and Evolution*, 7:73–79, 1991.

[66] D. Penny, M. D. Hendy, E. A. Zimmer, and R. K. Hamby. Trees from sequences: Panacea or Pandora's Box? *Australian Systematic Botany*, 3:21–38, August 1990.

[67] D. Penny and M.D. Hendy. Turbotree: a fast algorithm for minimal trees. *CABIOS*, 3(3):183–187, 1987.

[68] D. Penny and M. A. Steel. Combinatorial optimization: The great deluge algorithm and evolutionary trees. Preprint, October, 1992.

[69] K. K. Popper. *Objective knowledge: An evolutionary approach.* Oxford University Press, Oxford, England, 1972.

[70] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, second edition, 1992.

[71] M. Richards. Personal communication. Palmerston North, New Zealand, April 1993.

[72] D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[73] F. Rodríguez, J. L. Oliver, A. Marín, and J. R. Medina. The general stochastic model of nucleotide substitution. *Journal of Theoretical Biology*, 142:485–501, 1990.

[74] A. Rzhetsky and M. Nei. A simple method for estimating and testing minimum-evolution trees. *Molecular Biology and Evolution*, 9(5):945–967, 1992.

[75] N. Saitou and T. Imanishi. Relative efficiencies of the Fitch-Margoliash, maximum parsimony, maximum-likelihood, minimum-evolution, and neighbor-joining methods of phylogenetic tree construction in obtaining the correct tree. *Molecular Biology and Evolution*, 6(5):514–525, 1989.

[76] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[77] S. Sattath and A. Tversky. Additive similarity trees. *Psychometrika*, 42(3):319–345, 1977.

[78] G. F. Simmons. *Introduction to Topology and Modern Analysis.* McGraw-Hill, 1963.

[79] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 28:1409–1438, 1958.

[80] J. Sourdis and M. Nei. Relative efficiencies of the maximum parsimony and distance-matrix methods in obtaining the correct phylogenetic tree. *Molecular Biology and Evolution*, 5(3):298–311, 1988.

[81] M. Steel and M. Charleston. Five surprising properties of parsimoniously coloured trees. *Bulletin of Mathematical Biology*, 1994.

[82] M. A. Steel. Distribution of the symmetric tree difference metric on phylogenetic trees. *SIAM Journal of Discrete Mathematics*, 1:541–551, 1988.

[83] M. A. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

[84] M. A. Steel. Personal communication. Palmerston North, New Zealand, November 1992.

[85] M. A. Steel. Distributions on bicoloured binary trees arising from the principle of parsimony. *Discrete Applied Mathematics*, 41:245–261, 1993.

[86] M. A. Steel. The maximum likelihood point for a phylogenetic tree is not unique. *Systematic Biology*, 1993. (submitted).

[87] M. A. Steel, M. D. Hendy, and D. Penny. Loss of information in genetic distances. *Nature*, 336:118, nov 1988.

[88] M. A. Steel, M. D. Hendy, and D. Penny. Parsimony can be consistent! *Systematic Biology*, 1993. (in press).

[89] J. A. Studier and K. J. Keppler. A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular Biology and Evolution*, 5(6):729–731, 1988.

[90] D. L. Swofford. Phylogenetic analysis using parsimony (PAUP), version 3.0.

[91] D. L. Swofford and G. J. Olsen. Phylogeny reconstruction. In David M. Hillis and C. Moritz, editors, *Molecular Systematics*, chapter 11, pages 411–501. Sinauer Associates, Sunderland, 1990.

[92] F. Tajima and M. Nei. Estimation of evolutionary distance between nucleotide sequences. *Molecular Biology and Evolution*, 1(3):269–285, 1984.

[93] Y. Tateno, M. Nei, and F. Tajima. Accuracy of estimated phylogenetic trees from molecular data 1: Distantly related species. *Journal of Molecular Evolution*, 18:387–404, 1982.

[94] J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: An improved likelihood model of sequence alignment. *Journal of Molecular Evolution*, 34:3–16, 1992.

[95] P. J. M. Van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* Reidel, Boston, 1987.

