COMPUTATIONAL METHODS FOR PHYLOGENETIC ANALYSIS

A THESIS

Submitted in partial fulfilment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

in

ELECTRONICS AND COMPUTER ENGINEERING

by

MOHD. ABDUL HAI ZAHID

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

APRIL, 2007

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled COMPUTATIONAL METHODS FOR PHYLOGENETIC ANALYSIS in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy and submitted in the Department of **Electronics and Computer Engineering** of Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from January 2004 to April 2007 under the supervision of **Dr. R. C. Joshi,** Professor and **Dr. Ankush Mittal,** Associate Professor, **Department of Electronics and Computer Engineering** of Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Zahid

(MOHD. ABDUL HAI ZAHID)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

[illegible] [illegible] 5/7

(Dr. Ankush Mittal)
Associate Professor
Dept. of E&CE.

(Dr. R. C. Joshi)
Professor
Dept. of E&CE.

Date: 5/4

The Ph.D. Viva-Voce Examination of Mohd. Abdul Hai Zahid, Research Scholar has been held on ....

Signature of Supervisor(s) Signature of External Examiner

Abstract

The huge volume of genetic data generated through biological experiments is not useful until it is analyzed and classified properly. A single human genome project ensued in 3.2 million base pairs of nucleotide sequence data. Manual analysis of this kind of huge data is impossible [1]. The quest of classifying the genetic material leads to one of the most important field of biology called phylogenetics. Phylogenetics is the study of relationship among species or genes with the combination of molecular biology and mathematics. The large applications and availability of genetic data indicate the serious requirement for accurate, fast and generic phylogenetic analysis tools to process genomic data. This is presented in detail through citing recent works in the first part of the thesis.

It is well known that the network representation of the evolutionary relationship provides a better understanding of the evolutionary process and the non-tree like events such as horizontal gene transfer, hybridization, recombination and homoplasy. The second part of the thesis proposes a pattern recognition based approach for the construction of phylogenetic network due to recombination. Unlike other works [2, 3, 4, 5, 6, 7], we used both similarity and dissimilarity of Single Nucleotide Polymorphism (SNP) sites for classifying the nodes into mutation and recombination nodes. The use of both distance measures helps to overcome the information loss due to converting of sequence data into distances.

The proposed algorithm [8] conducts a row-based search to detect the recombination nodes which significantly reduces the complexity of the proposed algorithm. Comparisons with existing algorithms show the superiority of our approach both in network visualization and time complexity.

Third part of the thesis presents the problem of merging a set of given smaller phylogenetic trees into a bigger tree called supertree. There exist nearly 1.7 million known species. Constructing tree of life consisting of these species is impractical. A method which exploits the features of distance and character based phylogenetic reconstruction methods is proposed to construct phylogenetic tree of life. We developed a variant of well known Unweighted-Pair-Group Method with Arithmetic mean (UPGMA) [9] for constructing the rooted supertree [10]. The algorithm satisfies all the desirable properties of the supertree algorithms and gives the better visualization of the supertree than the existing supertree methods. We also consider the problem of supertree reconstruction for unrooted input trees [11].

The fourth part of the thesis introduces the problem of incorporating additional evolutionary information for constructing supertrees. Most of the existing supertree methods combine the input trees based on the topological information carried by each of the input tree and other evolutionary information is usually ignored [12]. If the available evolutionary information is considered with tree topology for amalgamating the input collection of trees, the resulting supertree would be more accurate and resolved. In this part, we propose a novel supertree method [13], which incorporates relative time divergence information with tree topologies. This method returns a supertree even for incompatibilities such as conflicts in divergence dates and incompatibility between topology and divergence dates. The conflicts are resolved based on graph theoretic concepts [14].

Another advantage of the algorithm is that the resulting supertree represents all nestings present in the input collection, which is not possible with other existing algorithms.

Fifth part of the thesis explores the problem of merging smaller trees with some of the labelled internal nodes. Generally, most of the existing supertree methods are developed based on the implicit assumption that only leaf nodes are labelled in the input tree collection. On the other hand, the phylogenies constructed based on morphological studies often contain the labelled internal nodes, thus requiring for a more generalized supertree approach. In this part of the dissertation, we propose an optimization based divide and conquer method [15] to combine semi-labelled trees. The algorithm returns a supertree even for (descendent level) incompatible input trees. Moreover, it also preserves all the nestings present in the input tree collection. On the other hand, most of the existing methods are neither capable of handling incompatible input trees nor the resulting supertree represent all the nestings present in all the input trees.

Finally, the contributions made in the thesis are summarized and scope for the future work is outlined.

Acknowledgements

First and the foremost, I would like to thank my supervisor Prof. R. C. Joshi for his invaluable advices, guidance, encouragement and for sharing his broad knowledge. He has been very generous in providing the necessary resources to carry out my research. He is an inspiring teacher, a great advisor, and most importantly a nice person.

I would like to express my deep sense of gratitude to my co-supervisor Dr. Ankush Mittal. I had an opportunity to assist him in Operating System labs and tutorials. It was a wonderful learning experience. His 40 hours-a-week study schedule, weekly reporting and seminar fundas helped me to stay focused and motivated throughout my stay in IITR as a research scholar. I could still remember how my first draft of my first research paper turned red after his corrections and the way he encouraged me to work and to correct it. No matter what I write here, I will be incomplete to acknowledge his help, encouragement, guidance, and support. Every word in my technical papers and thesis owe him thanks for his efforts in reading and discussing them. Dr. Ankush Mittal's insistence on preparing and giving the best possible presentation, writing the best possible paper, and being the best possible student, definitely changed the way of my thinking. His encouragement and moral support in every phase of my PhD made it possible for me to learn a lot and complete the dissertation in the scheduled time. Most importantly, Dr. Ankush Mittal's technical soundness along with spirituality and wonderful social behavior made him an ideal supervisor, a visionary and a wonderful teacher for me.

I specially thank my research committee members for their patience during evaluations and discussions. I am thankful to the Ministry of Human Resource Development (MHRD) and the All India Council for Technical education (AICTE) for providing financial help to complete my thesis.

The department of Electronic and Computer Engineering has provided an excellent environment for my research in bioinformatics. I spent many enjoyable hours with department members and fellow students. I enjoyed sharing ideas over a cup of tea, especially with Krishan Saluja, R. C. Gangwar and Soloman Raju. Though, I am quite younger to them they responded positively to my criticism on several issues. Without this friendly environment and freedom many of my ideas would not have come to fruition. I would also like to thank all the friends, who helped me directly or indirectly during my stay at Roorkee.

On a personal note, I owe everything to the Almighty and my parents. My father's hand written mails and long telephonic conversations with my mother and other family members provided me the mental support I needed. Evidently, work on a PhD thesis comes with a daily dose of stress and uncertainty. To evade this, I involved myself in various administrative activities at the institute. I am thankful 'Ravindra Bhawan' inmates and all the Students Affair Council (SAC 2006) members for their support.

Mohd. Abdul Hai Zahid

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is human nature that whenever man sees any object, whether living or non-living, he tries to name it and associate it with the related objects. The quest to name living organisms and finding the relationship between them leads to one of the most important field of biology called *Phylogenetics*. It sheds light on how the species or genes have evolved, giving a greater insight into the biological system.

1.1 An overview

Phylogenetics is the study of relationship among species or genes with the combination of molecular biology and mathematics. Two major developments, computer algorithms and availability of molecular data, have changed the way of finding the relationships between the species. The increase in the availability of DNA and protein sequence data has increased the interest in molecular phylogenetics and classification. Molecular phylogenetics overcomes the limitations of morphological phylogenetics such as, convergent evolution, finding the relationship among bacteria and the comparison of distinctly related organisms [16].

Large number of research projects in the field of phylogenetics shows the importance and need for the computational methods for the fast and accurate construction of the phylogenies.

Tree Of Life (TOL) project (refer to http://www.tolweb.org/tree) presents one of the greatest challenges of science in reconstructing the evolutionary history of every living organism on the earth. Under this many smaller projects such as "The Green Tree of Life", and "Assembling the Fungal Tree of Life (AFTOL)" are also proposed. Huge support is also provided from different funding agencies and countries to the development of computational methods to reconstruct the tree of life (refer to http://www.phylo.org).

Development and use of phylogenetic systems to enhance food safety and food security project is proposed to produce an evolutionary framework to understand the ecology, virulence and epidemiology of Listeria, Clostridium, and Enterococcus species. It will contribute to the development and implementation of effective control strategies for these species [17]. This project is supported by the United States Department of Agriculture. Similarly, research projects are also proposed and carried out to the reticulate evolution in the species.

Reconstruction of ancestral relationships from contemporary data is widely used to provide evolutionary and functional insights into biological system. These insights are largely responsible for the development of new crops in agriculture, drug design and in understanding the ancestors of different species. The other applications of phylogenetics includes:

1. Understanding of the process of evolution of different species, constraints, behavior and evolutionary time [18, 19, 20].

2. Conservation studies to reduce the phylogenetic diversity [21].

3. Discovering functional relationship between the cells due to similarity in the function of genes [22, 23].

4. Performing functional prediction of genes. Most of the similarly looking genes show the similar functionality. Looking at the evolutionary history of the newly sequenced gene will help in predicting its functionality [24].

5. Finding protein-protein interactions [25].

6. Helping ligands predication [26].

7. Development of vaccines, antimicrobial and herbicides [27, 28].

8. Forensics studies [29].

Since last three decades, phylogenetics has taken a new look due to the availability of molecular data, computational infrastructure and methods for phylogenetic tree construction. Many fields as given above are conceptually united with molecular phylogenetics.

1.2 Motivation

The large applications and huge available sequence data, reflect the need for developing fast and accurate methods for constructing the phylogenies. Most of the existing methods accepts molecular sequence data or distance data for the construction of the phylogenetic history. Usually evolutionary history is represented as an acyclic graph called a phylogenetic tree.

Advances in molecular biology and genomic lead to the large amounts of data, both in number of taxa as well as sequence length, available for phylogenetic construction. Moreover, most of the inferences in comparative biology depend on accurate estimates of evolutionary relationships [30, 31]. In this scenario character based methods, such as maximum likelihood, parsimony search, or quartet puzzling, play an important role. But they are very slow and cannot scale up to the size of the genetic data available. These methods compare trees according to a specific criterion. Ideally, all possible trees should be compared. However, the number of trees with n sequences are very large (i.e, $\Pi_{k=3}^{n}(2k-5)$). For example, given 20 sequences this number is of the order of 10^{20}. Thus, these methods go intractable for 15 to 30 taxa. On the other hand, distance based methods suffer with the problem of information loss due to converting sequences into distances. This information loss leads to inaccuracies in the phylogenetic tree reconstruction. This leaves a huge gap to be filled for fast and accurate phylogenetic tree with large number of taxa.

Phylogenetic tree reconstruction methods cannot be used to represent extensive phylogenetic relationships. There are situations or reticulate events where trees are not sufficient to represent phylogenetic relationships. They are: (1) horizontal or lateral gene transfer; (2) hybridization between the species; (3) microevolution of local populations within the species; (4) homoplasy, the portion of phylogenetic similarity resulting from evolutionary convergence [32]. A more generic framework called phylogenetic network is suitable for representing nontree or reticulate events.

After generating the genomic data of thousands of living organisms, now comes the task of classifying them and making a single tree of life. This is a challenging task and till date no algorithm exists that can compute most accurate tree of life. The existing phylogenetic reconstruction algorithms cannot be used for this task as they suffer with poor efficiency and computational hardness problems. The problem with the phylogenetic tree construction approach is that they cannot be used for the classification of 1.7 million described species on the earth. The existing methods cannot be used for constructing the tree of life due to aforementioned shortcomings.

In summary, efficient algorithms for constructing phylogenetic network due to recombination has to be developed. Phylogenetic network representation provides deeper insights and direction for designing drug for different viruses [33, 34, 35]. The methods for combining trees, either rooted or unrooted, and robust methods to add additional information like ancestral time divergence and semi-labelled trees are demanded by the biologists. The above discussion reflects the urgent need for the fast and accurate phylogenetic networks and supertrees construction methods.

1.3 Problem statement

The objective of this thesis is the development of algorithms for constructing phylogenetic network, supertree and supertree variants. This problem can be divided into the following subproblems:

1. *Given a set of binary sequence M, finding a phylogenetic network N that minimizes the number of recombination events with some biologically motivated structural properties.*

2. *Given a collection of small phylogenetic trees T combine them into single tree in such away that no branching information carried by any input tree is lost.*

3. *Given a set of divergence date statements D and a collection of small phylogenetic trees T, combine the input tree collection into single tree in such*

a way that no branching information carried by any input tree is lost and divergence date information is also preserved.

4. *Given a collection of small phylogenetic trees T combine them into single tree in such a way that the resulting tree preserves all the ancestral relationships described by the input collection.*

A general strategy is followed in order to solve the aforementioned problems. The steps are as follows:

1. Investigating the key existing methods of phylogenetic networks and supertrees.
2. Exploring the major inadequacies of the existing systems.
3. Proposing an efficient method that addresses the problems of the existing methods.
4. Exploring the correctness of the proposed approach.
5. Evaluating and performance using analytical and biological data sets.

1.4 Organization of the thesis

The thesis is organized as follows. Chapter 2 presents a brief literature review on Phylogenetic networks, supertrees and their variants. An efficient algorithm for constructing the phylogenetic network due to constrained recombination is proposed in chapter 3. Chapter 4 presents an efficient distance based supertree algorithm. An algorithm for constructing supertree with additional evolutionary information, such as fossil records, molecular dates and relative divergence dates, is given in chapter 5. In chapter 6, an algorithm for constructing semi-labelled supertree from a nested of semi-labelled input tree collecting is proposed. Chapter 7 finally concludes the thesis by presenting our contributions and directions for the future work.

Chapter 2

Review and Background

The term phylogenetics is derived from the Greek. Phylon means race or old family, and genos means birth or origin, and gennetikos means related to generation or the genesis of something. The goal of the phylogenetic study is to provide insights into the evolutionary relationships, divergence time, and the patterns and constraints of evolution process. This will help to understand the process of evolution and most of biological predications are based on this relationship. This chapter presents the background information required to understand the phylogenetic reconstruction methods and also gives a survey of prominent phylogenetic network and supertree methods.

2.1 Phylogenetic Networks

Traditionally, evolutionary relationship is represented by a tree model. However, recent developments in molecular phylogenetics indicated that the exchange of genetic material between the lineages needs a more generic framework to represent the evolutionary relationship [36]. Different conflicting phylogenetic signals such as horizontal gene transfer, hybridization, and homoplasy can only be represented with the help of a framework called networks rather than the trees [37, 38, 39]. In this section, we briefly present the biological behavior behind reticulate evolution along with the review of existing phylogenetic network reconstruction algorithms.

In evolutionary biology, any perfect phylogenetic history leads to a tree like representation of evolutionary history. The events such as horizontal gene transfer, hybridization, homoplasy or micro-evolution are known as reticulate events. The evolutionary history during the presence of reticulate events cannot be represented by a simple tree structure. The genes involving in reticulate event leads to a different tree than the one constructed using the genes that are not involved in the reticulate event. A network like structure is used to represent these events and both the trees.

2.1.1 Preliminaries

Biologists have recognized long before that the tree representation of evolutionary relationship oversimplifies the view of the process of evolution, as it cannot take into account the non-tree like or reticulate events. In this section, we cover both biological aspects as well as mathematical representation of the phylogenetics networks.

Reticulate events

Horizontal Gene Transfer (HGT): The transfer of genetic material between the lineages takes place through the sexual process. The changes might take place in the offspring gene sequence during the replication of genes. This leads to the change in some of the characters of the offspring. On the other hand, if the genetic material is transferred directly from one lineage to another, it is called horizontal gene transfer. In this process few genes or sometimes only a part of the gene is transferred. Horizontal gene transfer does not result in a new lineage or species. A network representing HGT is shown in Figure 2.1(a), and its two induced trees are shown in Figure 2.1(b) and 2.1(c).

