uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Zaky Adam

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.d. (Computer Science)

GRADE / DEGREE

School of Information Technologie and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Evolutionary Ancestor Inference via Genome Rearrangement

TITRE DE LA THÈSE / TITLE OF THESIS

David Sankoff

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Anne Bergeron (U.Q.A.M)

Marcel Turcotte

Prosenjit Bose

Lucia Moura

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Evolutionary Ancestor Inference *via* Genome Rearrangement

by

Zaky Adam

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in Computer Science

Computer Science Department
Faculy of Graduate and Postdoctoral Studies
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-61379-5
Our file Notre référence
ISBN: 978-0-494-61379-5

University of Ottawa

Faculy of Graduate and Postdoctoral Studies

Advisor

Dr. David Sankoff

Examining Board

Dr. Anne Bergeron

Dr. Prosenjit Bose

Dr. Marcel Turcotte

Dr. Lucia Moura

The thesis by

Zaky Adam

entitled

Evolutionary Ancestor Inference *via* Genome Rearrangement

is accepted in partial fulfillment of the

requirement for the degree of

Ph.D. in Computer Science

August 25, 2009

Date

Dr. Devinder Gandhi

Chair of Examination Board

Contents

1 Introduction **1**

1.1 Background 2

1.2 Gene Order Evolution 3

1.3 Genome Rearrangement 3

1.4 Signed and Unsigned Genomes 7

1.5 Phylogenies 7

1.6 The Small Phylogeny Problem with Parsimony 9

1.7 The Large Phylogeny Problem with Parsimony 10

1.8 Using Dynamic Programming in Phylogeny Inference 11

1.9 Publication Record and Collaborations 14

1.9.1 Papers Refereed and Published in Journals 14

1.9.2 Papers Refereed and Published in Conferences 15

2 Reconstructing Phylogenomic Trees **16**

2.1 Using Common Intervals 16

2.2 Notation and Definitions 18

2.3 The Steiner Tree Problem for Common Intervals 18

2.4 The General Dynamic Programming Solution for the Small Problem . . 19

2.5 Embedding the Small Problem for Permutations in a Larger Space 20

2.6 The small Problem for a Single Zero-One Variable . . . 21
2.7 Binary Trees . . . 23
2.8 Handling Incompatible Subsets in $\mathcal{P}_n$ with PQ-Trees . . . 23
2.9 Unequal Gene Complements . . . 29
2.10 Duplicate Genes . . . 30
2.11 Application to Chloroplast Genomes . . . 30
2.12 Validation Through Simulations . . . 35
2.13 Implementation . . . 37
2.14 PQ-Trees . . . 37
2.15 Does the Concept of Common Intervals Suffer from any Drawback? . . . 39
2.16 Using Parameterized Gene Clusters . . . 40
2.16.1 Max-gap Clusters . . . 41
2.16.2 Definitions . . . 42
2.16.3 The Approach Using Parameterized Gene Clusters . . . 44
2.17 Summary of the Approach . . . 45

3 Using Double Cut and Join Operation 47

3.1 MGR: Multiple Genome Rearrangement Algorithm . . . 50
3.2 The Small Phylogeny Problem under Rearrangement Distance . . . 51
3.3 Notation . . . 52
3.4 The Median Algorithm . . . 54
3.5 Escape from Local Minima . . . 54
3.6 The Campanulaceae cpDNA Data Set . . . 56
3.7 Data Set on Mammals . . . 56
3.8 Implementation . . . 58

3.9 Evidence for Excision-Circularization-Linearization-Reincorporation . . . 58
3.10 Discussion . . . 60

4 The Benefit of a Polynomial Time Algorithm for the Breakpoint Median Problem 61
4.1 Research Opportunity Based on an Overlooked Approach to Phylogenomics . . . 61

5 Which Metric is Better: Breakpoint or DCJ? That is the Question 67
5.1 Introduction . . . 67
5.2 A Fair Comparison . . . 69
5.2.1 Branch Lengths . . . 69
5.2.2 Node Dispersion . . . 70
5.2.3 Distance to True Genome . . . 71
5.3 Breakpoints and Rearrangements . . . 71
5.4 The Empirical Studies . . . 73
5.4.1 The Data . . . 73
5.4.2 The Sample of Optimal Reconstructions and Breakpoint Re-use . . . 73
5.4.3 Simulated Data Experiments . . . 74
5.4.4 The Measurements . . . 75
5.5 Results . . . 75
5.5.1 Average Branch Length . . . 76
5.5.2 Maximum Intranode Distance . . . 76
5.5.3 Average Distance Between Reconstruction and True Ancestor . 77
5.6 Discussion . . . 77

6 Conclusion 89
6.1 Future Work . . . 91

List of Tables

4.1 Status of complexity questions for five problems related to ancestral genome reconstruction: pairwise distance, halving problem, double distance, median problem, and halving problem problem. 62

5.1 Excess rate, in %, for each reconstruction, measured by competing criterion. Average branch length results. 76

5.2 Excess rate, in %, for each reconstruction, measured by competing criterion. Maximum intranode distance data. 77

5.3 Excess rate, in %, for each reconstruction, measured by competing criterion. Average distance between reconstructed and simulated ancestor results. 78

List of Figures

2.1 Illustration of first phase of the dynamic programming approach to infer ancestors using common intervals. 25

2.2 Illustration of second phase of the dynamic programming approach to infer ancestors using common intervals. 27

2.3 Illustration of first phase of the dynamic programming approach to infer the gene content of ancestors. 32

2.4 Illustration of second phase of the dynamic programming approach to infer the gene content of ancestors. 34

2.5 Best tree for six algae and the land plants. Rooting and branch lengths arbitrary. 35

2.6 Illustration of random generation of terminal genomes on an unrooted binary tree. 36

2.7 The PQ-tree T and a subset of the collection of permutations represented by T. 39

2.8 Illustration of the need to disregard an intervening gene in order to recover a common interval disrupted by that gene. 40

2.9 Illustration of the need to disregard two intervening genes in order to recover a common interval disrupted by those two genes. 40

2.10 Finding max-gap clusters between two genomes. 42

2.11 Constructing graphs from two genomes using neighbourhood parameter $\theta = 3$. 43

2.12 Constructing graphs from two genomes using neighbourhood parameter $\theta = 3$, and inferring the set of optimal edges and the set of near-optimal edges of their ancestor. 45

3.1 Illustration of adding a new genome to the partially reconstructed tree in MGR-Median. 51

3.2 Phylogeny for Campanulaceae data set. Rooting and edge lengths arbitrary. 57

3.3 Phylogeny for mammalian data set. Rooting and edge lengths arbitrary. 57

3.4 Diagnostics for transposition and block interchange. 59

4.1 The weight matrix of Example 2. 64

4.2 Phylogeny for Campanulaceae. 66

5.1 Strategy for comparing different metrics. 70

5.2 Phylogenies for mammalian data sets. 74

5.3 Graphical representation of computing average branch lengths of trees inferred using DCJ and breakpoint. 80

5.4 Graphical representation of computing maximum DCJ distances between ancestors and their counterparts of trees inferred using DCJ, and computing maximum breakpoint distances between ancestors and their counterparts of trees inferred using breakpoint. 81

5.5 Graphical representation of computing maximum pairwise DCJ distances between ancestors and their counterparts of trees inferred using breakpoint, and computing maximum pairwise breakpoint distances between ancestors and their counterparts of trees inferred using DCJ. . . 82

5.6 Graphical representation of computing maximum pairwise DCJ distances between ancestors of trees inferred using breakpoint and their counterparts of trees inferred using DCJ, and computing maximum pairwise breakpoint distances between ancestors of trees inferred using breakpoint and their counterparts of trees inferred using DCJ. 83

5.7 Graphical representation of computing average pairwise DCJ distances between true ancestors and their counterparts of trees inferred using DCJ, and computing average pairwise breakpoint distances between true ancestors and their counterparts of trees inferred using breakpoint. 84
5.8 Graphical representation of computing average pairwise DCJ distances between true ancestors and their counterparts of trees inferred using breakpoint, and computing average pairwise breakpoint distances between true ancestors and their counterparts of trees inferred using DCJ. 85
5.9 Competing criterion average branch lengths versus reconstructing criterion. 86
5.10 Competing criterion maximum intranode distance versus reconstructing criterion. 87
5.11 Distance according to competing criterion between simulated ancestor and reconstruction. 88

Abstract

Inferring ancestral gene orders in a phylgenomic tree is an important topic in comparative genomics. In this thesis, three different approaches have been used to infer ancestors, first, using common intervals in a model-free approach and extending it to using common clusters and neighbourhood parameter; second, using double cut and join operation (DCJ); third, using breakpoint distance. A statistically fair comparison between the performance of DCJ and breakpoint criteria ends the thesis.

Away from any assumptions or considerations, probabilistic or combinatorial, about specific processes involved in rearranging genomes, we present a new phylogenetic reconstruction method based solely on common intervals. The objective function to be optimized is simply the sum over the tree branches of the symmetric difference between the two sets of intervals associated with the genomes at the two ends of the branch. To achieve this goal, we use dynamic programming optimization to determine the presence of common intervals at the ancestral nodes of the phylogeny.

Noticing the drawback that the concept of common intervals suffers from, we introduce the concept of generalized adjacency to find common clusters using a neighborhood parameter that turns out to be closely related to the bandwidth parameter of a graph. Our focus will be on how this parameter affects the characteristics of clusters: how numerous they are, how large they are, how rearranged they are and to what extent they are preserved from ancestor to descendant in a phylogenetic tree. Again, we use dynamic programming optimization to determine the presence of individual edges at the ancestral nodes of the phylogeny.

The DCJ (double cut and join) operation introduced by Yancopoulos *et al.* in 2005 is the most inclusive operation to date as it can generate all the movement rearrangements. One year later, Bergeron *et al.* restated the DCJ model and produced a simplified (linear) algorithm, which is now the most general existing algorithm to transform one genome into another using genome rearrangements events. Motivated by both, the most inclusive operation, DCJ, and its most general algorithm, we study the small phylogeny problem in the space of multichromosomal genomes under the DCJ metric. This is similar to the existing MGR (multiple genome rearrangements) approach, but it allows, in addition to inversion and reciprocal translocation, operations of transposition and block interchange.

Thanks to Tannier *et al.*, the first polynomial solution to the median problem has been found in only one context, namely the case of breakpoint distance on multichromosomal genoms where chromosomes are unconstrained as to linearity or circularity. This motivated us to study the small phylogeny problem using breakpoint median as a third approach, that is different both biologically and computationally from the common intervals and DCJ approaches, and then to compare statistically the performance of both criteria, breakpoint and DCJ.

Keywords: phylogenetic tree, genome rearrangment, inversion, reciprocal translocation, transposition, block interchange, common intervals, generalized adjacency, neighborhood parameter, graph bandwidth, multiple genome rearrangement (MGR), double cut and join (DCJ), breakpoint(BP), excess explanatory rate.

Acknowledgements

I would like to express my sincere gratitude to Dr. David Sankoff, my advisor for the past years. He generously presented me his academic advice, encouragement, time, patience and dedication for completion of this thesis.

Also, I would like to thank my mother, my father, and my wife, Sonya, for their encouragement, and continuous support.

Chapter 1

Introduction

The genetic identity of individuals and species resides in their genomes. Over generations, genomes of species evolve through large numbers of small local changes, either *point mutations* or *copy number variation*, and very occasionally by major *genome rearrangements*, whose scope may include most or all of a chromosome or several chromosomes.

The reconstruction of the evolutionary history of a group of modern species, traditionally based on data on the different mutations in a single gene in different species, is called *phylogenetics* or *phylogeny*. The gene-level data may be the DNA sequence of the gene in question or it may be the amino-acid sequence of the protein coded by that gene.

There has been a longstanding debate on how to handle different phylogenetic results based on data from different genes. With the advent of genome sequencing, generating data from tens of thousands of genes, the question of scaling up phylogenetics to be accountable for more than a single gene, has become a major preoccupation. While much of the ensuing methodological developments have involved balancing or reconciling the results from many genes, whole genome sequencing has made an entirely different kind of approach to phylogeny feasible, namely the analysis of the accumulation of genome rearrangements over time. This latter kind of approach is the subject of this thesis.

Genome rearrangements include a limited repertoire of operations. Mathematically these may be considered as operating on the space of signed and fragmented permutations, representing chromosomes with identified genes located along their lengths. The operations (to be explained in more detail in Section 1.3) include inverting a fragment of a chromosome, while reversing the sign of all the genes in the fragment, exchanging the prefixes or suffixes

of chromosomes, fission or fusion of chromosomes, movement of a fragment of a chromosome to a new location, and excising or reinserting a circularized fragment of a chromosome.

Instead of the usual alignment distance comparing two genes or proteins, the study of genome rearrangements requires new distances, like the edit distances counting the minimum number of operations necessary to transform one genome into another, or the breakpoint distance measuring the number of gene adjacencies in one genome not present in the other. Other comparisons of genomes do not invoke the idea of distance, but count the number of common intervals, segments or gene clusters in two genomes.

Once we have the space of genomes and the means of comparing them, we can attack the phylogeny problem, namely how to find the family tree of a number of given genomes. This may be decomposed into two parts. The small phylogeny problem starts with a topologically fixed tree and tries to optimally locate the ancestral nodes in the space of genomes. The large phylogeny problem tries to find the best tree topology and necessarily calls the small phylogeny problem as a subroutine.

In this thesis, I explore the small phylogeny problem in various versions of the space of genomes and with various approaches to comparing two genomes. The big phylogeny problem will be considered but with a small enough set of genomes so that we can find the optimum simply by exhaustively considering the set of all trees.

1.1 Background

The two main approaches to genome comparison, which themselves overlap to some extent, are the distance-based approach and the cluster/segment/interval-based approach. We will be considering both of these. Within the distance-based approach, we may distinguish breakpoint analysis, mentioned above, and rearrangement analysis. Within the latter we can distinguish the classical *reversals/translocations* model, sometimes called the Hannenhalli-Pevzner theory [1] after the authors of the first polynomial-time algorithm for this class of problems, and the recent *double-cut-and-join* model [2], which incorporates a wider class of operations and is computationally easier.

These distinctions are all reflected in approaches to the small phylogeny problem, as we shall see in the ensuing sections.

An alternative approach to genome-based phylogeny is to simply use pairwise genomic

distances as input into one of the various distance-matrix clustering methods, like neighbour joining. This is a perfectly valid approach, but one which loses most of the genomic information at the outset and does not directly concern itself with inferring ancestral genomes, which is one of our major concerns.

1.2 Gene Order Evolution

Prokaryotes genomes consist usually of a single circular chromosome, whereas eukaryotes genomes consist of multiple linear chromosomes. In genome rearrangement approach, we focus on the general structure of a chromosome instead of the internal nucleic structure of each gene. We assume that the identity of each gene and its homologs, paralogs or orthologs, among a set of genomes have been already determined, such that each gene is represented by an integer number. In the realistic case, a sign (+ or -) must precede each gene indicating on which of the two complementary strands of DNA the gene resides, and then we say that the genome is signed, otherwise it is unsigned. Thus, if a chromosome consists of n genes, it is represented by an ordered set of n integers, and the whole genome is represented as a set of ordered chromosomes.

In this labeling, orthologous genes in different genomes receive the same number, as generally do paralogous genes in the same genome.

Note that for mathematical purposes a chromosome is sometimes not represented by a set of ordered genes, rather by a set of ordered blocks of genes contiguous in the N species being considered, a set of ordered *conserved chromosomal segments*, or a set of ordered *synteny blocks* of genes. A *conserved segment* among a set of genomes is defined to be a region found in all these genomes in which homologous genes keep the same order. A *synteny block* is a set of genes that appear together, but not necessarily in the same order, in all the genomes being compared.

1.3 Genome Rearrangement

There are many criteria to compare gene orders in two genomes, for example:

1. The *breakpoint distance* between two genomes G_1 and G_2 measures the amount of disruption between conserved segments in G_1 and G_2, that is the number of pairs of

genes *a, b* that are adjacent in one genome (it contains the conserved segment "*a b*") but not in the other (it does not contain neither "*a b*", nor "*-b -a*"). For example:

$$G_1 = 1\ .\ 2\ .\ 3\ .\ 4\ .\ 5\ .\ 6$$
$$G_2 = \text{-}6\ .\ \text{-}5\ \ 3\ \ 1\ .\ 2\ \ 4$$

We notice that the conserved pairs of genes are: "1, 2" and "5, 6", and the number of breakpoints is 3: "2, 3", "3, 4", "4, 5". Thus, the total distance between G_1 and G_2 is 3. The time complexity of computing the breakpoint distance is linear in the length of genome. Watterson *et al.* [3] suggested this metric for the first time. This metric has been used for a while to reconstruct phylogenomic trees [4, 5].

2. The *rearrangement distance* between two genomes measure the minimum number of rearrangement events necessary to transform one genome into another. There are many types of rearrangement operations, some of them are specific to multichromosomal genomes only.

   - In unichromosomal genomes, we have the following rearrangement operations:

     1) A *reversal* or an *inversion* of an interval of genes in a unichromosomal genome is an operation that reverses the order of genes in that interval and changes the signs of its genes. In the example below, the interval "3 1 -9 6" is reversed to become "-6 9 -1 -3":

        -5 2 3 1 -9 6 4 7 8

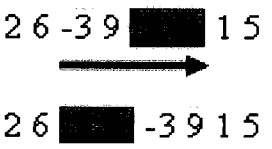
        ⟶

        -5 2 -6 9 -1 -3 4 7 8

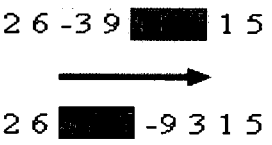
        The *reversal distance* between two genomes is the minimum number of reversals it takes to get from one genome to the other. For a given pair of genomes, the reversal distance is unique, but there are usually many possible reversal scenarios with this distance. The reversal distance has been used to infer phylogenomic trees as in [6].

     2) A *transposition* of one segment of genes from one site of a genome to another. It could be defined as swapping two adjacent segments of genes. In the

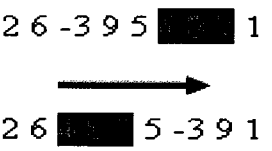
example below, the segment "-3 9" is transposed by swapping it with the segment "4 8 7":



3) An *inverted transposition* which is the same as transposition but the order of genes on the transposed segment is reversed. In the example below, the segment "-3 9" is transposed by swapping it with the segment "4 8 7" after being inverted to "-9 3":



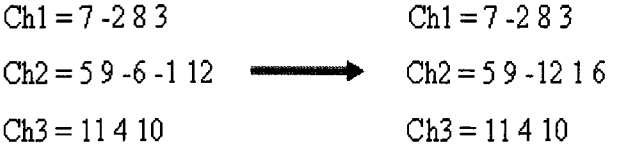
4) A *block interchange* is a transposition operation where the two segments are not adjacent.



In the above example, the segment "-3 9" is transposed by swapping it with the segment "4 8 7". The gene "5" separates both segments.

- In multichromosomal genomes:

The previous intrachromosomal rearrangements like reversal, transposition, and block interchange might occur on one chromosome of the multichromosomal genome. In the example below, a reversal of the interval "-6 -1 12" on the second chromosome becomes "-12 1 6":

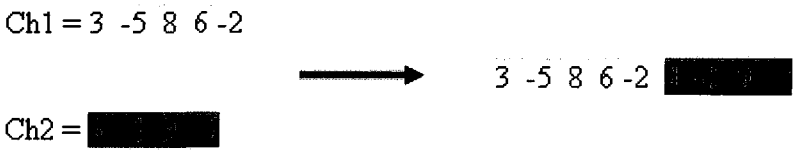


Also, the block interchange can occur between two segments belonging to two different chromosomes in multichromosomal genomes. In addition, the following interchromosomal rearrangements are specific to multichromosomal genomes:

1) A *translocation*, also called a *reciprocal translocation*, is an operation involving two chromosomes where a prefix of one exchanges with a prefix of the other, or a suffix of one exchanges with a suffix of the other, as shown below,

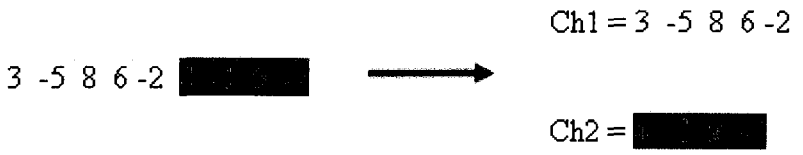
or a reversed suffix of one exchanges with a reversed prefix of the other as shown below:

2) A *fusion* is a merging of two chromosomes into one, as shown below:



A fusion could be considered as a special case of a translocation where a null prefix of one chromosome is exchanged with the whole other chromosome resulting in two chromosomes fusion.

3) A *fission* is a split of one chromosome into two, as shown below:



A fission could be also considered as a special case of a translocation where a null chromosome exchanges with a prefix of the other resulting in a chromosome fission.

We note that a chromosome and its entire reversal are equivalent in the mathematical formulation. This corresponds to the biological fact that the chromosome remains unchanged if we invert the whole thing. Thus:

$$1\ 2\ 3\ \ldots\ n = -n\ -(n\text{-}1)\ -(n\text{-}2)\ \ldots\ -2\ -1$$

Finally, there are two evolutionary events that affect the gene content and indirectly the gene order on a uni- or multi-chromosomal genome: insertions, including duplications, and deletions of single genes or segments of genes.

1.4 Signed and Unsigned Genomes

From a complexity point of view, the difference between the signed and the unsigned cases of genome representations is far from being trivial. It has been shown that the problem of reversal distance in the unsigned version is NP-complete [7], whereas the problem is solved by the exact polynomial algorithm of Hannenhalli and Pevzner in the signed version [1]. Also, the problem of the *syntenic distance*, defined to be the minimal number of translocations required to transform one genome into another, has been shown to be NP-complete in the unsigned version [8]. The problem of reversals and translocation distance was also solved by Hannenhalii and Pevzner in 1995 for the unsigned case [9]. Fortunately, with many genomes currently being sequenced, it is likely that many comparative maps (corresponding to unsigned genomes) will soon be replaced by sequencing data (corresponding to signed genomes).

1.5 Phylogenies

A *phylogeny* is a reconstruction of the evolutionary history of a set of organisms. A phylogeny is represented as a tree where the leaves represent current organisms and internal nodes represent ancestors. The edges represent evolutionary relationships between ancestors and their descendants. Thus, a phylogeny is considered as a "family tree of species".

Reconstructing phylogenies of organisms is one of the earliest and widely used applications of computational biology. A phylogeny is inferred from data derived from a set of present-day organisms. This can be either morphological data, e.g., macroscopic measurements, counts or patterns, molecular data, e.g. DNA or protein sequences, or structural genomic data, e.g. the linear order of genetic elements on chromosomes. When the data is molecular, the tree is called *phylogenetic*, but when the data is genomic, the tree is called *phylogenomic*. Among the advantages of using genomic data we can mention:

- The whole genome is studied so that we avoid the problem of different genes producing different phylogenies.
- While alignment is still needed at the molecular level to identify similarity regions, there is no need for alignment at the genomic level.
- Genome rearrangement events and duplications are much rarer than nucleotide mutations as mentioned in [10], which helps us to trace evolutionary history farther back than molecular data.

However, using genomic data imposes some challenges:

- The lack of data: the number of genomes sequenced up to now is not enough to build large phylogenies.
- There is no good model of evolution for the genomic data, for example we are still not sure of the relative probabilities of the various rearrangement events or of the distribution of sizes of the chromosomal segments involved, or of the favored sites for rearrangement.
- The mathematical complexity of gene orders, which combined with the lack of good model of evolution create significant mathematical challenges that researchers still strive to solve.

Thus, reconstructing phylogenies based on gene-order data is much more challenging mathematically than reconstructing phylogenies based on molecular data. For example, the computation of a pairwise genomic distance, which is the basis for the phylogeny reconstruction process, was by itself a challenging problem.

The construction of phylogenomic trees using genomic data was given impetus by Hannenhalli and Pevzner's discovery of a polynomial time algorithm for converting one genome to another using reversal operations [1].

Reconstructing phylogenomic trees based on gene order has used two types of objective functions for optimizing the inferred ancestral nodes: The first is based on breakpoints, or non-conserved adjacencies in the gene orders of two genomes [11], the second is based on the number of genome rearrangement events intervening between two genomes [7, 12, 13, 14]. The problem with the breakpoint approach is that it is a very local and superficial expression of genome rearrangements, and thus it can not reflect them directly, rather it reflects their final outcome, whereas a genome rearrangement operation might span an unrestricted segment of a genome, and thus it is able to reflect genome rearrangements. However, both breakpoint distance and the reversal distance suffer from the fact that they underestimate the actual number of steps that occurred biologically, therefore a corrected distance needed to be suggested for each one in [15].

In order to express the similarity between two genomes by exploring similar regions larger than a single breakpoint in the context of genome rearrangement analysis, researchers have introduced the notions of conserved intervals and common intervals into rearrangement analysis [16, 17]. The use of these notions has been extended to reconstructing ancestral gene orders in a given phylogeny [18, 19].

Armed with algorithms for computing pairwise genomic distances, researchers could proceed to reconstruct matrix-based phylogenomic trees. Nevertheless, the main reconstruction method used is parsimony, where each tree is scored in terms of its total evolutionary distance and the best tree, the most parsimonious, is the one with the minimum distance. Here we should distinguish between two types of parsimony problems: the *small parsimony problem* and the *large parsimony problem.*

1.6 The Small Phylogeny Problem with Parsimony

We assume that a binary-branching (not necessarily, but usually) tree structure is given where the leaves are assigned genomes of current species and we have to infer the genomes at the internal nodes such that the total tree distance will be minimized. Genomes at the leaves have the same gene-content but not the same gene-order. Thus we can formalize the

problem as follows:

Small Phylogeny problem

Input : Tree T where each leaf is labeled by a different genome consisting of the same n genes.

Output : labeling of internal nodes of the tree T minimizing the parsimony score.

The parsimony score is the total distance of the tree which is the sum of the lengths of its edges; the length of an edge is the pairwise distance computed according to one of the genomic distance computation algorithms. Thus, if the length of an edge (v, w), where v and w are two adjacent nodes in T, is $d(v, w)$, then:

$$\text{The parsimony score of } T = \Sigma_{(v,w) \text{ in } T} \; d(v, w)$$

The main algorithmic approach used to infer genomes of internal nodes is *dynamic programming.* In 1971, Walter Fitch [20] was the first to suggest a solution to the small phylogeny problem using dynamic programming, although he did not present a proof of the correctness of his algorithm. In 1975, David Sankoff [21] also suggested a dynamic programming algorithm to solve the same problem. It turned out that the two algorithms are similar but Sankoff's algorithm is more general, has a proof, and tells us why it works correctly.

Both algorithms are designed for rooted binary trees, however a similar approach could be used for unrooted binary trees.

1.7 The Large Phylogeny Problem with Parsimony

A set of m genomes representing m species are given. We have to find the binary-branching tree structure and infer the genomes at the internal nodes of this tree such that the total tree distance will be minimized over all possible trees and all possible labeling of internal nodes. Genomes at the leaves have the same gene-content but not the same gene-order. Thus we can formalize the problem as follows:

Large Phylogeny problem

Input : A set of m genomes, all of which consist of the same n genes but with different gene-orders.

Output : Tree T and the labeling of its internal nodes such that the parsimony score of this tree will be minimized over all possible trees and all possible labeling of internal nodes. The leaves of T are labeled with the m genomes.

A naïve solution is to reconstruct all possible tree topologies with m leaves, solve the small parsimony problem for each topology to find its parsimony score, and select the one with the minimal score. These trees could be rooted or unrooted.

The problem is that the number of all possible topologies grows exponentially with the number of leaves m. For example, for unrooted trees, this number is:

$$\frac{(2m-5)!}{2^{m-3}(m-3)!}$$

and for rooted trees, this number is bigger:

$$\frac{(2m-3)!}{2^{m-2}(m-2)!}$$

However, this does not justify the fact that the large parsimony problem is NP-complete, rather it is a formal proof by Day *et al.* [22] that projects the vertex cover problem, which is known to be NP-complete, onto the parsimony problem. Therefore, heuristic methods are necessary when m is larger than 7 or 8. Among the heuristic methods usually used is the *nearest neighbor interchange* found by David Robinson [23] in 1971.

1.8 Using Dynamic Programming in Phylogeny Inference

Dynamic programming is a powerful technique for algorithm design. Its basic idea is to exploit the "Principle of Optimality", where it applies, by decomposing a problem into a set of solutions to smaller problems plus an incremental term due to the extension of the problem by one element.

Richard Bellman was the pioneer in the systematic study of dynamic programming [24], but the first dynamic programming algorithm for biological macromolecular sequence comparison was designed by Saul Needleman and Christian Wunsch in 1970 [25].

Since that time, many bioinformaticians frequently use dynamic programming algorithms not only for pairwise sequence alignment, but also to solve the small phylogeny problem with

parsimony [21, 26] as well as many other molecular data-related problems such as multiple sequence alignments, RNA secondary structure, exon chaining problem, etc. Favoring this method is due to its advantages surveyed in [27], among which we can mention:

- Use of a simple objective function,
- Global optimality,
- Stable parameters and soft limits,
- Versatility and consistency.

As for using dynamic programming to label internal nodes of an unrooted binary-branching tree in the process of solving the small phylogeny problem, it usually follows the following scheme.

First, we determine a distance measure d to be the metric over the space S of possible solutions. We know that the degree is 1 for terminal nodes and 3 for internal nodes. We arbitrarily choose an internal node ρ to be the root. We also set a recurrence formula F to infer some possibilities for each internal node based on F for the two of its three direct neighbors remote from the root, "its descendants" (Which nodes are descendants of which others depends on the choice of root.) The three neighbors of the root are all its descendants.

Dynamic programming requires two passes over the nodes of the tree, the first is a recurrence, and the second is a trace-back. For each node v in the tree, the recurrence finds the optimal value of the objective function F_v for all eligible points in the space S. The key to tractability is to be able to restrict the eligible portion of S as much as possible. The order of visiting internal nodes to calculate F is not important as long as an internal node v is not visited before its two descendants β and γ, which is always possible in a tree. The recurrence for $v \neq \rho$ is:

$$F_v(x) = min_{\,y \in S}\,[d(x,y) + F_\beta(y)] + min_{\,z \in S}\,[d(x,z) + F_\gamma(z)].$$

For $v = \rho$, with descendants α, β and γ:

$$F_\rho(x) = min_{\,w \in S}\,[d(x,w) + F_\alpha(w)] + min_{\,y \in S}\,[d(x,y) + F_\beta(y)] + min_{\,z \in S}\,[d(x,z) + F_\gamma(z)].$$

To initialize, at each terminal node τ we set the objective function $F_\tau(g) = 0$ at the point g in the space representing the observed data value, and $F_\tau(x) = \infty$ for $x \neq g$. At the same time, we define set-valued pointer functions Y and Z:

$$Y_v(x) = \{y|\ d(x,y) + F_\beta(y)\ is\ minimal\},$$
$$Z_v(x) = \{z|\ d(x,z) + F_\gamma(z)\ is\ minimal\},$$

and

$$W_\rho(x) = \{w|\ d(x,w) + F_\alpha(w)\ is\ minimal\}.$$

Once the recurrence has been applied to ρ, giving functions F_ρ, W_ρ, Y_ρ and Z_ρ, a solution to the small phylogeny problem is found by choosing any x where $F_\rho(x)$ is minimal over S to be the root genome, and following the pointer functions, i.e., choosing any $w \in W_\rho, y \in Y_\rho$, and $z \in Z_\rho$ to be the genomes at nodes α, β and γ, then following the pointer functions for each of these nodes, until all internal nodes have been assigned genomes.