[96] L. M. Vigilant, M. Stoneking, H. Harpening, K. Hawkes, and A. C. Wilson. African populations and the evolution of human mitochondrial DNA. *Science*, 253:1503–1507, 1991.

[97] P. J. Waddell, D. Penny, M. D. Hendy, and G. Arnold. The sampling distributions and covariance matrix of phylogenetic spectra. *Molecular Biology and Evolution*, 1993. (in press).

[98] H. T. Wareham. On the computational complexity of inferring evolutionary trees. Master's thesis, Department of Computer Science, Memorial University of Newfoundland, St. John's, NF, Canada, A1C 5S7, March 1993.

[99] M. S. Waterman, J. Joyce, and M. Eggert. Computer alignment of sequences. In M. M. Miyamoto and J. Cracraft, editors, *Phylogenetic analysis of DNA sequences*, pages 59–72. Oxford University Press, 1991.

[100] C.-I. Wu and W.-H. Li. Evidence for higher rates of nucleotide substitution in rodents than in man. *Proceedings of the National Academy of Sciences*, 82:1741–1745, 1985.

Index

T_G (generating tree), 13, 53, 56, 67
ϵ (minimum edge length), 54, 56
σ (maximum path length), 54, 56
a (maximum internal edge length), 56
b (maximum pendant edge length), 56
c (sequence length), 10, 153
$d(T)$ (diameter of tree T), 56
n (number of taxa), 10, 154
r (ratio of maximum internal to maximum pendant edge length), 54, 56
s' (observed bipartition spectrum), 53
D (distance matrix), 23, 53
q (edge length spectrum), 53
s (expected bipartition spectrum), 53

a.t.e.l. (all trees equally likely), 54, 87
accuracy, 18
    definition of, 18
additive tree, 18
ancestral sequence, 69
ancient DNA, 21

bifurcation, 71
bipartition
    definition of, 8
bipartition spectrum, 30
    expected, 57, 62
    sampling from, 58, 59
bootstrap, 41
branch and bound, 29–31
    savings from, 39

Cavender model, 14
character, 10
character sequence, 10
character state, 10
city-block distance, 30
cladogram, 7
closest tree method (CT), 2, 22, 28
cluster, 12
    size of, 12
clustering algorithm, 23
Co, 22, 30, 31, 46
    complexity of, 30, 47
    consistency of, 47, 61
CoDH, 47
    consistency of, 61
colour, 10
compatibility method
    in maximum parsimony, 2
compatibility method (Co), 2, 22, 29, 46
compatible
    definition of, 8
complexity, 31, 41, 47
    definition of, 9
computers, 49
connectedness, 6

nsistency, 18, 25, 27, 47, 79
  definition of, 18
›SH, 47
  consistency of, 61
›SO, 47
  consistency of, 61
Γ, 22, 28, 31
  complexity of, 29
ΓDH
  consistency of, 61
ΓSH
  consistency of, 61
ΓSO
  consistency of, 61
cle, 6

ta
  contaminated, 21, 126
  random error in, 21
  sampling error in, 18, 50, 51
gree, 6
stance matrix (**D**), 23, 53
  calculation from bipartition spectrum, 60
  calculation of, 205
stance spectrum, 44, 157
  calculation of, 44, 208
stance Wagner method, 160
NA, 4, 21

ge, 6
  degree of, 6
  depth of, 5, 148
  incidence, 6
  inferring length of, 60
  length of, 10, 53, 153
  loop, 6
  negative length, 11
  pendant, 6
edge length, 76
  choice of, 57
  maximum internal (a), 56
  maximum pendant (b), 56
  probability distribution of, 54, 56, 57, 91, 157
  undefined, 83
edges
  compatible, 9, 29, 202
efficiency, 18
  definition of, 18
  statistical use of, 19
Euclidean distance, 30
evolution
  different rates of, 25, 53
evolutionary model, 14, 49
  Cavender, 14
  contradictions with, 50
  deviations from, 50
  inadequate, 50
  Jukes-Cantor, 14, 31
  Kimura, 14
  violation of, 50
evolutionary tree, 4
exhaustive search, 28