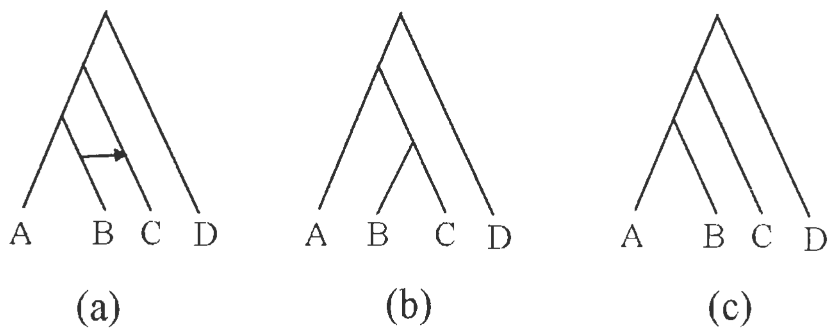


Figure 2.1: (a) Network representing HGT between B and C. (b) and (c) are induced trees of network (a).

The tree represented in Figure 2.1(b) is appropriate for the genes acquired through horizontal gene transfer, and the tree shown in Figure 2.1(c) is appropriate for the genes inherited from the ancestor of the species. Horizontal gene transfer is common phenomenon in bacteria. They develop the ability to adapt to new environments by acquiring the new genes by the process of horizontal gene transfer. The major mechanisms for horizontal gene transfer are:
Transformation: Bacteria take the DNA fragments from the environment. It mediates an exchange of any part of chromosome. Only short sequences are transformed in this way.
Conjugation: Conjugal plasmids or conjugal transposes mediate it. Long fragments of DNA can be transferred by conjugation.
Transduction: This is the transfer of DNA by phase. It requires the donor and

recipient share cell surface receptors for phage binding. This happens in limited and closely related bacteria.

Hybridization: In hybridization, two lineages recombine to create a new species. This non-tree evolutionary event is shown in Figure 2.2(a), and Figure 2.2(b) and 2.2(c) shows the induced trees of 2.2 (a).

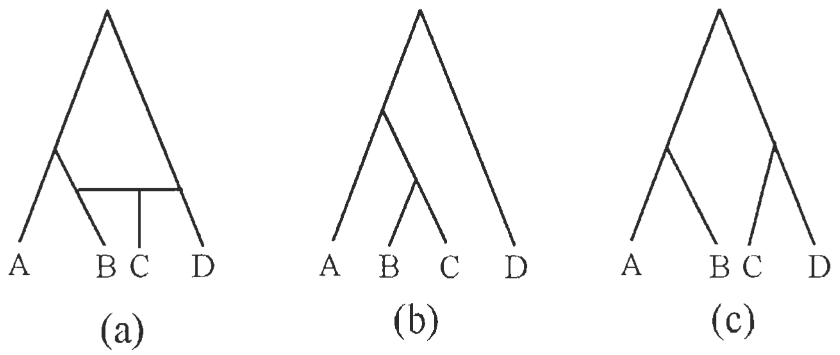


Figure 2.2: (a) Network representing hybridization between B and C. (b) and (c) are induced trees of network (a).

If the new species have the same number of chromosomes as the parent species then the process is called *diploid hybridization.* The *polypolide hybridization* results in a species that has the number of chromosomes equal to the sum of the number of chromosomes of its parents. The major mechanisms of hybridization are:

Autopolyploidazation is a speciation event involving the doubling of chromosomes within a single species. It produces a bifurcating speciation event in phylogenetic tree.

Allopolyplodization is hybridization between two species, when the offspring acquires a complete diploid chromosome complements of the two parents.

Diploid hybrid speciation is a normal sexual event that takes place between plants of different but related species.

Hybridization is common in plants, fishes, amphibians and reptiles, and is absent in other groups, particularly in birds, mammals and most anthropods.

Homoplasy: The resemblance of body structure and functions of the organs of two different species, which do not have the common ancestral origin, is called homoplasy. This is the result of convergent evolution, thus the two species are close to each other when compared on the anatomical basis, whereas genetically they are not.

Genetic Recombination: The exchange of genetic material between the homologous chromosomes is called the genetic recombination. It is the process which gives the new combinations of genetic material. Recombination creates a nontree evolution in the lineages. This may occur at any part of the chromosome. Homologous chromosomes are paired during the prophase of meiosis. In crossing over, two chromosomes swap a portion of their genetic material. After the separation member of a pair of chromosomes contain a part of its partners genetic material.

Genetic recombination has a great influence on the genetic structure of the population and explains a considerable amount of genetic diversity in the populations of sexually reproducing species.

Mathematical modelling of reticulate events

Mathematically, the reticulation events are visualized as Directed Acyclic Graphs (DAGs). Strimmer et al. [40] proposed the use of DAG model with some properties, called Acyclic Recombination Graph (ARG), for applying maximum likelihood to directed splits graphs. However, splits graphs are representations of possible incompatibilities in sequence data sets and not phylogenetic networks.

Hallett and Lagergren [41] used DAGs with a set of simple assumptions that were more biologically realistic than splits graphs. They created a method for inferring lateral gene transfer events when one is attempting to reconcile gene trees and species trees. Linder et al. [42] proposed a model of phylogenetic network based on DAGs, following Strimmer [40], to represent the topology of phylogenetic networks, adding a set of simple conditions, similar to Hallett [41] to ensure that resulting DAGs reflect the properties of biological reticulation. The mathematical model is formally described as follows:

A phylogenetic network is directed acyclic graph $N = (V, E)$, where the nodes are partitioned as tree and network nodes. Similarly, edges are also partitioned. Let a node $v \in V$ be tree node then if indegree of v is 0 and outdegree is 2 then v is root of the tree. If indegree is 1 and outdegree is 2 then v is internal node and if the indegree is 1 and outdegree is 0 then v is a leaf node. For v being a tree node the indegree ≤ 1. A node $v \in V$ is said to be network node if indegree is 2 and outdegree is 1. Similarly, an edge $e \in E$ is said to be network edge if it is incident on a network node, otherwise it is a tree edge. A simple DAG that depicts a phylogenetic network is shown in Figure 2.3. Nodes X, Y are tree nodes, R is a root node and Z is a network node. Doted edges represents network edges and solid edges indicates the tree edges.

Neighbor-Net [4], SplitsTree [43], and T-Rex [44] are some popular methods that use a simple graph with cycles to represent the phylogenetic network.

Approaches for phylogenetic network construction The network construction methods can be broadly classified into four categories [42]. They are as follows:

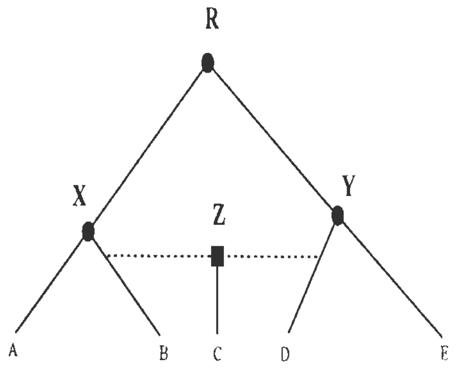


Figure 2.3: Simple DAG depicting phylogenetic network.

- First approach is to identify the genes involved in gene transfer then construct phylogeny after removing them. Most of the phylogenies constructed are based on this approach.

- Second approach is to use tree as underlying structure for a network construction. First, a phylogenetic tree is constructed based on existing algorithms and then the non-tree like edges are added to the tree based on some optimization criteria.

- Third approach is to construct many trees using different partitions of data or different algorithms then reconcile them. Whenever reconciliation fails there will be a conflict and can be represented a reticulation edge.

- Identifying the incompatibilities in the data beforehand and constructing all possible resolutions through reticulation from which biologist will choose the most relevant one.

2.1.2 Phylogenetic Network construction algorithms

Reticulated network [3] reconstruction method, implemented in T-REX package [44], uses a phylogenetic tree topology as a basic structure for reconstruction of the phylogenetic network. In this algorithm, first a phylogenetic tree is constructed for the given distance matrix using any existing tree reconstruction algorithm. Next new branches, known as reticulated branches, are added one at a time in order to minimize the least square criterion given in below equation, yielding weighted least-square loss function.

$$Q = \sum_{i \in X} \sum_{j \in X} (d(i,j) - \delta(i,j))^2$$

where $\delta(i,j)$ and $d(i,j)$ are the given path length and the minimum path length after adding a reticulate branch between the nodes i and j respectively. The length of newly added reticulate branch is calculated using the following equation:

$$l(xy,k) = \frac{\sum_{p=k+1}^{m} \sum_{(i,j) \in A_p} (\delta(i,j) - Min\{\delta(i,x) + \delta(j,y); \delta(j,x) + \delta(i,y)\})}{\sum_{p=k+1}^{m} |A_p|}$$

where l is the possible path length of the reticulated branch between x and y node, k is number of reticulation branches and A is a set of pairs of taxa such that adding a new reticulate branch will change the minimum path length between them. The set A is partitioned into m subsets, for the partitioning criteria for mathematical details, refer to [3] and [32]. This formula is used to compute the length of the new reticulate branch on the fixed interval $l_k \leq l \leq l_{k+1}$.

The total number of nodes in a binary unrooted phylogenetic tree will be $2n - 2$. Therefore, the maximum number of branches on a reticulated network can be $(2n-2)(2n-3)/2$ and the maximum number of branches in a complete graph is $n(n-1)/2$. Thus, any of these two is the maximum number of nodes in a network. If latter is considered as limit, then the degree of freedom of the network with N network branches can defined as $((n(n-1)/2) - N)$.

A penalty function opposing the loss of degree of freedom to the gain in fit is considered as the goodness of fit or stopping criterion as given in the following equation:

$$Q_1 = \frac{\sqrt{\sum_{i \in X} \sum_{j \in X} (d(i,j) - \delta(i,j))^2}}{\frac{n(n-1)}{2} - N}$$

The function Q_1 has only one minimum value over the interval $[2n-3, n(n-1)/2]$ amongst possible values of N. This minimum value defines a stopping rule for addition of new branches to the reticulate phylogeny. Many other stopping criteria are given in [43].

T-REX computes the reticulation network by first computing a phylogeny and subsequently forming a network by adding branches, which minimizes certain least square loss function. This restriction is time consuming and causes problem if the data is not tree like.

Split Decomposition [43] used the concepts of Split Graphs [45] and it is a transformation approach. Generally a phylogenetic tree T is made up of nodes (vertices), and branches (edges). The leaf nodes in a phylogenetic tree represent the living species at a particular time period. A tree reconstruction for X taxa is equivalent to compatible splits and determining the weight of each split.

A split in a phylogenetic tree T is the one in which when an edge is removed from T, it splits the set of taxa X into two non-empty groups. The split is called compatible if, for any two splits $S_1 = \{A_1, A_1'\}$ and $S_2 = \{A_2, A_2'\}$, one of the following four intersections is empty:

$$\{A_1 \cap A_2\}, \{A_1' \cap A_2'\}, \{A_1' \cap A_2\}, \{A_1 \cap A_2'\}$$

Any tree will represent the compatible system of splits. To obtain a network a system with fewer restrictions must be considered which is referred as weakly compatible split. For any three splits S_1, S_2, S_3 and $A_i \in S_i$, for $i = 1$, 2 and 3, at least one of the following four intersections is empty:

$$\{A_1 \cap A_2 \cap A_3\}, \{A_1 \cap A_2' \cap A_3'\}, \{A_1' \cap A_2 \cap A_3'\}, \{A_1' \cap A_2' \cap A_3\}$$

A split graph representing a weakly compatible split system is a graph $G = (V, E)$ whose leaf nodes $v \in V$ are labelled by a set of taxa X and edges $e \in E$ are straight lines representing the split. Split is represented by a band of parallel edges, in such a way that deleting all the parallel edges splits the graph into two components A and A'. The length of the edges, representing a split S, indicates its weight or support.

Split Decomposition is quite conservative. It only represents splits of taxa with positive isolation index. Many splits with negative isolation index are removed. But they may represent some conflicting information.

Neighbor-Net [4] combines the features of Neighbor Joining [46] and SplitsTree [6] algorithms. In Neighbor-Net two taxa with least dissimilarity are not added immediately instead they are kept waiting till the new node is paired up with these two taxa. Then these three nodes are linked and the distance matrix is reduced. There are three important steps as in all network reconstruction algorithms: criteria for selection of nodes, reduction of distance matrix, and finally estimation of edge lengths.

The selection of taxa to be grouped together is made as follows. Let A be the set of nodes, $N = |A|$ and d be the distance function defined between pair of nodes in A. Let $C_1, C_2, C_3, ..., C_n$ be the clusters of either size one or two. Then the distance for each pair of cluster can be calculated by the function given in the following equation:

$$d(C_i, C_j) = \frac{\sum_{x \in C_i} \sum_{y \in C_j} d(x,y)}{|C_i||C_j|}$$

The clusters are selected based on the minimization of the following function.

$$Q(C_i, C_j) = (n-2)d(C_i, C_j) - \sum_{k \neq i} d(C_i, C_k) - \sum_{k \neq j} d(C_j, C_k)$$

The nodes are selected based on the minimization of the following function.

$$Q(x, y) = (n-2)d(x, y) - \sum_{k \neq i} d(\{x\}, C_k) - \sum_{k \neq j} d(\{y\}, C_k)$$

where $x \in C_i$ and $y \in C_j$. The reduction of distance is done as follows. If x and z are neighbors for y, Neighbor-Net algorithm will replace x, y, z into two new nodes u and v. The distance form u to v is computed as:

$$\begin{aligned} d(u,a) &= \alpha d(x,a) + \beta d(y,a) \\ d(v,a) &= \beta d(y,a) + \gamma d(z,a) \\ d(u,v) &= \alpha d(x,a) + \beta d(y,a) + \gamma d(z,a) \end{aligned}$$

Theory of circular decompositions is used for the estimation of the branch lengths. The branch length can be computed as follows:

$$d(x,y) = \sum_{s \in S(\theta)} \alpha_s \delta_s(x,y)$$

where $x, y \in X, s \in S(\theta), d$ is circular distance, and θ represents the circular order. α_s is a positive constant and δ_s is a split metric. δ_s is 0 if the pair of nodes are in same part of the split S, otherwise its value will be 1. δ_s in terms of d is given in below equation.

$$\delta_s = \frac{1}{2}(d(x_i, x_j) + d(x_{i-1}, x_{j-1}) - d(x_{i-1}, x_j) - d(x_i, x_{j-1}))$$

Netting [47] is a distance based method, which first generates all the parsimonious trees for the given data and then connects the leaf nodes to form a network. The algorithm starts with connecting the sequences that are most similar. A new node is added to the tree such that the three pairwise distances are satisfied. The ancestral distance between two species is considered to be equal to the number of differences between them. A new edge is added between the nodes if homoplasy encounters between them. Gaps and invariant positions in the sequences are not considered for the analysis. Since the method tends to satisfy the distances among the species, the number of dimensions required to represent the relationship may be more, therefore the method leads to the complex network representation.

In the **median network** [48, 49] sequences are first transformed into binary data and constant sites are not taken into consideration. Each split is encoded as binary characters 0 or 1. Sites supporting the same split are clustered into one site and then the site is weighted as the number of sites clustered. As a result the species are represented as vectors of binary numbers. The vectors are used for the construction of network. Using this method, it is difficult to represent the network of 30 or more species. The median network can represent at the most parsimonious trees and results in complex network if the diversity between the sequences is more.

In summary, most of the existing [4, 43, 47, 50, 51] network construction methods are based on distance data. Converting sequences into distances leads to loss of information. Very less work has been done on the computational aspects of the phylogenetic network using sequence data except the work by Wang et al. [52] and Gusfield et al. [7].