Inferring a genome out of three others is called the *median problem* and all indications are that it is likely to be NP-hard in any rearrangement metric space [7, 14], except in the case of breakpoint distance on multichromosomal genomes where chromosomes are unconstrained as to linearity or circularity, where Tannier *et al.* [53] found a polynomial time solution for it.

Now, a question might arise: Can we use dynamic programming to label internal nodes based on all types of genome rearrangement events, or at least one of them, e.g. reversals?

Actually, the answer is no. Because the solution space of sorting by reversals alone is so huge even for a small permutation, such that tracing all possible solution by dynamic programming will be exponential in the size of the genome. For example, in order to transform the permutation (-12, 11, -10, 6, 13, -5, 2, 7, 8, -9, 3, 4, 1), into the identity (1, 2, 3, ..., 13), the number of all possible solutions that have the same minimum number of reversals required to the transformation is 8278540. If we have a rooted binary tree which terminal vertices are assigned genomes of size 13 each, then inferring an ancestor u of two terminal descendants, x and y, using reversals would give a number of optimal solutions of the order of 8278540 from each descendant. Using dynamic programming to infer v, the ancestor of u and w, would require tracing a number of optimal solutions of the order of 2 x (8278540 + 8278540) x 8278540. Siepel [28] found an $O(n^3)$ algorithm that represents, in a compact notation, the set of all optimal solutions to transform a given signed permutation into the identity, where n is the size of the permutation.

One solution for this is to write a recurrence for a simplified version of the problem and to check, during the trace-back, whether its solution is also valid for the original problem.

In this thesis, we used this strategy in two ways. One makes use of PQ-trees during the trace-back, and the other makes use of a bandwidth calculation.

1.9 Publication Record and Collaborations

The following list of publications have been produced solely or partly due to the work described in this thesis. In each case, I append an assessment of the contribution of myself and of my collaborators, except for my supervisor, David Sankoff, who has played his advisory role in all aspects of this research.

1.9.1 Papers Refereed and Published in Journals

1. Qian Zhu, Zaky Adam, Vicky Choi, and David Sankoff. 2008. Generalized gene adjacencies, graph bandwidth and clusters in yeast evolution. *Transactions on Computational Biology and Bioinformatics.* In press.

   This is a full-length journal version of the conference version listed below. My main contribution to this, aside from those to the original version, was the idea to carrying out the simulation study. I also participated in the discussions about improving the bandwidth analysis, the second of the two major components of the extended version. The actual computing, however, was carried out by Zhu. This paper represents sections 2.15 and 2.16 of chapter 2.

2. Zaky Adam and David Sankoff. 2008. The ABCs of MGR with DCJ. *Evolutionary Bioinformatics Journal*, 4: 69-74.

   This was entirely my research project. This paper represents the bulk of chapter 3.

3. Zaky Adam, Monique Turmel, Claude Lemieux, and David Sankoff. 2007. Common intervals and symmetric difference in a model-free phylogenomics, with an application to streptophyte evolution. *Journal of Computational Biology*, 14(4): 436-445.

   A somewhat extended version of the conference paper listed below. This was entirely my research project; the molecular biologists Turmel and Lemieux suggested the application and provided the data. This paper represents the bulk of chapter 2.

1.9.2 Papers Refereed and Published in Conferences

1. Zaky Adam and David Sankoff. 2009. A statistically fair comparison of ancestral genome reconstructions, based on breakpoint and rearrangement distances. The 7th Comparative Genomics Satellite Workshop (RECOMB CG 2009). *Lecture Notes in Computer Science*, 5817: 193-204.

   This was entirely my research project. This paper represents the bulk of chapter 5.

2. Chunfang Zheng, Qian Zhu, Zaky Adam, and David Sankoff. 2008. Guided genome halving: hardness, heuristics and the history of the Hemiascomycetes. *Proceedings of the 16th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2008).*

   These are preliminary results of the study mentioned in the Future Work section below. My contribution was part of the algorithm to infer ancestral genomes, namely the part which requires a median routine. For this we modified the algorithm in my "The ABCs of MGR with DCJ" paper.

3. Qian Zhu, Zaky Adam, Vicky Choi, and David Sankoff. 2008. Generalized gene adjacencies, graph bandwidth and clusters in yeast evolution. The 4th International Symposium on Bioinformatics Research and Applications (ISBRA 2008). *Lecture Notes in Computer Science*, 4983: 134-145.

   My contribution to this paper was the dynamic programming of the small phylogeny problem, inspired by the method I used in the "Common intervals" paper, but without the PQ-trees component. Instead, we used bandwidth calculations. I participated fully in the discussion of how to do the latter calculations. This paper represents sections 2.15 and 2.16 of chapter 2.

4. Zaky Adam, Monique Turmel, Claude Lemieux, and David Sankoff. 2006. Common intervals and symmetric difference in a model-free phylogenetics, with an application to streptophyte evolution. The 4th Comparative Genomics Satellite Workshop (RECOMB CG 2006). *Lecture Notes in Computer Science*, 4205: 63-74.

   This was entirely my research project; the molecular biologists Turmel and Lemieux suggested the application and provided the data. This paper represents the bulk of chapter 2.

Chapter 2

Reconstructing Phylogenomic Trees

2.1 Using Common Intervals

Phylogenetics based on gene order has used two types of objective functions for optimizing the inferred ancestral nodes. One is based on breakpoints, or non-conserved adjacencies in the gene orders of two genomes, dating from 1997 [29] and related to the quantitative pre-genomics literature on conserved segments [30]. The second is based on the number of inversions or other genome rearrangements intervening between two genomes [31, 7, 14, 32].

The scope of a genome rearrangement operation may be unrestricted across the genome; a breakpoint is a very local structure. To emphasize local similarities spanning regions larger than a single breakpoint within a rearrangement analysis, Bergeron and colleagues have integrated more general notions of conserved intervals and common intervals with rearrangements [16, 17]. Further, they used these integrated concepts in phylogeny, including the reconstruction of ancestral gene orders [18, 19].

Definition 1. An interval $[i, j]$ is a common interval between two or more signed permutations, if the elements of $[i, j]$ appear consecutively in all permutations, but not necessarily in the same order, or in the same sign.

Example 1. In the example below, the interval [2, 5] from permutation D is common between the two permutations D and F.

$$D = (1\ 2\ 3\ 4\ 5\ 6)$$

$$F = (1\ 6\ 4\ -2\ 5\ -3)$$

In this chapter, we put even more importance on common intervals, basing a phylogenetic reconstruction method solely on them. This analysis is model-free, in the sense that it dispenses completely with assumptions or considerations, probabilistic or combinatorial, about specific processes involved in rearranging genomes. The objective function we will optimize is simply the sum over the tree branches of the symmetric difference between the two sets of intervals associated with the genomes at the two ends of the branch. The motivation is simply that the more related two genomes are, the more common intervals they will have, while as evolutionary distance increases, more and more intervals from each will be lacking from the other.

Optimizing a tree on the space of subsets of the power set of $\{1, \cdots, n\}$ means assigning a set of subsets of $2^{\{1,\cdots,n\}}$ to each ancestral node in such a way as to minimize the sum of the branch lengths (symmetric difference) using dynamic programming [20, 33, 26]; this optimization is easy; the set of intervals representing a genome, however, is constrained to be compatible with a permutation. We do not know the complexity of optimizing the ancestral nodes of a tree under this constraint, but conjecture that it is hard. Thus we model our optimization heuristic after the unconstrained case, and add the constraint in the traceback of the algorithm, where a greedy procedure is used to test the addition of possibly conflicting intervals to the subset of intervals representing a genome, represented by a PQ-tree.

Our methods carry over directly to the case where some or all genomes do not contain the full set of genes. All we require is a preliminary assignment of genes to ancestral nodes, rapidly achieved by dynamic programming, and an adjustment of a test for identity of two intervals to be restricted to the reduced intervals containing only the genes in common in the two genomes. The same sort of extension of the model works when duplicate genes and gene families are allowed, but only under certain conditions; the general case will require further work.

We apply our method to the biologically controversial questions of streptophyte phylogeny and recover results comparable to previous work on both gene order phylogeny and DNA sequence-based phylogeny.

2.2 Notation and Definitions

For a permutation Π on $(1, \cdots, n)$, we write $\Pi = (\pi(1), \cdots, \pi(n))$. Let $\mathcal{S} = \{\Pi_1, \cdots, \Pi_N\}$ be a set of such permutations. For each Π, the interval set determined by $\pi(h)$ and $\pi(k)$, for $1 \leq h < k \leq n$, is $\{\pi(h), \pi(h+1), \cdots, \pi(k)\}$. For each $J = 1, \cdots, N$, let $\mathcal{I}_J \subset 2^{\{1,\cdots,n\}}$ be the $\binom{n}{2}$ interval sets of $2^{\{1,\cdots,n\}}$ determined by Π_J. We define the projection $B(\Pi_J) = \mathcal{I}_J$. The set of common intervals of Π_J and Π_K is $B(\Pi_J) \cap B(\Pi_K) = \mathcal{I}_J \cap \mathcal{I}_K$, the intersection of the $\binom{n}{2}$ interval sets defined by Π_J and those defined by Π_K.

Let $X \subseteq \mathcal{I}_J$ and $Y \subseteq \mathcal{I}_K$. We say X is compatible with $B(\Pi_J)$ and Y is compatible with $B(\Pi_K)$ and define the metric

$$\begin{aligned} d(X,Y) &= |X \cup Y \setminus X \cap Y| \\ &= |X| + |Y| - 2|X \cap Y|. \end{aligned} \tag{2.1}$$

In particular,

$$\frac{1}{2}d(\mathcal{I}_J, \mathcal{I}_K) = \binom{n}{2} - |\mathcal{I}_J \cap \mathcal{I}_K|. \tag{2.2}$$

Note that not all subsets of $2^{\{1,\cdots,n\}}$ are permutation-compatible, i.e., are subsets of $B(\Pi)$ for some permutation Π. E.g., $\{\{1,2\},\{2,3\},\{2,4\}\}$ is not permutation-compatible. For any permutation-compatible X, we define $B^{-1}(X)$ to be the set of permutations Π for which $X \subseteq B(\Pi)$.

Example 2. Let us have the two sets of intervals:

$$S1 = \{\{1,4\},\{1,3\},\{2,3\}\}$$
$$S2 = \{\{1,4\},\{1,3\},\{1,2\}\}$$

Then the size of their symmetric difference is:

$$d(S1, S2) = |\{\{2,3\},\{1,2\}\}| = 2$$

2.3 The Steiner Tree Problem for Common Intervals

The Steiner tree problem with input $\mathcal{S}$ is to find a tree graph $T = (V, E)$, where the vertices in V are permutation-compatible subsets of $2^{\{1,\cdots,n\}}$ and $\{B(\Pi)\}_{\Pi \in \mathcal{S}} \subseteq V$ such that the tree length

$$L(T) = \sum_{XY \in E} d(X, Y) \tag{2.3}$$

is minimal.

In a Steiner tree, we can distinguish two types of vertices in V, terminal vertices, i.e., of degree 1, which are all projections of permutations in $\mathcal{S}$, and non-terminal, or "ancestral", vertices, some or all of which, the "unknown vertices" may be projections of permutations not in $\mathcal{S}$ or other permutation-compatible subsets of $2^{\{1,\cdots,n\}}$. We require that unknown vertices have degree 3 or more, a condition which, by the metric property, does not affect the minimal value of L in (2.3).

As with phylogenetics methods in general, the search for the Steiner tree is often divided into two problems, the inner, or "small", problem, and the outer, or "big", problem. The big problem is essentially a search through the set of all trees satisfying the above description. For each tree examined during this search, the small problem is to optimally identify each of the unknown vertices in V as the projection of some permutation on $(1, \cdots, n)$ or some other permutation-compatible subset of $2^{\{1,\cdots,n\}}$.

2.4 The General Dynamic Programming Solution for the Small Problem

A general method for the small problem, i.e., for optimizing the position of the ancestral nodes of a tree in a metric space was given in [26].

Condition 1. It suffices to consider trees where there is a path between any two unknown vertices not passing through a vertex in $\mathcal{S}$; otherwise the problem decomposes into subproblems in an obvious way.

The solution requires choosing, arbitrarily, one of the unknown vertices r as the "root", and directing all edges in E away from the root. We write $v \rightarrow u$ for an edge directed from ancestor v to "daughter" u. For any given position of the ancestral vertices, let

$$l(v) = \sum_{v \rightarrow u} l(u) + d(u, v), \tag{2.4}$$

with initial condition $l(v) = 0$ if v is a terminal vertex. Note that the tree structure and Condition 1 ensure that recurrence (2.4) determines $l(v)$ for all non-terminal vertices. Then

for any tree T with specified positions of the ancestral vertices we may rewrite (2.3) as

$$L(T) = l(r). \tag{2.5}$$

For a Steiner tree, for a given position of v, we obtain from (2.4)

$$l(v) \quad = \quad \sum_{v \to u} \min_{u} [l(u) + d(u, v)]. \tag{2.6}$$

If all the minimizing u are stored at each application of (2.6), then a Steiner tree T may be recovered by tracing back the recurrence from the root to the terminal nodes.

Depending on the metric space, the minimizing u in (2.6) may be more or less difficult to calculate. Such is the case that interests us here, where the vertices are projections of permutations, and the search for optimizing u is not straightforward. However, we can easily solve a closely related problem, which provides a lower bound on $L(T)$ and suggests how to solve the problem on permutations.

2.5 Embedding the Small Problem for Permutations in a Larger Space

For a given $\mathcal{S}$ consider the Steiner tree problem with input $\{B(\Pi_1), \cdots, B(\Pi_N)\}$ in the metric space (M_n, d), where $M_n = 2^{2^{\{1,\cdots,n\}}}$, the set of all subsets of the power set of $\{1, \cdots, n\}$, and d is as before.

The set M_n is larger than $\mathcal{P}_n$, the set of all permutation-compatible subsets of $2^{\{1,\cdots,n\}}$, in that there are elements of $X \in M_n$ which are not of the form $X \subseteq B(\Pi)$ for any permutation Π.

Example 3. Let $X = \{\{1,2\},\{1,3\},\{1,4\}\}$. Then there is no permutation Π for which $X \subseteq B(\Pi)$ for any permutation Π.

Then the solution to the Steiner tree problem in (M_n, d) is a lower bound to the solution of the problem in $(\mathcal{P}_n, d)$.

The Steiner tree problem in $(\mathcal{P}_n, d)$ is harder than in (M_n, d) because the intervals sets of a permutation, or sets compatible with a permutation, are highly constrained, whereas in (M_n, d) it suffices to treat each subset of $\{1, \cdots, n\}$ separately. To see this, note that the

symmetric difference between two points $X, Y \in M_n$ may be written

$$d(X,Y) = \sum_{\sigma \subset \{1,\cdots,n\}} |\chi_\sigma(X) - \chi_\sigma(Y)|, \tag{2.7}$$

where χ_σ is the indicator function of σ. Then, writing $X(v)$ for the element of M_n associated with vertex v of a tree, the length of any tree T satisfies

$$\begin{aligned} L(T) &= \sum_{uv \in E} d(X(u), X(v)) && (2.8) \\ &= \sum_{uv \in E} \sum_{\sigma \subset \{1,\cdots,n\}} |\chi_\sigma(X(u)) - \chi_\sigma(X(v))| && (2.9) \\ &= \sum_{\sigma \subset \{1,\cdots,n\}} \sum_{uv \in E} |\chi_\sigma(X(u)) - \chi_\sigma(X(v))| && (2.10) \end{aligned}$$

so that the minimization of $L(T)$ may be done component-wise, i.e., separately for each $\sigma \subset \{1, \cdots, n\}$. In Section 2.6 then, we will discuss only whether or not each ancestral vertex contains σ or not, which can be phrased as the tree optimization problem for a single zero-one character.

2.6 The small Problem for a Single Zero-One Variable

Consider that the data at each terminal vertex consist of either a zero or a one, representing the presence or absence of a set σ. Dynamic programming for ancestral node optimization requires two passes. In the forward pass, from the terminal nodes towards the root, the value of the variable (the presence or absence of σ) may be established definitely at some ancestral vertices, while at other vertices it is left unresolved until the second, "traceback" pass, when any multiple solutions are also identified.

Suppose ancestral vertex v has p daughter vertices $u_1, \cdots, u_p$, where $p \geq 2$. (If $v \neq r$, then v has degree $p+1$, if $v = r$, then v has degree $p \geq 3$.) In practice, it usually suffices to allow only $p = 2$ for ancestral vertices and $p = 3$ for the root (binary trees), and indeed our algorithm is programmed for this case. Nevertheless, in this section we will give the general solution, which may be required in some contexts, and is of mathematical interest in any case. In the next section we will discuss the simplification to binary trees.

Suppose for each daughter u_i, we have already decided whether $\sigma \in u_i$ definitely, possibly,

or definitely not. Let

$$q(\sigma) = \#\{i|\sigma \in u_i \text{ definitely}\} \tag{2.11}$$

$$Q(\sigma) = \#\{i|\sigma \in u_i \text{ definitely or possibly}\}. \tag{2.12}$$

If $v \neq r$ and $q(\sigma) \geq \frac{p}{2} + 1$, then $\sigma \in v$ definitely. This is true because then

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \leq \frac{p}{2} - 1 \tag{2.13}$$

whereas if σ were not in v,

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \geq \frac{p}{2} + 1 \tag{2.14}$$

which is clearly not optimal, no matter whether σ is in the ancestor of v or not.

On the other hand, if $v \neq r$ and $Q(\sigma) \leq \frac{p}{2} - 1$, then definitely $\sigma \notin v$. This is true because then

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \leq \frac{p}{2} - 1 \tag{2.15}$$

whereas if σ were in v,

$$\sum_{v \to u} |\chi_\sigma(u) - \chi_\sigma(v)| \geq \frac{p}{2} + 1 \tag{2.16}$$

which is clearly not optimal, no matter whether σ is in the ancestor of v or not.

This leaves the cases where both

$$q(\sigma) < \frac{p}{2} + 1 \tag{2.17}$$

$$Q(\sigma) > \frac{p}{2} - 1, \tag{2.18}$$

where we say $\sigma \in v$ possibly.

As for r, if $q(\sigma) > \frac{p}{2}$, then $\sigma \in r$ definitely; if $Q(\sigma) < \frac{p}{2}$, then $\sigma \notin r$ definitely; otherwise $\sigma \in r$ possibly.

While applying the traceback of the recurrence, we reassign the "possible" memberships in ancestral vertices either to definite memberships or to definite exclusions. For ancestral vertex v, whether or not $\sigma \in t$, the ancestor of v, has no bearing if $\sigma \in v$ definitely or $\sigma \notin v$ definitely. Otherwise, if $\sigma \in t$ and $1 + q(\sigma) > \frac{p+1}{2}$, then we reassign $\sigma \in v$ definitely. If $\sigma \notin t$ and $Q(\sigma) < \frac{p+1}{2}$, then we reassign $\sigma \notin v$ definitely.

In the remaining cases if $\sigma \in t$ and $1 + q(\sigma) \leq \frac{p+1}{2} \leq 1 + Q(\sigma)$, or if $\sigma \notin t$ and

$Q(\sigma) \geq \frac{p+1}{2} \geq q(\sigma)$, then we can assign or exclude σ from v, without affecting $L(T)$. Note that if $\sigma \in r$ possibly, we are also free to assign it to r or not at the beginning of the traceback.

Note that while the dynamic programming can find a solution in time linear in N, the number of different solutions may be exponential in N. Considering all possible sets σ, the number of solutions is also exponential in n.

2.7 Binary Trees

The case of binary branching trees, where $p + 1 = 3$ except at the root where $p = 3$, is somewhat simpler in the traceback as there is at most one free choice of assignment, and that is at r. For any other ancestral vertex v, membership or not of σ in t, the ancestor of v, always determines its membership, or not, in v.

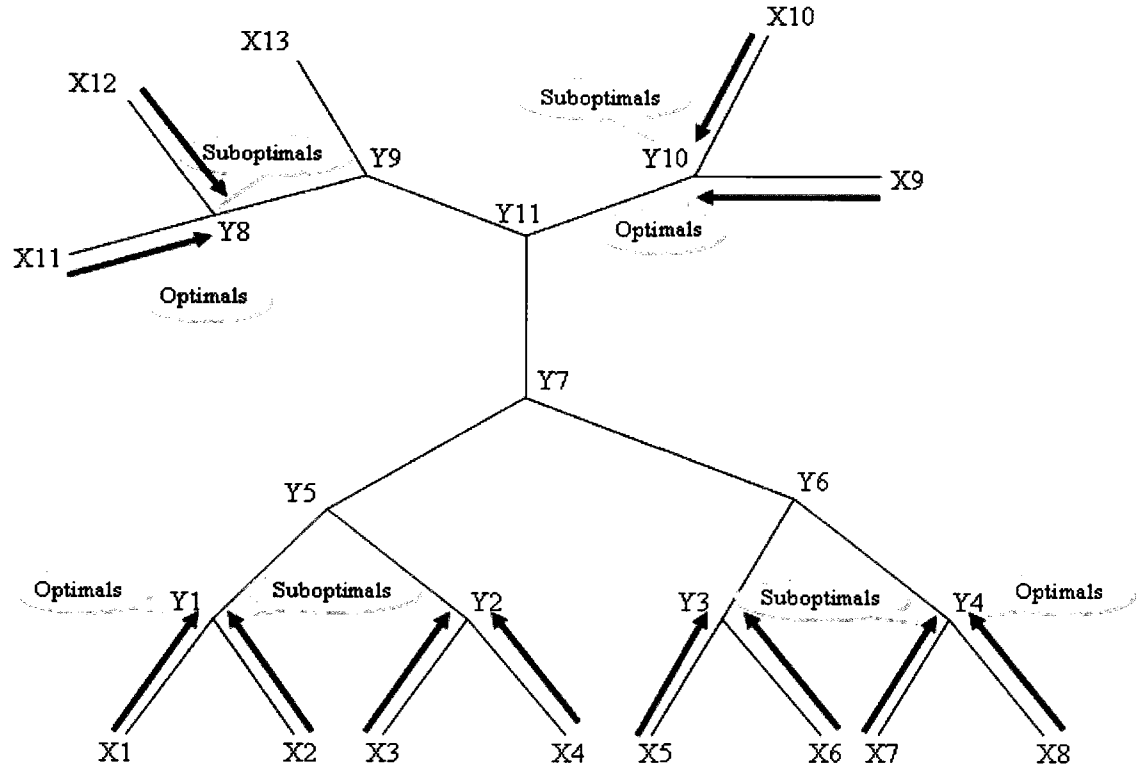
Moreover, for the big phylogeny problem, it suffices to search for the optimal binary tree. If the optimal tree is not binary, this will still show up as a binary tree with one or more edges of length $d = 0$.

Nevertheless, even for the small problem, because of the possible choice at r, when we consider all σ, the number of solutions may be exponential in n.

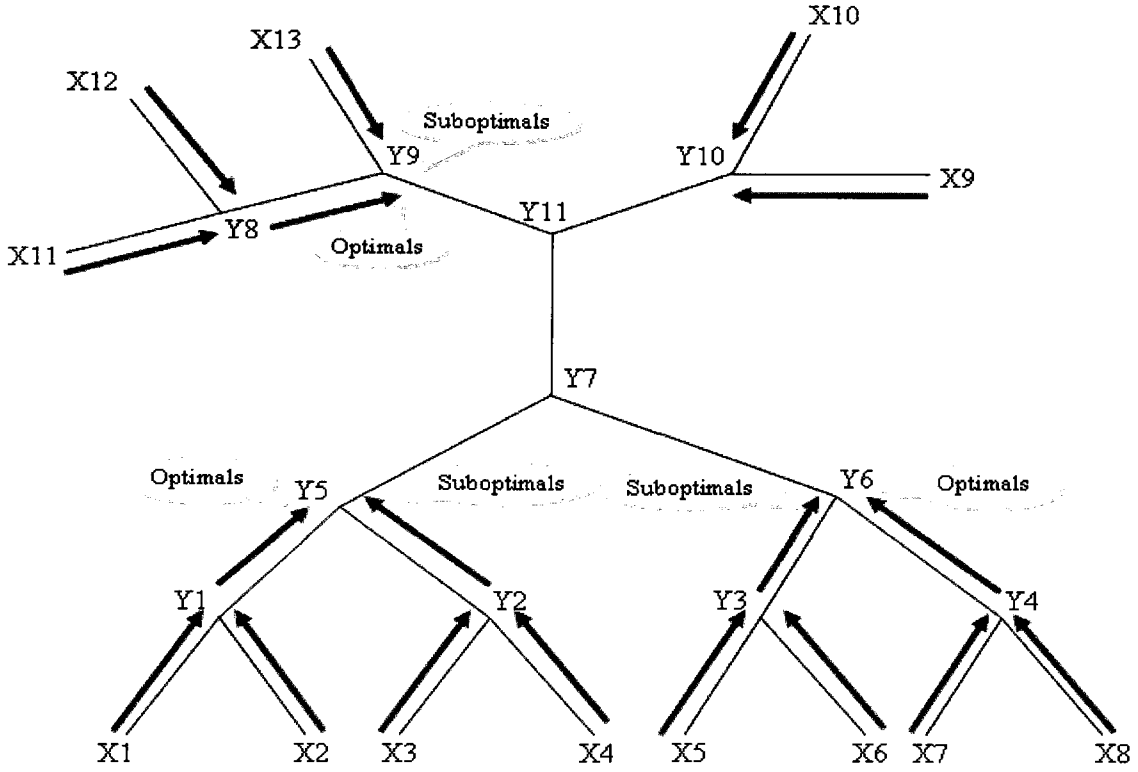
Figures 2.1 and 2.2 show graphically the steps of the first phase of the dynamic programming approach and the second phase of the dynamic programming approach, respectively, on an unrooted binary tree.

2.8 Handling Incompatible Subsets in $\mathcal{P}_n$ with PQ-Trees

In the case of binary trees, after the forward pass of the dynamic programming, those σ for which $q(\sigma) = 2$ at any ancestral vertex v must all be in $B(\Pi)$ for some permutation Π, namely any terminal vertex permutation Π for which v is an ancestor. In the case of r, those σ for which $q = 3$ must be in the interval sets of all the terminal vertex permutations in $\mathcal{S}$. In this special case, the solution in (M_n, d) is also a solution for $(\mathcal{P}_n, d)$.