F-M methods
  complexity of, 41
Faith's method, 160
falsifiability, 18, 131
  definition of, 20
Farris' method, 160

Fitch's algorithm, 30, 37, 237
complexity of, 31
Fitch-Margoliash method, 160
Fitch-Margoliash methods, 39
fossils, 21

generating tree (T_G), 13
graph
connected, 6
cycle in, 6
definition of, 6

Hadamard conjugation, 11, 28, 29, 31, 61
algorithm, 197
complexity of, 57, 59
complexity of with four colours, 58
definition of, 58
Hamming distance, 11
heuristic, 23
heuristic search, 28, 165
great deluge, 166
hill-climbing, 166
hitch-hiking, 166
simulated annealing, 166
tabu, 166
hill-climbing, 30

i.i.d., 25, 50
information
loss of, 4

Jukes-Cantor correction, 50, 60, 207, 249
Jukes-Cantor model, 14, 31

loop, 6

maximum likelihood, 39
maximum parsimony method (MP), 22, 30
maximum path length (σ), 54, 56, 71
metric, 76
minimum edge length (ϵ), 54, 56
molecular clock, 18, 21, 25, 26
definition of, 11
monophyly, 12
MP, 22, 30
complexity of, 31
consistency of, 30, 61
MPDH
consistency of, 61
MPSH, 31
consistency of, 31, 61
MPSO, 31
consistency of, 31, 61, 81, 107
multiple changes
correction for, 11, 12, 50, 60, 207, 249

nearest neighbour, 13
neighbour-joining method (NJ), 3, 22, 27
neighbouring pair, 13, 23
neighbourliness (ST), 3
neighbourliness method (ST), 22, 26
net divergence, 27
NJ, 22, 27
best weighting for net divergence, 27
complexity of, 28
consistency of, 27, 28
NJa, 48

complexity of, 48
consistency of, 48
.e, 6
se
pink, 50–52, 126, 132, 133, 159
white, 21, 50, 51, 115, 132, 133, 158
m, 30
leotide, 10

-to-one, 171
o, 171
imality, 23
global, 23
local, 23

simony length, 37
tition distance, 76
h, 6
length of, 6
R, 4
'logenetic methods, 43
agreement between, 76, 78, 158
classes of, 21, 43
consistency of, 44
constructive, 23
desirable characteristics of, 1, 18
distance-based, 21
finding more, 43
number of, 43
search, 28
sequence-based, 21
ylogeny
generating, 5
non tree-like, 20
reconstruction, 4
Poisson process, 50
polymerase chain reaction (PCR), 4
power, 18

random number generation, 57
recombination, 20
representation of findings, 76
RNA, 4, 21
robustness, 18
definition of, 20
root distribution, 69
rooted tree
choice of, 67
example of, 72

sampling error, 18, 25, 50, 51, 78, 154
search
exhaustive, 2, 28
heuristic, 28, 165
semi-metric, 20
sequence alignment, 10
sequence length (c), 10
sequencing error, 115
simulation, 4
choice of parameters with large n, 71
choice of parameters with small n, 53
general approach, 52
growing data, 70
large n, 66
small n, 53
simulation
growing data, 66
SL, 22, 47
complexity of, 47

consistency of, 47
ST, 22, 26
complexity of, 27
consistency of, 27
support, 26, 29

taxa, 4
large number of, 19
number of, 10, 156
taxonomy
central problem of, 4
TD, 22, 25, 47
complexity of, 26
consistency of, 26, 47
topology, 171
transition, 11
transition matrix, 49, 70, 153
transversion, 11
tree, 6
binary, 7
definition of, 6
diameter of, 89
number of topologies, 52, 54
rooted, 7, 67
rooted binary, 7
topology, 153, 154, 203
unrooted binary, 7
tree topology, 87, 155
definition of, 7
trees
number of, 9, 28, 31, 108
number with same topology, 54
TTDN (tree topology description notation), 171

unweighted pair-group method with arithmetic mean (UPGMA), 22
UPGMA, 22, 25, 47
complexity of, 25
consistency of, 25, 79, 81

vertex, 6
pendant, 6
violation of model, 50

weta, 21
white noise, 21, 115