Wang et al. [52] showed the problem of finding a perfect phylogenetic network, a network with minimum number of recombination nodes, is NP-hard. They gave an algorithm for a restricted problem, called node disjoint network, with $O(n^4)$ computing time, where n is number of leaf nodes or species. The restriction was that in the merged path of the recombination node, there is no node that is in the merge path of a different recombination node. In other words, no node can be shared by two recombination cycles, if underlying undirected graph has cycles. A recombination cycle that shares no node with any other recombination cycle is called a "gall" and a phylogenetic network with disjoint recombination cycles is called a "gall tree" [7]. Gusfield et al. [7] showed that the algorithm in [52] is incomplete and does not constitute the necessary test conditions for the existence of the gall tree. The most efficient solution for the gall tree problem, till date, is given in [7]. It uses the conflict graph as the main tool for the detection of the conflicting sites. The components of the conflict graph are used to construct galls in the gall tree. Finally, all the galls are connected, leading to the final gall tree. The algorithm can compute the gall tree, if one exists, in $O(nm+n^3)$ time, where m is the length of the binary sequence [7].

In chapter 3, we propose an efficient algorithm for the construction of the gall tree. Converting the binary sequences into distance matrix based on dissimilarity leads to the information loss. To avoid this effect we use both similarity and dissimilarity between the binary sequences as a major tool for detecting and constructing the gall tree.

2.2 Phylogenetic consensus and supertrees

Many smaller overlapping phylogenetic trees are combined in a supertree in such a way that no branching information is lost. There may exist an exponentially large number of supertrees for a given set of trees. The optimal tree is selected based upon different optimality criteria.

If all the input trees classify the same set of taxa, the result of amalgamating them will be a consensus tree. Many of the supertree algorithms use the existing consensus methods as the base for the their supertree construction. In this section, we start with the basic terminology, desirable properties of the supertree algorithms, then give a survey of prominent consensus tree methods. Finally this section ends with the critical survey of the supertree methods.

Basic Terminology

A tree is an acyclic connected graph and can be represented as $T = (V, E)$. A vertex $v \in V$ is internal if the degree of v is greater than two, otherwise v is leaf and a distinct vertex with degree two is called the root. An edge e, connected to the vertices u and v, is internal if both u and v are internal vertices, otherwise it is an external edge. The set of leaf nodes of a tree is represented as $L(T)$.

A phylogenetic or evolutionary tree is a tree T having a single internal node with degree two and rest of the internal nodes have degree three or more. If the degree of each internal node is three except root then it is called a binary rooted tree. Two rooted phylogenetic trees on set of species Y, $T^{'} = (V^{'}, E^{'})$ and $T^{''} = (V^{''}, E^{''})$ are considered as identical if there exists a bijection function $f : V^{'} \rightarrow V^{''}$, which includes a bijection from $E^{'}$ to $E^{''}$ and fits the set of leaf nodes Y. Except the root, the labelling of the internal nodes is unimportant in phylogenetics. The internal nodes represent the hypothetical ancestors of the descendents. These hypothetical ancestors are known as Least Common Ancestors (LCA) of their immediate descendents. A phylogenetic tree is shown in Figure 2.4, where clusters represent the LCAs of their descendents.

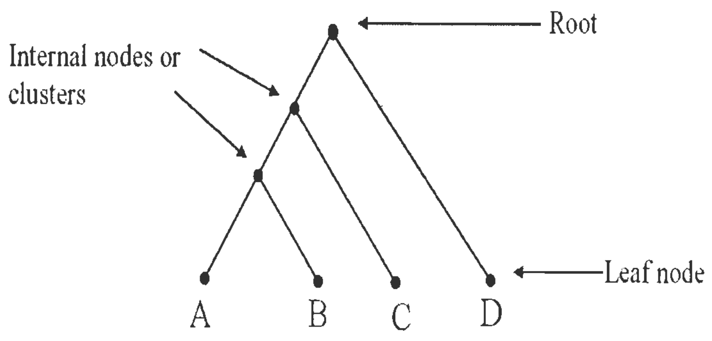


Figure 2.4: Phylogenetic tree with least common ancestors.

Restriction on phylogenetic trees

Let T be the rooted tree with the leaf set X. Given a set X' such that $X' \in X$, the topological restriction of the tree T on the taxa X' is the tree obtained by deleting the nodes, which are not in the path from root to any node in X' and then contracting the internal edges whose degree is two. The topological restriction is represented as $T|_{X'}$. An example is shown in Figure 2.5. T' is called the induced subtree of T by X'.

Compatibility of phylogenetic trees

A rooted phylogenetic tree T displays a rooted phylogenetic tree T' if T' can be obtained from an induced subtree of T by contraction and is represented as $T \leq T'$. According to Steel [53], a collection of rooted phylogenetic trees is compatible if there is a phylogenetic tree that displays all of them. Figure 2.6 shows T' and T'' in which T'' can be obtained from T' by contraction.

Clusters, Triplets, splits and quartets

A group or set is said to be a cluster on a tree if and only if it contains all

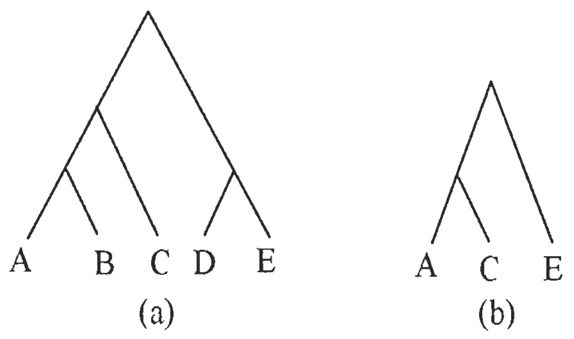


Figure 2.5: (a) T is phylogenetic tree on taxa {A,B,C,D,E}, (b) represents the restriction $T|_{\{A,C,E\}}$.

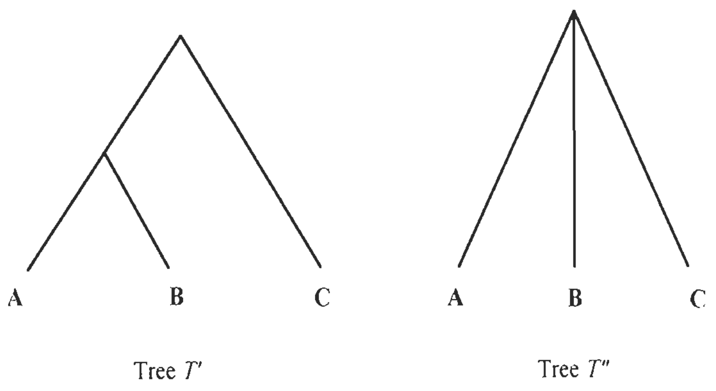


Figure 2.6: Tree T' and T'' are compatible according to Steel M.

the descendents of its most recent common ancestor as shown in Figure 2.4. A collection of clusters C is compatible if and only if for each pair of clusters A and B in C, that is $A, B \in C$, either A is in B or B is in A, or A and B are disjoint. The compatibility condition is as follows:

1. $A \subseteq B$ or $B \subseteq A$
2. $A \cap B = \emptyset$

A cluster A is compatible with the tree T if it is compatible with every cluster in T. A rooted tree T refines another rooted tree T' on the same set of taxa if every cluster in T' is a cluster of T.

A rooted triplet is the binary rooted tree with three leaf nodes. Any two leaf nodes share a least recent common ancestor. A triplet is denoted as $AB|C$, where the least common ancestor of the taxa A and B are descendents of least common ancestor of A, B and C. Triplets are used for the construction and estimation of the rooted trees.

Removing a branch in an unrooted tree will divide the tree into two connected graphs. Each group represents a subset of the set of taxa X in the tree. Let A is the group of taxa on side and B is the group of taxa on the other side, then the split can be represented as $A|B$. Splits are replacement of clusters for unrooted trees.

A quartet is a binary unrooted tree with four leaf nodes. The two pairs of taxa share a common ancestor each. The quartet is represented as $AB|CD$, where the taxa A, B and C, D share the common ancestors and there is an edge between the common ancestors of these two groups. The quartet $AB|CD$ is said to belong to tree T if there is a split in T having A and B on one side and C and D on the other.

Let T be a collection of phylogenetic trees $\{T_i, T_2, ..., T_k\}$ with leaf sets $L(T_1)$, $L(T_2)$, ..., $L(T_k)$. A supertree ST is a tree with the leaf set $L(T_1) \cap L(T_2) \cap ... \cap L(T_k)$ such that each tree T_i is an induced tree of ST. If the $L(T_1) \cap L(T_2) \cap ... \cap L(T_k) = L(T_i)$ then the result of amalgamation is called consensus tree, otherwise the result is called supertree.

Desirable properties and limitations of the consensus and supertree algorithms

There are some inherent limitations of the concensus and supertree methods given in [54]. McMorris [55] discussed three properties that cannot simultaneously be satisfied by any consensus method, Steel et al. added some more properties [53, 54]. These desirable properties are as follows:

1. The method should be independent of the order of the input trees.
2. Renaming or relabelling of the species should not result in a different tree. The output should be the old tree with the corresponding species renamed.
3. If there exist at least one parent tree for the given collection of input trees, then the output tree is one of those parent tree.
4. Each leaf that is present in at least one of the input trees should be present in the output tree.
5. The resulting tree should be computed in polynomial time.

Property 1 and 2 are essential and can be achieved simultaneously. However, third property is not easy to achieve. Therefore the alternatives are: (1) neglect the Property 3. (2) Dealing with restrictive inputs, such as rooted trees (3) Resulting a collection of supertrees that lists all the possible trees that can be drawn from the collection of input tree.

The first option can be achieved by always giving the star like tree as output. This option leads to the loss of biological information loss.

The second option is promising. Many algorithms for rooted trees satisfying all the three desirable properties exist and are discussed in the coming sections. These methods follow two additional properties.

Property 4 says that no taxa can be removed from the output tree even though the input trees are not compatible. On the other hand, property 3 is applied when the input trees are compatible.

The third option is also attractive as parsimony based methods always result in a set of more than one tree. For highly unresolved input tree the set of output tree may grow exponentially [56].

McMorris [57] has shown that any consensus tree for unrooted trees does not satisfy the following three properties simultaneously:

1. If all input trees displays $ij|kl$, then the output tree displays $ij|kl$.
2. If all the input trees display $ij|k$, then the output tree should display $ij|k$.
3. If at least one input tree display $ij|k$ and no input tree displays $ik|j$ or $kj|i$ then the resulting output tree should display $ij|k$.

The property 3 cannot be satisfied by any consensus method for the rooted phylogenetic trees [55] .

2.2.1 Phylogenetic consensus and supertree algorithms

In this section, we present a critical review on phylogenetic consensus and supertrees.

Strict consensus method

Given a collection of phylogenetic rooted or unrooted trees, this method returns the rooted or unrooted tree with the clusters or splits common to all the trees in the collection of input trees [57].

Formally the strict consensus can be defined as: Let $P = T_1, T_2, ..., T_n$ be a collection of unrooted phylogenetic trees on L and $\sum(T)$ represents all the splits in T. The strict consensus tree displays the splits present in all the input trees, represented as follows:

$$\sum(C\_Utree) = \bigcap_{T_i \in P} \sum(T_i)$$

A similar formulation can be done for the rooted trees.

Majority rule consensus tree

The majority consensus rule tree contains exactly those clusters or splits that appear in more than half of the input trees [58, 59]. Therefore, all the splits or clusters appearing the strict consensus tree are also the part of the majority rule consensus tree. In other words, the majority rule tree refines the strict consensus tree. Formally it can be defined as follows:

$$\sum(P) = \bigcup_{T_i \in P} T_i$$

For $\sigma \in \Sigma$, let $n(\sigma)$ is the number of times the split σ found in the collection of input trees. The majority rule states that:

$$\sum(M\_Utree) = \{\sigma \in \Sigma : \frac{n(\sigma)}{|P|} \geq 0.5\}$$

Similar equation can be derived for the clusters.

Loose consensus tree or semi-strict consensus tree

The loose consensus method will result in a tree with the splits or cluster found in the input trees, which are compatible with every tree in P [59, 60]. Therefore it refines the strict consensus tree. Formally it can be defined as:

$$\sum(P) = \sum(C\_Utree) \cup \{A|B \in \bigcup_{T_i \in P} (T_i) \sum(C\_Utree)\}$$

The disadvantage of this method is that the output tree displays the splits or clusters found in only one input tree [61]. The splits or clusters appearing in more than one tree are considered as reliable, but the split or cluster found in a single tree may or may not be reliable.

Adam's consensus tree

This is first consensus tree method and is defined for only rooted input trees [62]. In this method first the trees are divided into partitions and their product is determined. Let $P_1, P_2, ..., P_n$ be the partition of the set of taxa. The product of these partitions is that the taxa a and b are in the same block if the appear in the same block of all the partitions. The partitions represent the maximal clusters in the input trees.

The maximal cluster of a rooted tree T is the largest proper cluster in the tree T. The maximal cluster partition for a tree T is the partition $P(T)$ of the set of taxa with blocks equal to the maximal clusters of T. For further details Adam's paper for consensus trees [62] can be referred. The algorithm is as follows:

```
ALGORITHM: AdamsTree($T_1, T_2, ..., T_k$)
begin
1.  if the tree $T_1$ contains only one leaf then
        return leaf.
2.  for each $i = 1, ..., k$ let $\pi_i$ be the partition given by the leaf
        set of maximal subtrees of $T_i$.
3.      let $\pi$ be the partition product of $\pi_1, \pi_2, ..., \pi_k$.
4.  for each block $B$ of $\pi$ construct
        AdamsTree($T_{1|B}, T_{2|B}, ..., T_{k|B}$).
5.  Append the roots of these trees to a new vertex.
        return the new tree.
end
```

Some methods have been designed based on rooted triplets and quartets. The advantage of these methods is that the two trees sharing triplet or quartet share much more information than that in a shared cluster or split. Two trees can share lots of triplets but may not share a single cluster.

Local consensus tree

Let T is the collection of input trees, and the R is the collection of triplets of each tree of T, $R = \bigcup_{T_i \in T} r(T_i)$. The set R is said to be compatible if there exist a tree T' such that $R \subseteq r(T')$.

For any compatible set of triplets and the set of leaf nodes Aho's algorithm [63] can be used to find the local consensus tree. The algorithm is as follows:

```
Input: Collection of rooted trees $T$ and the set of leaf nodes $L$.
Output: A rooted phylogenetic tree $T'$ that displays $T$ or
        incompatibility message.
ALGORITHM: BUILD($T, L$)
begin
1.  Set $S$ to be label set of $P$.
2.  if $|S| = 1$, then the rooted tree consists of one vertex and labelled
        by an element of $S$.
3.  if $|S| = 2$ , construct $[P, S]$.
4.  Let $S_1, S_2, ..., S_k$ denote the vertex set of the components of $[P, S]$.
        if $k = 1$ then halt and return $T$ is not compatible.
5.  for each $i \in \{1, 2, ..., k\}$,
    call $BUILD(T', V_i)$, where $T'$ is the restriction of $T$ on $V_i$.
        if returns a tree, call it $R_i$ else
        return incompatible input trees.
```

```
6.  Construct a new tree $R$ by connecting the roots of
      the tree $R_i$ to a new node $r$.
7.  **return** $R$.
**end**
```

Another class of methods converts the input trees into the distance or sequence data. This converted data is used for the consensus tree construction. Some of the prominent methods are Matrix Representation with Parsimony (MRP) [64, 65], and Average consensus tree [66]

Matrix Representation with Parsimony (MRP)

This method is basically derived for the input trees, which classify the overlapping set of leaf nodes (supertrees). Here we will consider the special case, with the same leaf nodes in the input trees.