X10
X13
X12
Suboptimals
Suboptimals
Y9
Y10
X9
Y11
Y8
X11
Optimals
Optimals
Y7
Y5
Y6
Optimals
Y1
Suboptimals
Y2
Y3
Suboptimals
Y4
Optimals
X1
X2
X3
X4
X5
X6
X7
X8

(a)


X10
X13
X12
Suboptimals
Y9
Y10
X9
Y11
Y8
X11
Optimals
Y7
Optimals
Y5
Suboptimals
Suboptimals
Y6
Optimals
Y1
Y2
Y3
Y4
X1
X2
X3
X4
X5
X6
X7
X8

(b)

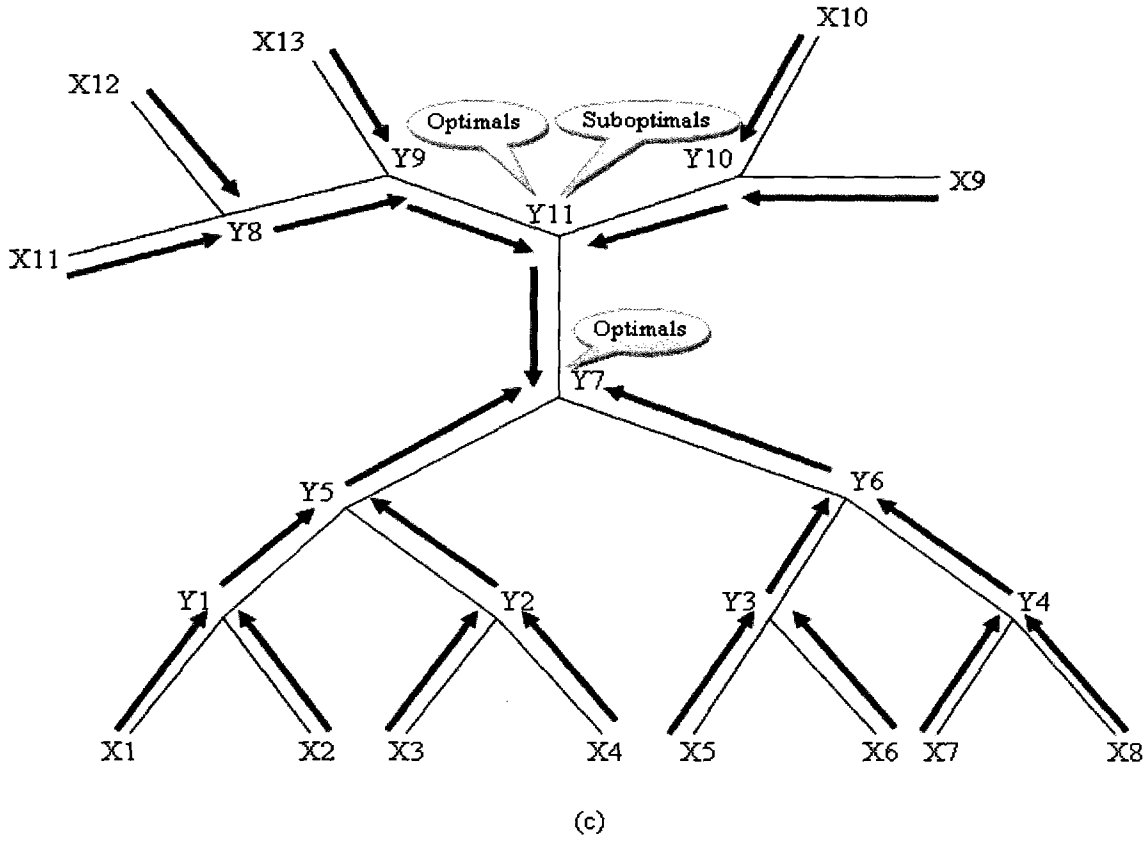
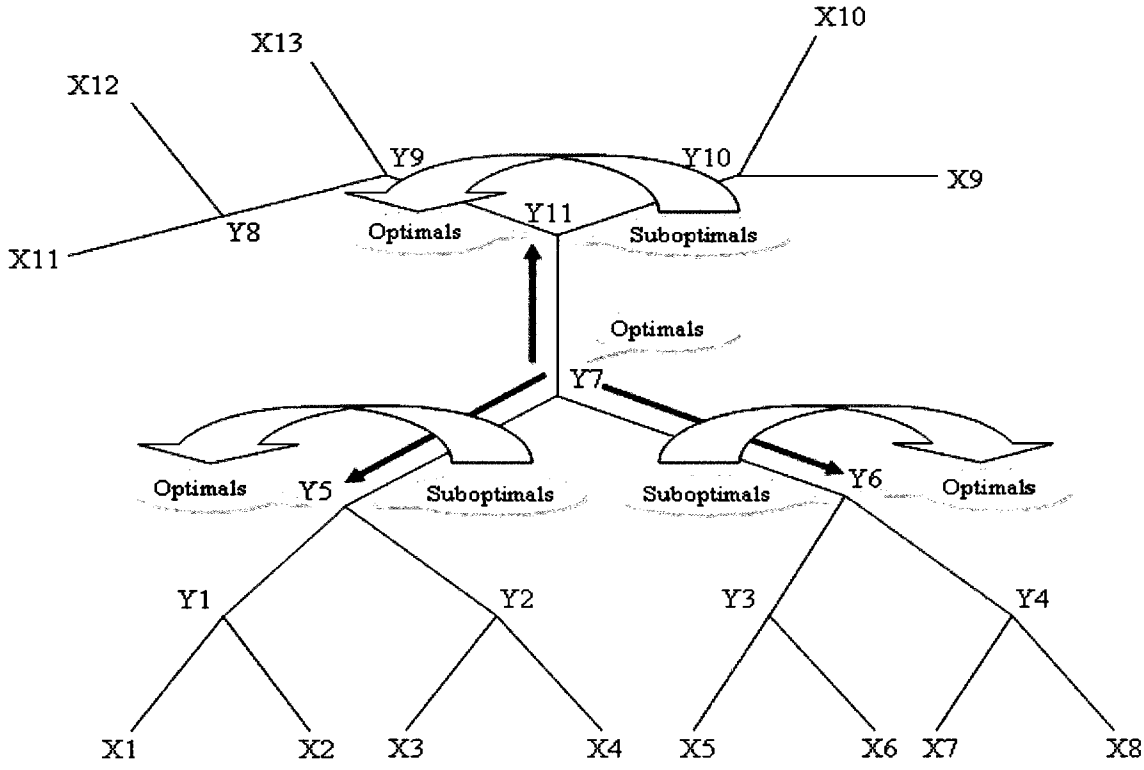
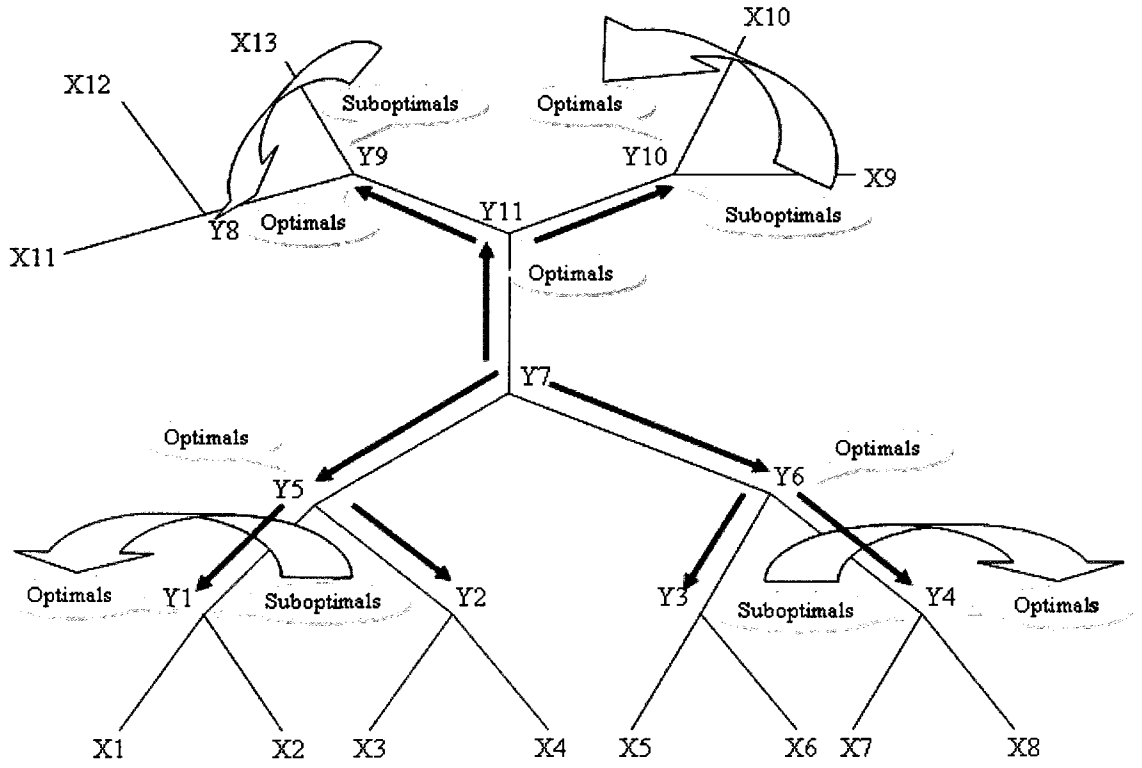


Figure 2.1: Illustration of first phase of the dynamic programming approach. The Xs represent current species and the Ys represent the ancestors to be inferred. (a) and (b) represent computing the set of optimal intervals (Optimals) and the set of suboptimal intervals (Suboptimals) for the first level of ancestors and for the second level of ancestors of an unrooted binary tree, respectively. Due to lack of space, the two sets of Y2 and Y3 are not shown. (c) represents computing the set of optimal intervals (Optimals) and the set of suboptimal intervals (Suboptimals) for the third level of ancestors (Y11) and for the "root" (Y7) of the unrooted binary tree.


X10
X13
X12
Y9
Y10
X9
Y11
Y8
Optimals
Suboptimals
X11
Optimals
Y7
Optimals
Y5
Suboptimals
Suboptimals
Y6
Optimals
Y1
Y2
Y3
Y4
X1
X2
X3
X4
X5
X6
X7
X8

(a)


X10
X13
Suboptimals
Optimals
X12
Y9
Y10
X9
Y8
Optimals
Y11
Suboptimals
X11
Optimals
Y7
Optimals
Optimals
Y5
Y6
Optimals
Suboptimals
Y1
Y2
Y3
Y4
Suboptimals
Optimals
X1
X2
X3
X4
X5
X6
X7
X8

(b)

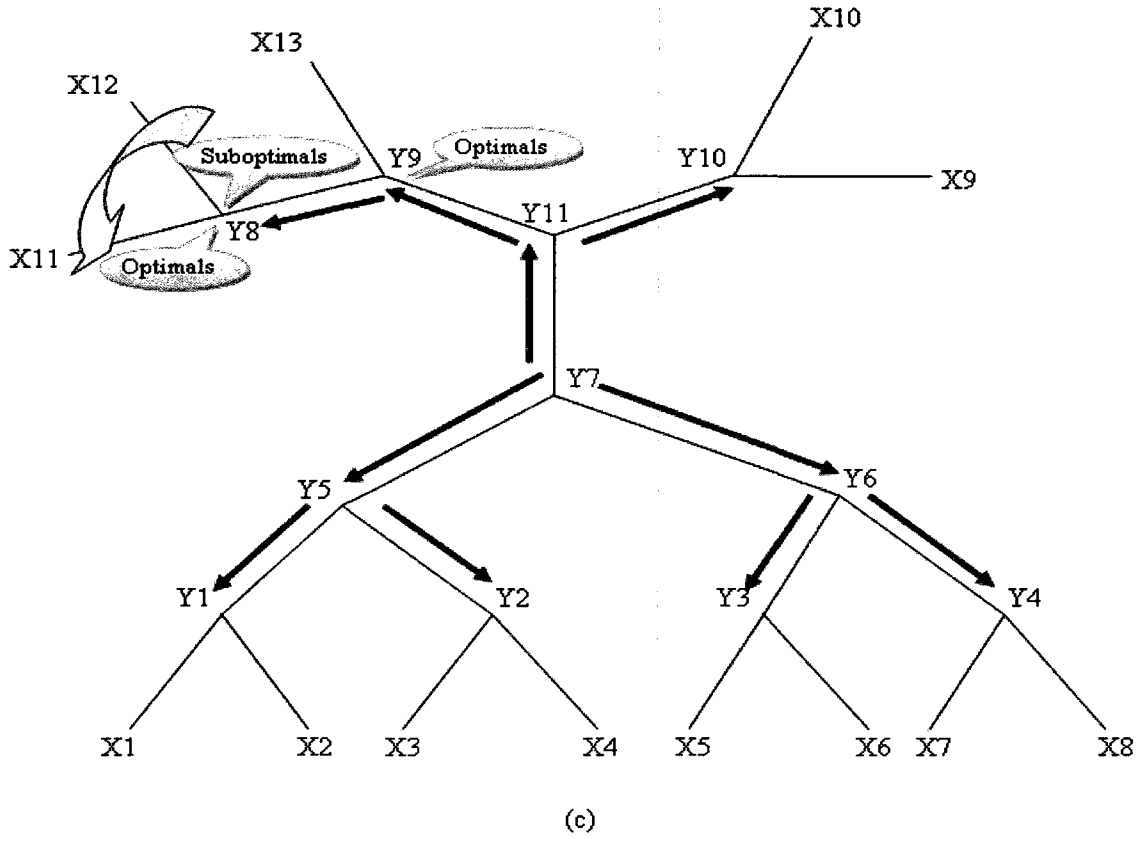


Figure 2.2: Illustration of second phase of the dynamic programming approach. (a) represents updating the set of optimal intervals of each descendant of the root by the intervals resulting from the intersection of the set of optimal intervals of the root and the set of suboptimal intervals of that descendant. (b) and (c) represent the recursive implementation of the step (a) on the descendants of the root. The recursion stops at the first level of ancestors. Due to lack of space, the two sets of Y2 and Y3 are not shown.

In general, however, this is not the case. In the following example the Steiner tree in M_n is shorter than that in $\mathcal{P}_n$.

Example 4. Let $\mathcal{S} = \{(4,1,2,3),(2,1,3,4),(3,1,4,2)\}$, and suppose T has a single ancestor r.

Then in M_n, we calculate

$q(\{1,2\}) = q(\{1,3\}) = q(\{1,4\}) = 2,$

$q(\{1,2,3\}) = q(\{1,3,4\}) = q(\{1,4,2\}) = 2,$

$q(\{2,3,4\}) = 0$,

$q(\{2,3\}) = q(\{3,4\}) = q(\{4,2\}) = 1$,

so that the only proper subsets in r are $\{1,2\}, \{1,3\}, \{1,4\}, \{1,2,3\}, \{1,3,4\}, \{1,4,2\}$. Each of the terminal vertices is missing two of these subsets but contains one that is not in r, so that $\sum d = 3 \times (2+1) = 9$.

In $\mathcal{P}_n$, there are multiple solutions to optimizing r, including the permutations $(4,1,2,3)$, $(2,1,3,4)$ or $(3,1,4,2)$ themselves. Each one of the latter shares only two interval sets with each other, so that $\sum d = 12$. Thus the difficulty with working in $\mathcal{P}_n$ is the requirement that the set of subsets at an ancestral vertex has to correspond to a permutation, or at least be compatible with some permutation, while this restriction does not apply in M_n.

There is a data structure particularly well-suited for representing a set of subsets compatible with a permutation, namely the PQ-tree [34]. A PQ-tree is just a rooted tree with terminal vertices $1, \cdots, n$, where the daughter vertices of some special non-terminal vertices may be "ordered", though the left-right or right-left direction of the order is not specified. For an ordered vertex x, the blocks of terms spanned by the various daughters of x must be in the same order for any permutation compatible with the tree, while there is no such constraint on the vertices which are not ordered.

Given a PQ-tree and a new subset X, it is possible to rapidly check whether X is compatible with the other subsets previously used to build the PQ-tree and to update the PQ-tree to include the additional ordering information, if any, in X. PQ-trees have already been utilized in gene order phylogenetics [19, 35].

To ensure optimality in our procedure, we would have to interpret u and v in recurrence (2.6) as PQ-trees. The difficulty would be to carry out the minimization over all possible $u_1, \cdots, u_p$, i.e., to find a constraint limiting the set of possible PQ-trees for u that could be in a solution if a given PQ-tree for v is. This is a problem even for binary trees; the search space of all possible pairs of PQ-trees cannot be reduced to a very limited set as we did in Equation (2.10) and Section 2.6. The actual calculation of the symmetric difference induced by two PQ-trees is not time-consuming. One approach to this problem might be found in the concept of PQR-trees [36, 37], which could include a limited number of intervals incompatible with one of the two descendants of v, but we have not explored this.

Instead, we proceed heuristically, constructing a PQ-tree at each vertex only during the traceback. This is motivated by the observation in our test data, that in the traceback it is

rare for the following configuration to occur:

- t is the ancestor of v, and v is the ancestor of u_1 and u_2.
- σ_1 and σ_2 are both definitely in the PQ-tree at t, after the traceback step at t.
- σ_i is in u_j only for $i = j$ but not for $i \neq j$, before the traceback.
- either one of σ_1 and σ_2 can be added to PQ-tree defined by the pre-traceback definites at t, but not both.

If this never happens, then we can be sure our result is optimal. If it does happen, we assign as many such σ to the definites of v as possible, using a greedy approach with larger intervals tried before smaller intervals.

2.9 Unequal Gene Complements

Doing gene order phylogeny on realistic sets of genomes which contain different sets of genes is recognized as a substantially more difficult than the case with identical gene sets in all species [38, 39].

There is a natural solution within our framework. There are two steps to this solution.

First, the presence or absence of each gene at each vertex is determined by dynamic programming to minimize the number of gene deletions or insertions across all edges of the tree. This is done gene-by-gene, analogous to the set-by-set procedure in Section 2.5. Figures 2.3 and 2.4 show graphically the steps of the first phase of the dynamic programming approach and the second phase of the dynamic programming approach, respectively, on an unrooted binary tree to determine the gene content of ancestors.

Second, in our main algorithm, when the definites for a vertex v are being decided in the recurrence step, any interval σ in one of its descendants u_1 containing a gene g absent from the other genome u_2 is assigned to be a definite of v if $\sigma \backslash \{g\}$ is present in u_2. This includes the case where $\sigma = \{g, h\}$ and gene h has previously been decided to be present in u_2.

In building the PQ-tree for u_2 during the traceback, if σ is definite for v, and $\sigma \backslash \{g\}$ is a possible for u_2, then it is a candidate for inclusion in the PQ-tree. In calculating the symmetric difference between the sets of intervals from two genomes, we first delete from consideration any gene that is not present in both genomes, and remove any null sets or duplicate sets.

It can be shown that this generalizes the minimization of the sum of symmetric differences across the tree, given the correctness of our dynamic programming solution for gene presence or absence at ancestral vertices. In particular, in the case of identical gene complements, it reduces to our original method.

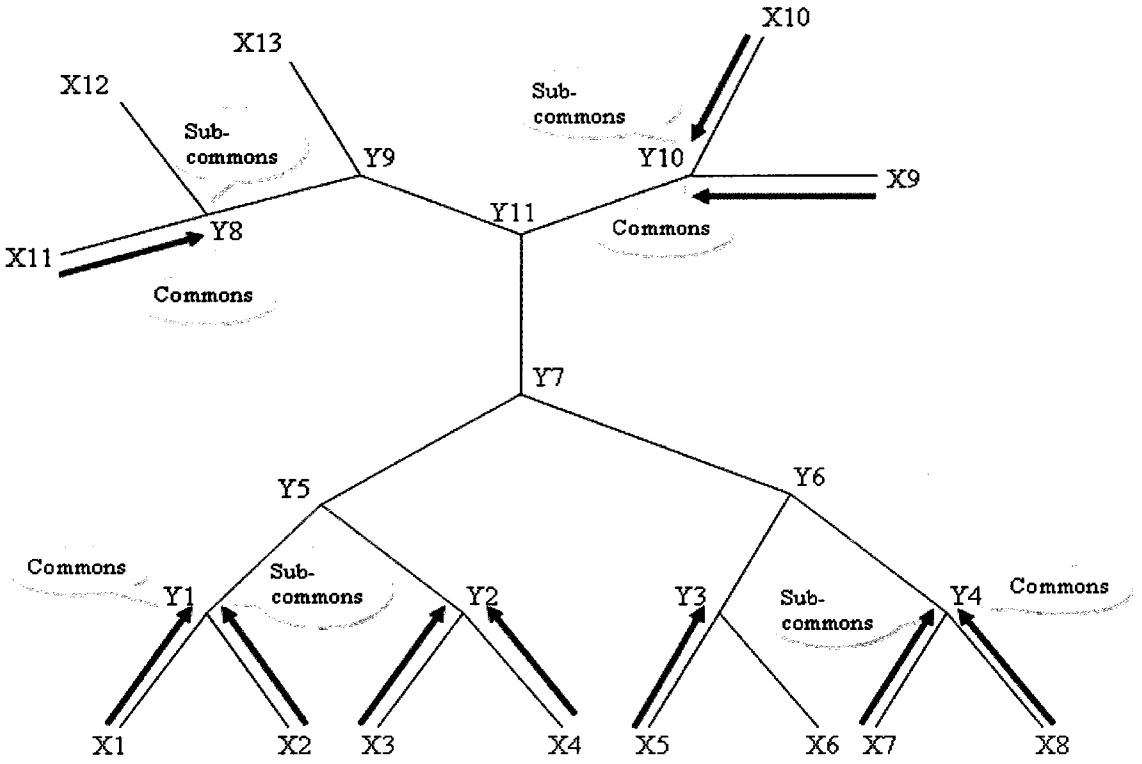
2.10 Duplicate Genes

The introduction of gene duplicates and gene families into genome comparisons causes even more difficulty than unequal gene complements. This either calls for changing the analytical framework from comparing permutations to comparing strings, or integrating gene-tree/species-tree methodology into our procedures, or both. The method proposed in this chapter, however, formally applies directly to data containing occasional duplicate genes. The construction and comparison of sets of intervals is not complicated by duplicates, except that some convention must be adopted for intervals containing two or more copies of the same gene, i.e., only include one copy per interval or all copies, requiring the same number of copies for identity of two intervals. We did not encounter this problem with our test data, so we have not yet explored it further.

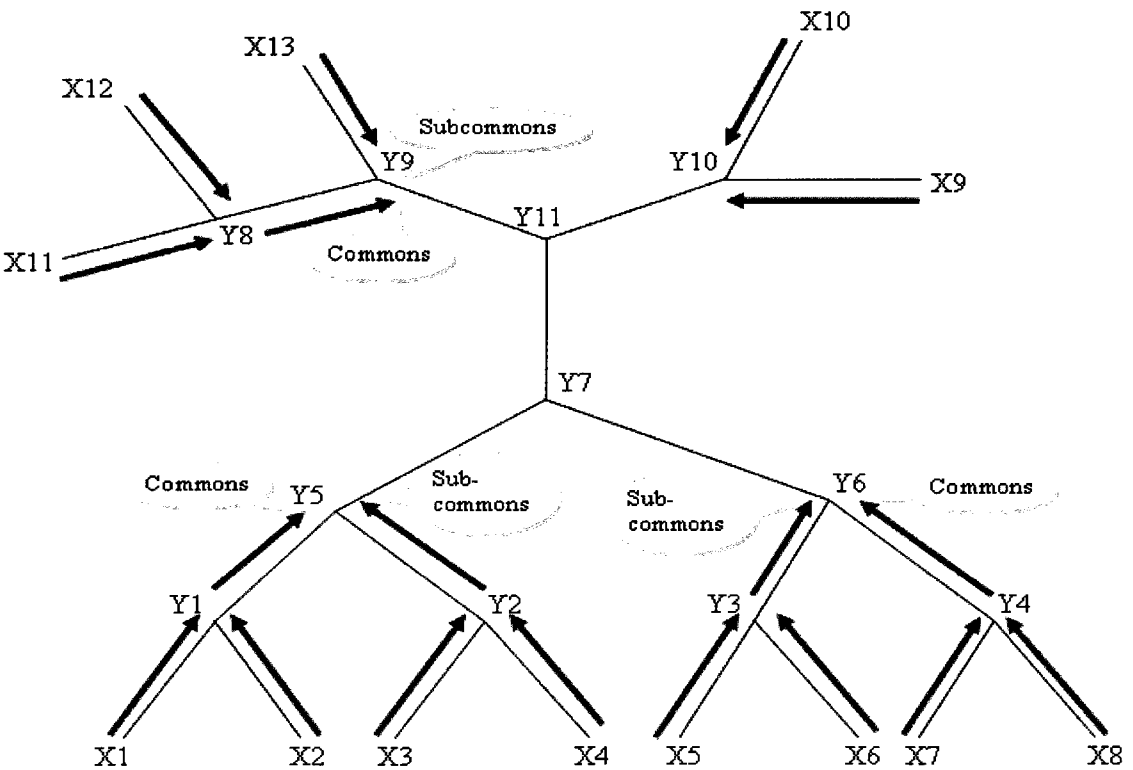
2.11 Application to Chloroplast Genomes

New sequences of the chloroplast genome allow the exploration of the evolution of the streptophytes, a phylum containing several classes of algae but also the familiar multicellular land plants. Our data includes gene orders from *Mesostigma viride* [40], *Chlorokybus atmophyticus* [unpublished data], *Staurastrum punctulatum* [41], *Zygnema circumcarinatum* [41], *Chaetosphaeridium globosum* [42], *Chara vulgaris* [43], as well as the land plant *Marchantia polymorpha* [44], with 120-140 genes per genome. The first six of these represent five of the six major charophycean (i.e., other than land plants) lineages.

We use the complete gene order of these genomes, including duplicate genes and genes not present in some organisms. There are 148 distinct genes, 35 of which were absent from one or more of the genomes. Five of the genomes had six or eight duplicated genes, largely the same ones in each, while the remaining two genomes had one or zero duplications, respectively.


X10
X13
X12
Sub-
commons
Sub-
commons
Y9
Y10
X9
Y11
Y8
X11
Commons
Commons
Y7
Y5
Y6
Commons
Y1
Sub-
commons
Y2
Y3
Sub-
commons
Y4
Commons
X1
X2
X3
X4
X5
X6
X7
X8

(a)


X10
X13
X12
Subcommons
Y9
Y10
X9
Y11
Y8
X11
Commons
Y7
Commons
Y5
Sub-
commons
Sub-
commons
Y6
Commons
Y1
Y2
Y3
Y4
X1
X2
X3
X4
X5
X6
X7
X8

(b)

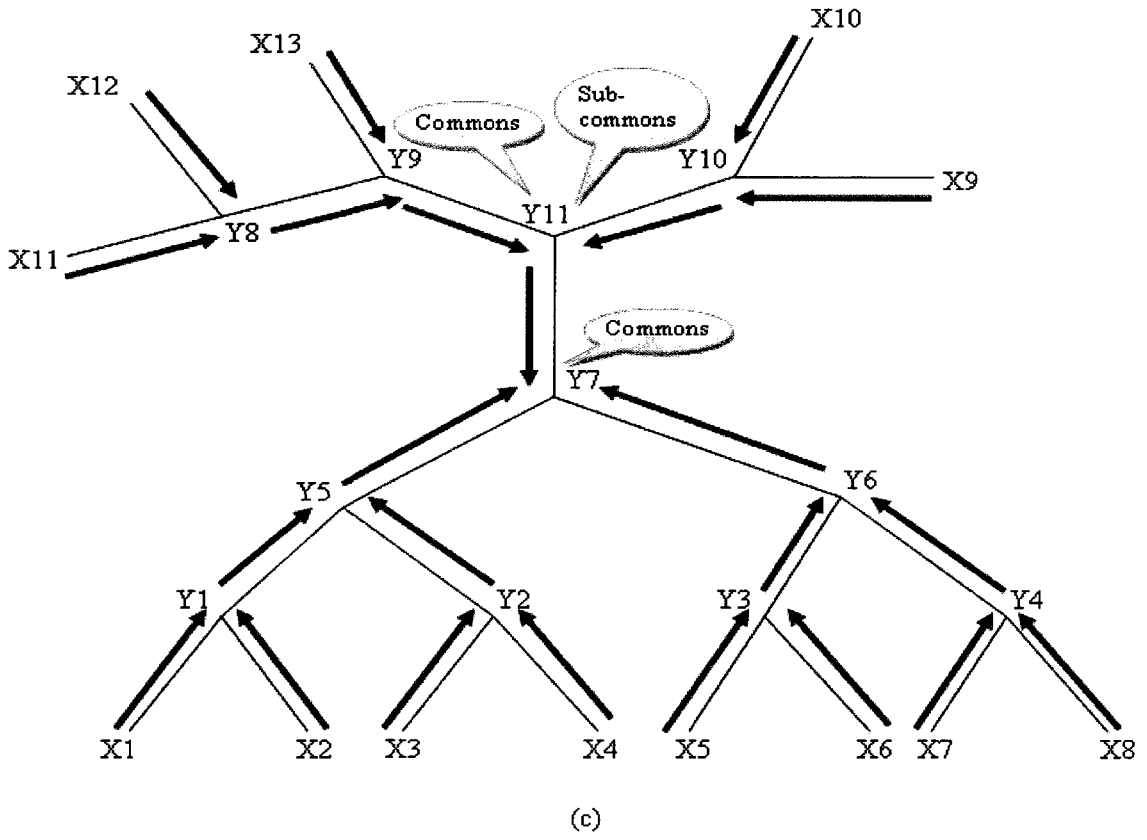
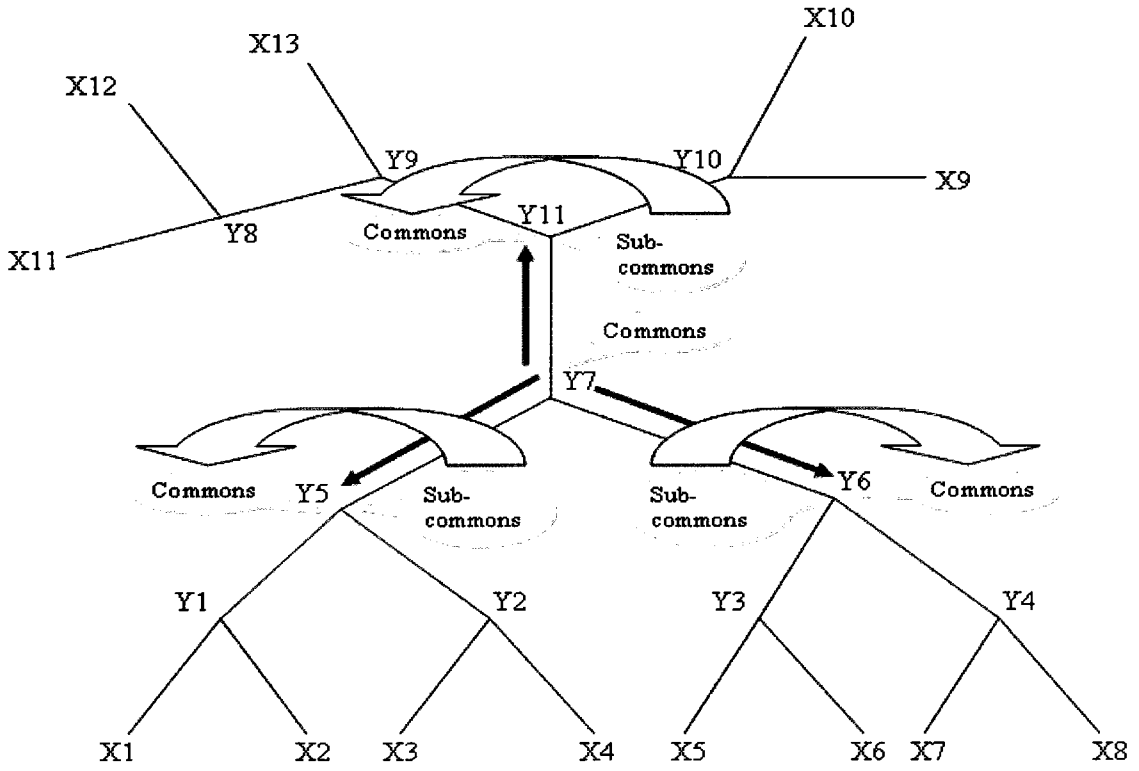
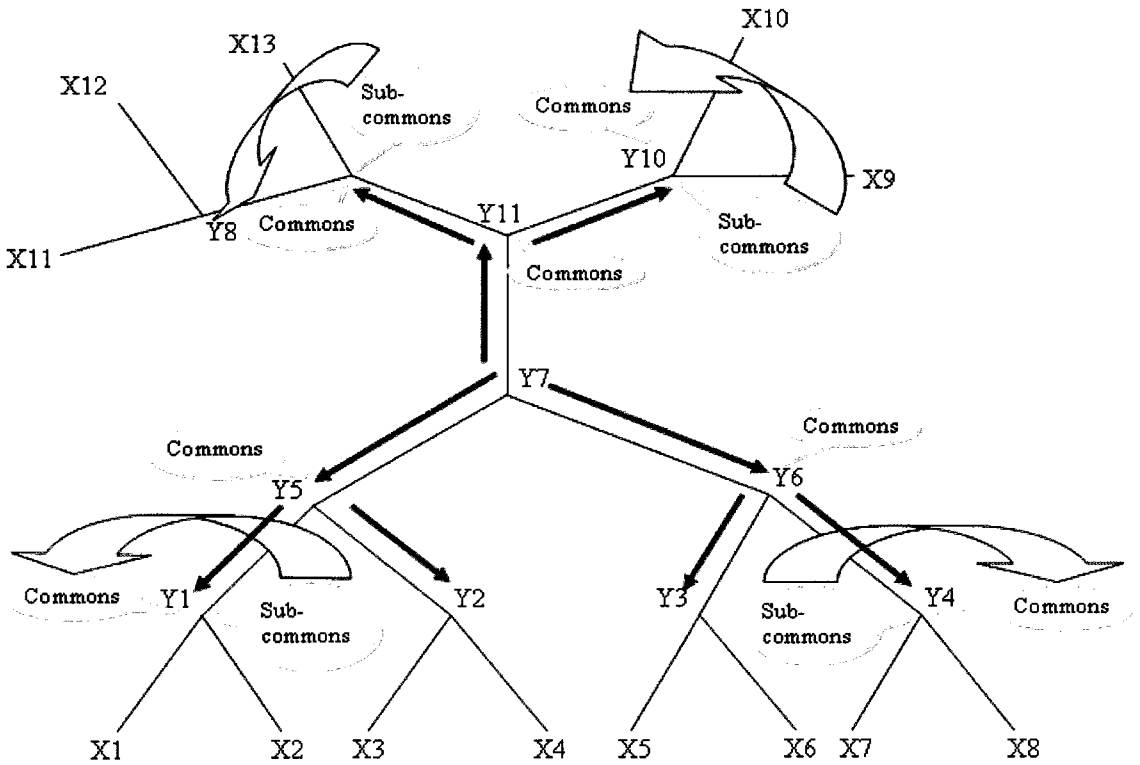


Figure 2.3: Illustration of first phase of the dynamic programming approach to infer the gene content of ancestors. The Xs represent current species and the Ys represent the ancestors to be inferred. (a) and (b) represent computing the set of common genes (Commons) and the set of subcommon genes (Subcommons) for the first level of ancestors and for the second level of ancestors of an unrooted binary tree, respectively. Due to lack of space, the two sets of Y2 and Y3 are not shown. (c) represents computing the set of common genes (Commons) and the set of subcommon genes (Subcommons) for the third level of ancestors (Y11) and for the "root" (Y7) of the unrooted binary tree.


X10
X13
X12
Y9
Y10
X9
Y11
Y8
X11
Commons
Sub-
commons
Commons
Y7
Commons
Y5
Sub-
commons
Sub-
commons
Y6
Commons
Y1
Y2
Y3
Y4
X1
X2
X3
X4
X5
X6
X7
X8

(a)


X10
X13
Sub-
commons
X12
Commons
Y10
X9
Y11
Y8
Commons
Sub-
commons
X11
Commons
Y7
Commons
Commons
Y5
Y6
Commons
Y1
Sub-
commons
Y2
Y3
Sub-
commons
Y4
Commons
X1
X2
X3
X4
X5
X6
X7
X8

(b)

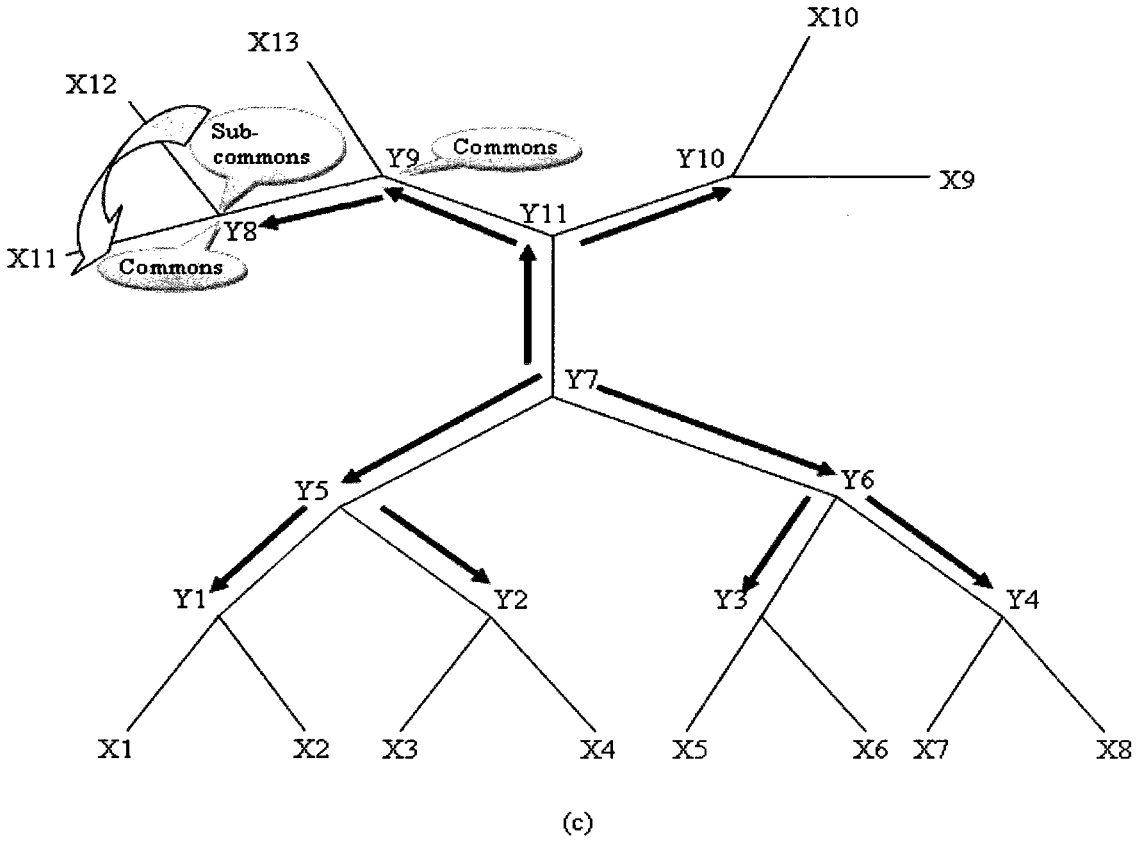


Figure 2.4: Illustration of second phase of the dynamic programming approach to infer the gene content of ancestors. (a) represents updating the set of common genes of each descendant of the root by the intervals resulting from the intersection of the set of common genes of the root and the set of subcommon genes of that descendant. (b) and (c) represent the recursive implementation of the step (a) on the descendants of the root. The recursion stops at the first level of ancestors. Due to lack of space, the two sets of Y2 and Y3 are not shown.

There has been some controversy among phycologists about the origin of land plants, in particular whether they represent a sister grouping to the Charales, represented here by *Chara vulgaris* or a sister grouping to the Coleochaetales, represented here by *Chaetosphaeridium globosum*. The best tree we obtained is depicted in Figure 2.5. We note the correct grouping together of the two Zygnematalean algae *Staurastrum* and *Zygnema*. The tree also correctly depicts the early branching of *Mesostigma* and *Chlorokybus*, though it does not bear on the controversy of whether *Mesostigma* is actually a streptophyte, or whether its

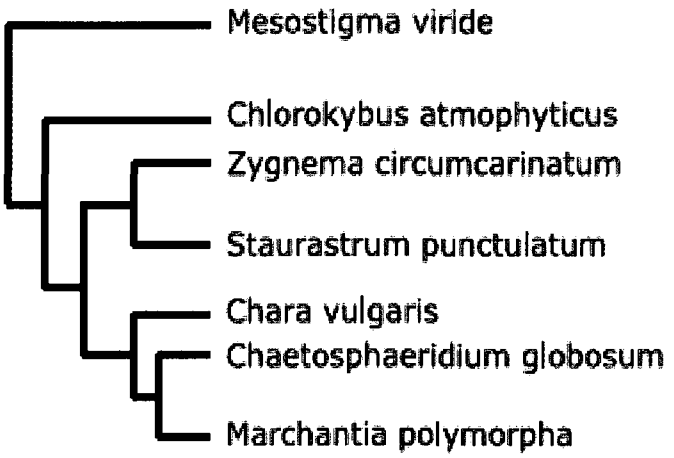


Figure 2.5: Best tree for six algae and the land plants. Rooting and branch lengths arbitrary.

origin predates that of the streptophytes in algal evolution. Of particular interest is the grouping of *Marchantia* with *Chaetosphaeridium* instead of with *Chara*. Previous analysis of the same genomes, at both the sequence and gene-order levels, suggested that *Chara* branches early and that the Zygnematale-Coleochaetale lineages group with the land plants [43]. The early branching of the Zygnematales was seen to be a much less parsimonious solution. Nevertheless, the results in Figure 2.5 represent an additional line of evidence in the still unsettled problem of streptophyte phylogeny.

2.12 Validation Through Simulations

We stressed that our method is not motivated by any particular model of genome rearrangement. To test its behavior, however, we generate multiple data sets on the tree in Figure 2.5, using a particular rearrangement model and then see how often our method reconstructs the underlying tree. We calculate the edge lengths for this tree by implementing a minimal rearrangements algorithm [52] similar to MGR [12].

Once the edge lengths are available, we generate new data sets. For simplicity's sake, we do not take account of the processes of gene duplication and gene loss seen occasionally in the real chloroplast genomes. We simply assume that all seven simulated genomes have the same 128 genes, representing the average number in the real data. Starting with an

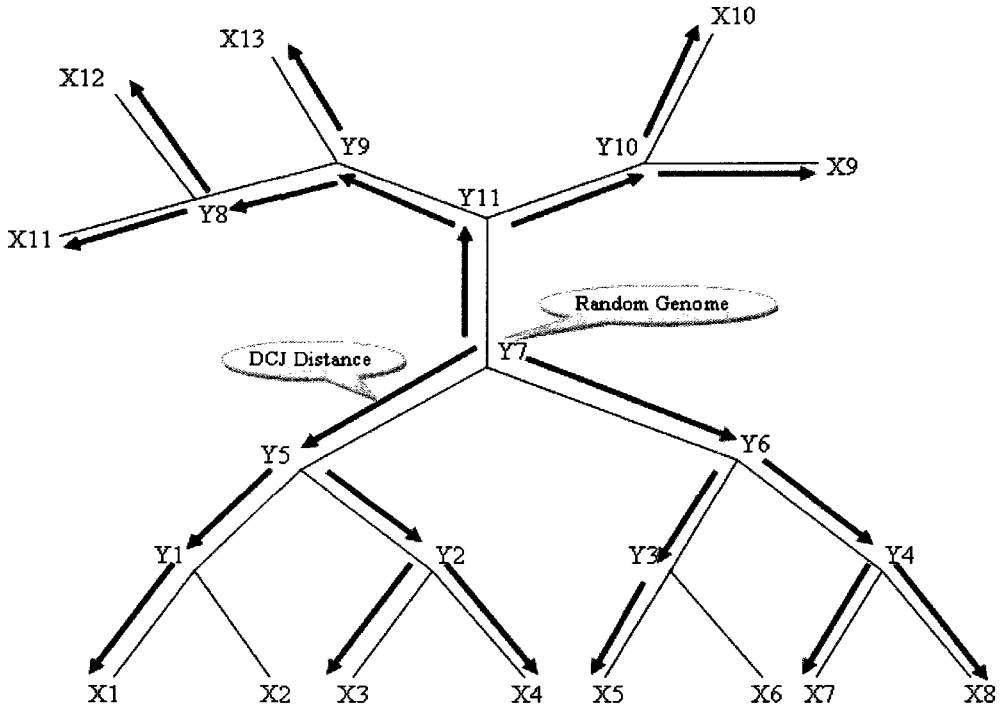


Figure 2.6: Illustration of random generation of terminal genomes on an unrooted binary tree.

arbitrary gene order in the most ancestral vertex, we generate new gene orders for its three neighbouring vertices, by means of the same number of random rearrangements (i.e., random inversions) inferred for the connecting edge in the real data. For each of the other ancestral vertices, once its gene order is established, we similarly determine the gene order of its two as yet undetermined neighbour vertices, using the same number of random rearrangements inferred for the connecting edges in the real data.

This strategy makes no effort to conserve local structure; the breakpoints of the inversions are chosen independently at random.

Figure 2.6 shows the graphical illustration of random generation of terminal genomes on an unrooted binary tree.

The data sets are then analyzed using the symmetric difference criterion to find the best tree. Out of 40 data sets generated according to the tree of Figure 2.5, 91% reconstructed the same tree, while the remaining 9% reconstructed a tree which differed only in suggesting an (implausible) paraphyletic configuration of the *Zygnematales*, with *Staurastrum* branching

off earlier than *Zygnema*.

Thus, at least for the particular random rearrangement model and specific tree examined here, when random data sets generated on the tree are analyzed according to the symmetric difference on conserved interval sets, the results are quite accurate. This holds even though our method does not make use of strandedness information about the genes, information crucial to other methods for reconstructing the rearrangement history of the genomes.

This performance also holds for other trees we have examined, with the proviso that no interior edge length is too short; otherwise all methods tend to produce two or more trees with little difference in the optimality criterion.

2.13 Implementation

Our current implementation of the method includes an exhaustive search of binary trees only for the "big" problem, plus heuristics when N is larger than 9 or 10. The algorithm for the "small" problem is also the binary tree version. We have tested it for values of n in the range of 100-200. The PQ-tree construction notifies when a conflict is detected and resolved, and the pre-calculation of gene content of ancestral nodes is also implemented. The ability to handle duplicate genes is inherent in the algorithm and requires no special consideration.

2.14 PQ-Trees

What is the relationship between PQ trees and reconstructing phylogenomic trees?

In the second phase, trace-back, of the dynamic programming algorithm used in [45] where we traverse the tree top-down, we try to optimize each internal node u by reassigning its intervals with "possible" memberships either definite memberships, if they appear in direct ancestor v of u, or definite exclusions, otherwise.

The problem is that some of u's intervals with "possible" memberships that should be reassigned definite memberships are not compatible with the u's intervals that have already definite memberships. Fortunately, this problem has a solution: PQ-trees is a data structure invented in 1976 by Booth and Lueker [34] to solve the general consecutive arrangement problem that is defined as follows: Given a finite set U of n elements and a collection S of subsets of U, does there exist a permutation π of U in which the members of each subset

$s \in S$ appear as a consecutive substring of π ?

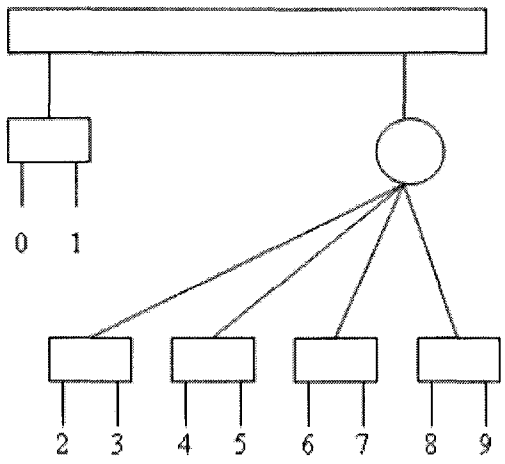
In [34] Booth and Lueker found a linear time algorithm to solve this problem. A PQ-tree is a rooted tree whose internal nodes are of two types: P and Q. The children of a P-node can be in any order, while the children of a Q-node can be in a left-to-right or in a right-to-left order.

A P-node is represented as circle and a Q-node as a rectangle. The leaves of a PQ-tree are labeled bijectively by the n elements of U, such that the sequence resulting from reading the labels of the tree leaves from left to right is a permutation of U, which is called the *frontier* of a PQ-tree.

Example 5. Let us have a set $U = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and a collection of subsets $S = \{\{0, 1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}, \{8, 9\}, \{5, 4, 2, 3\}, \{7, 6, 4, 5\}, \{3, 2, 8, 9, 7, 6\}, \{9, 8, 4, 5, 3, 2, 7, 6\}, \{1, 0, 4, 5, 9, 8, 2, 3, 7, 6\}, \{0, 1, 6, 7, 2, 3, 9, 8, 4, 5\}, \{8, 9, 2, 3, 6, 7, 4, 5, 0, 1\}\}$. The PQ-tree T in Figure 2.7 can represent each subset of S in a consecutive substring of its frontier.

Actually, the tree T can represent more subsets than those in S, however this does not mean that T can represent any additional subset that we might add to S after all subsets of S have been processed by T. For example, T can not represent the permutation: π = 6 7 2 3 0 1 9 8 4 5, because the root node in T is a Q-node, so its children can be in a left-to-right or in a right-to-left order, which means that the two elements 0 and 1 should appear either at the beginning or at the end of any permutation represented by T. The algorithm found by Booth and Lueker helps to detect such an incompatible permutation.

Since 1976, PQ-trees have been used for many applications such as testing the consecutive ones property in matrices, interval graphs, and graph planarity. However, PQ-trees received recently more consideration from some researchers in bioinformatics as they needed it in their algorithms [16, 19, 34, 46, 47]. During the traceback of the dynamic programming algorithm used in [45], we build a PQ-tree at each internal node u. The PQ-tree of u is reconstructed based on u's intervals with definite memberships, then we add u's intervals, whose memberships are to be changed from "possible" to definite, one by one to u's PQ-tree. Actually, the order of adding those intervals is not a trivial issue as adding an interval s_1 safely to a PQ-tree might prevent adding another interval s_2 to it, whereas adding them in the opposite order could be done safely for both. But an algorithm that finds the optimal order of adding intervals is not deterministic, so we use a greedy approach by sorting intervals



$\pi_1 = 0\ 1\ 6\ 7\ 2\ 3\ 5\ 4\ 9\ 8$
$\pi_2 = 1\ 0\ 9\ 8\ 2\ 3\ 7\ 6\ 4\ 5$
$\pi_3 = 6\ 7\ 4\ 5\ 8\ 9\ 2\ 3\ 1\ 0$
$\pi_4 = 4\ 5\ 6\ 7\ 9\ 8\ 3\ 2\ 0\ 1$
$\pi_5 = 8\ 9\ 2\ 3\ 4\ 5\ 6\ 7\ 0\ 1$
.....

Figure 2.7: The PQ-tree T and a subset of the collection of permutations represented by T.

in decreasing order in terms of interval size, such that larger intervals will be tried before smaller ones.

2.15 Does the Concept of Common Intervals Suffer from any Drawback?

The strict definition of common intervals might cause loosing some information. For example, in Figure 2.8 the interval a = [2 3 7 4 5] of permutation D would be considered a common interval with the interval b = [4 -2 5 -3] of permutation F, if 7 did not intervene between the two elements 3 and 4 in a. Therefore, if we disregard the existence of 7 in a, then a and b would be common intervals, but in order to keep the definition of common interval intact, we would choose another name and say that a and b are "common clusters".

The concept of common clusters is more flexible than the one of common intervals, as we can change the allowed number of intervening elements. For example, if two intervals belonging to two different permutations could not be considered common intervals just because two consecutive elements intervene between two other elements in one of the two intervals, and do not exist in the other, then we can consider the two intervals common clusters. In Figure 2.9 the interval c = [2 3 4 7 8 5] of permutation E would be a common interval with d = [4 -2 5 -3] of permutation H, if 7 and 8 did not intervene between 4 and 5 in c. In this

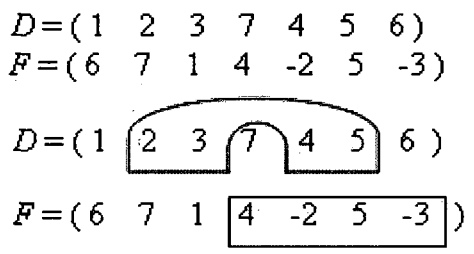


Figure 2.8: By disregarding 7, the two intervals [2 3 7 4 5] and [4 -2 5 -3] from D and F respectively would be considered as common intervals.

case, we say that c and d are common clusters.

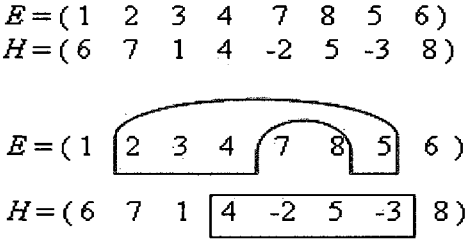


Figure 2.9: By disregarding 7 and 8, the two intervals [2 3 4 7 8 5] and [4 -2 5 -3] from E and H respectively would be considered as common intervals.

Therefore, we need to introduce a new parameter representing the allowed number of intervening elements: *neighborhood parameter*. More about that will be discussed in the next section.

2.16 Using Parameterized Gene Clusters

The definition of synteny blocks, gene clusters or similar constructs from the comparison of two or more genomes entails an important trade-off: if we place emphasis on identical content and order of the genes, segments or markers in a block or cluster, only relatively small regions of the genome will satisfy this restrictive condition, giving rise to many small blocks while missing large regions common to the genomes. On the other hand, by allowing unrestricted scrambling of genes within blocks, as with max-gap clusters [48] or gene teams [49], we miss local genome rearrangement, an important aspect of evolutionary history, or

we give up the possibility of detecting any regions of extensive local conservation. This motivates us to find a parameterized definition of gene cluster that allows us to control the amount of emphasis we put on conserved order within a cluster [50]. We want this parameter θ to allow us to explore characteristics of clusters such as how numerous they are, how large they are, how rearranged they are, and to what extent they are conserved from ancestor to descendent in a phylogenomic tree. Although, the motive to find such a parameter is biological, we discover that this parameter is closely related to the bandwidth parameter in graphs. Before presenting the approach, let us get an idea about the max-gap criterion in order to understand why it does not help detect regions of highly local conservation.

2.16.1 Max-gap Clusters

The concept of max-gap clusters is used when a pairwise comparison between two genomes is implemented in order to identify maximal clusters of homologous genes. First, a distance between two genes in this model is defined as the number of genes intervening between them; second, a single parameter g is used to denote the maximum number of genes allowed to intervene between two genes in order for these latters to still belong to the same cluster; third, a chain is defined as a set of genes such that for any gene a in the set there is at least one other gene b in the set such that the distance between a and b is not greater that g; finally, a max-gap cluster is defined as a max-gap chain that is not included in any larger chain (i.e., the chain is maximal). Each cluster is characterized by its size and length, where the size of a cluster is simply the number of marked genes (genes shared between two clusters in the two genomes) it contains and the length is the total number of marked and unmarked genes (genes that are not shared between two clusters in the two genomes). The size of a cluster is same among the compared genomes, whereas the length might vary from one genome to the other.

Figure 2.10 shows an example of finding max-gap clusters between two unichromosomal genomes each of length $n = 20$. The maximum gap allowed is $g = 2$. Two max-gap clusters (outlined by dark boxes) have been found. The marked genes are boldfaced and underlined and their total number is $m = 10$. A gap of size 3 separates the two clusters in (a), whereas a gap of size 5 separates them in (b). We notice that the smaller cluster has the same size 3 between the two genomes but not the same length which is 3 in (a) and 4 in (b), whereas the other cluster has the same size 7 and length 10 in both (a) and (b) of Figure 2.10.

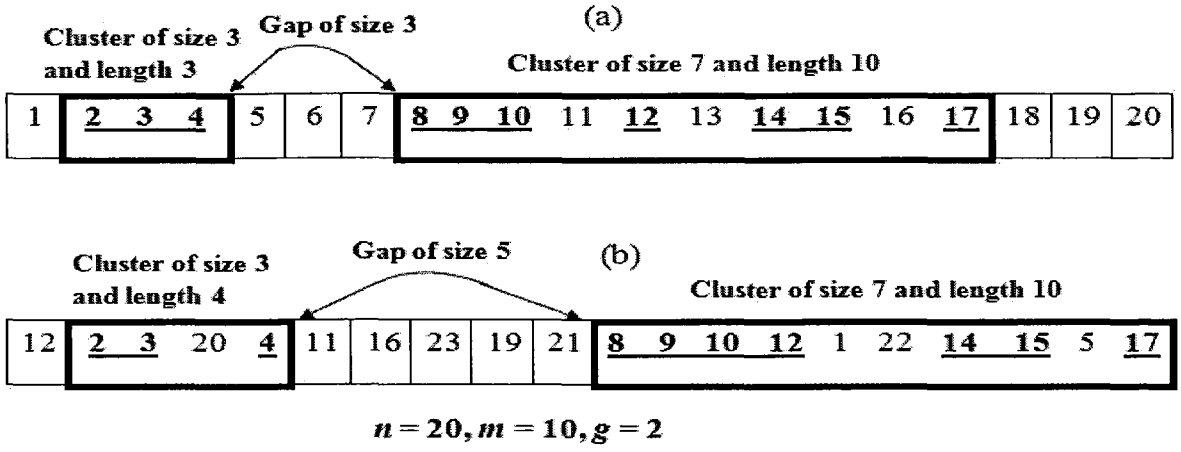


Figure 2.10: The result of finding max-gap clusters between two genomes each of length $n = 20$. When the maximum gap allowed is g = 2, two clusters are found. The total number of marked genes that are boldfaced and underlined is $m = 10$.

2.16.2 Definitions

The following definitions are necessary to understand the approach in the next subsection 2.16.3, where our characterization of gene clusters is made up of a general part that identifies clusters of vertices common to two graphs, and a specific part where a graph is determined by the proximity of genes on the chromosomes of a genome.

Definition 2. Let $G_S = (V_S, E_S)$ and $G_T = (V_T, E_T)$ be two graphs with a non-empty set of vertices in common $V = V_S \cap V_T$. We say a subset of $C \subseteq V$ is an ***ST*-cluster** if it is the vertex set of a connected component of $G_{ST} = (V, E_S \cap E_T)$.

Definition 3. For the purposes of genome comparison, we may consider V_X to be the set of genes in the genome X. For genes g and h in V_X on the same chromosome in X, let $gh \in E_X$ if the number of genes intervening between g and h in X is less than θ, where $\theta \geq 1$ is a fixed *neighbourhood parameter.*

Example 6. In Figure 2.11 we have two genomes, S consisting of three chromosomes and T consisting of two chromosomes. For each genome a graph is drawn based on neighbourhood parameter value $\theta = 3$ as follows: In each chromosome of each genome, every pair of genes are connected by an edge if they are neighbors, i.e. the number of intervening genes is less than θ. Thus, an edge connects genes 1 and 2 in genome S because there is zero intervening gene, an edge connects genes 1 and 3 because there is one intervening gene, namely gene

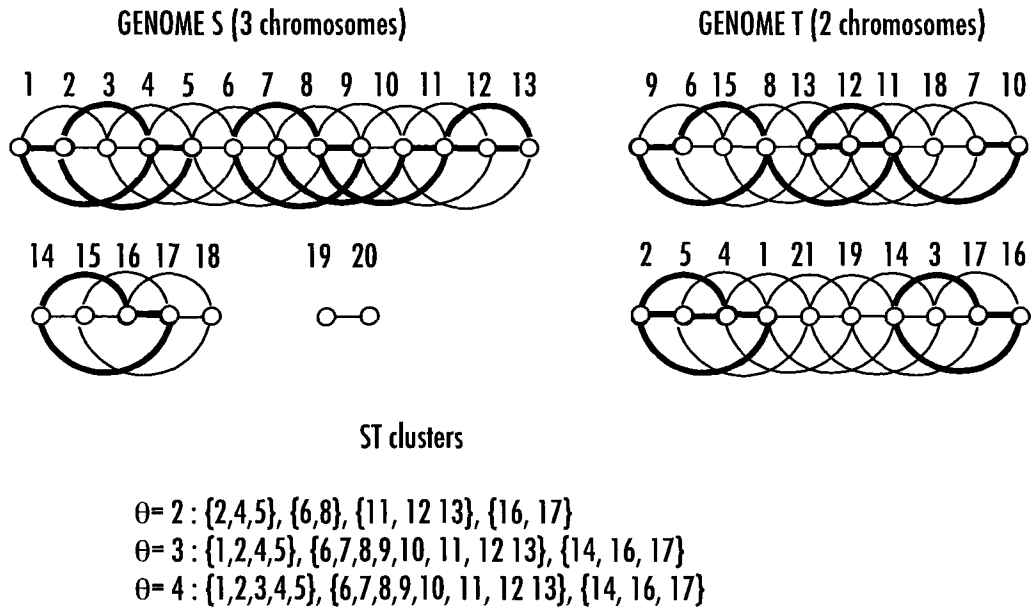


Figure 2.11: Graphs constructed from two genomes using neighbourhood parameter $\theta = 3$. Thick edges determine clusters. ST-clusters listed for $\theta = 2$ and $\theta = 4$ as well.

2, and an edge connects genes 1 and 4 because there are two intervening genes, namely genes 2 and 3, whereas genes 1 and 5 are not connected by any edge because the number of intervening genes is $3 \not< \theta$. The edges drawn in thick line are the edges shared between the two graphs and determine clusters. ST-clusters for $\theta = 2$ and $\theta = 4$ are listed as well.

These definitions of edge sets and ST-clusters decompose the genes in the two genomes into identical sets of disjoint clusters of size greater or equal to 2, and possibly different sets of singletons belonging to no cluster, either because they are in V, but not in $E_S \cap E_T$, or because they are in $(V_S \cup V_T \setminus V)$. For simplicity, we do not attempt to deal with duplicate genes in this paper. When $\theta = 1$, a cluster has exactly the same gene content and order (or reversed order) in both genomes. When $\theta = \infty$, the definition returns simply all the synteny sets, namely the sets of genes in common between two chromosomes, one in each genome.

Since the neighbourhood parameter was found to be closely related to the bandwidth in graphs, let us recall what the formal definition of the graph bandwidth is. Let Π be the set of all orderings of V. The **bandwidth** of a graph $G(V, E)$ is defined to be

$$B(G) = \min_{p \in \Pi} \max_{uv \in E} |p(u) - p(v)|. \tag{2.19}$$

In a genome S each chromosome χ determines a physical order among the genes it contains.

2.16.3 The Approach Using Parameterized Gene Clusters

In this approach, the chromosomal gene order in each genome is represented by a graph, and a dynamic programming approach is used to infer ancestral gene orders based on edges shared between descendents' graphs. In the forward pass, edges shared between descendents' graphs are considered definite for their ancestor, and called *optimals*, whereas edges non-shared between descendents' graphs are called *near-optimals* and should wait for the second, trace-back, pass to determine if they will be definite for their ancestor or discarded. For the root R, with three descendants, an edge is *optimal* if it is shared between at least two of them. We do not consider *near-optimals* for R. In the trace-back, starting from R down to the leaves, if an edge e is *optimal* for parent and *near-optimals* for its descendant, then e is promoted to the optimal status and considered definite for that descendant. Figure 2.12 shows an example of inferring the set of optimal edges and the set of near-optimal edges at the ancestor of two genomes, S and T, at two terminal nodes during the forward pass. The graphs are constructed from S and T using parameter $\theta = 3$.

But the problem is that some clusters representing connected components of the graphs at internal nodes inferred as mentioned might not be compatible with one genome. Checking whether this problem occurred is known in graph theory as the *graph bandwidth problem.* Therefore, at each internal node we check the *bandwidth* of its graph made of its definite edges after each promotion of an edge from the *near-optimal* to the *optimal* status. If the bandwidth $> \theta$, then the corresponding edge should be discarded because this means that no genome could be build where the vertices in the connected components can be linearly disposed such that each edge has less than θ genes intervening between the two endpoints. We have applied this approach to a set of yeast genomes [50]. Among the results, we find strong evidence for setting a certain fixed value to the cluster parameter. We also find that we can recover almost all the clusters that can be found without order constraints, i.e., by the max-gap criterion, indicating that local order conservation is a lot greater than that unconstrained definition would suggest.