MRP [64, 65] encodes the input trees into binary characters and the supertree or consensus tree is constructed from the resulting data matrix using a parsimony tree building method. The method for converting the trees to binary data is as follows:

Let T is a collection of unrooted trees. For each spilt $A|B$ in each tree a binary character is assigned to each taxon in the tree. All the taxa in the split A is assigned the value zero (0), and the taxa appearing in the split B are assigned to one. The MRP consensus tree is the strict consensus tree of the set of tree resulting from parsimony analysis.

Average consensus tree [66]

This can be viewed as the distanced version of MRP. A distance matrix based on the path length, which is the sum of weights on each branch of the path, between the taxa is constructed for each tree. Then the average of all the distance matrices is considered for the construction of the consensus tree.

The least square difference between the two trees $T1$ and $T2$ is computed as follows:

$$\partial(T_1, T_2) = \sum_a \sum_b \{d_1(a,b) - d_2(a,b)\}^2$$

where d_1 and d_2 represents the path length distance between a and b in the tree T_1 and T_2. For a given set of input trees the average consensus tree T_c is a tree which minimizes the following function:

$$\partial(T_c, T) = \sum_{i=1}^{k} \partial(T_c, T_i)$$

This method suffers with the disadvantages such as there is no efficient algorithm to construct the tree. It is not known whether the splits appearing in all the input trees will appear in the resulting consensus tree or not.

These disadvantages are addressed in the Buneman consensus tree [67], and an efficient algorithm for its construction is given by Berry and Bryant [68].

Phylogenetic supertree methods

Supertree methods summarize the collection of trees without losing branch information. In other words, it is an algorithm which takes the collection of phylogenetic trees on the overlapping sets of taxa as an input and returns the single tree on the taxa present in all the input trees. Most of the methods use any of the consensus method as the base method for the construction of the supertrees. There are different approaches to solve this problem. Some methods solve it using graph theoretic approach, others are subtree based and yet others are recoding based approach. We present some of the popular methods in the forthcoming sections.

Mincut Algorithm

This is a subtree based algorithm and has all the advantages of using triplets and quartets. Mincut method [53] is one of the very few methods that satisfy all the desirable properties of supertree. This method is an extension of the BUILD [63]. BUILD method reports an error when the input trees are incompatible. Semple [53] modified the BUILD algorithm in such a way that if the input trees are incompatible then some of the edges are removed from the trees based on the minimum loss criteria. The minimum cost is identified using the edges which have minimum weights and when removed, they will result in a disconnected modified graphs. Therefore the algorithm results in a single tree even if the input trees are not compatible.

Modified Mincut method

This is also a subtree based method. Mincut method results in highly unresolved trees when applied to the input tree representing polytomy. Page [69] modified the Mincut algorithm to work for a general case. The basic idea was to divide the edges of the modified graph into three categories: unanimous, uncontradicted and contradicted. Whenever the modified graph is fully connected, instead of removing the edges from the minimal cut set of the graph, the contradicted edges are removed. As it is an extension of the Mincut method, it also satisfies the properties that Mincut satisfies.

RankTree

RankTree [12] method is the extension of the BUILD, which includes the ancestral time divergence of certain pair of species along with the input tree collection. The absolute divergence time of the species is not available. An alternative approach of relative time divergence is considered in this method.

A rank function is defined for the collection of input tree, which assigns the ranks to the internal vertices based on the order of speciation. A ranked tree is shown in Figure 2.7.

The Figure 2.7 indicates that the least common ancestor of the species A and

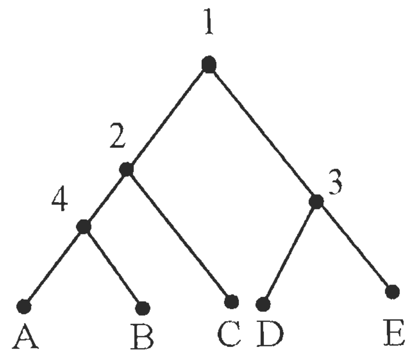


Figure 2.7: A ranked phylogenetic tree

B, LCA(A,B), has diverged after the LCA(D,E).

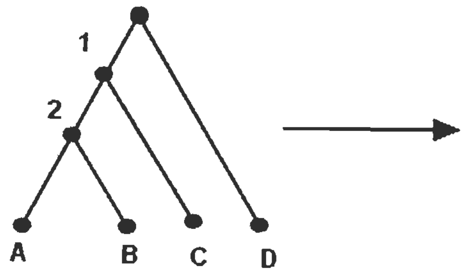
RankTree modified the BUILD in such a way that the algorithm assigns ranks to the internal nodes and at the end when the supertree is constructed the ranking is preserved. For incompatible input trees, the algorithm results in an error message.

The other variants of the BUILD are Semi-labelled and AncestralBuild [70]. The Semi-labelled supertree is constructed when some of the internal nodes of the input trees are labelled. This imposes the restriction that the children of the labelled internal node cannot be changed in the resulting tree. AncestralBuild given an alternative approach to RankTree and can be applied to the incompatible input trees.

In recoding based techniques, the trees are converted to distance data and then any of the tree construction method can be used for the construction of the supertrees. Some of the prominent methods are discussed here.

Matrix Representation with Parsimony (MRP) and its variants

This is the most widely used phylogenetic supertree method defined by Baum [64] and Ragan [65] independently. MRP represented a universally applicable method that could combine even an incompatible set of input trees. In this method binary coding of the components of each input tree is used to generate a pseudocharacter matrix representation of the trees [71]. An example is given in Figure 2.8.



Taxa	Nodes 1	Nodes 2
A	0	0
B	1	0
C	1	1
D	1	1
E	?	?

Figure 2.8: A phylogenetic tree and its binary character matrix.

The character matrices of all the trees are combined. The leaves that are not present in a given tree are marked as '?', i.e, missing. The final matrix is then analyzed with reversible parsimony procedure [72] to produce one or more parsimonious trees.

Irreversible MRP [73] differs only in its use of base method, which is irreversible parsimony [66]. Purvis MRP [74] uses a reversible parsimony but the matrix elements represents sister group relations rather than binary characters of the components. This method uses the parsimony tree construction method, which is computationally intractable problem, making MRP intractable.

The most similar supertree (MSS) method [75] also uses distance matrix, but the optimization criterion here is the minimization of the weighted sum of the absolute differences in the path lengths between the supertree and each of the input trees.

MinFlip supertree methods

Supertree methods developed by Chen et al. [76] and Eulenstein [77] are motivated by the concept of error correction. In MRP taxa present in the same cluster are scored as 1, those absent are scored as 0, and those which are not sampled are marked '?' as shown in Figure 2.8. One notion of error in such cluster system is the presence of an incorrect label in a cluster or the absence of one that should be present. This type of errors are called flips $0 \rightarrow 1$ or $1 \rightarrow 0$.

This formulation of the problem can be mapped to an optimization problem of finding the minimum number of flips that converts the matrix into a matrix which is consistent with a phylogenetic tree. This is an NP-hard problem and [76, 77] gave approximate algorithms for it.

Another approach to the matrix representation is to employ the distances between the taxa of each tree and finally combine them into a single average matrix, called as average consensus method [76]. Any distance based tree construction method can then be used to construct the supertree. This is the only method which considers the branch lengths for supertree construction, while all the other methods discussed above just use the tree topology.

The disadvantage of recoding based approaches is that they lead to hard optimization problems and are intractable for even a small number of taxa. On the other hand, the extensions of BUILD suffer with the problem of all or nothing, given incompatible input tree collection the algorithms fail to return a supertree. In chapter 4 we present an efficient method for supertree construction that addresses the problems of both recoding based methods and the methods based on BUILD. In chapter 5 we discuss the issues involved in the supertree construction with additional available evolutionary information. In chapter 6, we proposed a conflict graph based method for constructing supertrees with internal labelled node overcoming the problems of existing methods.

Chapter 3

Phylogenetic Network with Constrained Recombination

Pattern recognition has emerged as a major tool for bioinformatics applications such as DNA sequence analysis and DNA Microarray analysis [78]. It has also been applied to different graph theory applications [79]. In general, given a new DNA sequence, it is compared with the sequences that has already been studied and analyzed. The sequences that are similar would probably have similar functional and structural properties in case of genes and proteins. Relationship between the homologous sequences has an important implication in phylogenetics. Sequence alignment is one of the methods of sequence comparison, similar to the string matching, which is studied extensively in pattern recognition. Sequence alignment is a procedure for comparing two or more sequences by searching a series of individual characters or character patterns in the sequences. The distance based phylogenetic methods make use of the dissimilarity between the sequences to find the evolutionary relationships between the species. We use both similarity and dissimilarity for the classification of the nodes into mutation and recombination nodes. A similar approach for the clustering of symbolic objects is given in [80]. This chapter proposes a pattern recognition based $O(n^2)$ time approach for constructing the phylogenetic network, where n is the number of nodes or sequences in the input data. The network is constructed with the restriction that no two cycles in the network share a common node.

3.1 Introduction

The recombination event originates from modelling mutation in DNA sequences [81]. For example, consider that each species is assigned a binary sequence. When recombination occurs, the child gets some parts of genetic sequence from one ancestor and rest of the sequence from another ancestor, as shown in Figure 3.1. Thus, the recombination event cannot be modelled with a tree structure. Instead it should be represented as network where some of the nodes may have two parents. The nodes with two parents are called as the recombination nodes.

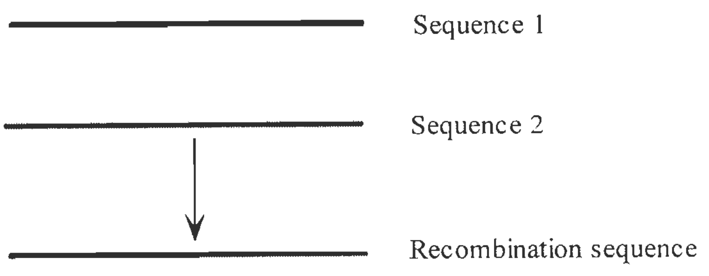


Figure 3.1: The recombination event.

Presence of recombination allows different parts of a single sequence to display different evolutionary histories. This violates the traditional assumption of single evolutionary history underlying the sequences. A study on HIV-1 [82] has shown that the most frequent recombination event has made it difficult to design a drug for HIV. Recombination in HIV is recognized as an important mechanism by which the virus escape the attack against the drug [82]. Since long time, the consequences of recombination are ignored, and phylogenies were constructed by neglecting the recombination events. Schierup and Hein [83, 84] and Posada [85] have shown the effect of negligence of recombination while constructing the phylogeny. The effects shown by Schierup and Hein include the long terminal branches in star like trees and that the rate of heterogeneity among the sites is wrongly inferred. Despite above facts, very little has been published on robust methods for recombination.

3.2 Preliminaries

In this section, we introduce the basic terminology, in addition to those given in chapter 2, and assumptions made for the development of the algorithm. We follow the terminology given in [7, 86] for simplicity.

If a node u is reachable from a node v via a directed path, then v is an ancestor of u, and u is a descendent of v. Each node in the phylogenetic network is represented with a binary number of some specified length m, each element in the array of m binary numbers is called a site or characteristic. The node with all zeros in its sequences is called the root of the network N. In the perfect phylogeny, the transformation of a site from 0 to 1 occurs at most once for each site. The nodes in perfect phylogenetic networks are organized in such a way that there is a unique node having state 1 in site i whereas, every other node having state 1 at site i is descendent of this unique node. The transformation from 0 to 1 is possible in case of recombination where the crossovers can change the state from 0 to 1. A phylogenetic network with recombination is said to be perfect if it has minimum number of recombination nodes and follows all the restrictions mentioned above.

A set of binary sequences represents a phylogenetic network N, if and only if each sequence labels exactly one leaf of the network N. A phylogenetic network on a set of three binary sequences is shown in Figure 3.2. The biological interpretation of a phylogenetic network N for n sequences represents the possible history of the sequences under the following assumptions: (1) there is a single known ancestral sequence, (2) the change in one site, from 0 to 1, is permitted only once (called mutation), (3) two sequences are permitted to recombine as a result of recombination event, (4) each site in the sequence represents a SNP (single nucleotide polymorphism), a site where two of the four possible nucleotides appear in the population with the frequency above some threshold [87].

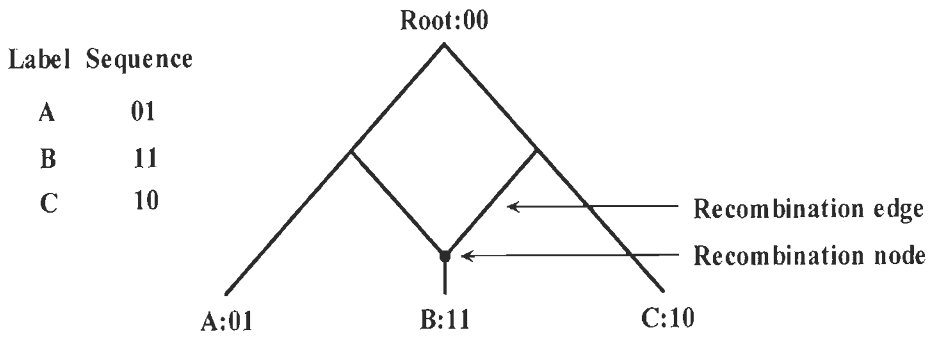


Figure 3.2: A Phylogenetic network for three binary sequences.

Given a set of species and their binary sequences, a perfect phylogenetic network, always exists with $O(mn)$ recombination nodes, where n is number of species and m is the length of binary sequences. However, recombination is a rare event in the evolutionary process, therefore a phylogenetic network with minimum number of recombination nodes is biologically significant and informative.

A node x in a phylogenetic network N is said to be coalescent if it has two paths out of it that meet at recombination node r. Those two paths together with the coalescent node and recombination node represent a recombination cycle.

A recombination cycle in a phylogenetic network that does not share any node with any other recombination cycles is called a gall. But the path from root to any of the gall or a node, which is not on the gall, can pass through the gall. A phylogenetic network is said to be gall tree if every recombination cycle is a gall in the network.

A simplified constrained recombination network with recombination, coalescent and mutation nodes is shown in Figure 3.3.

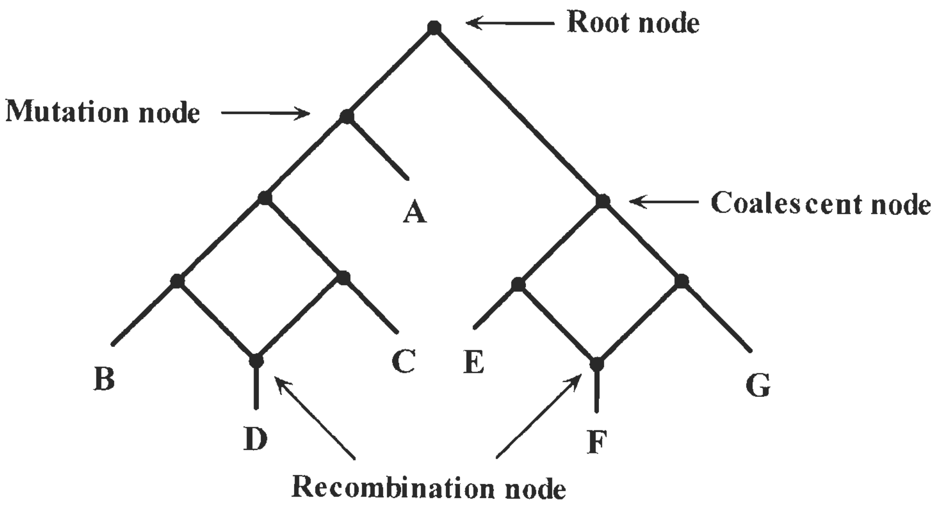


Figure 3.3: A gall tree with recombination, coalescent and mutation nodes.