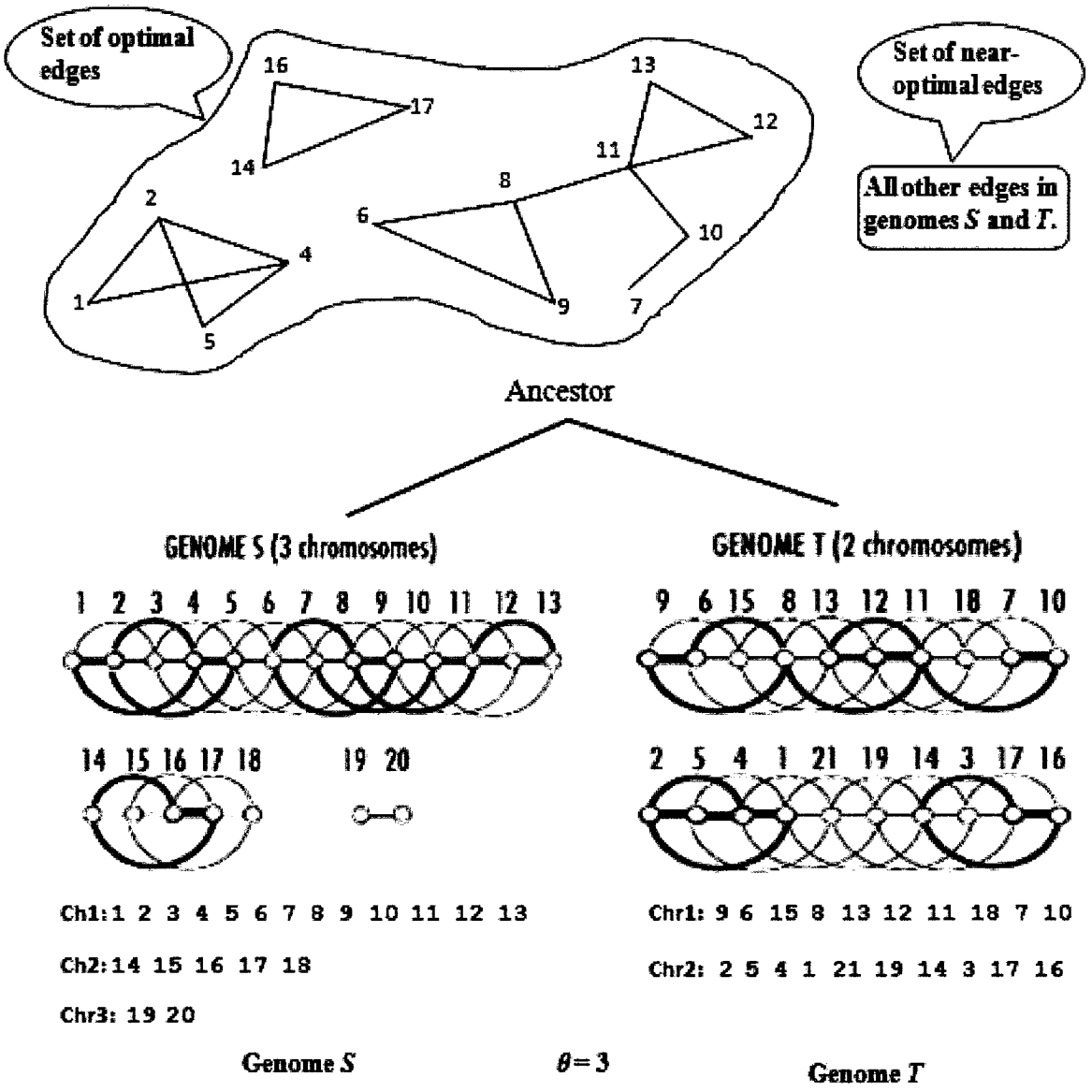


Figure 2.12: Graphs constructed from two genomes using parameter $\theta = 3$. Thick edges determine clusters. The set of optimal edges represent edges shared between descendants S and T. The set of near-optimal edges represent all other edges in S and T.

2.17 Summary of the Approach

We have proposed and implemented a new method based on common intervals of two or more genomes for constructing a phylogenetic tree. This has the advantage (or disadvantage!) of being independent of any particular model of genome rearrangement or rearrangement

weighting. A maximum of emphasis is laid on the commonality of gene order at the local level, the objective function for the tree being merely the sum, over all the tree branches, of the symmetric difference between the two sets of intervals associated with the genomes at the two ends of the branch.

We have only begun to explore the algorithmic possibilities in this approach. The use of PQR-trees during the recurrence part of the algorithm might allow for a global and efficient optimum in the general case. Failing that, more sophisticated heuristics are certainly available for the construction of PQ-trees at ancestral vertices during the traceback.

One issue we have not examined is the interpretation of branch length. The number of intervals at terminal vertex is $O(n^2)$, but in our test data the number of intervals at ancestral vertices is typically $O(n)$, so that terminal branches would appear unduly distant from the cluster of ancestral vertices, were the untransformed symmetric distance considered meaningful as a clocklike measure of evolution.

Note that this chapter deals only with unsigned genomic data. There is no particularly natural way to extend it to signed genomes, except of course to simply drop the sign. Our approach, however, does seem particularly well-suited to situations with gene families, and even to string, instead of permutation, representations of genomes.

The focus in one of our projects on inferring phylogenomic trees is on giving even more importance to common intervals than in previous studies by making reconstructing phylogenomic trees rely solely on them [45]. This approach is a model-free in the sense that it makes no assumptions regarding rearrangement event types that ancestors underwent to produce the current genomes. The objective function optimizes the sum over all the tree branches of the symmetric difference between the two sets of intervals associated with the genomes at the two ends of the branch. The idea is that more related genomes share more common intervals and vice versa.

We have applied this method to recently sequenced chloroplast genomes of streptophytes in order to contribute to the controversial questions of streptophye phylogeny, adding to previous work on both gene order phylogeny and DNA sequence-based phylogeny.

We have also applied this method to a set of yeast genomes using the concept of parametrized gene clusters instead of common intervals as explained in Section 2.14.

Chapter 3

Using Double Cut and Join Operation

Based on reversals and translocations, Bourque and Pevzner devised the MGR (Multiple Genome Rerrangement)[12] algorithm for inferring the median of three genomes and extended it to reconstructing a genome-scale phylogenetic tree and inferring ancestral gene orders. MGR is applicable for unichromosomal or multichromosomal genomes.

In 2005, Yancopoulos *et al.* [2] introduced the double cut and join (DCJ) operation as a general operation for all rearrangement events including transposition, inversion, and translocation, resulting in a simpler model for the edit distance and a simpler algorithm to convert one genome into another. A double cut and join operation simply cuts the chromosome in two places and joins the four ends of the cut in a new way. Recently, Bergeron *et al.* [51] improved the DCJ model even further by producing a simplified linear algorithm, which is currently the most general algorithm for genome rearrangement movements.

We have thus been motivated to solve the median problem and its extension to inferring the ancestral gene order in a phylogenomic tree using DCJ, i.e., allowing, in addition to inversion and reciprocal translocation, operations of transposition and block interchange. The results, in the space of multichromosomal genomes, appear in [52]. The approach used is similar to MGR, i.e., based on iterations of a rearrangement median problem, namely the inference of an ancestral genome based on its three neighbors in a binary phylogeny. All indications point to the median problem in any rearrangement metric space being NP-hard [7, 14]. I developed a heuristic for this problem that attempts to avoid local minima using

a variant of simulated annealing, by setting the loss threshold to zero.

I tested this algorithm on chloroplast and mammalian data sets and found solutions as good as or better than MGR when transpositions are not allowed. Allowing these operations gives quantitatively better solutions where, however, part of the reconstructed ancestral genomes is included in circular chromosomes.

The DCJ approach tends to reconstruct one or more small circular chromosomes at speciation points although there is no current biological evidence for the durability over evolutionary time of circular chromosomes in the nuclear genomes of higher eukaryotes. This raises the question of the biological significance of these chromosomal circles.

In this chapter we discuss a version of the small phylogeny problem in the metric space of multichromosomal genomes under a rearrangement distance metric. The particular metric we use is the double cut and join metric (DCJ) [2]. This is similar to the existing MGR approach [12] but it allows, in addition to inversion and reciprocal translocation, operations of transposition and block interchange.

Models of genome rearrangement processes have permitted different repertoires of operations. Certainly, realistic models must account for inversion. They also must allow reciprocal translocations, and processes of chromosome fusion and fission, all of which involve transferring an entire telomeric (i.e., suffix or prefix) region of at least one chromosome.

Other movements of chromosomal fragments, usually not involving telomeres, are widely attested, and grouped together under the label of transpositions. They are produced by a variety of processes, such as gene duplication followed by the loss of the original copy, or retrotransposition, or recombination errors.

Of the three true movement rearrangements, inversion, translocation and transposition, only the first two, separately or in combination, have proved very amenable to mathematical modeling, as exemplified by the Hannenhalli-Pevzner formula for the edit distance between two genomes, i.e., the minimum number of operations required to transform one genome into another, and the efficient algorithm for producing such a series of operations. No formula or efficient algorithm exists for transposition, either by itself or in combination with the other two operations. As for other structural genome modifications, such as duplication of genes or of chromosomal segments, or deletions and insertions, while they are also aspects of genomic plasticity and often consequences or causes of movement rearrangements, mathematical models of rearrangement are not easily extended to encompass them.

The DCJ model, however, allows for the generation of a new kind of movement operation, a generalized transposition called block interchange, which is not represented in the biological genome rearrangement literature, though it has long been studied in the mathematical literature on rearrangement. Both transposition and block interchange can be thought of as the excision of a fragment, its circularization, together counting as one DCJ operation, followed by a second set of cuts, where the circle is not necessarily cut in the same place it was originally created through a join, and then reincorporated at a new site in the chromosome. Transpositions and block interchanges thus count as two DCJ operations whereas inversions and translocations each count as one.

We postpone the question of the biological significance of these chromosomal circles to Section 3.9. Yancopoulos *et al.*'s original publication [2] pointed out that the running time of their algorithm could be reduced to linear if circles were not constrained to be reincorporated into linear chromosomes as soon as they were generated. Bergeron *et al.* [51] recently restated the DCJ model and produced a simplified (linear) algorithm ignoring the reincorporation constraint and, as in the mathematical justification of DCJ in [2], without any explicit mention of the particular operations of inversion, translocation, transposition, interchange, fusion and fission. It is thus the most general existing algorithm for movement rearrangements. As it has a form which lends itself well to constraints on the operations allowed, it can largely emulate other algorithms, e.g., the Hannenhalli-Pevzner algorithm (but without taking into account "hurdles" and "knots") or the Yancopoulos-Attie-Friedberg algorithm (at the cost of losing its computational efficiency).

Solutions of the small phylogeny problem in rearrangement metric spaces are generally based on iterations of a rearrangement median problem, namely the inference of an ancestral genome based on its three neighbours in a binary phylogeny. Before the discoveries in [53], all indications were that the median problem in any rearrangement metric space is likely to be NP-hard [7, 14]. Thus in Section 3.1 we give a brief idea of MGR, and in Section 3.2 we present the general algorithm for basing a rearrangement phylogeny on the median problem, while in Section 3.4 we present a heuristic for the median problem in DCJ space. Section 3.5 discusses ways of avoiding local minima of the small phylogeny problem. The rest of the chapter is devoted to applications to chloroplast and mammalian data sets.

3.1 MGR: Multiple Genome Rearrangement Algorithm

MGR [12] is a heuristic algorithm that reconstructs a phylogenetic tree of a set of species represented by their genomes, such that the sum of the rearrangements is minimized over all the edges of the tree. It can also infer the ancestral gene orders. MGR works for the three types of genomes: unichromosomal circular, unichromosomal linear, and multichromosomal genomes. When genomes are multichromosomal, MGR uses reversals and translocations as rerrangement operations, whereas when genomes are unichromsomal, MGR relies only on reversals.

In the process of inferring a phylogenetic tree, MGR uses a heuristic subroutine called, MGR-Median, which is the MGR solution to the median problem of three genomes and is also presented in [12].

For a given set of genomes, MGR infers the tree as follows: it starts by finding the median of the three closest genomes based on the reversal distance and using the MGR-Median, then it goes into a loop that, in each iteration, adds one more genome to an edge of the partially reconstructed tree until the full phylogeny is constructed. The simplest of such partially reconstructed tree is the one that consists of three genomes. The genome to be added at each iteration is chosen such that the tree resulting from adding this genome to the partially reconstructed tree will be of minimum total length as in Figure 3.1. The genome will be thus added to a certain edge such that the resulting tree will have that minimum total length.

As for the MGR-Median, given three genomes G_1, G_2, and G_3, it tries to find their median M such that the total distance $D = d(M, G_1) + d(M, G_2) + d(M, G_3)$ will be minimized. Finding M is achieved by iteratively implementing *good reversals* in the genomes G_1, G_2, and G_3, such that these genomes will get gradually closer to each other until they are all transformed eventually into an identical genome which would be the sought median. A reversal is *good* if it makes a genome closer to its ancestor which is unknown here, and consequently finding a good reversal is not an obvious task. Therefore, MGR considers a reversal in G_1 good if it reduces the reversal distance between G_1 and G_2 and the reversal distance between G_1 and G_3.

When no good reversals are available, MGR-Median tries to find the *best reversal* which

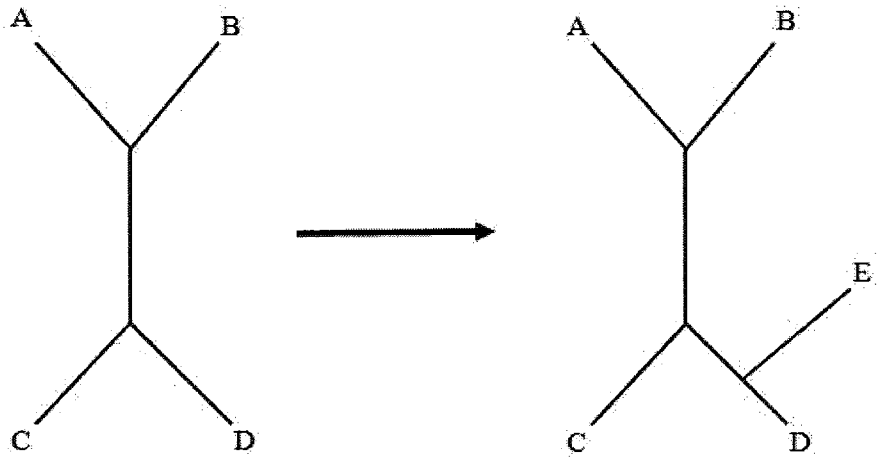


Figure 3.1: Illustration of adding a new genome to the partially reconstructed tree in MGR-Median. After trying to add the new genome E to every edge of the tree consisting of A, B, C, and D, and computing the total length of the resulting tree, E has been finally added to the edge leading to genome D as the tree resulting in is of minimum total length.

will be the result of a depth k search minimizing the total pairwise reversal distances, where k is the total number of available reversals.

Although the MGR-Median is theoretically independent of the labeling order of G_1, G_2, and G_3, in practice this is not guaranteed. There is no conceptual difficulty in extending the MGR-Median approach to more than 3 genomes, but the authors do not explore this.

Without going into all the details of MGR, a local optimization step is required to be implemented on the ancestors of the tree after inferring them. That step is not mentioned in [12] but would be added later to the software implementing MGR.

3.2 The Small Phylogeny Problem under Rearrangement Distance

Recall the discussion of the small phylogeny problem in Chapter 1. A more detailed description is as follows. Given the quintuple $(N, \mathcal{P}, n, \mathcal{G}, d)$, where

$\mathcal{P}$ is a phylogeny with N labeled terminal nodes,

$\mathcal{G}$ is a set containing N genomes, each made up of markers (signed positive or negative)

$1, \cdots, n$, partitioned among one or more circularly or linearly ordered chromosomes, and each genome associated with one of the terminals of $\mathcal{P}$, and

d is a metric (satisfying non-negativity, reflexivity, symmetry and the triangle inequality) on the set of all possible genomes with n markers,

the small phylogeny problem is to construct a set of genomes $\mathcal{H}$ to associate with the non-terminal nodes of $\mathcal{P}$, such that the phylogenetic tree length

$$L(\mathcal{H}) = \sum_{XY \in \mathcal{B}} d(X, Y) \tag{3.1}$$

is minimal, where $\mathcal{B}$ is the set of branches in $\mathcal{P}$.

In this chapter, we consider the simplest structure for $\mathcal{P}$, namely an unrooted, binary-branching tree. All nodes are of degree one (terminals) or three (non-terminals). The overall structure of our (heuristic) algorithm for minimizing L is as shown on the next page.

The initialization step can be important in reducing the computing time in the **while** loop and in the **Escape** routine. An easy initialization, but one which does not favour rapid convergence, consists of choosing a different random genome for each genome in $\mathcal{H}$. A better choice is to set the genome equal to the genome of one of the nearest terminal nodes.

The **Median** algorithm is the subject of Section 3.4.

The choice of ordering of $\mathcal{H}$ can be fixed at the outset once and for all, or it may change before each pass of $\mathcal{H}$ in the hope of avoiding a poor local minimum. We have not experimentally verified the importance of choosing an ordering strategy.

The **Escape** routine is the subject of Section 3.5.

3.3 Notation

We use the following notation to represent the adjacencies in a genome [51]. A genome is a sequence of chromosomes. A chromosome is a sequence of genes. A chromosome can be circular or linear. A gene is represented by two vertices, its *tail* and its *head*. The tail of a gene x is denoted by x_t, and its head is denoted by x_h. Depending on which of the two complementary strands of DNA it is on, a gene is read from left to right or from right to left. Thus, if the gene x is read from left to right on the genome, it will be optionally preceded by a positive sign ($+x$) and represented as $x_t x_h$, whereas if it is read from right to left, it will be preceded by a negative sign ($-x$) and represented as $x_h x_t$. Since two consecutive genes

Algorithm Small Phylogeny

input $\mathcal{P}, \mathcal{G}$

set $L = \infty$

initialize $\mathcal{H}$

calculate $L' = L(\mathcal{H})$

while $L' < L$

set $L = L'$

choose an ordering of the elements of $\mathcal{H}$

for each $G \in \mathcal{H}$, with neighbours A,B,C

$G =$ **Median**(A, B, C)

$L' = L(\mathcal{H})$

end while

Escape from local minimum

output

do not necessarily have the same orientation, an *adjacency* of two consecutive genes x and y, depending on their respective orientation, can be of four different types:

$$\{x_h, y_t\}, \{x_h, y_h\}, \{x_t, y_t\}, \{x_t, y_h\}.$$

An extremity that is not adjacent to any other gene is called a *telomere*, represented by a singleton x_h or x_t. Thus, a genome is a set of adjacencies and telomeres such that the tail or the head of any gene appears in exactly one adjancency or telomere.

Example 1. A genome G consisting of 7 genes {1, 2, 3, 4, 5, 6, 7} might have two chromosomes as follows: G = -4, -2, 7# 3, -6, 5, -1#

the # sign denoting the ends of successive chromosomes. G is represented as a set of adjacencies and singletons as follows:

$$\text{G} = \{\{4_h\}, \{4_t, 2_h\}, \{2_t, 7_t\}, \{7_h\}, \{3_t\}, \{3_h, 6_h\}, \{6_t, 5_t\}, \{5_h, 1_h\}, \{1_t\}\}$$

3.4 The Median Algorithm

According to the notation presented in section 3.3, if two vertices a and b from different genes are adjacent in a genome, we represent this by an edge $\{a, b\} = \{b, a\}$; for a vertex c at the end of a chromosome and hence adjacent to no other vertex, we construct the singleton $\{c\}$. Then any rearrangement operation can be represented by an operation on one or two terms in the representation, such as $\{a, b\}, \{c, d\} \rightarrow \{b, d\}, \{a, c\}$ or $\{a, b\} \rightarrow \{b\}, \{a\}$ or $\{a, b\}, \{c\} \rightarrow \{b\}, \{a, c\}$. **Algorithm Median** successively transforms all three genomes into their median.

This algorithm is inspired by the MGR algorithm [12] in its strategy of seeking operations which move each genome toward the other two as much as possible at each step. The details of which operations are prioritized are slightly different, as are the final steps towards the median. The use of the DCJ paradigm makes the coding straightforward, as can be deduced from the accompanying pseudocode in page 54. A consequence of the DCJ approach is that the median can contain circular chromosomes, even if the three neighbouring genomes have only linear chromosomes, whereas previous methods exclude the presence of circles in the median. The **Algorithm Median** is shown on the next page.

3.5 Escape from Local Minima

Once the small phylogeny algorithm converges, we seek a better minimum as follows. Again we iterate over all ancestral nodes until convergence. At each node V, we examine the adjacencies defining V's current genome. Those adjacencies and singletons that are in all three or in any two of the neighbours constitute the invariant part of V.

Consider the set U containing just those adjacencies or singletons of V that are in only one of the neighbours. Our approach to finding a better minimum is to pick any two vertices at random in U, to perform a DCJ operation on the two adjacencies or singletons containing these two vertices and to add the resulting adjacencies or singletons to U, replacing the current adjacencies and singletons in V. If the resulting genome has better or equal median distance than the current minimum, it replaces the current genome. This is repeated a large number of times, 5000 in our experiments. When there is no longer any change in the total tree length, the algorithm terminates.

Algorithm Median

input three genomes with same gene content.

while it is possible to do an operation $\{a, b\}, \{c, d\} \rightarrow \{b, d\}, \{a, c\}$ that creates two adjacencies $\{b, d\}$ and $\{a, c\}$ in one genome that are already shared by the other two, execute such an operation.

endwhile

while it is possible to do an operation $\{a, x\}, \{b, y\} \rightarrow \{a, b\}, \{x, y\}$ that creates an adjacency $\{a, b\}$ in one genome that is already shared by the other two, let S be the set of such operations. (N.B., either x or y or both may be null elements so that, e.g., $\{a, x\}$ is just the singleton $\{a\}$.) For each operation in S, associate a score defined to be the increment in $|S|$ were that operation to be applied. Choose an operation in S to apply with maximum positive score.

if $|S| = 0$ and it is possible to do operations $\{a, x\}, \{b, y\} \rightarrow \{a, b\}, \{x, y\}$ and $\{a, w\}, \{b, z\} \rightarrow \{a, b\}, \{w, z\}$ in two genomes that not only create an adjacency $\{a, b\}$ in each that already exists in the third, but also create an element of S, execute such a pair of operations. (N.B., not all of x, y, w and z can be null.)

else if $|S| = 0$ and it is possible to do operations $\{a, x\}, \{b, y\} \rightarrow \{a, b\}, \{x, y\}$ and $\{a, w\}, \{b, z\} \rightarrow \{a, b\}, \{w, z\}$ in two genomes that create an adjacency $\{a, b\}$ in each that already exists in the third, execute such a pair of operations that minimize the genomic distance between the two affected genomes. (N.B., not all of x, y, w and z can be null.) **endif**

endwhile

for all adjacencies $\{a, b\}$ in any genome where $\{a\}$ and $\{b\}$ are singletons in both other genomes, carry out the operation $\{a, b\} \rightarrow \{a\}, \{b\}$.

endfor

(At this point all three genomes have been transformed to the same structure, the median.)

for all pairs of adjacencies and/or singletons in the median we carry out all possible DCJ operations on the pair and see if it reduces the sum of the distance between the median and the three original genomes; if so, we adjust the median accordingly. Whenever such an adjustment is found, we repeat this search.

endfor

output median genome.

By retaining alternative medians of equal median distance at each step, this approach effectively searches far from the original solution. MGR [12] also includes a (somewhat different) post-processing step for escaping from local minima.

3.6 The Campanulaceae cpDNA Data Set

The well-known Campanulaceae chloroplast dataset consists of 13 cpDNAs with 105 markers each. Each genome consists of one circular chromosome. The data were first collected by E. Cosner and have been studied by Cosner *et al.* [54] and Moret *et al.* [55]. Using GRAPPA, Moret *et al.* reconstructed 216 tree topologies of Campanulaceae with a total distance of 67 reversals each. Bourque and Pevzner [12] used MGR to reconstruct one of these 216 trees, that shown in Figure 3.2, with a total distance of 65 inversions.

We ran our program on this data set using the tree reconstructed by MGR, without allowing the appearance of additional circular chromosomes (i.e., no transpositions or block interchanges), and obtained 64 DCJ operations.

Running the program unconstrained, we obtained a total distance of 59 DCJ operations. Only four ancestors had an extra circular chromosome, but there is no biological evidence in the Campanulaceae, or other higher plants, of chloroplast genomes consisting of two or more circles.

3.7 Data Set on Mammals

The mammalian data set, drawn from [56], consists of the genomes of human, rat, mouse, cat, dog, pig and cow. Each genome consists of 307 HSB (homologous synteny blocks). In [56], the total distance of the tree in Figure 3.3 is 487 reversals, obtained using MGR.

Running DCJ on this data set using the same tree topology, without allowing the ancestors to have any circular chromosome also resulted in a total distance of 487 DCJ operations. The first local minimum was 495, but the **escape** routine brought it down to 487.

When we allowed ancestors to have circular chromosomes in addition to linear ones, we obtained a total distance of 486 DCJ operations. The number of circular chromosomes that appeared in each ancestor ranged between 1 and 5, but only in the immediate ancestors of the seven data species.

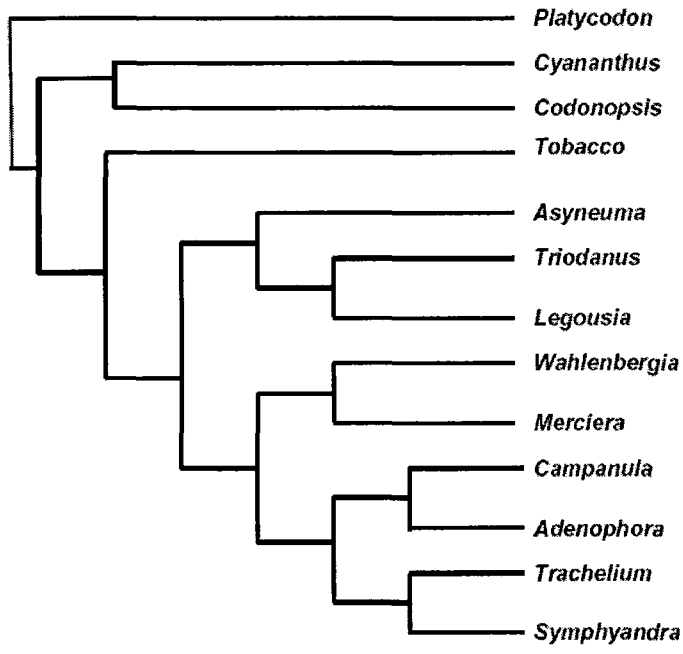


Figure 3.2: Phylogeny for Campanulaceae data set. Rooting and edge lengths arbitrary.

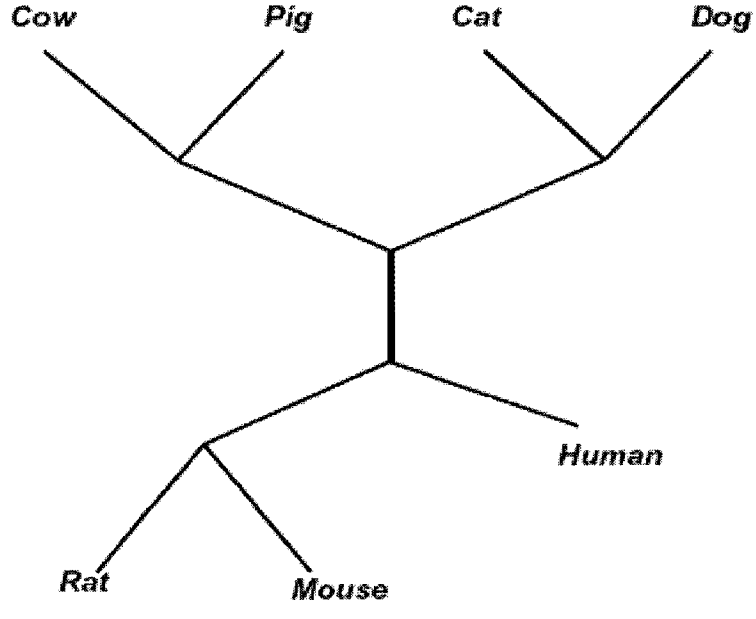


Figure 3.3: Phylogeny for mammalian data set. Rooting and edge lengths arbitrary.

3.8 Implementation

Our experimental software was oriented to achieve the maximum accuracy through the **escape** routine and the median improvement steps, with little regard to the size of problems beyond the ones considered here. However, there is much room for improving the code in view of larger data sets. For example, the look-ahead could be made more efficient by storing rather than repeating certain steps. A bounding procedure could be used to prune some search paths. Also, the local optimization step could be carefully calibrated to cut down on the number of attempts necessary to escape the local minimum.

3.9 Evidence for Excision-Circularization-Linearization-Reincorporation

The DCJ approach can reconstruct circular chromosomes at speciation points although there is no current biological evidence for the durability over evolutionary time of circular chromosomes in the nuclear genomes of higher eukaryotes. While circularization is well-known and understood in the functioning of the immune system, in somatic cell tumors, classical "double minutes", and various very small DNA molecules like episomes, and while ring chromosomes are a relatively common genetic abnormality, the existence of circular chromosomes as part of the normal genomic complement of a species, including in homozygotes and participating in normal meiosis, is unattested.

We have noted in our real examples, however, that when the DCJ operations are constrained, the algorithm produced solutions that are exactly as good as MGR solutions. This validates the suggestion in [51] that the notation and algorithm proposed in that article can serve as basis for exploring the effects of constraints on genome rearrangement problems.

The question remains, what is the evolutionary significance of these chromosomal circles, especially circular intermediates? Circular DNA structures abound in all sorts of organisms, even eukaryotes. Circular chromosomes are well-known in clinical studies [57] and the process of excision, circularization, linearization and reincorporation is exactly what happens in the configuration of the immune response in higher animals. And circular intermediates within germ line cells could play a role in rearrangement without becoming fixed in a population. But because the evolutionary consequences of block interchange could have come about in

other ways, e.g. various combinations of nested inversions, there has been no reason to look for evidence of this process or even to notice it. The question of the existence or importance of block interchange remains open.

How would we detect a transposition or a block interchange in closely related genomes? Figure 3.4 shows how the flanking markers of the transposed segment in one genome are adjacent in the other genome and vice versa. In genomes that are farther apart, we could expect some aspects of this pattern to be disrupted by subsequent rearrangements.

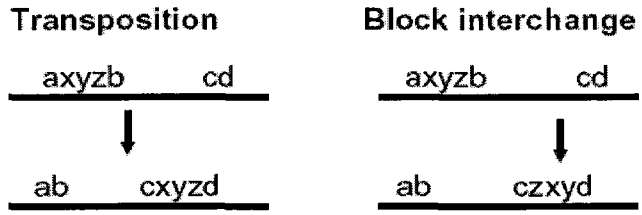


Figure 3.4: Diagnostics for transposition and block interchange. The "interchange" terminology can be understood from the interchange of xyz and bc in the transposition event and the interchange of xy and bc in the block interchange event.

Still, a few of these may survive, or be clearly visible despite subsequent rearrangements. For example, one of the circular chromosomes at the ancestor of pig and cow in Figure 3.3 is made up of markers 127 and 128 in the numbering system of [56]. The segments at the beginning of chromosome 1 of pig, with the flanking segments outlined on either side of the boldface segments from the circular chromosome, are -137,-136,-135,-134,-133,-132,[-131,-130], -129,[125],**127,128**,[-126],271,272,..., while chromosome 9 of cow is [125,126],129,[130],**-128, -127**,[131],132,133,134,135,136,137. This fits the pattern for transposition in Figure 3.4, aside from the subsequent inversion of segment 126.

Segments 205 and 206 can form another circle in this ancestor; chromosome 5 of cow and pig are: 211,-208,[-207,-210],-209,212,293,[294],**-206,-205**,[295],296 and -296,[-295,-294], -293,208,209,[210],**-206,-205**,[207],-212,-211, respectively, which fits the diagnostic pattern exactly.

These and other examples can, of course, be interpreted in other ways. But their existence is rather improbable without postulating transposition. This suggests that a systematic search for such patterns is warranted within a statistical model allowing a certain

degree of post-transposition rearrangement.

3.10 Discussion

We have explored the small phylogeny problem under the DCJ paradigm and found that not only can it emulate MGR, and even do better in some circumstances, but by allowing circular constructs it effectively serves as a lower bound for all procedures with a constrained set of operations.

We raise the problem of the biological significance of transpositions and block interchanges and suggest that current evidence warrants a systematic study of the (existence and) prevalence of this operation.

Finally, we point out that the study of genome rearrangement is highly sensitive to the quality of the data and the degree of resolution of the procedures for demarcating conserved syntenic regions. Without a high degree of completion and correctness of genome assemblies, translocations between chromosomes may be confused with transpositions. And with increasing analytical resolution, not only do the number of conserved blocks increase, but the relative proportions of different kinds of rearrangements may shift unpredictably [58].

Chapter 4

The Benefit of a Polynomial Time Algorithm for the Breakpoint Median Problem

4.1 Research Opportunity Based on an Overlooked Approach to Phylogenomics

For 10 years, since the NP-completeness proofs of Bryant [59], Pe'er and Shamir [60] and Caprara [7], it has been taken for granted that the median problem, in all its genome distance formulations, is NP-hard. Indeed, this thesis work has been oriented to either develop heuristics for this problem for cases where exact solutions would be impossibly lengthy, or to get around it by using a dynamic programming formulation of genomic phylogeny, displacing the complexity to the traceback phase, where this complexity may not come into play in many instances or may be subject to better heuristics than the median problem.

Recently, in surveying a set of problems involving more than two ordinary genomes, Tannier *et al.* [53] found one definition of genome space where many of these problems, including the median problem, can be solved in polynomial time. In the Table 4.1, drawn from the Tannier *et al.* paper, these problems are cross-tabulated against the genomic space in which they can be posed.

Thus to complete my exploration of various ways of approaching phylogenomics based

problem context	distance	halving	double distance	median	guided halving
breakpoint uni	P	open	open	NP [60, 59]	open
breakpoint general multi	P new	P new	P new	P new	P new
breakpoint linear multi	P new	open P?	P new	NP new	NP [64]
DCJ uni	P [2, 51]	P [62]	open	NP [7]	open
DCJ general multi	P [2, 51]	P [63, 65]	NP new	NP new	NP new
DCJ linear multi	P [2]	open	open	open NP?	open NP?
RT uni	P [1]	open	open	NP [7]	open
RT multi	P [9, 66, 67, 68]	P [61]	open NP?	open NP?	open NP?

Table 4.1: Status of complexity questions for five problems related to ancestral genome reconstruction, for eight genomic distances in the unichromosomal and multichromosomal contexts. Note that unichromosomal problems require that both input and output genomes be unichromosomal, so all problems involving doubled genomes are computationally defined in the circular case, when the doubled genome consists of a single circular chromosome composed of two successive occurences of the ordinary genome. Other versions of the halving problem are less restrictive [61, 62, 63]. RT stands for the Hannenhalli-Pevzner versions of the problems, permitting reversals and translocations, including fission and fusion. P and NP stand for polynomial and NP-hard, respectively, and when followed by ?, represent the conjectures of Tannier *et al.*

on gene order evolution, I study the phylogeny in this particular breakpoint distance context. Note that the median problem based on breakpoint distance has been proved to be NP-complete for unichromosomal genomes, either circular or linear, and is likely to be hard in most multichromosomal cases. The one context where the median problem is not NP-hard, namely the case of breakpoint distance on multichromosomal genomes where chromosomes are unconstrained as to linearity or circularity.

This will be my third approach, different both biologically and computationally from the common intervals and DCJ approaches.

For the purpose of breakpoint distance, a gene is represented by two vertices, its *tail* and its *head*. The tail represents the start of the gene and the head represents its end. For a gene x read from left to right on the genome, it will be represented as $x_t x_h$, whereas if it is read from right to left, it will be represented as $x_h x_t$. For a genome consisting of n

genes, when two genes x, y are consecutive on the genome, the right-hand vertex of gene x and the left-hand vertex of gene y constitute an *adjacency.* The left-hand vertex of the first gene, gene 1, as well as the right-hand vertex of the last gene, gene n, are considered to be *singletons* or *telomeres.*

Example 1. A genome G consisting of 10 genes {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} might have three chromosomes as follows: G = -2, -1, 9# 10, -6, 4, -3# 8, 5, 7#
the # sign denoting a break between the ends of successive chromosomes. G is represented as a set of adjacencies and singletons as follows:

$\{T2_h, 2_t1_h, 1_t9_t, 9_hT, T10_t, 10_h6_h, 6_t4_t, 4_h3_h, 3_tT, T8_t, 8_h5_t, 5_h7_t, 7_hT\}$

The letter T denotes a telomere. The number of telomeres is $2m$ where m is the number of chromosomes, and the number of adjacencies is $n - m$.

For the purpose of computing the median M of three genomes Π_1, Π_2, and Π_3, defined on a set of n genes, each gene x will be represented by four elements, x_h, x_t, t_{xh}, and t_{xt}, where t_{xh} represents the case when x_h is a telomere, and t_{xt} represents the case when x_t is a telomere. A complete weighted graph $G = (V, E)$ is then built on a set of vertices V, whose size is $\mid V \mid = 4n$, where each gene x is represented by its four elements. Each edge $(x, y) \in E$ in G is weighted by the number of times (x, y) appears as an adjacency in the three genomes Π_1, Π_2, and Π_3. Thus the weight could be 0, 1, 2, or 3. Each edge (x_h, t_{xh}) or $(x_t, t_{xt}) \in E$ in G is weighted by half the number of times x_h or x_t, respectively, appears as a telomere in the three genomes Π_1, Π_2, and Π_3. Thus the weight could be 0, $\frac{1}{2}$, 1, or $\frac{3}{2}$.

Example 2. Let Π_1 = 1, 2, 3# 4, 5#; Π_2 = -5, 2# 3, 4, -1#; Π_3 = 2,-3,-4# 5,-1#; be three genomes. Each consists of two linear chromosomes. Then the complete weighted genome G will be represented by a weight matrix shown in Figure 4.1.

The matrix is symmetric, therefore the values in the lower triangle will not be considered, so they are all assigned 0s.

Tannier *et al.* [53] pointed out that in the context of no constraints on linearity of chromosomes, any perfect matching on the $4n$ vertices of G defines a genome. A median genome M of minimum breakpoint score can be constructed for G by finding the appropriate perfect matching in G. What is the relationship between the weight of the perfect matching and the score of median genome?

In their paper, Tannier *et al.* argued that the pairwise breakpoint distance d_{BP} between two genomes Π and Γ, of size n genes each, is $d_{BP}(\Pi, \Gamma) = n - a - \frac{e}{2}$, where a is the

	1_t	1_h	t_{1t}	t_{1h}	2_t	2_h	t_{2t}	t_{2h}	3_t	3_h	t_{3t}	t_{3h}	4_t	4_h	t_{4t}	t_{4h}	5_t	5_h	t_{5t}	t_{5h}
1_t	0	0	1.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_h		0	0	0	1.0	0	0	0	0	0	0	0	0	1.0	0	0	0	1.0	0	0
t_{1t}			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{1h}				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2_t					0	0	0.5	0	0	0	0	0	0	0	0	0	1.0	0	0	0
2_h						0	0	0.5	1.0	1.0	0	0	0	0	0	0	0	0	0	0
t_{2t}							0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_{2h}								0	0	0	0	0	0	0	0	0	0	0	0	0
3_t									0	0	0.5	0	0	1.0	0	0	0	0	0	0
3_h										0	0	0.5	1.0	0	0	0	0	0	0	0
t_{3t}											0	0	0	0	0	0	0	0	0	0
t_{3h}												0	0	0	0	0	0	0	0	0
4_t													0	0	1.0	0	0	0	0	0
4_h														0	0	0	1.0	0	0	0
t_{4t}															0	0	0	0	0	0
t_{4h}																0	0	0	0	0
5_t																	0	0	0.5	0
5_h																		0	0	1.0
t_{5t}																			0	0
t_{5h}																				0

Figure 4.1: The weight matrix of Example 2.

number of common adjacencies between Π and Γ and e is the number of common singletons (telomeres) between Π and Γ. Thus, for any genome Π_i, $d_{BP}(\Pi_i, M) = n - (a_i + \frac{e_i}{2})$, where a_i and e_i are the number of common adjacencies and the number of common telomeres respectively between Π_i and M. When the weight of the perfect matching M is computed, every adjacency or telomere shared between M and Π_i will increase the weight by 1 or $\frac{1}{2}$ respectively, and the total weight will be $a_1 + a_2 + a_3 + \frac{e_1+e_2+e_3}{2}$. Thus, the total distance (score) between the median genome M defined by the perfect matching M and the three genomes Π_1, Π_2, and Π_3 is: $d_{BP}(\Pi_1, M) + d_{BP}(\Pi_2, M) + d_{BP}(\Pi_3, M) = 3n - (a_1 + a_2 + a_3 + \frac{e_1+e_2+e_3}{2})$. This means that when the weight of the perfect matching is maximized, the score of the median defined by that matching will be minimized. And since the maximum weight perfect matching has a polynomial time algorithm, then the breakpoint median problem has also a polynomial time solution.

Going back to our example, the maximum weight perfect matching is:

The edge: $1_t - t_{1t}$, weight: 1.5

The edge: $1_h - 2_t$, weight: 1.0

The edge: $t_{1h} - t_{4h}$, weight: 0.0

The edge: 2_h - 3_h, weight: 1.0

The edge: t_{2t} - t_{3h}, weight: 0.0

The edge: t_{2h} - t_{3t}, weight: 0.0

The edge: 3_t - 4_h, weight: 1.0

The edge: 4_t - t_{4t}, weight: 1.0

The edge: 5_t - t_{5t}, weight: 0.5

The edge: 5_h - t_{5h}, weight: 1.0

Total maximum weight matching = 7.0

The median defined by this matching consists of the following set of adjacencies and telomeres: {1t} {1h,2t} {2h,3h} {3t,4h} {4t} {5t} {5h}, which yields the following linear order: 1, 2, -3, -4# -5#. This genome consists of two chromosomes, the first is linear and the second is circular.

There are several ways of implementing maximum weight perfect matchings. We have adopted one [69] and have made it the engine of a median calculating routine used iteratively for the small phylogeny problem we have already coded in [52].

Although the maximum weight perfect matching algorithm is polynomial, the execution time of the version that we used is not negligible, being at least cubic in the size of the set of graph vertices, which is $\mid V \mid = 4n$ where n is the number of genes in the genome.

Nevertheless, this method is a lot more efficient than the other techniques we have explored for this thesis. Results on the same data sets we have been using for DCJ median, Campanulaceae (phylogeny reproduced below) and mammals, suggest that the small phylogeny problem based on breakpoint median is solved more than 100 times faster than that based on DCJ median. The ancestral gene orders converge after 4-6 tree traversals, and each traversal takes 1-2 minutes, whereas DCJ median takes more than 20 traversals to converge and each traversal, especially the first one, takes many hours to finish.

The speed of this new approach makes it very attractive for routine use. However, there remain questions of comparability and variability to investigate. Many biological researchers would ascribe more credibility to DCJ analysis than breakpoint analysis. It becomes important then to systematically assess how different are the results of the two methods. Also previous work [70] suggests that breakpoint-based phylogeny is more variable and less precise than DCJ phylogeny.

We have thus designed a series of computational analyses on two data sets: a mammalian

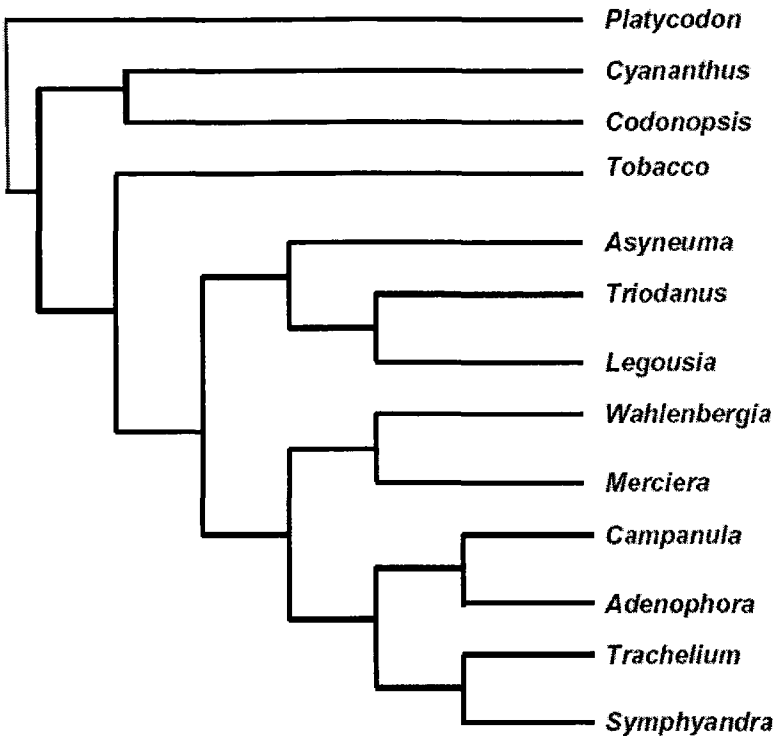


Figure 4.2: Phylogeny for Campanulaceae.

data set that we have used previously, plus a new mammalian data set that also includes the opossum genome. This has been complemented by simulation experiments. The variability of breakpoint results have been compared to the variability of DCJ results, with variability being generated by different initializations of the heuristics. The details are the subject of the next chapter.

Chapter 5

Which Metric is Better: Breakpoint or DCJ? That is the Question

5.1 Introduction

Breakpoint distance and rearrangement distances provide alternative ways of evaluating phylogenetic trees and reconstructing ancestral genomes based on whole genome data. When applied to a set of sufficiently diverse genomes, these approaches will generally lead to different results. Ancestral genomes optimal under the breakpoint criterion will not necessarily minimize the total rearrangement distance on the edges of a given tree, and optimality according to rearrangement distance will not necessarily minimize the total number of breakpoints.

Can we say that one of these methods is superior to the other? Lacking any widely accepted probability model for genomes evolving through rearrangements, we have no analytic framework for the statistical properties of reconstructions, in particular the accuracy and reliability of these reconstructions. Even assessment through simulations, though informative [6], is highly dependent on the assumptions necessary for generating the data, assumptions that are either highly simplified such as uniform weights on a small repertoire of rearrangement events or highly parametrized models pertinent to limited phylogenetic domains at best.

Is there any sense, then, in which we could affirm that one objective function on re-

constructions is better than the other? In this chapter we introduce a way of evaluating two mathematically independent optimization approaches relative to each other, namely how good or bad each is with respect to the other's criterion. Of course, breakpoints and rearrangement distances provide upper and lower bounds for each other, but between these bounds they are not mutually constrained. The idea is that *the approach that comes the closest to satisfying the other's criterion as well as its own, is more desirable.*

We will illustrate this method on two data sets on mammalian evolution, as well as three different simulations modeling each of these data sets in a different way, eight data sets in total. We use a given, well-accepted phylogeny for each data set and each simulation. We reconstruct all the ancestral genomes, once minimizing the total breakpoints over all tree branches (using the polynomial-time median method in [53]), and once with the minimum rearrangement distance [52]. Because optimal reconstructions are not unique, we sample five different reconstructions in each case.

We then apply our "fair" method to four aspects of these reconstructions. First we assess all the branch lengths in each reconstruction and then compare, in two different ways, the dispersion (or compactness), for each tree node, of the five different reconstructions. Finally, for the simulated data sets, we measure the distances of the reconstructed ancestors from the simulated ancestors.

For the analysis of the branch lengths and the dispersion of reconstructed ancestors, the results show a clear and systematic advantage of rearrangement distance over breakpoint distance. Despite this, the two methods prove to be equally good at reconstructing the known simulated ancestor genomes.

In Section 5.2, we formalize our proposal for a fair comparison of metrics, illustrating with four aspects of the "small" phylogeny problem, branch lengths, node dispersion (looked at two ways) among optimal solutions, and distance between reconstructed and true ancestral genomes in simulations. In Section 5.3 we formalize the breakpoint distance and the rearrangement distance, and sketch how they are used in solving the small phylogeny problem. In Section 5.4.1, we briefly describe the two real data sets on mammalian evolution as well as the simulations carried out under various "breakpoint re-use" conditions. The results are presented in Section 5.5 and commented in the Discussion.

5.2 A Fair Comparison

Let $\mathcal{P}$ be a phylogeny where each of the M terminal nodes is labeled by a known genome, and let d_A and d_B be two metrics on the set of genomes. Each branch of $\mathcal{P}$ may be incident to at most one terminal node and at least one of the N ancestral nodes. Consider reconstructions $R_A = (G_1^A \ldots, G_N^A)$ and $R_B = (G_1^B, \ldots, G_N^B)$ of the set of genomes to label the ancestral nodes such that

$$L_A(R_A) = \sum_{\text{branch } XY \in \mathcal{P}} d_A(X^A Y^A) \tag{5.1}$$

is minimized, and

$$L_B(R_B) = \sum_{\text{branch } XY \in \mathcal{P}} d_B(X^B Y^B) \tag{5.2}$$

is also minimized. Without taking into account any additional, external criteria, there is no justification for saying one of d_A or d_B is better as a criterion for reconstructing the ancestral genomes. In general, unless there is some monotonic relationship between d_A and d_B, we can expect that

$$L_A(R_A) < L_A(R_B), \tag{5.3}$$

i.e., strict inequality holds and

$$L_B(R_B) < L_B(R_A). \tag{5.4}$$

5.2.1 Branch Lengths

Inequalities (5.3) and (5.4) are not necessarily inherited by the individual terms in the sums, e.g., $d_A(X^A Y^A)$ may not always be less than $d_A(X^B Y^B)$. We call $e_A(X^B Y^B) = d_A(X^B, Y^B) - d_A(X^A, Y^A)$ the *excess length* of branch $X^B Y^B$ with respect to distance d_A, though it may sometimes be less than zero.

As schematized in Figure 5.1, we will calculate the least squares fit to $d_A(X^B Y^B) = \alpha_{B/A} d_A(X^A, Y^A)$ and to $d_B(X^A Y^A) = \alpha_{A/B} d_B(X^B, Y^B)$. Then $1 - \alpha_{B/A}$ is the excess rate of B with respect to d_A and $1 - \alpha_{A/B}$ is the excess rate of A with respect to d_B. A metric that induces a reconstruction with a lower excess rate with respect to the other metric may be considered superior, i.e., if the excess rate of A with respect to d_B is less than the excess rate of B with respect to d_A, then d_A is better in the sense of being more universal or less "parochial": the branches in the A reconstruction are closer to optimal length according to

incommensurable	compare	regression	commensurable
d_A	$d_B(X^AY^A) \leftrightarrow d_B(X^BY^B)$	$d_B(X^AY^A)=\alpha_{A/B}d_B(X^BY^B)$	$1 - \alpha_{A/B}$
d_B	$d_A(X^BY^B) \leftrightarrow d_A(X^AY^A)$	$d_A(X^BY^B)=\alpha_{B/A}d_A(X^AY^A)$	$1 - \alpha_{B/A}$

Figure 5.1: Strategy for comparing different metrics.

d_B than the branches in the B reconstruction are according to d_A.

5.2.2 Node Dispersion

Among the N_s different optimal solutions sampled for the reconstruction under d_A, let $G_i^A = \{G_{i1}^A, \ldots, G_{iN_s}^A\}$ be set of reconstructions of ancestral genome G_i. Then

$$V^A(G_i^A) = \max_{0<j<k\leq N_s} d_A(G_{ij}^A, G_{ik}^A) \tag{5.5}$$

is the dispersion of the N_s reconstructions. Since the reconstructions may be expected to cluster around the "true" value of G_i as measured by d_A, we may also expect that

$$V^A(G_i^A) \leq V^A(G_i^B), \tag{5.6}$$

where

$$V^A(G_i^B) = \max_{0<j<k\leq N_s} d_A(G_{ij}^B, G_{ik}^B) \tag{5.7}$$

since the genomes in G_i^B are not necessarily close to the true G_i as measured by d_A, and similarly

$$V^B(G_i^B) \leq V^B(G_i^A). \tag{5.8}$$

In both (5.6) and (5.8), the inequality would normally be strict.

As in Section 5.2.1, we will calculate the least squares fit to $V^A(G_i^B) = \alpha_{B/A}V^A(G_i^A)$ and to $V^B(G_i^A) = \alpha_{A/B}V^B(G_i^B)$ over all ancestral nodes. Then $1 - \alpha_{B/A}$ is the excess rate of B with respect to d_A and $1 - \alpha_{A/B}$ is the excess rate of A with respect to d_B. As with the branch lengths, we can see whether one of the two metrics has a systematically lower excess rate.

5.2.3 Distance to True Genome

In contrast to the real data sets, with the simulated data sets, we actually know the ancestral genomes G_i. We can expect

$$d_A(G_i, G_i^A) \leq d_A(G_i, G_i^B) \tag{5.9}$$

and

$$d_B(G_i, G_i^B) \leq d_B(G_i, G_i^A) \tag{5.10}$$

As in Sections 5.2.1 and 5.2.2, we will calculate the least squares fit to $d_A(G_i, G_i^B) = \alpha_{B/A} d_A(G_i, G_i^A)$ and to $d_B(G_i, G_i^A) = \alpha_{A/B} d_B(Gi, G_i^B)$ over all ancestral nodes. Then $1 - \alpha_{B/A}$ is the excess rate of B with respect to d_A and $1 - \alpha_{A/B}$ is the excess rate of A with respect to d_B. As with the branch lengths and node dispersion, we can see whether one of the two metrics has a systematically lower excess rate.

5.3 Breakpoints and Rearrangements

Given genomes $g_1, \ldots, g_M$ associated with the terminal nodes of $\mathcal{P}$, the small phylogeny problem is to construct a set of genomes $G_1, \ldots, G_N$ to associate with the non-terminal nodes of $\mathcal{P}$, such that the phylogenetic tree length L is minimal, as in Section 5.2. We consider the simplest structure for $\mathcal{P}$, namely an unrooted, binary-branching tree. All nodes are of degree one (terminal) or three (non-terminal). Our algorithms for searching for a minimimum L depend on median algorithms as shown in the pseudocode presented as Algorithm 3 here.

We recall the median problem that we have discussed in previous chapters: considering three genomes H, J, K as points in some metric space (E, d), find another genome $C \in E$ such that $d(C, H) + d(C, J) + d(C, K)$ is minimal.

There is a large literature on median problems in comparative genomics [53], with all studied versions except one proving to be NP-hard. This is summerized in Table 4.1. For our purposes we take E to be the set of oriented multichromosomal or unichromosomal genomes on the same set of n elements or genes. The genomes in this set may have circular as well as linear chromosomes, a property, which has little consequence for the numerical results of the median problem, but has computational advantages for both the two metrics we study; the double cut and join (DCJ) distance [2, 4], where we have implemented highly accurate code

Algorithm 1: Outline of phylogenetic reconstruction based on medians.

Algorithm for Small Phylogeny Using Metric d

input $\mathcal{P}, g_1, \ldots, g_M$

set $L = \infty$

initialize $G_1, \ldots, G_N$

calculate $L' = L(G_1, \ldots, G_N)$

while $L' < L$

set $L = L'$

for each $G_1, \ldots, G_N$, with neighbours H, J, K

$G =$ **Median Algorithm for** $d(H, J, K)$

$L' = L(G_1, \ldots, G_N)$

end while

output

[52] and the breakpoint distance, which gives rise to the only known version of the median problem that is of polynomial complexity [53].

The breakpoint distance between two genomes is defined to be

$$\begin{aligned} d_{BP} &= n - \text{ the number of common gene adjacencies in the genomes} \\ &\quad + \frac{1}{2} \text{ the number of common chromosomal endpoints (telomeres)} \end{aligned} \tag{5.11}$$

where gene adjacency requires conservation of their relative orientation (strandedness). We have carried out the first implementation of the Tannier *et al.*'s algorithm. The median problem turns out to be directly transformable to a version of the maximum weight perfect matching problem. We made use of the code in [69] for this purpose. Although the the maximum weight perfect matching algorithm is polynomial, the execution time is not negligible, being at least $O(n^3)$.

The DCJ metric d_{DCJ} counts the minimal number of operations necessary to transform one genome into another, where the repertoire of operations includes inversions, reciprocal translocations, chromosome fissions and fusions, as well as block interchanges, which count as two operations. Transposition of chromosomal segments from one site to another are

a special case of block interchange. Our version of the median solver in [52] is based on the MGR algorithm [12], with some differences due to the different rearrangement distances used, but also because of extensive use of additional routines to escape from local minima.

In the breakpoint method, the ancestral gene orders converge after 4-6 iterations within Algorithm 3, and each iteration takes 3-5 minutes for the data we consider in the next section. The current implementation of the DCJ method requires extensive computing time, taking more than 20 iterations to converge on the same data, where each iteration, especially the first one, takes many hours to finish.

5.4 The Empirical Studies

5.4.1 The Data

The first data set, drawn from [56], consists of the placental mammalian genomes from human, rat, mouse, cat, dog, pig and cow. Each genome consists of 307 HSB (homologous synteny blocks). The second data set includes the marsupial opossum along with the placental mammalian genomes human, rat, mouse and dog. These data are from the supplementary information for reference [71]. Each genome consists of 603 HSB. The given phylogenies are shown in Figure 5.2. Although these data sets are obviously not independent, the differences in the species involved, and the large number of extra HSB induced by the presence of the opossum genome, assures that these problems are rather different from the computational point of view.

5.4.2 The Sample of Optimal Reconstructions and Breakpoint Re-use

For each of the two data sets, we ran the breakpoint phylogeny algorithm 40 times with different random initializations of $G_1, \ldots, G_N$. We retained the runs that gave the five minimum total tree lengths, usually the same value. For each branch of the tree and for each of the five results, we computed the corresponding DCJ distance between the two breakpoint-inferred genomes determining that branch, and then computed the breakpoint re-use [12, 72] quotient $r = 2d_{DCJ}/d_{BP}$.

We ran the DCJ phylogeny algorithm five times only with different random initializations

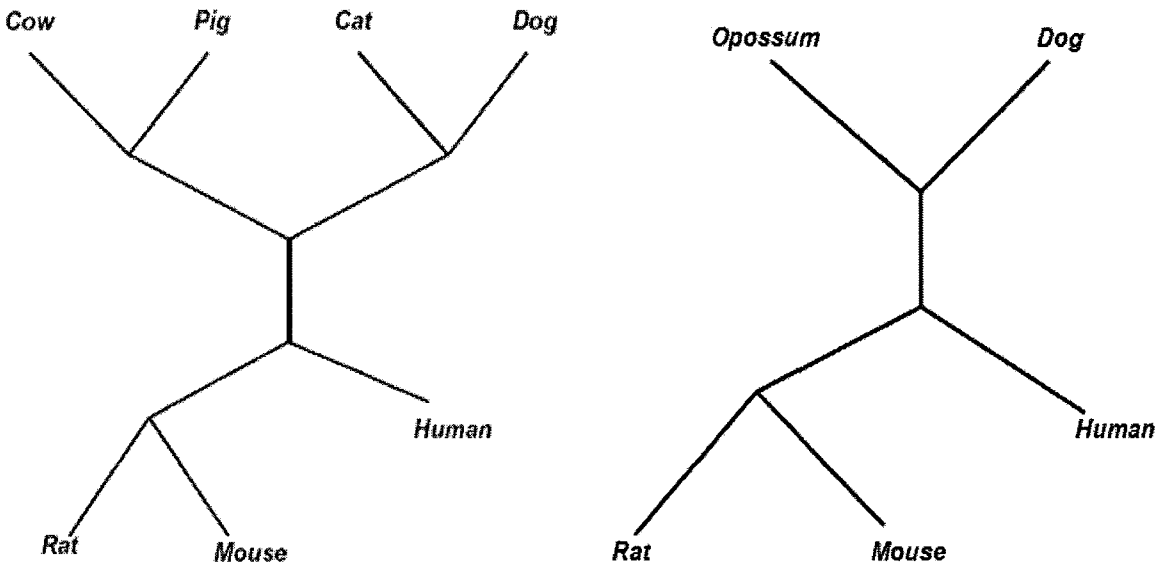


Figure 5.2: Phylogenies for mammalian data sets.

of $G_1, \ldots, G_N$. For each branch of the tree and for each of the five results, we computed the corresponding breakpoint distance between the two DCJ-inferred genomes determining that branch, and then computed the quotient $r = 2d_{DCJ}/d_{BP}$.

In the simulations we make use of the average $\bar{r}(X, Y)$ of the ten values of the re-use statistic calculated for each branch XY in these two ways, as well as the average of the ten branch lengths $\bar{d}_{DCJ}(X, Y)$.

5.4.3 Simulated Data Experiments

For each of the two data sets, we generated a random genome at an arbitrarily chosen "root", distributing the HSB over $\chi = \text{mean}_i^M \chi(g_i)$ chromosomes and then generated its descendants along the tree branches using 90% inversions and 10% reciprocal translocations for the $\bar{d}_{DCJ}(X, Y)$ random rearrangements on branch XY. We ran three separate simulations, constraining the rearrangements on a branch XY in three different ways: once assuring each operation on XY used two new breakpoints so that $r = 1$, once assuring each successive rearrangement on a branch used one new breakpoint and re-used one existing breakpoint so that $r = 2 - 1/n$, and once assuring $r = \bar{r}(X, Y)$. We then implemented the breakpoint and DCJ phylogeny algorithms exactly as for the real data, obtaining five reconstructions for each criterion.

5.4.4 The Measurements

For each of the $8 = 2$ criteria $\times$ (1 real + 3 simulated) data sets, we calculated the following aggregate statistics over the reconstructions, in each case measured by *both* criteria, the one used to infer the reconstructions and the opposing one:

1. average branch length $d_A(X^A, Y^A)$ and $d_B(X^A, Y^A)$, for each branch XY,
2. maximum intranode distance, $\max_{jk} d_A(G_{ij}^A, G_{ik}^A)$ and $\max_{jk} d_B(G_{ij}^A, G_{ik}^A)$, for each G_i,
3. maximum intranode, intercriteria, distance, $\max_{jk} d_A(G_{ij}^A, G_{ik}^B)$, for each G_i,
4. (simulations) average distance between reconstruction and true ancestor, $d_A(G_i, G_i^A)$ and $d_B(G_i, G_i^A)$, for each G_i.