3.3 Conditions for the detection of recombination nodes

In this section, we formulate the necessary and sufficient conditions for the detection of the recombination nodes. We use the similarity and dissimilarly between the sequences as major tool for the detection of recombination nodes. The similarity and dissimilarity of the sequences is computed with respect to 1. For example, let S_1 : 00010 and S_2 : 00110 are two binary sequences, the similarity is 1, due to 1 in both sequences at site 2 from right, and dissimilarity is 1 as the site 3 in S_2 is 1 and in S_1 is 0.

Lemma 3.1 is crucial for the detection of the gall in the given binary sequences. It says that the similarity and dissimilarity between the sequences, which share a common parent should be computed after removing the parent's characteristics from each child. This avoids the misleading similarity between the species.

Lemma 3.1. *Let S and S' be the sequences of the children of node v. If S' is not the result of the mutation or recombination in S then the similarity between S and S' is due to common ancestry.*

Proof. Let S' not be a child of S, then S' is not reachable from S, therefore, all the sites or characters of S' are different from the characters of the node S. Let S and S' be children of node v, then both S and S' are reachable from node v, and show similarity by at least one character (or site) with the parent node v. Therefore both the nodes S and S' show the similarity with each other at the parent characters or sites. Hence, this proves that the distinct nodes will show similarity due to common ancestry. □

A set of binary sequences can be assigned to each node in the network based on the following properties. For a non-recombination or the mutation node, the sequence is same as its parent, except at a single site i, where the mutation has occurred and the value has changed from '0' to '1'. This gives the basis for the assumption that if we consider each sequence as a binary number then the parent will always have less value than its children. As recombination is the process of exchanging the genetic material between the species, the sequence of recombination node will either be only two substrings of both the parents (for single crossover) or many smaller substrings of both the parents (called multiple crossovers).

The similarity and dissimilarity measures give important structural details about the gall trees. We show that the nodes can be classified into mutation and recombination nodes using the distance measure. We construct the gall tree and prove that this algorithm displays minimum number of recombinations needed by the sequence matrix, which has all 0's in the ancestral sequence.

Lemma 3.2. *If a node v' is the result of mutation from its parent v then $v < v'$, when the sequences are considered as the binary numbers.*

Proof of lemma 3.2 is quite obvious as only one site is changed from 0 to 1 during mutation and the back mutation is not permitted.

Lemma 3.3 plays an important role in the detection of the recombination nodes. It states that if a node is the result of recombination then it should be greater than at least one of the parents. This can be used to classify the input sequences in mutation and recombination classes.

Lemma 3.3. *Let v be a recombination node with sequence S. If P' and P'' are two parent nodes of v, with sequences S' and S'' respectively, then any of the following will hold.*

(a) $S' < S$ and $S'' < S$

(b) $S' > S$ and $S'' < S$

(c) $S' < S$ and $S'' > S$

Proof. Let us consider the proof for the binary sequences of length two. Let three binary sequences, which give a recombination node v are 01, 10, 11. These sequences can be placed in only three different ways to represent the recombination as shown in Figure 3.4 . It is proved by Gusfield et al. [7] that these are the only possible ways of representing the above sequences. In all the cases, the sequence with all 0's is considered as root. The case of child having less value than both of its parent is not possible because back mutation is not permitted and the assumption that the root of the gall tree will have all 0's in it. Let us now consider the cases (a), (b), and (c) individually below:

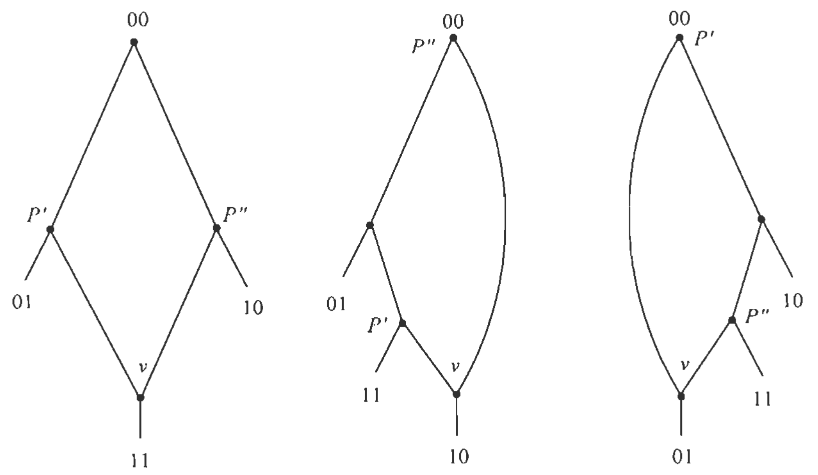


Figure 3.4: Three cases for Lemma 3.3.

(a) Here, the two mutations from root node lead to the species 01 and 10. The recombination node v, with sequence 11 is the result of recombination of 01 and 10. Clearly, v is greater than its two parents. This is shown in leftmost network of the Figure 3.4.

(b) In this case, the sequence 01 mutated from root and the sequence 11 is mutated from 01. The recombination node v is the result of recombination between root and P'. It satisfies the case (b) stating, $S' > S$ and $S'' < S$. This is shown in middle network of the Figure 3.4.

(c) In this case, the sequence 10 is mutated from root and the sequence 11 is mutated from 10. The recombination node v is the result of recombination between root and P''. It satisfies the case (c) stating, $S' < S$ and $S'' > S$. This is shown in rightmost network of the Figure 3.4.

It can be easily seen that the proof follows for binary sequences of length greater than two. □

Converting the binary sequences into distance matrix based on similarity or dissimilarity leads to the information loss. For example, consider the sequences S_1: 00010, S_2: 00100, S_3:10010 and S_4:10100. The distance or dissimilarity between S_1 and S_2 is 2, and the distance between S_3 and S_4 is also 2. According to the definitions of child parent relationship mentioned in section 2 and 3, the sequences S_1 and S_2 do not share any relationship with each other. On the other hand, the sequences S_3 and S_4 share a child parent relationship even though they have the same dissimilarity, i.e., 2 as for S_1 and S_2. This clearly indicates loss of information due to transition. Its effects when applied to different methods, which construct the network based on distance data, are shown in section 3.6. To avoid this loss, we consider both similarity and dissimilarity for the construction of gall trees.

The theorem 3.1 uses the lemma 3.2 and 3.3 for the detection of the recombination nodes. It helps in finding the parents of the recombination node, which is particularly useful when any one of the parent is greater than the child. It states that the similarity between the parents of the recombination node is 0. Given three nodes indicating a recombination event, using lemma 3.2 and 3.3

we can find one parent easily and the other parent can be can be detected with aforementioned fact.

Theorem 3.1. *Let M be the given sequence matrix representing the gall tree. A species or sequence is said to be a result of recombination if any one of the following conditions holds good:*

1. *If two species have $0 < similarity <= 100\%$ and $dissimilarity > (100/m)\%$, where m is the length of the sequence, one sequence represents parent and other sequence represents the child, which is the result of recombination.*
2. *Parents of the recombination node have $similarity = 0$.*

Proof. 1. Suppose that at some node x, mutation occurred at site i, which represents the change at only site i from 0 to 1 in the mutated node y. If we calculate the similarity and dissimilarity between the nodes x and y corresponding to the value 1 at each site, the similarity between them will be 100% and dissimilarity will be $(100/m)\%$. This indicates that only one site has modified its value from 0 to 1. By the assumptions we made for phylogenetic network, there is no provision for back mutation, that is transformation from 1 to 0 and in addition, the mutation of more than one site at the same instance of time is ruled out. This restricts the dissimilarity as $(100/m)\%$ for mutation. But in case of recombination the two aforementioned restrictions of the mutation are ruled out due to the fact that the resulting sequence will carry a part of the sequence from one of its parents and rest will be copied from the other parent (in single crossover). This fact indicates that the similarity can be $0 < similarity <= 100\%$ and $dissimilarity > (100/m)\%$. Hence the condition (1) holds true.

2. Let us prove this by contradiction. Suppose the recombination node v has two parents with the sequences S' and S'' showing some similarity. From the assumptions made in section 3.2 and lemma 3.1, the similarity between the species in node disjoint recombination networks is due to following three reasons: (1) common parent, (2) child parent relationship with a single mutation, and (3) recombinations. If S' and S'' show some similarity then one of the above relations holds. The relation 1 is eliminated by removing the parent characteristics while computing the similarity between the children. Since we focus on a constrained recombination problem, where two recombination nodes are disjoint, relation 3 is avoided. Let S'' is an immediate child of S', due to mutation in S'. If the nodes S' and S'' are the parents for recombination node N, then the sequence of recombination node will be same as any of its parents, $N \in (S', S'')$, instead of a new sequence as shown in Figure 3.5.

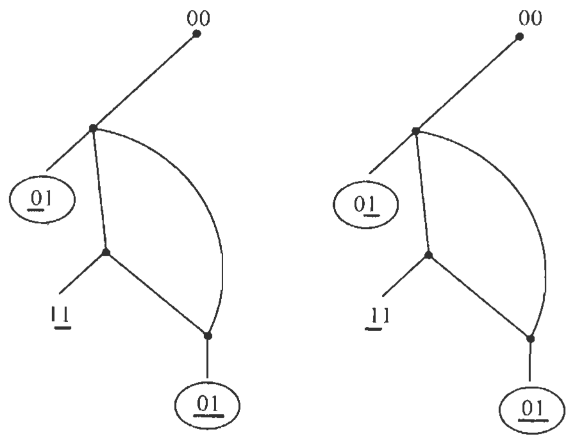


Figure 3.5: Result of recombination between parent and child.

Hence, the parents of the recombination node will be dissimilar to each other.

□

Theorem 3.2 gives a strong basis for the detecting the galls in the given input data. Any data satisfying the conditions given in theorem 3.2 will have gall tree. Otherwise, the data will not represent the gall tree structure.

Theorem 3.2. *If C, C', and C'' are child list of nodes S, S', and S'' then the following conditions should hold for the gall trees:*

1. *Recombination node will be $C \cap C' \neq \emptyset$ and $|C \cap C'| = 1$.*

2. *If the number of recombination nodes in any of the parents is greater than 1, and $C \cap C' \neq \emptyset$ then ($C \cap C'' = \emptyset$ or $S'' \cup C'' \subseteq C$) and ($C' \cap C'' = \emptyset$ or $S'' \cup C'' \subseteq C'$) .*

Proof. 1. This is proved by contradiction. Let $|C \cap C'| > 1$ represent that the nodes S and S' are involved in more than one recombination with each other. Each recombination node represents a cycle in the gall tree. The path from root node to the recombination node always has two alternatives, one from each of its parents. If there are more than one recombination node for the single pair of parents S and S', then there are two paths for each recombination node which involves the same set of parents S and S'. In other words, the parent nodes are shared by two recombination cycles. But according to the definition of gall tree, the nodes in one cycle should not be shared with other recombination cycle. This contradicts the assumption made and hence proves the condition.

2. The proof is similar as in case (1). Let $C \cap C' \neq \emptyset$ and the number of recombination nodes in C be two. If $C \cap C' = X$ and $C \cap C'' = Y$, then there exists a path from root node to the recombination cycle of node X and node Y, which passes through the node S. This violates the node disjoint

rule of gall trees. Let child list C'' and the node S'' itself be a subset of the child list of node S. If the similarity and dissimilarity between the children of S are computed after removing the S's characteristics from each node then the recombination node will not show S in its parent list, according to lemma 3.1. This parent relationship with the recombination node is due to the common ancestor of all the nodes in the recombination cycle. Thus, when there is common ancestor for the parents of a recombination node then the parent of all the nodes in that cycle is also added as the parent to the recombination node.

$\square$

If child list is associated with every node that indicates its potential children, then any two nodes, having child list C and C', share more than one child, i.e., $|C \cap C'| > 1$, and it indicates that the parents are involved in more than one recombination resulting into a non gall tree structure. The condition (1) in theorem 3.2 restricts the two parents from having more than one recombination node.

Similarly, the condition (2) states that only the node, which is parent of the two or more galls in the networks can have more than one recombination nodes in its child list.

3.4 The Node Class Algorithm

In this section, we propose an algorithm for the phylogenetic network reconstruction with constrained recombination. We prove that this algorithm results in minimum number of galls in the resulting network.

The root sequence contains all zeros in it and only a state change from 0 to

1 gives information about mutation or recombinations. Thus the similarity and dissimilarity with respect to 1 are considered. The similarity and dissimilarity with respect to 1's in the sequence can be calculated using AND and XOR binary number operations respectively. For example, consider the sequences S_1 and S_2, having zero similarity and 2 dissimilarities as shown below.

0 0 0 1 0	: S_1	
0 0 1 0 0	: S_2	
0 0 0 0 0	AND	(Similarity)
0 0 1 1 0	XOR	(Dissimilarity)

The number of 1's in the result represents similarities or dissimilarities with respect to the operation performed. Similarly, the parent's characteristics can be eliminated from children using the XOR operation. This approach significantly reduces the number of comparisons performed to compute the similarity and dissimilarity, hence reduces the complexity of the algorithm.

The algorithm makes the child of each node given in the data matrix based on similarity and dissimilarity. We assume that all the sequences represent a unique leaf node in the network. The arrangement of nodes starts with the root node, assumed to have all 0s in it. Each mutation will lead to change at only one site and recombination may lead to more than one change.

The algorithm accepts a $n \times m$ binary matrix as input, where each row represents a node in the phylogenetic network. Similarity and dissimilarity matrices are generated based on the input matrix and are computed corresponding to the value 1 at the sites. The distances (similarity and dissimilarity) between the siblings are measured after removing the parent's characteristics from the children. The parent node is considered as the root to all the nodes in the child list except the recombination node. If the data does not represent gall tree, then the algorithm terminates by reporting an error message. We use a special data structure *Node* with three variables *Lable*, *Type* and *Count*. *Label* is used to represent node label, *Type* for representing the type of the node, and *count* is used to represent number of parents of the node. There are three types of node *Recombination*, *Mutation* and *Null*. *Recombination* is assigned to the node which is the result of recombination, *Mutation* is assigned to the node which results from mutation, and *Null* is used to represent the node, which is the child of the root node or is not a *Mutation* or *Recombination* node. The algorithm is as follows:

DATA STRUCTURE:

Input : Binary matrix of $n \times m$.

Output : Parent and Child list for each node.

$D \longleftarrow$ An input matrix of size $n \times m$,where n is number of species and m is length of sequences. Each row is considered as a Node

$Sim_{ij} \longleftarrow$ The similarity with respect to 1's on comparing rows i and j.

$Dis_{ij} \longleftarrow$ The dissimilarity with respect to 1's on comparing rows i and j.

$Node \longleftarrow$ A record with three variables: Label, Count, and Type. Initially the Count and Type variables of a *Node* are set to zero and Null respectively. The variable Label is set to the labels of rows in the input matrix.

$Child_i \longleftarrow$ An array of child labels for $Node_i$ initially set to *Null*.

$Parent_i \longleftarrow$is an array of parent labels for $Node_i$ initially set to *Null*.

```
ALGORITHM: $Node\_Class(D)$
begin
```

```
1. for each Row (Node) $i$ in $D$ do
2.   for each Row (Node) $j$ in $D$ do
3.     if $0 < sim_{ij} \leq 100\%$ and $Dis_{ij} \geq 100/m\%$ then
4.       if $Node_i < Node_j$ then
           $prnt \leftarrow i$
           $chld \leftarrow j$
         else
           $prnt \leftarrow j$
           $chld \leftarrow i$
         end if(4)
         $Node_{chld}.Count \leftarrow Node_{chld}.Count + 1$
         $Child_{prnt} \leftarrow Child_{prnt} \cup Node_{chld}.Label$
5.       if $0 < Node_{chld}.Count \geq 2$
           $Node_{chld}.Type \leftarrow Recombination$
         else
           $Node_{chld}.Type \leftarrow Mutation$
         end if(5)
       end if(3)
     end for(2)
   end for(1)
 end Algorithm
```

The next function *Test_Gall* takes Child list and Node record list as input and based on theorem 3.2, it verifies whether gall tree exists in the given data. If the data does not represent the gall tree then the function terminates giving an error message as an output, otherwise it returns a message confirming presence of gall tree. The function is as follows:

```
FUNCTION: Test_Gall ( $Child, Node$ )
Begin
1. for each Row (Node) $i$ in $D$ do
2.   for each Row (Node) $j$ in $D$ do
3.     if $|Child_i \cap Child_j| > 1$ then
         return Gall tree does not exist
       end if(3)
     end for(2)
   end for(1)
4. for each Row $l$ with more than one recombination node as child do
5.   for each child $c$ of $child_l$ do
6.     if $Node_c.Label \cup Child_c \subseteq Child_l$
         Add a new node in the Child list same as $Node_c$
         Add the Children of $Node_c$ to the new node
         Remove the children of $Node_c$ from Child list of $Node_l$
         Add the new $Node$ as the child of $Node_c$
       else
         return Gall tree does not exist
       end if(6)
     end for(5)
   end for(4)
end Function
```

The following theorem proves that the algorithm results in a gall tree, if one exists, with the minimum number of galls in it.