At the end of this chapter, Figure 5.3 represents graphically measurement 1, Figure 5.4 represents graphically measurement 2, Figures 5.5 and 5.6 represent graphically measurement 3, and finally Figures 5.7 and 5.8 represent graphically measurement 4. The figures show the first mammalian data set, but the measurements were implemented for each of the two mammalian data sets mentioned in Section 5.4.1.

5.5 Results

Before examining the results, we reiterate that the key to this methodology is that it is fair, i.e., not inherently biased either towards BP or DCJ. Each comparison is made according to a single criterion; we do not compare BP scores with DCJ scores. The BP measurements are slightly worse on the DCJ reconstructions and the DCJ measurements are slightly worse on the BP reconstructions. How bad they are is measured in normalized terms – slope of a least squares line anchored at (0,0). The excess of this slope over 1.0 we call the *excess explanatory rate* (of using the ancestral genomes reconstructed under criterion A instead of those reconstructed under criterion B, when B is used to make the measurements). The smaller this cost, the closer the A ancestors are to being solutions, not only under criterion A but also B. Thus we see that in the real data, the excess rate of the DCJ reconstruction is systematically lower than that of the BP reconstruction, for both real data sets and for all the simulations.

5.5.1 Average Branch Length

In Table 5.1, we see that the excess rate for DCJ (in boldface), which measures how far the DCJ reconstructions are from being BP-optimal, is only of the order of half the excess rate of the BP reconstructions. This is true in both real data sets and in all the simulations, regardless of re-use rate. To illustrate the derivation of the excess rates as detailed in Section 5.2.1 and Figure 5.1, we present scattergrams of competing criterion versus reconstructing criterion branch lengths in Figure 5.9 for the real data sets, corresponding to the top row in Table 5.1.

dataset	seven placentals		marsupial, placentals	
reconstruction	BP	DCJ	BP	DCJ
	real data			
	5.23	**1.94**	11.55	**6.92**
	simulated			
no re-use	6.31	**3.05**	7.22	**5.56**
re-use 2	8.55	**2.02**	8.65	**0.44**
actual re-use	6.06	**3.85**	8.69	**3.30**

Table 5.1: Excess rate, in %, for each reconstruction, measured by competing criterion. Average branch length results.

5.5.2 Maximum Intranode Distance

In calculating the intranode distances, no patterns could be discerned for the excess rates within the second data set, the one containing the opossum genome. In particular, there were only three points on each scattergram, making inference very sensitive to statistical fluctuation in any of them. The correlations were very poor, and sometimes even negative. We tracked this latter problem down to two very different DCJ reconstructions of the node closest to opossum, occurring in at least two of the sets of simulations. Generally local minima for these problems tend to be relatively close together and this is what justifies our using summary statistics for them. Exploring the large-scale structure of the set of optimal solutions is beyond the scope of this chapter, however, so we confined the study of intranode distances to the seven placentals data set. Here, Table 5.2 shows that in seven out of eight comparisons, the smaller excess rate (in boldface) is that for the DCJ reconstructions.

These rates are often negative, showing that the DCJ reconstructions are often more closely clustered in terms of d_{BP} than the BP reconstructions are.

Figure 5.10 illustrates the derivation of the excess rates for the real data. Note the smaller correlations compared with the average branch length comparisons.

dataset	intranode		intranode, intercriteria	
reconstruction	BP	DCJ	BP	DCJ
		real data		
	34.1	**-27.8**	11.55	**6.92**
		simulated		
no re-use	40.0	**-35.9**	59.2	**14.3**
re-use 2	8.09	**5.46**	**13.75**	26.4
actual re-use	5.96	**-3.26**	62.5	**16.6**

Table 5.2: Excess rate, in %, for each reconstruction, measured by competing criterion. Maximum intranode distance data.

5.5.3 Average Distance Between Reconstruction and True Ancestor

In the case of the simulated data sets, we actually know the ancestral genomes. Thus as in Section 5.2.3, we can assess the relative performance of d_{BP} and d_{DCJ} with respect to how close the reconstructed genomes are to the true genomes. Table 5.3 shows the results of combining the results of both data sets to estimate the $\alpha_{DCJ/BP}$ and the $\alpha_{BP/DCJ}$. Figure 5.11 illustrates the calculations of these values. We note the extremely small excess rates, surely within the noise level when we compare with Tables 5.1 and 5.2, so that it appears that the reconstructed genomes are equally close to the true simulated ancestors no matter which method was used to infer them, or used to measure the distance.

5.6 Discussion

It is often taken for granted that rearrangement distance is "better" than breakpoint distance in phylogenetic inference because the former is connected to a model of genome evolution, while the latter is not. This reasoning is specious, however, since

dataset	combined data	
reconstruction	BP	DCJ
	simulated	
no re-use	**-0.9**	1.1
re-use 2	0.4	**-0.1**
actual re-use	**0.4**	0.7

Table 5.3: Excess rate, in %, for each reconstruction, measured by competing criterion. Average distance between reconstructed and simulated ancestor results.

- the rearrangements inferred in reconstructing the phylogeny are virtually always far fewer than those that actually generated the tree, and they may be quite different rearrangements,
- the set of rearrangement operations in the evolutionary model cannot include all possible mechanisms,
- the uniform costs accorded to all operations is a weakness of the rearrangements approach
- breakpoint distance is in fact closely related to evolutionary models, especially those with relatively unconstrained rearrangement operations.

In fact, there is no *a priori* biological or statistical reason to prefer one approach over the other. This is what motivates our search for the criterion that does least poorly as judged by the competing criterion.

The general picture that emerges from our analysis is that the ancestral genomes reconstructed according to the breakpoint criterion are more dispersed in the space of genomes than those reconstructed under the rearrangements criterion. This explains the dispersion analyses and by extension the branch-length results, all of which give the impression that the alternative optimal reconstructions under the DCJ criterion are relatively more closely compact in the set of genomes.

The larger dispersion might lead us to expect that the breakpoint reconstructions are on the average farther from the true simulated ancestor. But this does not seem to be the case. One explanation might be that the high "dimensionality" of genome space allows the reconstructions to be far from each other, compared to the DCJ reconstructions, while still allowing them to be close to the real ancestor genome.

We found a number of reconstructions under one criterion that were better under the competing criterion than the reconstructions actually found under this competing criterion. This is partially due to the limited sample size and partially due to the fact that the whole analysis is optimized, not just one node or branch.

Our statistically fair approach to evaluating competing objective criteria, could be applicable in other setups where objects or structures optimal according to one criterion are evaluated by a competing criterion.

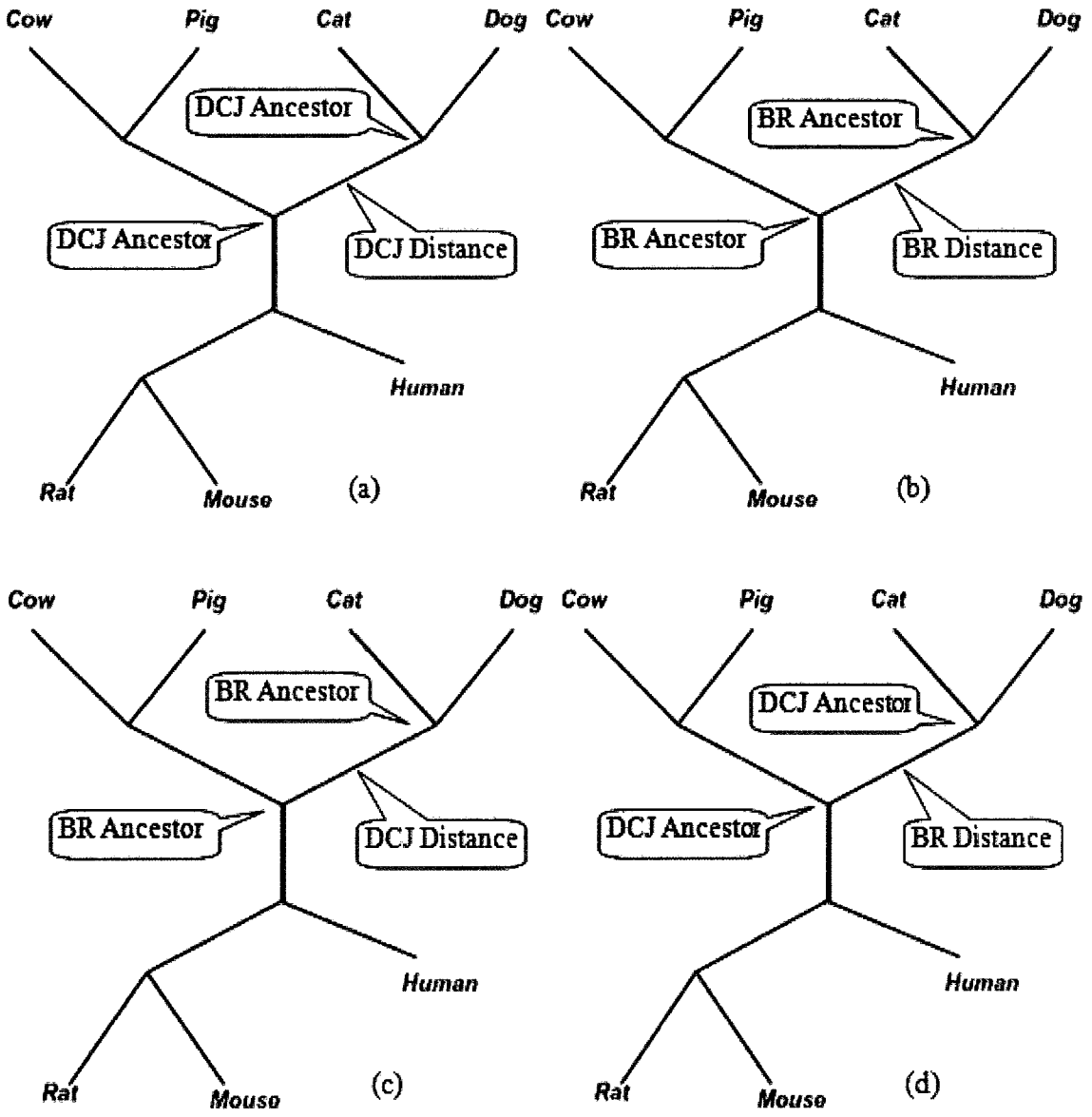


Figure 5.3: (a) Average DCJ branch lengths of trees inferred using DCJ. (b) Average breakpoint branch lengths of trees inferred using breakpoint. (c) Average DCJ branch lengths of trees inferred using breakpoint. (d) Average breakpoint branch lengths of trees inferred using DCJ. This process is repeated for the 4 data sets, the real data, and the 3 simulated ones, of each of the two mammalian data sets.

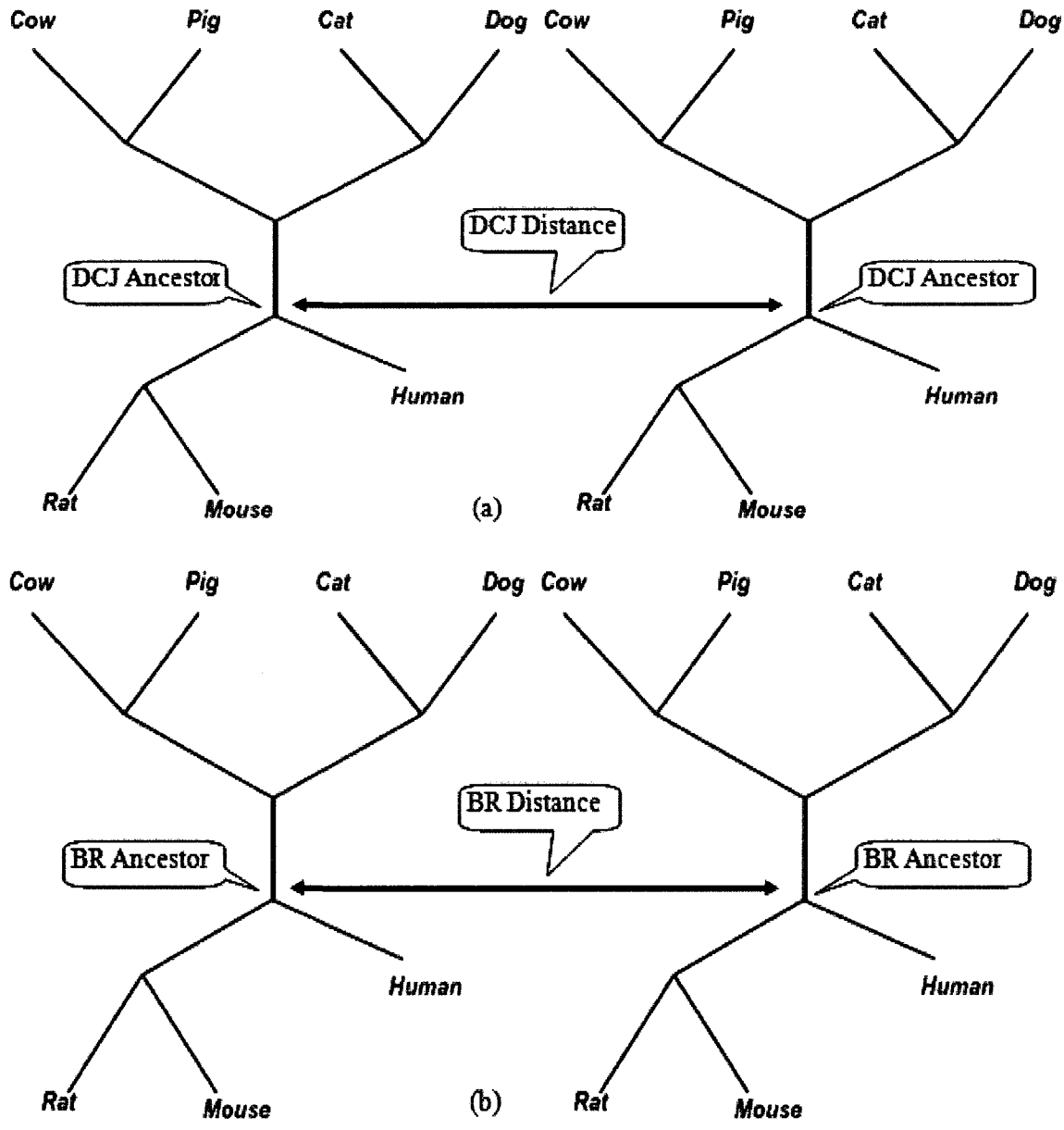


Figure 5.4: (a) Max pairwise DCJ distances between ancestors and their counterparts of trees inferred using DCJ. (b) Max pairwise breakpoint distances between ancestors and their counterparts of trees inferred using breakpoint. This process is repeated for the 4 data sets, the real data, and the 3 simulated ones, of each of the two mammalian data sets.

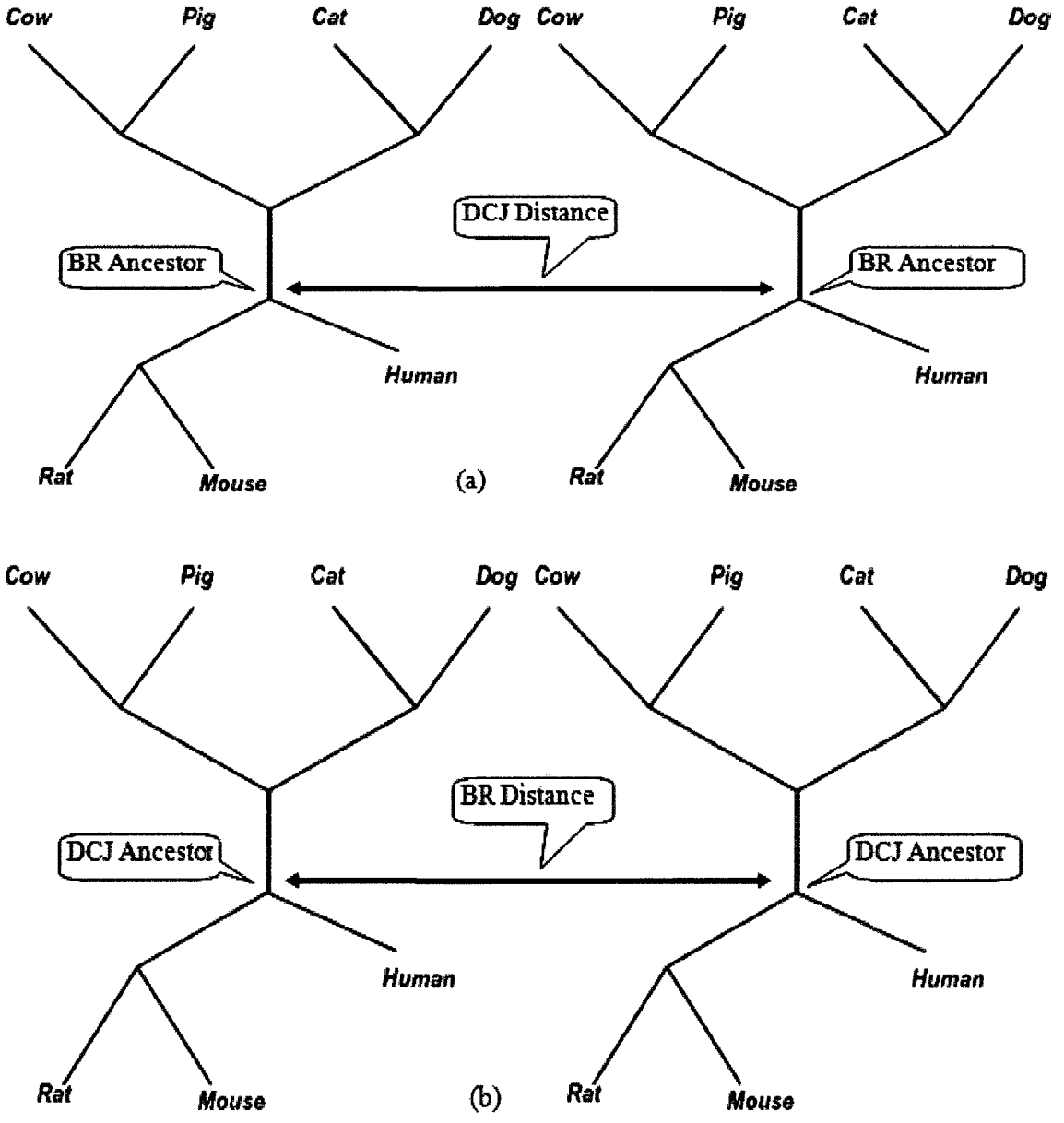


Figure 5.5: (a) Max pairwise DCJ distances between ancestors and their counterparts of trees inferred using breakpoint. (b) Max pairwise breakpoint distances between ancestors and their counterparts of trees inferred using DCJ. This process is repeated for the 4 data sets, the real data, and the 3 simulated ones, of each of the two mammalian data sets.

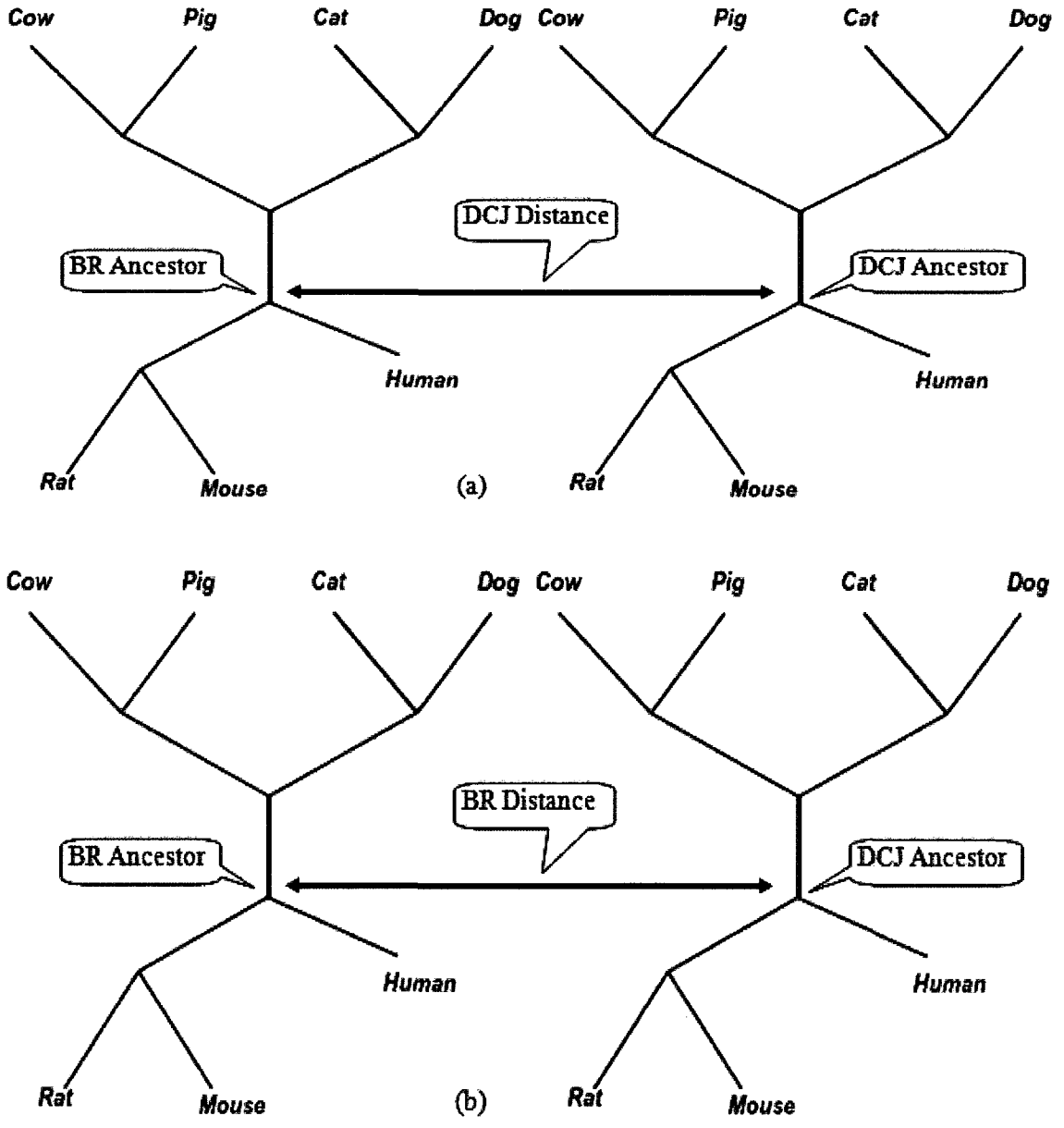


Figure 5.6: (a) Max pairwise DCJ distances between ancestors of trees inferred using breakpoint and their counterparts of trees inferred using DCJ. (b) Max pairwise breakpoint distances between ancestors of trees inferred using breakpoint and their counterparts of trees inferred using DCJ. This process is repeated for the 4 data sets, the real data, and the 3 simulated ones, of each of the two mammalian data sets.

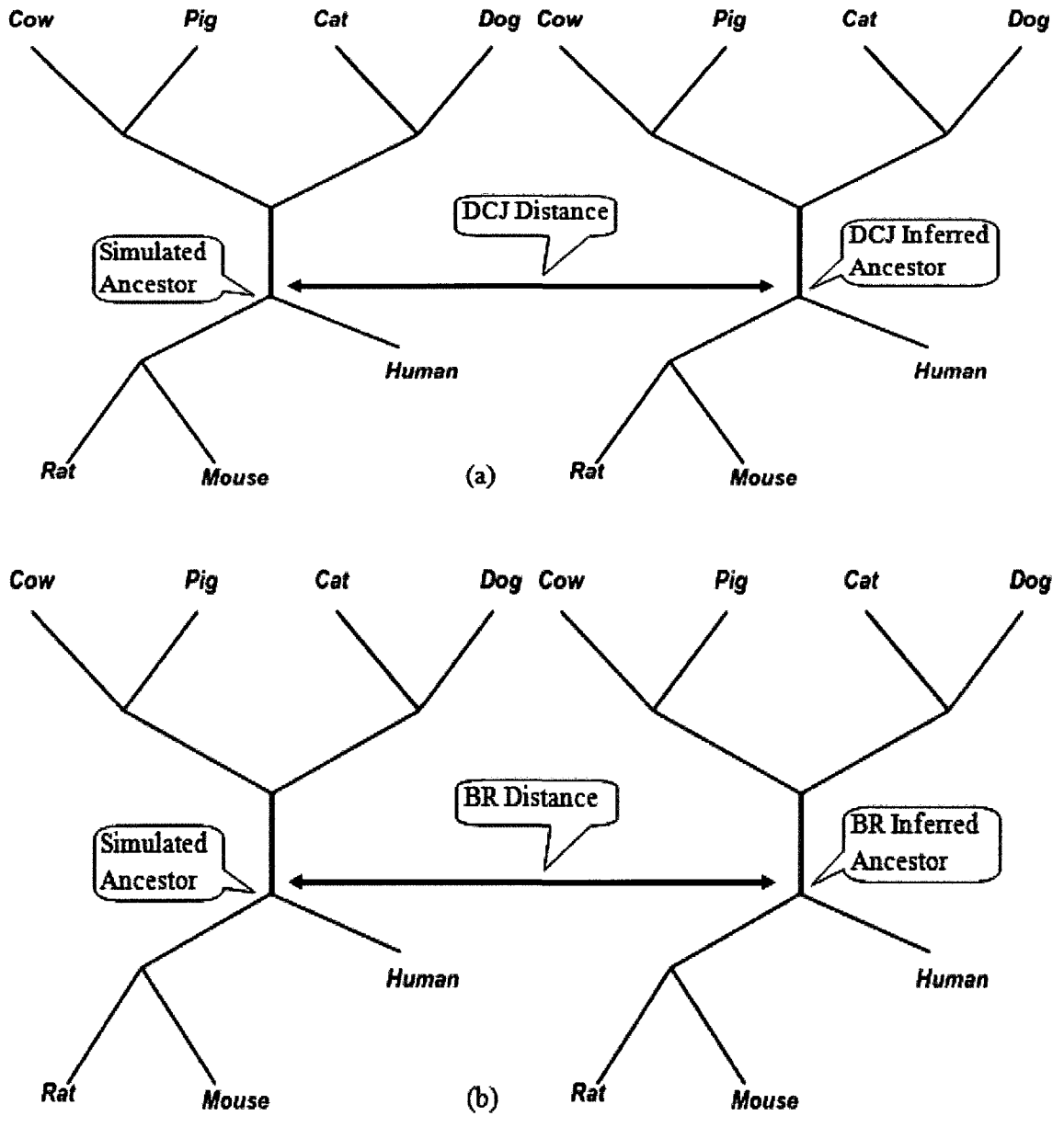


Figure 5.7: (a) Average pairwise DCJ distances between true ancestors and their counterparts of trees inferred using DCJ. (b) Average pairwise breakpoint distances between true ancestors and their counterparts of trees inferred using breakpoint. This process is repeated only for the 3 simulated data sets of each of the two mammalian data sets.

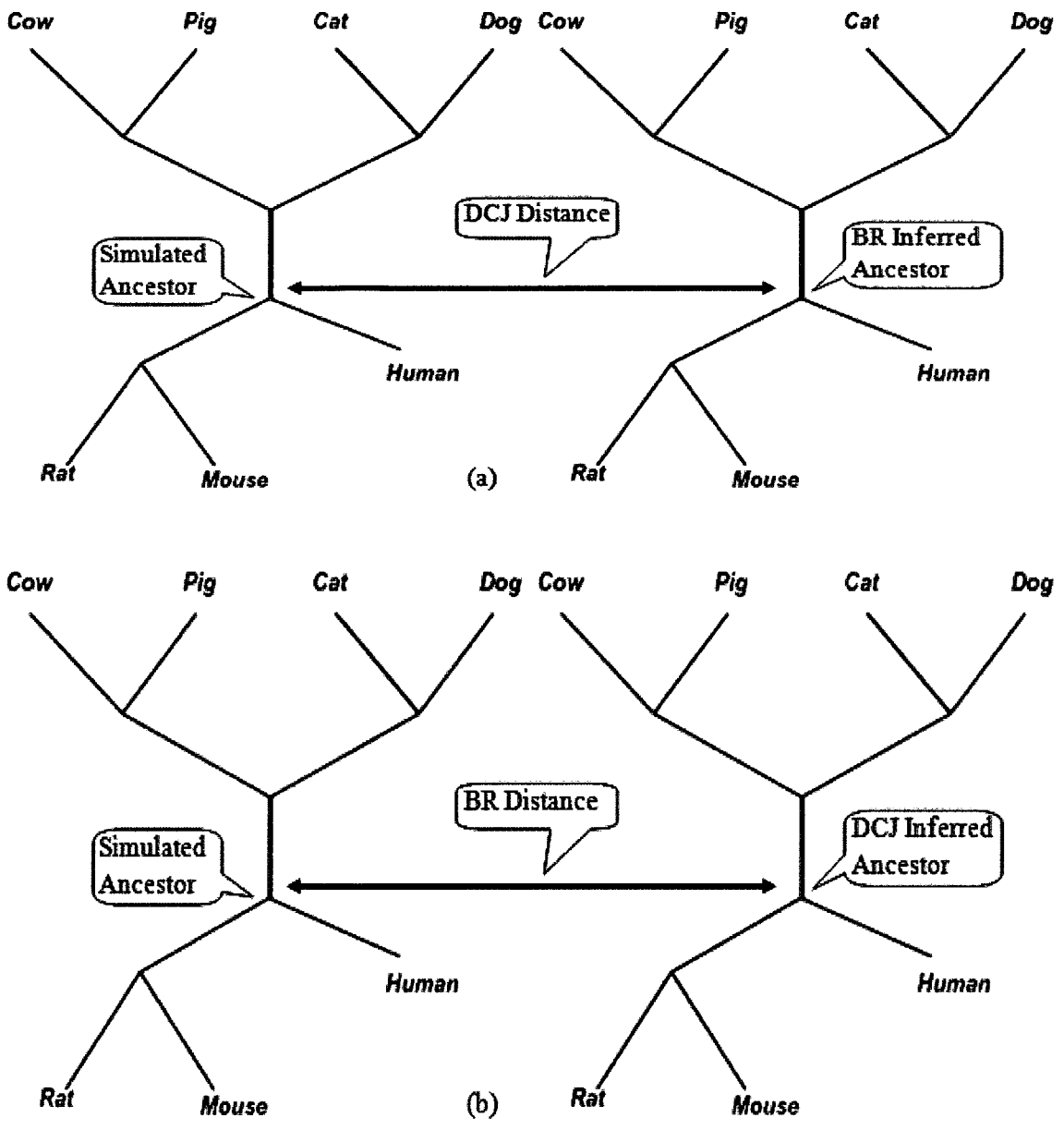


Figure 5.8: (a) Average pairwise DCJ distances between true ancestors and their counterparts of trees inferred using breakpoint. (b) Average pairwise breakpoint distances between true ancestors and their counterparts of trees inferred using DCJ. This process is repeated only for the 3 simulated data sets of each of the two mammalian data sets.