Theorem 3.3. *For the input matrix M, if there are k recombination nodes then any gall tree that minimizes the recombination will have exactly k galls or node* *disjoint cycles.*

Proof. Let T be a gall tree for the input binary matrix M. If there is a gall Q in T that contains only the mutation nodes, then the sequence labeling of the nodes on Q can be derived from the perfect phylogeny. The root of the gall Q is the sequence labeling the coalescent node of Q. Replacing Q with perfect phylogeny will result in a gall tree with one recombination less than the gall tree T. Hence, any gall tree using the minimum number of recombinations must have exactly one recombination node for each gall. Therefore, the minimum number of galls in a gall tree is exactly the number of recombination nodes. □

3.5 Correctness and time complexity analysis of the algorithm

All the results and facts in section 3.4 are based on the existence of the gall tree for the input matrix M. The theorems in section 3.3 and 3.4 show correctness of the algorithm. When the input data does not display a gall tree structure, the algorithm reports an error message and terminates. The gall tree computed by the algorithm have minimum number of recombinations.

The algorithm proposed in this chapter computes a gall tree, if one exits, in $O(n^2)$ time, where n is the number of nodes in the input data. The complexity is reduced with the help of small bookkeeping, which maintains a child list and a record for each node. There are three phases for finding a gall tree.

In the first phase, the similarity and dissimilarity matrices are computed based on the binary AND and XOR operations. The comparison is considered as a major operation contributing to the complexity of the computation and counting the number of ones in the result is taken as an elementary operation, this phase takes at most $O(n^2)$ time to compute the similarity and dissimilarity.

In the second phase, the nodes are classified into three types *Null*, *Mutation*, and *Recombination* using *Node_Class* algorithm. The first two steps for loops, i.e., step 1 and 2, are major steps contributing towards the complexity of the algorithm and thus the steps take $O(n^2)$ time.

The final phase is used to find a gall tree based on child list and node record using *Test_Gall* method. The first two for loops, i.e., steps 1 and 2, run for n times and takes n^2 time. The step 4 runs for each node with more than one recombination node, which is far less than the total number of nodes, i.e., n, and the step 5 runs for each child and its child list which is also less than the total number of nodes. The first two steps dominate the complexity of the *Test_Gall* method and lead to $O(n^2)$ time for the algorithm.

The total time required to compute the gall tree, if one exist, is $O(n^2+n^2+n^2)$ which is $O(n^2)$. This gives the better computing time than the best known algorithm [7] with time complexity of $O(n^3)$.

The first two 'for' loops, i.e., step 1 and step 2,of the *Test_Gall* method run for n times to check the existence of the gall tree. Therefore, it takes $O(n^2)$ time to find that there does not exist a gall tree for the given binary matrix.

3.6 Results and discussion

In this section, we illustrate the working of the proposed method with an example and compare the results with the existing methods, such as T-Rex [44], NeighborNet [4], and SplitsTree [6]. The comparison is made with these methods based on the resulting network and the time complexity.

3.6.1 An illustrative Example

The input matrix for the algorithm is shown in Figure 3.6, which consists of seven leaf nodes and a binary number to represent each of the nodes. The initial value for the variables of each Node record is set to Null. The similarity and dissimilarity matrix after removing the parent's characteristic are shown in Figure 3.7.

Label	Sequence
A	0 0 0 1 0
B	1 0 0 1 0
C	0 0 1 0 0
D	1 0 1 0 0
E	0 1 1 0 0
F	0 1 1 0 1
G	0 0 1 0 1

Figure 3.6: Input binary matrix with labels.

After processing each node the values assigned to each variable or properties of the nodes are shown in Table 3.1. The values for nodes A and C are 'Null' because they are mutated from the root node, and not from any other nodes. The nodes D and F are the result of the recombination and have two parents. The rest of the nodes are the result of mutation from their respective parents.

Table 3.2 shows the child list of each node. The nodes *D* and *F* do not have any child, thus their child lists carry *Null* values. On the other hand, the nodes *D* and *F* are in the child list of *B*, *C* and *E*, *G* nodes respectively; thus making them recombination nodes. The function *Test_Gall* results in Table 3.3 when applied to Table 3.3 and *Node* list. The child list is computed based on the

$$\begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ A & 1,0 & 1,1 & 0,2 & 0,3 & 0,3 & 0,4 & 0,3 \\ B & 1,1 & 2,0 & 0,3 & 1,2 & 0,4 & 0,5 & 0,4 \\ C & 0,2 & 0,3 & 1,0 & 1,1 & 1,1 & 1,2 & 1,1 \\ D & 0,3 & 1,2 & 1,1 & 2,0 & 0,2 & 0,3 & 0,2 \\ E & 0,3 & 0,4 & 1,1 & 0,2 & 2,0 & 1,1 & 0,2 \\ F & 0,4 & 0,5 & 1,2 & 0,3 & 1,1 & 3,0 & 1,1 \\ G & 0,3 & 0,4 & 1,1 & 0,2 & 0,2 & 1,1 & 2,0 \end{array}$$

Figure 3.7: Similarity and dissimilarity matrix for the input data shown in Figure 3.6. The first element represents similarity and second one represents dissimilarity.

Table 3.1: Values of each property of node record after processing the input matrix shown in Figure 3.6.

Node Label	Type	Count
A	Null	0
B	Mutation	1
C	Null	0
D	Recombination	2
E	Mutation	1
F	Recombination	3
G	Mutation	1

gall tree conditions proved in theorem 3.2. The child list for the node C has two recombination nodes D and F, and the child F has count value 3 indicating three parents. But, it satisfies second condition in theorem 3.2. The steps 4,5, and 6 of function *Test_Gall* are executed, which results in a new node C'. The node C' is added to child list of C. The child nodes E,F, and G are removed from the child list of node C and added as the children of node C'. The modified child list is shown in Table 3.3.

Table 3.2: Child list for the input matrix shown in Figure 3.6.

Node Label	Child List
A	B
B	D
C	D,E,F,G
D	NULL
E	F
F	Null
G	F

Given the child and node record list for each of the node in the input data, it is easy to construct the gall tree for it. The procedure starts by including a new node called Root, with all zeros in it's sequence. All the nodes with a *Null* entry in their *NodeType* are attached to the root node. The rest of the nodes are then added and connected accordingly to the child list entries. The child list is preprocessed and analyzed for coalescent node, therefore, the phylogenetic network can be constructed directly with the help of child list and node record itself. It is obvious that the network can be constructed with a single scan of child list and node record. Both the lists contain n entries and can be computed in linear time, i.e., $O(n)$.

Table 3.3: Modified Child list for the input matrix shown in Figure 3.6.

Node Label	Child List
A	B
B	D
C	D,C'
C'	E,F,G
D	NULL
E	F
F	Null
G	F

The network construction for the above example starts with the first entry in the node record. Here, the first node is A which has a *Null* entry in its *Node.Type* as shown in Table 3.1, so it is attached to the Root node of the gall tree. Node A's child list shows B as child, therefore, the node B is attached to node A as child. The nodes B and C have the node D as child, therefore, node D is attached to both of them. The whole network is constructed in similar fashion. The final gall tree for the input shown in Figure 3.6 is shown in Figure 3.8.

3.6.2 Comparison with other methods

The distance matrix used for the input to the different methods is shown in Figure 3.9. The distance between the nodes represent the number of mismatches in the sequences. For example, consider the sequences S_1:00010 and S_2:10010 the distance is 1 as only one site has a mismatch and all other sites have the matching values. When the input matrix is applied to different network construction

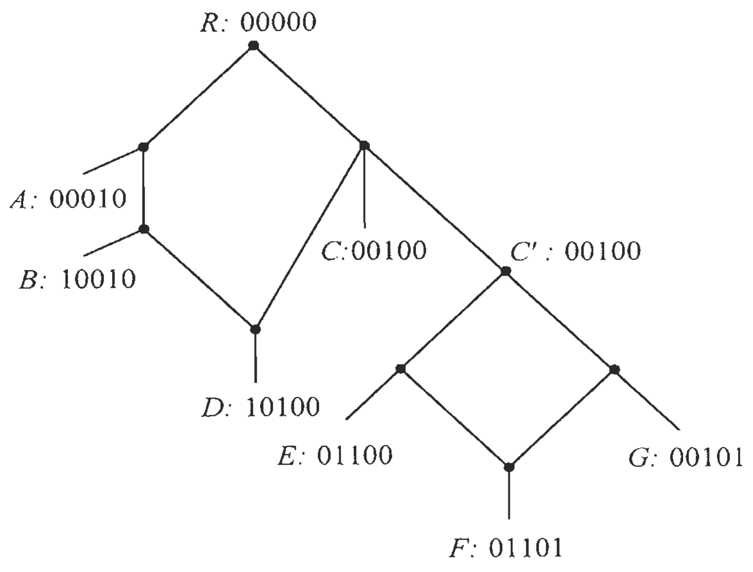


Figure 3.8: Final gall tree for the input data shown in Figure 3.6.

methods the resulting networks are shown in Figure 3.10 and Figure 3.11.

	A	B	C	D	E	F	G
A	0	1	2	3	3	4	3
B	1	0	3	2	4	5	4
C	2	3	0	1	1	2	1
D	3	2	1	0	2	3	2
E	3	4	1	2	0	1	2
F	4	5	2	3	1	0	1
G	3	4	1	2	2	1	0

Figure 3.9: The distance matrix used as input to T-Rex, NeighborNet, and SplitsTree.

Figure 3.10(a) shows the tree constructed using Neighbor Joining method and is used as the underlying tree structure by T-Rex for constructing phylogenetic network, shown in Figure 3.10(b). There is a reticulation edge between the nodes B, D and G, F. The nodes E and F have evolved from a common ancestor and F shares a reticulation edge with G. This indicates that the node F is the result of reticulate evolution. However, in the actual data F is the result of recombination between E and G. It can be observed that the other reticulation edge is missing between E and F. Similarly, a reticulation edge is missing between C and D as the node D is the result of recombination between B and C.

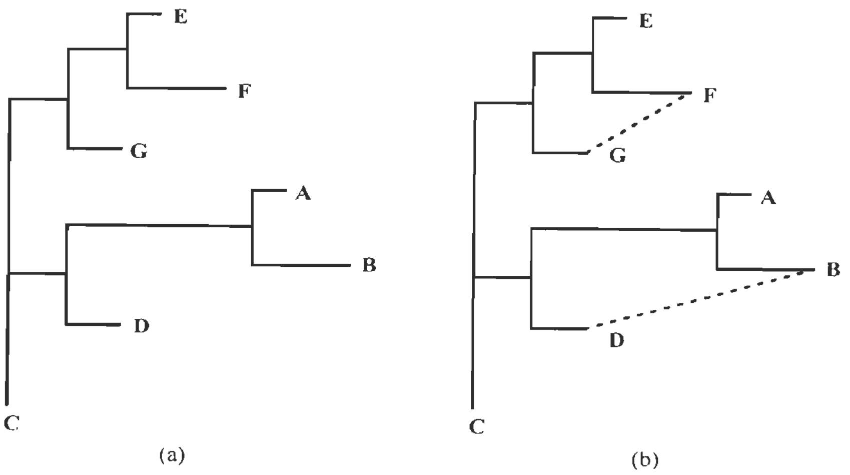


Figure 3.10: Underlying tree for T-Rex, constructed using neighbor joining method and the network constructed using T-Rex.

Figure 3.11 shows the network constructed using NeighborNet and SplitsTree, both of which give the same result. NeighborNet and SplitsTree give closer visualization to actual representation than T-Rex does. The evolution of node F is properly indicated as the recombination node of E and G. The evolution of D is also interpreted properly. However, NeighborNet and SplitsTree added an additional edge between the node A and C. In actual representation there is no relationship between A and C as they have evolved independently from root node. Thus, the output is not fully correct for these algorithms.

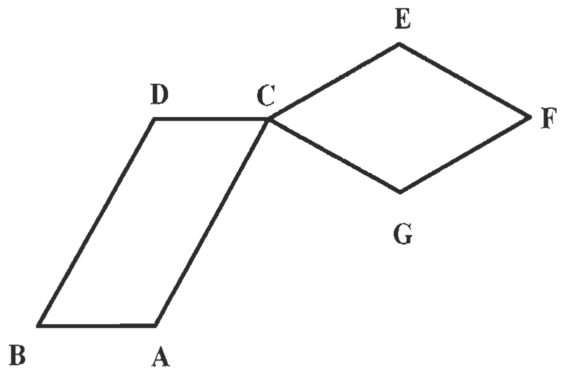


Figure 3.11: Phylogenetic network constructed using NeighborNet and SplitsTree.

The variation in the networks is due to the information loss while converting the sequences into distances. To avoid this our algorithm utilizes both similarity and dissimilarity measures for the construction of the network. This gives a nearest visualization of the data to the actual representation. The resulting network of the proposed method is shown in Figure 3.8. Our algorithm gives the correct output as verified by the actual data representation.

The time required to compute the phylogenetic network by SplitTree is $O(n^5)$, T-Rex is $O(n^4)$ and NeighborNet is $O(n^3)$, where n is the number of node. On the other hand the algorithm proposed in this chapter computes the phylogentic network in $O(n^2)$ time.

3.7 Conclusion

In this chapter, we proposed a pattern matching based approach for the construction of phylogenetic network with constrained recombination. The proposed algorithm, computes the gall tree in $O(n^2)$ time, where n is represents the number of nodes. We also formulated the necessary and sufficient condition for constructing the gall trees. There are two major changes in the network construction approach that reduces the complexity and improves the visibility of the proposed method. They are:

1. A row-based search to detect the recombination nodes
2. Use of both similarity and dissimilarity.

Other algorithms search the columns for the detection of recombination and the number of columns in a sequence may be far greater than the row, which increases the complexity of the previous algorithms. The use of both similarity and dissimilarity avoids the information loss due to converting sequences into distances. The comparison with the result of other algorithms like T-Rex, NeighborNet, and SplitsTree shows that our algorithm is accurate and efficient.

Chapter 4

Supertree algorithms

Phylogenetic tree of species can be constructed using distance and character based methods, as discussed earlier. In spite of the great progress in the phylogenetic tree constructing methods, it appears difficult for a single research team to construct the tree of life due to following reasons:

- Most of the individual researchers and research teams are concentrating on the evolutionary pathways of specific phylogenetic groups.

- Many efficient phylogenetic reconstruction methods, such as Maximum Parsimony [72] and Maximum Likelihood [88], are hard optimization problems and are limited to small number of taxa. On the other hand, distance-based [9, 89, 46] phylogenetic reconstruction methods are computationally efficient but while converting sequence data to distance data, loss of information occurs.

In this chapter, we propose a phylogenetic supertree method that exploits the features of both the methods in a way that smaller trees can be constructed using efficient character based methods and then these smaller trees can be merged using distance based methods.

4.1 UPGMA and its variant

In this section, we describe the standard UPGMA algorithm, which is a simple bottom-up data clustering method used for constructing phylogenetic trees. The failure of UPGMA with respect to the constraints on the tree distances is discussed with a novel solution to overcome the drawback of standard UPGMA.

4.1.1 Standard UPGMA

UPGMA is statistical based and simplest among the phylogenetic tree construction methods. It accepts the pairwise distance between the species to be classified and give a phylogenetic tree as output. It starts by considering each species as a cluster then combines two nearest clusters in each step. After combining two closely related clusters the algorithm recalculates the pairwise distances between the clusters. The process is repeated till there is only one cluster left. The method is as follows:

1. Take distance matrix as input. For example a distance matrix for three species is shown below.

Species	A	B	C
B	d_{AB}		
C	d_{AC}	d_{BC}	
D	d_{AD}	d_{BD}	d_{CD}

2. Species with the smallest distance amongst them will be clustered.

3. The new distance matrix is computed with the distance between new group and the remaining species as $d_{(AB)C} = 1/2(d_{AC} + d_{BC})$ for all the species.

4. Repeat until all the species have been grouped.

4.1.2 Failure of Standard UPGMA

This method applied to pairwise distance data returns a rooted phylogenetic tree. However, when applied to tree distance data leads to the clustering of wrong taxa. This will be clear with the following example. Let us consider the trees shown in Figure 4.1 as input trees and their average distance matrix shown in Figure 4.2. The distance between two species is computed as the average of path lengths between the species in each tree of input collection.

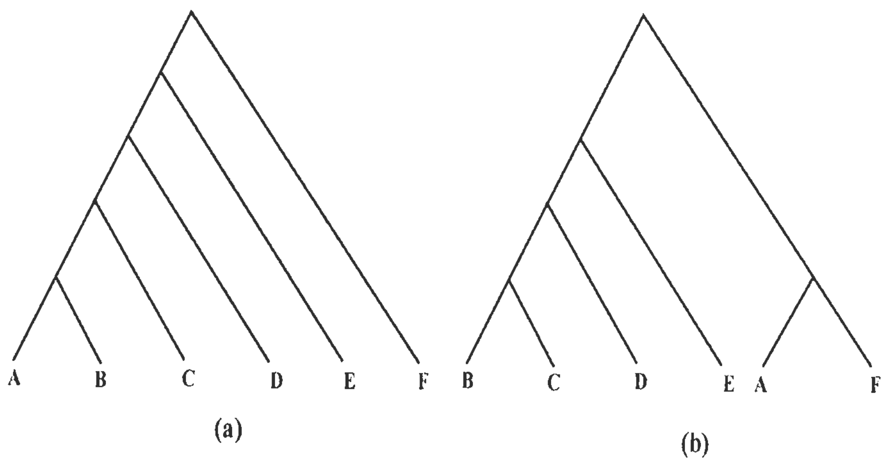


Figure 4.1: Input trees for UPGMA.

	A	B	C	D	E
B	4				
C	4.5	2.5			
D	4.5	3.5	3		
E	4.5	4.5	4	3	
F	4	6	5.5	4.5	3.5

Figure 4.2: Distance matrix for the trees shown in Figure 4.1

The distance matrix in Figure 4.2 shows that the distance between B and C is least and thus leads to the grouping of the B and C. The distance matrix after merging B and C is shown in Figure 4.3.

$$\begin{array}{ccccc}
 & A & BC & D & E \\
BC & 4.25 & & & \\
D & 4.5 & 3.25 & & \\
E & 4.5 & 4.25 & 3 & \\
F & 4 & 5.75 & 4.5 & 3.5
\end{array}$$

Figure 4.3: Modified distance matrix after grouping B and C

The species D and E at minimum distance in the matrix shown in Figure 4.3, but if the distance from the least common ancestor of the BC is considered then D is close to the group BC. The distance between the least common ancestor of BC and D is 2 as shown in Figure 4.1(a) and 4.1(b). This shows that employing the standard UPGMA algorithm leads to wrong clusters, thus leading to incorrect phylogenetic supertree.

4.2 UPGMA variant: A new distance calculation algorithm

In this section, we propose a variant of standard UPGMA that makes use of different distance measure for recalculating distance matrix to be used for clustering the species.

The algorithm is based on sets of clusters and therefore is used for rooted trees. It makes use of cluster information available in the given tree for distance calculation. Once the clusters shows least difference they are combined and considered as a single cluster. Then the distance from the newly formed cluster to all the other clusters is recalculated. The algorithm for calculating distances between two species or clusters is as follows:

```
ALGORITHM: Distance( A , B )
Input: clusters A and B
Output: Distance between the clusters A and B
begin
1. Find all clusters with A or B and make a set X
2. Find the minimum cluster M in which both A and B are present
3. Dist = 0
4.   for each cluster i in X
5.       if $X_i \cap M = X_i$ then
6.         Dist = Dist + 1
         end if(4)
     end for
7. Final distance is Dist = Dist-2 (for two nodes)
end Algorithm
```

In the first step, the algorithm finds all clusters with A or B where A and B can be a single node or a cluster. Once the clusters are available a cluster with minimum number of taxa that includes both A and B is searched. This minimum cluster is then used for finding the distance between the clusters. The working of the algorithm can be made clear with the help of following example.

For the tree shown in Figure 4.1(a), the distance between cluster BC and A can be computed as:

1. Clusters with BC or A are X = {{A}, {AB}, {ABC}, {ABCD}, {ABCDE}, {ABCDEF}}.

2. Minimum cluster with BC and A, M = {ABC}.
3. Clusters with $X_i \cap M = X_i$ are {{A}, {AB}, {ABC}, {ABC}}.
4. The final distance is number of clusters in step - 2, i.e. 4–2 = 2.

Similarly, the distance can be calculated in different trees and an average is taken for further combining of species.

There are at most $(2n - 1)$ clusters in each tree therefore the algorithm takes $O(n)$ time for distance calculation. The new modified distance matrix, shown in Figure 4.4, is used for further grouping instead of the distance matrix show in Figure 4.3.

	A	BC	D	E
BC	3.5			
D	4	2.5		
E	4	3.5	3	
F	3.5	5	3.5	3.5

Figure 4.4: Distance matrix computed using variant of standard UPGMA.

4.3 The supertree algorithm

In this section, we describe the new technique for the construction of the supertrees and establish the desirable properties of the supertree method. The advantage of this algorithm is that it will return a tree even for incompatible input tree collection. The algorithm is described below.

```
ALGORITHM: DISTSUPERTREE( T )
begin
1. Find common and distinct leaves in the given
```

```
   trees as $COMM = \cap_{i=1}^{k} L(T_i)$ and $DISTINCT_i = \{L(T_i) - COMM\}$
2. For all trees, find the restrictions of the input
   trees on the subsets $COMM$ and $DISTINCT$.
   $T_i' = T_i|\{COMM\}$ and $T_i'' = T_i|\{DISTINCT_i\}$ for all $i = 1$ to $k$
3. Construct the phylogenetic consensus tree $ST$ using UPGMA variant for
   the for the restriction trees constructed in step 2.
     /* for each tree the number of edges between the leaves
     is considered as the evolutionary distance between them.
     For the final tree the average of the distances of the
     trees is considered. */
4. if $COMM \neq \emptyset$
5.   for each $i = 1$ to $k$
6.     if $T''$ is a subtree of $T_i$ without performing contraction.
         The root of $T''$ is connected with root
         of the $ST$, and the whole tree is renamed as $ST$.
       else
7.       for each node $N_j$ in $DISTINCT_i$
           Add an edge between $N_j$ and its least common
           ancestor in $T_i$ in the tree $ST$.
         end for(7)
       end if(6)
     end for(5)
   end if(4)
end Algorithm
```

The first step of the algorithm finds the $COMM$ common and $DISTINCT$ leaf nodes in the collection of input trees. Restriction trees for the both $COMM$

and $DISTINCT$ sets are constructed in the second step of the algorithm. Step 3 takes the restriction trees of $COMM$ set as input and uses the variant of UPGMA to find the consensus tree for common leaf nodes. The distance between the species is considered as the average number of edges between them. Finally the distinct nodes are added to the consensus tree (which is the result of step 3), based on least common ancestor optimality criteria.

4.4 Properties of the DISTSUPERTREE

The proposed $DISTSUPERTREE$ algorithm shows all the properties given by Steel [54] for the supertree methods. Here we establish the desirable properties for the above algorithm.

The $DISTSUPERTREE$ method computes the supertree in polynomial time as each step in the algorithm takes polynomial time to complete the desired task. The key steps in the algorithm are 3 and 4, which compute the supertree of all the taxa. Step 3 takes $O(kl^3)$, where k is number of tree of trees and l is number of common leaf nodes, $l = |COMM|$, in the collection of the input trees. Step 4, which grafts the distinct nodes on the consensus tree resulting from step 3, takes $O(km)$, where m stands for the number of nodes in $DISTINCT$ set of each input tree $m = |DISTINCT_i|$, for $i = 1$ to k. The total complexity of the algorithm is $O(kn^3)$, where k is number of trees and n is the total number of leaf in all the input rooted trees. The following theorem formally proves the time complexity of the proposed algorithm.

Theorem 4.1. *Let $T = T_1, T_2, ..., T_k$ be a set of rooted phylogenetic trees. Then the $DISTSUPERTREE$ takes $O(kn^3)$ time to compute the supertree.*

Proof. The step three of the algorithm takes $O(kl^3)$ time for constructing the consensus tree of common nodes, where k is number of trees and l is number of common leaf nodes in the input tree collection. It can be elaborated as follow: the algorithm *distance* takes $O(l)$ time and it has to be computed l^2 times for pairwise distances between each node. Thus, the total time to compute the distances is $O(l^3)$. The whole computation has to be performed for each tree the complexity goes to $O(kl^3)$.

The step 4 is to add the distinct nodes of each tree to the result of step 3. It takes $O(km)$ time, where k is number of trees and m is distinct leaf nodes in the input tree collection.

In the worst case the value of l and m can be same as the total number of nodes in the input tree collection n. Therefore, the total complexity of the algorithm in worst case is $O(kn^3)$. □

It is evident that the algorithm satisfies the properties 3 and 4, which states that the result of the supertrees should not be affected if the order of the trees is changed or leaves are relabelled. The algorithm does not make use of the order or labelling information rather uses the structural topology for the construction of the supertree. Hence properties 3 and 4 are satisfied. The fact it returns a supertree is proved in the following theorem.

Theorem 4.2. *Let $T = T_1, T_2, ..., T_k$ be a set of rooted phylogenetic trees. Then the $DISTSUPERTREE$ applied to T returns a tree.*

Proof. To simplify the proof we consider two cases, the first case is of collection of input trees classifying the common species, and the second case is of the collection of the input trees classifying the overlapping set of species.

For the first case, it is clear that the $DISTSUPERTREE$ method returns a tree when applied to a collection of trees T, which classifies the common leaf nodes. The algorithm returns a consensus tree at step 4. UPGMA variant is used for the construction of the consensus tree which always results in a single rooted tree.

For the second case, the input collection of the tree classifies overlapping set of species. Step 4 uses the least common ancestor information carried by the input trees to ensure that all the leaf nodes are connected to the proper least common ancestor.

The leaf nodes of each input tree come under $COMMON$ set or $DISTINCT_i$ set. Both the sets are used for the construction of the supertree. The loop in step 4 ensures that all trees are processed for each node in the $DISTINCT$ set, which is added to the resulting supertree. This leads to a single rooted supertree. □

The proof of the theorem 4.3 depends on the result proved in theorem 4.2 of [90].

Lemma 4.1. *Let T and T' are two phylogenetic rooted trees then $T \leq T'$ if and only if $r(T) \subseteq r(T')$ and $L(T) \subseteq L(T')$.*

Theorem 4.3. *Let $T = T_1, T_2, ..., T_k$ be a set of compatible rooted phylogenetic trees. Then the $DISTSUPERTREE$ applied to T returns a tree, which displays all the input trees.*

Proof. If $T = T_1, T_2, ..., T_k$ is the set of compatible rooted collection of input trees. Let $r = \cup_{i=1}^{k} r(T_i)$ is set all the rooted triplet of the input trees, where $r(T_i)$ is the set of triplets of the tree T_i . Then by comparing the algorithm Mincut [53] with the $DISTSUPERTREE$, when the input collection of trees is compatible, the resulting tree of both the algorithms is identical (shown in section 4.5) when applied to $r(T_i)$. Therefore, as $r(T_i)$ is a subset of rooted triplets of the tree returned by Mincut, $r = \cup_{i=1}^{k} r(T_i)$ also holds. According to lemma 4.1 , ST displays all the trees in the collection T. □

4.5 Experimental Evaluation and comparisons

In this section, we discuss the results of the experiments conducted upon two different trees and compare them with the Mincut [53], and Adams consensus methods [91, 62]. The first data set consists of the trees, which shares common leaf nodes. The result will be a consensus tree. The input trees for this method are given in Figure 4.5.

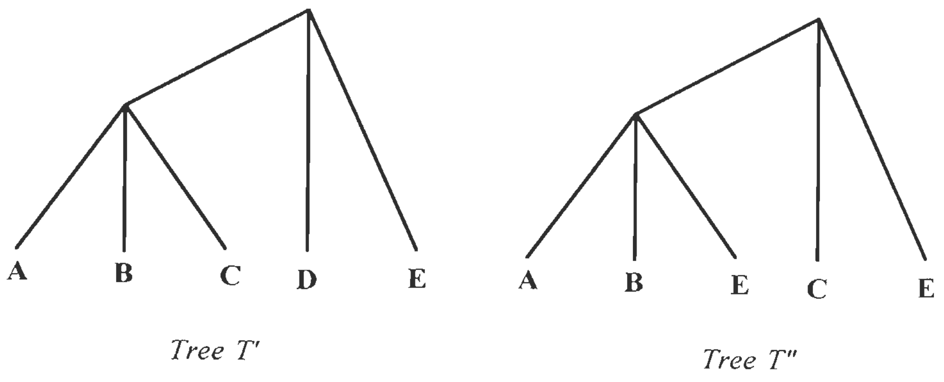


Figure 4.5: Trees T' and T'' are amalgamated to give a consensus tree using different methods.

If the trees in Figure 4.5 are given as input to *DISTSUPERTREE*, after the first step *COMM* contains all the leaf nodes and set *DISTINCT* is empty. The distance matrix computed for the trees is given in Figure 4.5.

The matrix represented in Figure 4.5 is constructed using the average number of edges between any two leaf nodes in two trees. Then the tree ST is constructed using these distances as shown in Figure 4.7(a). This tree returned by the algorithm as supertree after step 4 because the set *DISTINCT* is empty. Figure 4.7 also includes the results obtained by the Mincut and Adam's consensus method.

DISTSUPERTREE gives the identical tree to the Adam's consensus tree when applied to the common leaf nodes, indicating that the *DISTSUPERTREE*

	A	B	C	D
B	2			
C	2.5	2.5		
D	3	3	2.5	
E	2.5	2.5	3	2.5

Figure 4.6: Distance matrix obtained as a result of step 3 of DISTSUPERTEE.