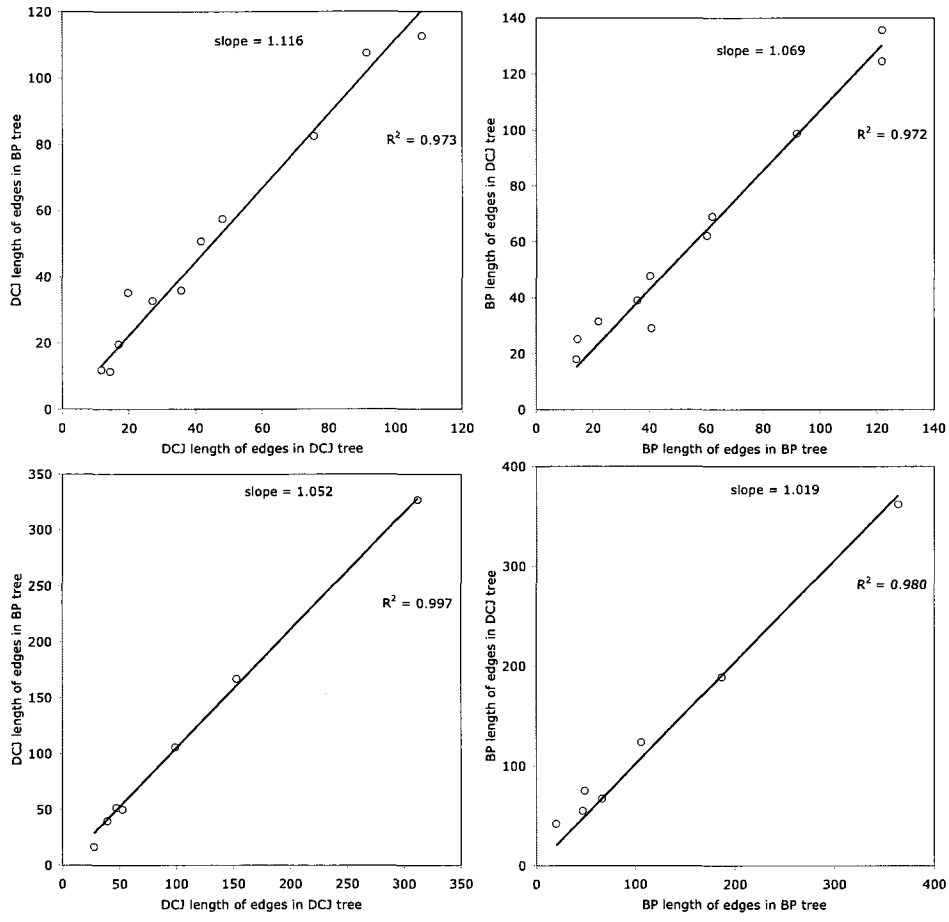


Figure 5.9: Competing criterion average branch lengths versus reconstructing criterion. Top: seven placentals data. Bottom: placentals plus opossum data.

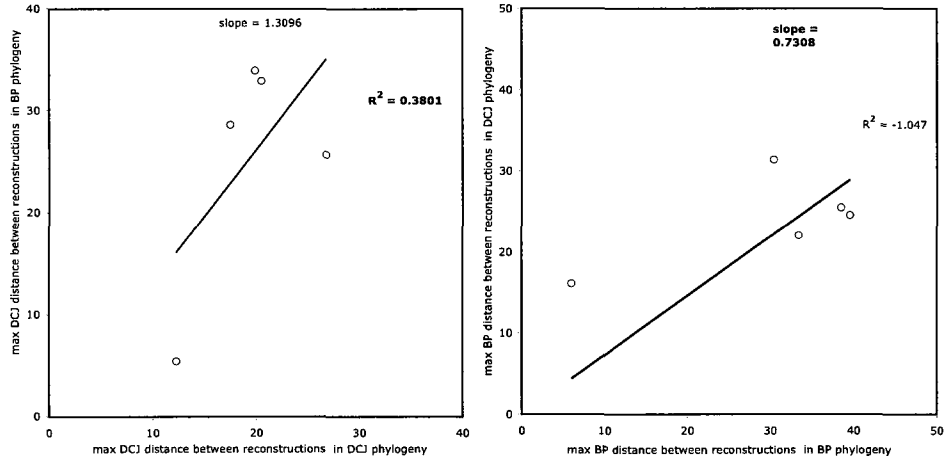


Figure 5.10: Competing criterion maximum intranode distance versus reconstructing criterion.

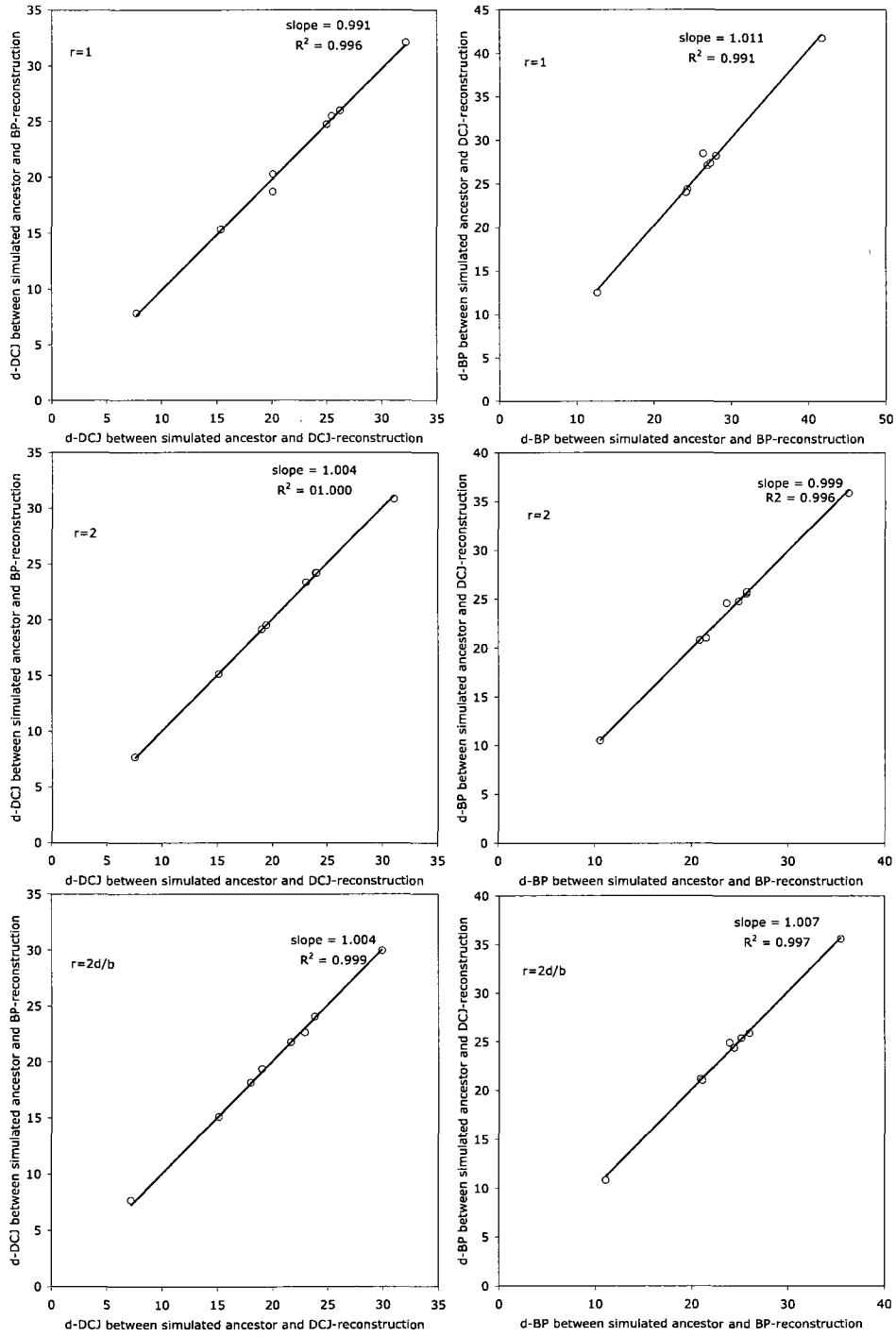


Figure 5.11: Distance according to competing criterion between simulated ancestor and reconstruction.

Chapter 6

Conclusion

In concluding this thesis, we refer to the title: "Evolutionary Ancestor Inference via Genome Rearrangement". In Chapters 2 to 4, we explored three approaches to this topic; the first uses the concept of common intervals; the second uses DCJ operations; and the third involves breakpoint distance. Chapter 5 compares DCJ distance to breakpoint distance.

1. Using common intervals [45], the purpose was to infer the tree structure and its ancestral genomes without any assumptions about the rearrangement events that created that ancestral and modern genomes on the phylogeny. Through our approach we were able to efficiently and accurately infer the tree topology (albeit with only a moderate number of species) and the ancestor genomes. The latter were not necessarily inferred as complete gene orders, but as sets of compatible intervals, which may or may not cover the entire genome. The algorithm was then extended to work in a more realistic context – genomes with unequal gene content. The main complication in the approach was the need to detect and resolve incompatible intervals. To do this we had recourse to a fairly specialized data structure, PQ- trees, to handle the intervals. While implementing PQ-Trees, we added a practical improvement to the Booth and Lueker [34] procedure, which is a potentially great increase in efficiency, namely sorting the intervals in decreasing order in terms of interval size, prior to being incorporated (or not) into the PQ-trees. As a distance measure, the size of the symmetric difference of two sets of common intervals was introduced and used for the first time in our work. In [50], the concept of common intervals was extended to common clusters where neighbourhood and adjacency are no longer synonymous, so that genes could be

neighbours without being strictly adjacent. However, this extension was not achieved without obstacles. As in [45], there was a need for a tool to detect incompatible edges. In this context, *graph bandwidth* turned out to be equivalent to the *neighborhood parameter* θ introduced in [50]. In order to compute the *graph bandwidth*, we explored the polynomial time exact algorithm in [73] and implemented its improved version in [74], after unsatisfactory experience with the heuristics, the Cuthill-McKee algorithm [75] and Reversed Cuthill-McKee algorithm [76]. In [50] we could recover common clusters even when they had been disrupted by some intrusive genes due to local rearrangements. By manipulating the neighbourhood parameter, we can manipulate the number of clusters and the size of clusters, and explore the appropriate balance between them. Also, we can determine the threshold at which graphs representing ancestral genomes start having too many incompatible edges. The neighbourhood parameter concept granted us much more flexibility in finding common clusters in comparison with common intervals and helped us to recover gene order conservation which the max-gap criterion cannot do.

2. In the chapters on DCJ, the purpose was to solve the small phylogeny problem using DCJ operations, which subsumes all the major types of genome rearrangement operations previously studied. Since MGR was already available to solve the same problem using reversal and translocation, our work [52] used MGR as a baseline for DCJ. We designed a heuristic to solve the median problem and extend it to infer ancestors in an unrooted binary tree. The DCJ median outperformed MGR by inferring ancestors with a better bound on the total tree length when there are no constraints on linearity or circularity of chromosomes in ancestral genomes. Even when multichromosomal genomes are not allowed to have circular chromosomes, or when unichromosomal genomes are not allowed to have more than one circular chromosome, the performance of DCJ was equal to MGR's. As an indirect but important result of the implementation of DCJ, we raised the question of the evolutionary significance of the chromosomal circles that appear in ancestral genomes due to the block interchange operation allowed by DCJ. Chromosomal circles are known to exist in some special circumstances, but our focus on DCJ leads us to wonder about their possible evolutionary significance and to call for biological comment or investigation on how widespread they are and even whether they exist as an evolutionary intermediate.

The main obstacle in this work was the optimization step, whose success allowed DCJ to outperform MGR. The major benefit of this work is that it provided us with a lower bound for the median problem. We could thus subsequently use it as a subroutine in other works, such as [64] and in the work presented in Chapter 5.

3. With respect to breakpoint distance, our purpose was first to implement the polynomial time algorithm for the breakpoint median problem found by Tannier *et al.* [53], and use it as a subroutine in the algorithm solving the small phylogeny problem, and second to statistically evaluate the performance of the two criteria, breakpoint distance and rearrangement distance represented by DCJ, relative to each other. We used the DCJ median designed in [52]. We showed that DCJ was preferable to breakpoints in the inference of branch lengths and in the inference of closely related alternative solutions. We determined this was to show that the DCJ constructions were closer to optimal according to the breakpoint criterion, and the breakpoint constructions were relatively farther from optimum using the DCJ criterion. However, in simulations, both methods inferred ancestors which were close to optimal according to the competing criterion. Another direct and useful result of this work is implementing the breakpoint median and using it in solving the small phylogeny problem because it is the fastest known median algorithm. The main obstacle in this work was the running time of the DCJ median that delayed reaching the final results and constrained the sample size of the alternative construction to 5, when 25 would have been more informative. This indicates that speed-up techniques for improving efficiency are needed before we can take real advantage of the accuracy of this method.

6.1 Future Work

My experience with heuristic improvements of ancestral gene order has stimulated an interest in exploring methods such as simulated annealing, taboo search, etc. in the context of the median problem in isolation, as integrated with the small phylogeny problem (infer ancestors, given tree topology) and the large phylogeny problem (find the topology). Given the NP-hardness of almost all versions of the median problem for rearrangement distances, and the suboptimal results produced by all the available gene order phylogeny software, exploring optimization methods, using them properly, and inventing new variants of them became a

pressing problem in the field, and a good research subject.

The breakpoint based phylogeny is an appealing area that still needs to be explored, therefore a lot of future work could be achieved. For example, since the total distance between the median and its three neighbors given by the polynomial time algorithm for the breakpoint median [53] in the context of multichromosomal genomes where chromosomes are unconstrained as to linearity or circularity, represents a lower bound, it would be interesting to solve the problem of the breakpoint median on multichromosomal genomes where chromosomes are constrained to be linear as this case is more realistic biologically. Though the latter problem is NP-hard, it would interesting to find an exact solver for it. For that purpose, a best-first branch and bound approach could be used. Then it would be interesting to compare the total distances of medians from both cases.

But the exact solver will be slow for large genome sizes, thus it will be useful for comparison purposes only. It would be interesting to work on solving the same problem by finding a heuristic algorithm for the breakpoint median, and using it as a subroutine to infer ancestral genomes in the small phylogeny problem. In inferring the breakpoint median, an approach similar to the one already used in inferring the DCJ median could be used. The algorithm and software will be useful for practical use as it represents a solution to the breakpoint median problem of multichromosomal genomes in a realistic case where chromosomes are constrained to be linear.

Phylogenetic methods are a promising approach to resolving the paralogy (multiple, non-contiguous copies of a gene) problems that plague theoretical models of genome rearrangement. By trading off gene duplication and loss costs against rearrangement costs, an approach pioneered in [77], gene family histories could be integrated with overall genome dynamics. Improvements and innovation in gene order phylogeny techniques, and new results in the gene-tree/species tree problem, together with the avalanche of new genomic data in the last few years, make this approach to paralogy an interesting and timely research subject.

Bibliography

[1] Hannenhalli, S. and Pevzner, P. 1995. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proceedings of the 27th Annual ACM-SIAM Symposium on the Theory of Computing*, ACM Press, p. 178-189.

[2] Yancopoulos, S., Attie, O., Friedberg, R. 2005. Efficient sorting of genomic permutations by translocation, inversion, and block interchange. *Bioinformatics*, 21: 3340-3346.

[3] Watterson, G.A., Hall, T.E., and Morgan, A. 1982. The chromosome inversion problem. *Journal of Theoretical Biology*, 99: 1-7.

[4] Blanchette, M., Kunisawa, T., and Sankoff, D. 1999. Gene order breakpoint evidence in animal mitochondrial phylogeny. *Journal of Molecular Evolution*, 49: 193-203.

[5] Moret, B.M.E., Tang, J., Wang, L.S., and Warnow, T. 2002. Steps toward accurate reconstructions of phylogenies from gene-order data. *Journal of Computer and System Sciences*, 65(3): 508-525.

[6] Moret, B.M.E., Siepel, A.C., Tang, J., and Liu, T. 2002. Inversion medians outperform breakpoint medians in phylgeny reconstruction from gene-order data. Proceedings of the 2nd Workshop on Algorithms in Bioinformatics (WABI 2002), *Lecture Notes in Computer Science*, 2452: 521-536.

[7] Caprara, A. 2001. On the practical solution of the reversal median problem. Proceedings of the 1st Workshop on Algorithms in Bioinformatics (WABI 2001), *Lecture Notes in Computer Science*, 2149: 238-251.

[8] DasGupta, B., Jiang, T., Kannan, S., Li, M., and Sweedyk, Z. 1997. On the complexity and approximation of syntenic distance. *Proceedings of the 1st Conference on Computational Molecular Biology (RECOMB 97)*, ACM Press, p. 99-108.

[9] Hannenhalli, S. and Pevzner, P. 1995. Transforming men into mice (polynomial algorithm for genomic distance problem). *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS 1995)*, p. 581-592.

[10] Rokas, A. and Holland, P.W.H. 2000. Rare genomic changes as a tool for phylogenetics. *Trends in Ecology and Evolution*, 15: 454-459.

[11] Sankoff, D. and Blanchette, M. 1997. The median problem for breakpoints in comparative genomics. *Lecture Notes in Computer Science*, 1276: 251-263.

[12] Bourque, G. and Pevzner, P. 2002. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Research*, 12: 26-36.

[13] Sankoff, D., Sundaram, G., and Kececioglu, J. 1996. Steiner points in the space of genome rearrangements. *International Journal of the Foundations of Computer Science*, 7: 1-9.

[14] Siepel, A. and Moret, B. 2001. Finding an optimal inversion median: experimental results. Proceedings of the 1st Workshop on Algorithms in Bioinformatics (WABI 2001), *Lecture Notes in Computer Science*, 2149: 189-203.

[15] Wang, L.S. and Warnow, T. 2001. Estimating true evolutionary distances between genomes. *Proceedings of the 33rd ACM Symposium on Theory of Computing*, p. 637-646.

[16] Bergeron, A. and Stoye, J. 2003. On the similarity of sets of permutations and its applications to genome comparison. Proceedings of the 9th International Computing and Combinatorics Conference (COCOON 2003). *Lecture Notes in Computer Science*, 2697: 69-79.

[17] Bergeron, A., Heber, S, and Stoye, J. 2002. Common intervals and sorting by reversal: a marriage of necessity. *Bioinformatics*, 18 (Supplement): 54-63.

[18] Bérard, S., Bergeron, A., and Chauve, C. 2005. Conservation of combinatorial structures in evolution scenarios. Proceedings of the 2nd RECOMB Comparative Genomics Satellite Workshop (RECOMB CG 2004), Lagergren J., ed. *Lecture Notes in Computer Science*, 3388: 1-14.

[19] Bergeron, A., Blanchette, M., Chateau, A., and Chauve, C. 2004. Reconstructing ancestral gene orders using conserved intervals. Proceedings of the 4st Workshop on Algorithms in Bioinformatics (WABI 2004). *Lecture Notes in Computer Science*, 3240: 14-25.

[20] Fitch, W. M. 1971. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20: 406-416.

[21] Sankoff, D. 1975. Minimal mutation trees of sequences. *SIAM Journal of Applied Mathematics*, 28: 35-42.

[22] Day, W. H. E., Johnson, D. S., Sankoff, D. 1986. The Computational Complexity of Inferring Rooted Phylogenles by Parsimony. *Journal of Mathematical Biosciences*, 81: 33-42.

[23] Robinson, D. F. 1971. Comparison of labeled trees with valency three. *Journal of Combinatorial Theory*, 11: 105-119.

[24] Bellman, R. E. 1957. Dynamic programming. *Princeton University Press.*

[25] Needleman, S. B. and Wunsch C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48: 443-453.

[26] Sankoff, D. and Rousseau, P. 1975. Locating the vertices of a Steiner tree in an arbitrary metric space. *Mathematical Programming*, 9: 240-246.

[27] Sankoff, D. and Kruskal, J. B. 1983. Time warps, string edits, and macromolecules: The theory and practice of sequence comparison. *Addison-Wesley Publication*, p. 45-49.

[28] Siepel, A. 2003. An algorithm to enumerate sorting reversals for signed permutations. *Journal of Computational Biology*, 10: 575-597.

[29] Sankoff, D. and Blanchette, M. 1997. The median problem for breakpoints in comparative genomics. Proceedings of the 3rd International Computing and Combinatorics Conference (COCOON 1997). *Lecture Notes in Computer Science*, 1276: 251-263.

[30] Nadeau, J.H. and Taylor, B. 1984. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences of U.S.A.*, 81: 814-8.

[31] Sankoff, D., Sundaram, G. and Kececioglu, J. 1996. Steiner points in the space of genome rearrangements. *International Journal of the Foundations of Computer Science*, 7: 1-9.

[32] Bourque, G., Pevzner, P.A. 2002. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Research*, 12: 26-36.

[33] Hartigan, J.A. 1973. Minimum mutation fits to a given tree. *Biometrics*, 29: 53-65.

[34] Booth K. and Lueker G. 1976. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13: 335-379.

[35] Parida, L. 2005. A PQ tree-based framework for reconstructing common ancestors under inversions and transpositions. *IBM Research Report* RC 23837.

[36] Meidanis, J. and Munuera, E.G. 1996. A theory for the consecutive ones property. *Proceedings the 3rd South American Workshop on String Processing (WSP 1997).* Carleton University Press, p. 194-202.

[37] Telles, G.P. and Meidanis, J. 2005. Building PQR trees in almost-linear time. Proceedings of the 2nd Brazilian Symposium on Graphs, Algorithms, and Combinatorics (GRACO 2005). *Electronic Notes in Discrete Mathematics*, 19: 33-39.

[38] Sankoff, D., Bryant, D., Deneault, M., Lang, F. and Burger, G. 2000. Early eukaryote evolution based on mitochondrial gene order breakpoints. *Journal of Computational Biology*, 7: 521-535.

[39] Tang, J., and Moret, B.M.E. 2003. Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. Proceedings of the 8th Workshop on Algorithms and Data Structures (WADS 2003). *Lecture Notes in Computer Science*, 2748: 37-46.

[40] Lemieux C, Otis C, Turmel M. 2000. Ancestral chloroplast genome in *Mesostigma viride* reveals an early branch of green plant evolution. *Nature*, 403: 649-652.

[41] Turmel, M., Otis, C. and Lemieux, C. 2005. The complete chloroplast DNA sequences of the charophycean green algae *Staurastrum* and *Zygnema* reveal that the chloroplast genome underwent extensive changes during the evolution of the *Zygnematales. BMC Biology*, 3: 22.

[42] Turmel M, Otis C, Lemieux C. 2002. The chloroplast and mitochondrial genome sequences of the charophyte *Chaetosphaeridium globosum*: insights into the timing of the events that restructured organelle DNAs within the green algal lineage that led to land plants. *Proceedings of the National Academy of Sciences of U.S.A.*, 99: 11275-80.

[43] Turmel M, Otis C, Lemieux C. 2006. The chloroplast genome sequence of *Chara vulgaris* sheds new light into the closest green algal relatives of land plants. *Molecular Biology and Evolution*, 23: 1324-38.

[44] Ohyama, K., Fukuzawa, H., Kohchi, T., and 13 co-authors. 1986. Chloroplast gene organization deduced from complete sequence of liverwort Marchantia polymorpha chloroplast DNA. *Nature*, 322: 572-574.

[45] Adam, Z., Turmel, M., Lemieux, C., and Sankoff, D. 2007. Common intervals and symmetric difference in a model-free phylogenomics, with an application to streptophyte evolution. *Journal of Computational Biology*, 14: 436-445.

[46] Parida, L. 2006. A PQ Framework for Reconstructions of Common Ancestors and Phylogeny. *Lecture Notes in Computer Science*, 4205: 141-155.

[47] Landau, G. M., Parida, L., Weimann, O. 2006. Using PQ Trees for Comparative Genomics. *Lecture Notes in Computer Science*, 3537: 128-143.

[48] Hoberman, R., Sankoff, D., and Durand, D. 2005. The statistical analysis of spatially clustered genes under the maximum gap criterion. *Journal of Computational Biology*, 12: 1081-1100.

[49] Bergeron, A., Corteel, S., and Raffinot, M. 2002. The algorithmic of gene teams. Proceedings of the 2nd Workshop on Algorithms in Bioinformatics (WABI 2002). *Lecture Notes in Computer Science*, 2452: 464-476.

[50] Zhu, Q., Adam, Z., Choi, V., and Sankoff, D. 2008. Generalized gene adjacencies, graph bandwidth and clusters in yeast evolution. The 4th International Symposium on Bioinformatics Research and Applications (ISBRA 2008). *Lecture Notes in Computer Science*, 4983: 134-154.

[51] Bergeron, A., Mixtacki, J., and Stoye, J. 2006. A unifying view of genome rearrangements. In Bucher, P., Moret, B., eds., Proceedings of the 6th Workshop on Algorithms in Bioinformatics (WABI 2006). *Lecture Notes in Computer Science*, 4175: 163-173.

[52] Adam, Z. and Sankoff, D. 2008. The ABCs of MGR with DCJ. *Evolutionary Bioinformatics Journal*, 4: 69-74.

[53] Tannier, E., Zheng, C., Sankoff, D. 2009. Multichromosomal Median and Halving Problems under Different Genomic Distances. *BMC Bioinformatics*, 10: 120.

[54] Cosner, M., Jansen, R., Moret, B., Raubeson, L., Wang, L., Warnow, T., Wyman, S. 2001. An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In Sankoff, D., Nadeau, J., eds., *Comparative Genomics*, Kluwer Academic. p. 99-121.

[55] Moret, B., Wang, L., Warnow, T., Wyman, S. 2001. New approaches for reconstructing phylogenies from gene order data. *Bioinformatics*, 17(Supplement): 165-173.

[56] Murphy W. J., Larkin D. M., Wind A. E., Bourque G., Tesler G., Auvil L., Beever J. E., Chowdhary B. P., Galibert F., Gatzke L., Hitte C., Meyers S. N., Milan D., Ostrander E. A., Pape G., Parker H. G., Raudsepp T., Rogatcheva M. B., Schook L. B., Skow L. C., Welge M., Womack J. E., O'Brien S. J., Pevzner P. A., Lewin H. A. 2005. Dynamics of Mammalian Chromosome Evolution Inferred from Multispecies Comparative Maps. *Science*, 309: 613-617.

[57] Kuttler, F. , Mai, S. 2007. Formation of non-random extrachromosomal elements during development, differentiation and oncogenesis. *Seminars in Cancer Biology*, 17: 56-64.

[58] Mazowita, M., Haque, L., Sankoff, D. 2006. Stability of rearrangement measures in the comparison of genome sequences. *Journal of Computational Biology*, 13: 554-566.

[59] Bryant, D. 1998. The complexity of the breakpoint median problem, Technical report CRM-2579. Centre de recherches mathématiques, Université de Montréal.

[60] Pe'er, I. and Shamir, R. 1998. The median problems for breakpoints are NP-complete. *Electronic Colloquium on Computational Complexity.*

[61] El-Mabrouk, N. and Sankoff, D. 2003. The reconstruction of doubled genomes. *SIAM Journal of Computing*, 32: 754-792.

[62] Alekseyev, M. and Pevzner, P. A. 2008. Colored de Bruijn graphs and the genome halving problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4: 98-107.

[63] Warren, R. and Sankoff, D. 2008. Genome halving with double cut and join. *Proceedings of the 6th Asia-Pacific Bioinformatics Conference*, 6: 231-240.

[64] Zheng, C., Zhu, Q., Adam, Z., and Sankoff, D. 2008. Guided genome halving: hardness, heuristics and the history of the Hemiascomycetes. *Proceedings of the 16th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2008).*

[65] Mixtacki, J. 2008. Genome halving under DCJ revisited. Proceedings of the 14$^{\text{th}}$ International Computing and Combinatorics Conference (COCOON 2008). *Lecture Notes in Computer Science*, in press.

[66] Tesler. G. 2002. Efficient algorithms for multichromosomal genome rearrangements. *Journal of Computer and System Sciences*, 65: 587-609.

[67] Ozery-Flato, M., Shamir, R. 2003. Two notes on genome rearrangement. *Journal of Bioinformatics and Computational Biology*, 1(1): 71-94.

[68] Jean, G. and Nikolski, M. 2007. Genome rearrangements: a correct algorithm for optimal capping. *Information Processing Letters*, 104(1): 14-20.

[69] Lau, H. 2006. A Java Library of Graph Algorithms and Optimization. Discrete Mathematics and its Applications Series (Editor Kenneth H. Rosen) Chapman & Hall. ISBN: 1584887184.

[70] Swenson, K.M., Marron, M., Earnest-DeYoung, J.V., and Moret, B.M.E. 2005. Approximating the true evolutionary distance between two genomes. *Proceedings of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX 2005)*, SIAM Press, p. 121-129.

[71] Mikkelsen, T.S., Wakefield, M.J., Aken, B., Amemiya, C.T., Chang, J.L., Duke, S., Garber, M., Gentles, A.J., Goodstadt, L., Heger, A., *et al.* 2007. Genome of the marsupial Monodelphis domestica reveals innovation in non-coding sequences. *Nature*, 447: 167177.

[72] Sankoff, D. 2006. The signal in the genomes. Public Library of Science Computational Biology 2, e35.

[73] Saxe J. 1980. Dynamic-programming algorithms for recognizing small-band-width graphs in polynomial time. *SIAM Journal of Algebraic and Discrete Methods*, 1: 363369.

[74] Gurari, E. M., Sudborough, I.H. 1984. Improved dynamic programming algorithms for bandwidth minimization and the mincut linear arrangement problem. *Journal of Algorithms*, 5: 531-546.

[75] Cuthill, E., Mckee, J. 1969. Reducing the bandwidth of sparse symmetric matrices. *Proceedings of the 24th National Conference of the ACM.* Association for Computing Machinery, p. 157-172.

[76] Sherman, A.H., Liu, W. 1976. Comparative Analysis of the Cuthill-McKee and the Reverse Cuthill-McKee Ordering Algorithms for Sparse Matrices. *SIAM Journal on Numerical Analysis*, 13(2): 198-213.

[77] Sankoff, D. and El-Mabrouk, N. 2000. Duplication, rearrangement and reconciliation. In Sankoff, D. and Nadeau, J.H. (eds.). Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map alignment and the Evolution of Gene Families. *Kluwer Academic*, p. 537-550.