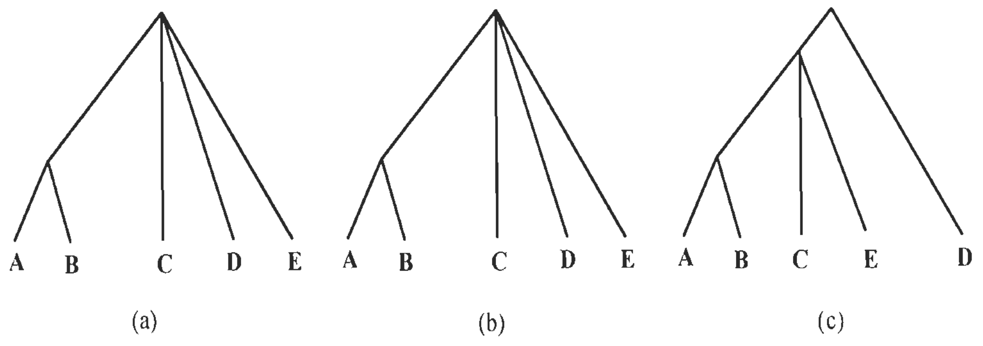


Figure 4.7: Consensus trees as the result of combining tree of Fig.3 (a) DISTSUPERTREE (b) Adam's consensus method (c) Mincut.

method is at least as good as Adam's consensus method, which preserves all the nesting represented by all the input trees. The reason for the difference in output trees shown in Figure 4.7 is that Mincut supertree method uses a local optimization principle and Adams consensus tree uses set theoretic operations. The disadvantage of the Mincut algorithm is that the result of the algorithm sometimes contains the triplets which express conflict in input trees, an example of this is given below. Even though the $DISTSUPERTREE$ uses an optimality criterion, it gives similar results to Adams consensus tree. But the results are appreciable when applied to the tree representing the polytomy, which results when two or more lineages diverge from the single ancestor at the same time.

An example is presented for the trees shown in Figure 4.8.

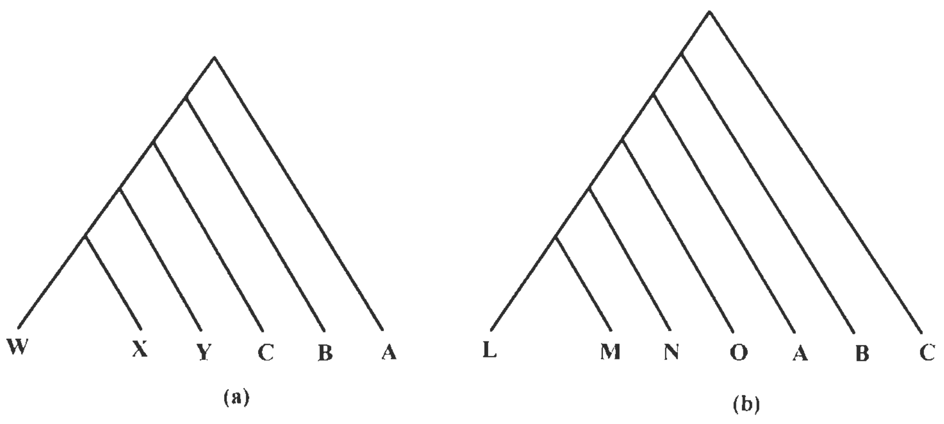


Figure 4.8: Trees $T^{'}$ and $T^{''}$ form an example of polytomy and are used for experiments.

In the trees shown in Figure 4.8(a) and 4.8(b), only three leaf nodes (A, B, and C) are common and the trees show a conflict on the relationship of these nodes. They also show resolved relationship for distinct leaf nodes. When trees shown in Figure 4.8 are given as input to the *DISTSUPERTREE*, the result of step 3 and step 4 is shown in Figure 4.9.

$$\begin{array}{ccc} & A & B \\ B & 3 & \\ C & 4 & 3 \end{array}$$

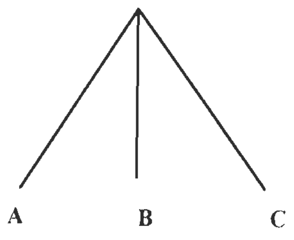


Figure 4.9: Average distance matrix for COMM of Figure 4.8, and a tree for the same.

The subtrees of DISTINCT represent subtrees of the corresponding tree without any common node. Therefore, the subtrees can be directly attached to the root node of the ST. The result of *DISTSUPERTREE* is shown in Figure 4.10

and the result of Mincut is shown in Figure 4.11.

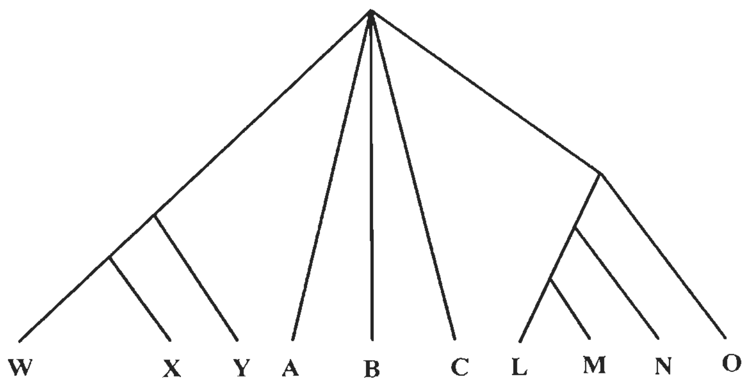


Figure 4.10: The supertree constructed using DISTSUPERTREE for the tree shown in Figure 4.8.

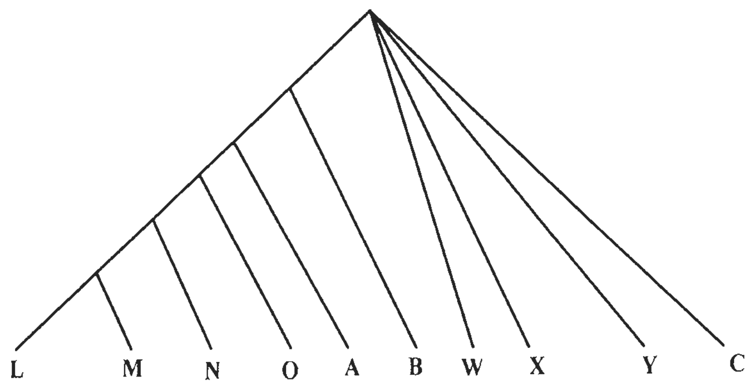


Figure 4.11: The supertree constructed using Mincut for the tree shown in Figure 4.8.

It is clear from the Figure 4.10 and Figure 4.11 that when the tree shown in Figure 4.8 are given as input to *DISTSUPERTREE* and Mincut, the *DISTSUPERTREE* gives a better fit than Mincut. The input trees disagree on the relationship of A, B, and C, but the supertree computed using Mincut method contains triplet $AB|C$, thus resulting in an unresolved supertree. Both Mincut and Modified Mincut are local minima algorithms. There exists an exponentially large number of supertrees for a given set of tree and the above algorithms stop at local minima. The above example shows that the Mincut algorithm is sensitive to the size of the input trees and in addition, it fails to include the contradictory information of the input trees. There is no existing supertree or consensus method that displays all the uncontradicted information [54]. As it is always desirable to maximize the uncontradicted information that a supertree displays, our method fills the lacuna.

4.6 Supertree for unrooted input phylogenetic trees

The phylogenetic tree construction methods typically generate urooted tree to represent evolutionary relationship. This representation has strong statistical and natural selection basis, such as maximum likelihood and maximum parsimony methods respectively, to prove its relevance in biology. If the smaller phylogenetic trees are constructed using these methods, resulting in smaller unrooted phylogenetic trees. The task of amalgamating these unrooted trees in such a way that the properties of supertrees mentioned in Chapter 2 are also satisfied is not possible [54].

The possible alternatives to produce a satisfactory solution or supertree are:

1. Ignore some of the properties.

2. Have restrictions on input trees such as rooted input trees, number of input trees, or restriction on the degree of input trees.

3. Return a collection of supertrees rather than a single supertree.

Each of possible alternative have its own disadvantages, such as the first alternative may lead to an uninformative star like tree. The second alternative is promising but converting unrooted trees to rooted trees and reverting back to original unrooted trees is difficult. Though, mathematical methods have been developed for finding roots in a graph yet their biological significance is very trivial [92]. The third alternative may result in an exponential number of trees making it difficult to choose the best among them.

In this section, we propose a heuristic algorithm for constructing supertrees for unrooted input trees. We follow the second alternative of converting the unrooted tree into rooted trees. We used $DISTSUPERTREE$ to construct rooted supertree. Finally the resulting rooted supertree is converted to unrooted tree using the unrooting information acquired during the initial converting process.

4.6.1 Converting trees

It is important to note that the root need not be the temporal or ancestral root of the tree. This property gives a greater freedom in selecting the root of an unrooted tree and also makes it difficult to all the possibilities. There are two possible approaches of converting the unrooted tree to a rooted tree.

The first approach is to select a taxon present in all the trees, consider it as "pseudo-root". Next the pseudo root is removed along with its incident edges from all the unrooted trees. The node that was incident to pseudo root is considered as root of each tree. If there are more than one taxon is common in the input collection, selecting different taxon leads to different resulting supertrees that leads to violation of some of basic properties of supertrees.

The second approach is to add new leaf node or taxon to all the input trees and consider its incident node as the root of the tree. It is computationally expensive to decide the nodes to which a new node can be attached.

To overcome these problems, we propose an alternative approach based on pendants vertex and pairs. A vertex of degree one in a rooted or unrooted tree is called a pendent vertex (i.e. leaf node) and the edge incident on one pendent vertex is called pendent edge. If a pair of pendant vertices are adjacent to a non-pendent vertex, which is not adjacent to any other pendant vertices then the pair is called pendent pair. Pendant pairs should be preserved while merging two trees for optimality.

The proposed heuristic approach divides the unrooted tree into two rooted trees by removing the edge that is incident to the nodes that divide the unrooted tree in to equal number of pendant pairs. This divides each tree T_i into two subtrees ST_{i1} and ST_{i2}. The incident vertices of the removed edged in ST_{i1} and ST_{i2} are connected through a new node called temporary root. The remaining pendant vertices are added based on their position in other trees. The algorithm is as follows:

```
ALGORITHM: ROOT(T)
begin
1.  Remove pendant vertices from each tree leaving pendant pairs.
2.  Remove the edge that divides the into two subtrees with equal pendants
3.  Add a new root node and attach the roots of the subtrees with this node
4.  for each tree $T_i$
5.      for each pendant pair $p_j$
            if $p_j \in T_k$ then
                attach the pendant vertex $p_j$ to the subtree consist of other
                pendant vetex of the pendant pair found in $T_k$.
            else
```

```
                Attach either the taxon to any subtree. The result will not be effected
            **end (if)**
        **end (for)**
    **end (for)**
**end (Algorithm)**
```

Working of this algorithm can be illustrated with the following example. The input trees are shown in Figure 4.12. Trees after removing the pendant vertices are shown in Figure 4.13. Input trees shown in Figure 4.12(a) have an additional pendant vertex E. The pendant vertex E is a part of pendant pair tree shown in Figure 4.13(b). Therefore, according to step 5, in the above algorithm, the vertex E is to be attached to the subtree consisting of node D as the taxa E and D share a pendant pair in tree shown in 4.13(b). The rooted tree for the input trees are shown in Figure 4.14.

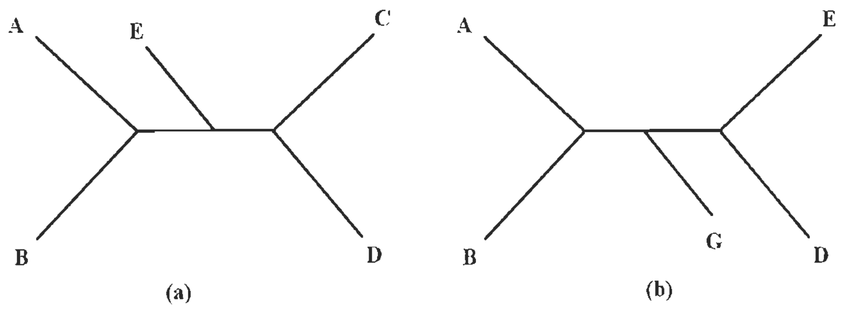


Figure 4.12: Two unrooted phylogenetic trees.

A generic supertree algorithm for amalgamating the converted rooted trees is as follows: The algorithm first computes the supertree then selects a suitable set of nodes to condense and remove the attached root nodes. The information retained in the form of equal pendants can be used to condense nodes. In

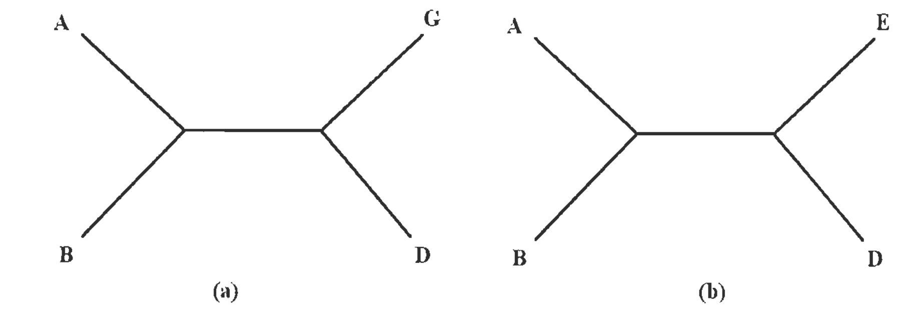


Figure 4.13: Trees after removing the pendant vertices from trees shown in Figure 4.12.

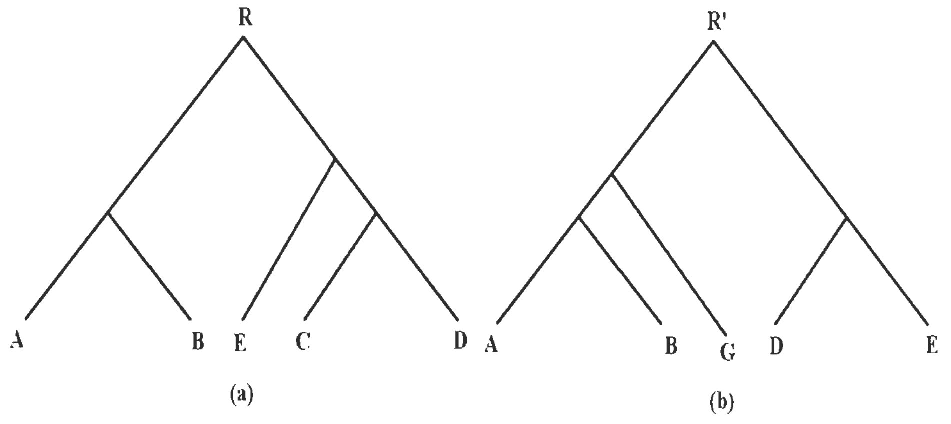


Figure 4.14: Rooted trees for the trees shown in Figure 4.12.

other words, either ST_{i1} or ST_{i2} is condensed. This leads to the final unrooted supertree.

```
ALGORITM: MERGE( X )
begin
1. $T = ROOT(X)$
2. $T = DISTSUPERTREE(T)$
3. for each tree $T_i$
       Condense either $ST_{i1}$ or $ST_{i2}$.
```

```
    end for
end
```

The result after step 2, when *MERGE* algorithm is applied to the input trees given in Figure 4.12 is shown in Figure 4.15. The final supertree is shown in Figure 4.16. As the input trees are compatible any subset of taxa in subtree ST_{i1} and ST_{i2} will not be on different path from root to the leaf node in all the trees. This indicates the condensing any sub tree will not effect the resulting supertree. Both the changes leads to same isomorphic graphs.

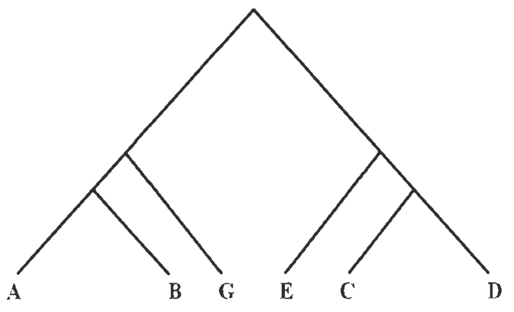


Figure 4.15: Result of DISTSUPERTREE when applied to the trees shown in Figure 4.12.

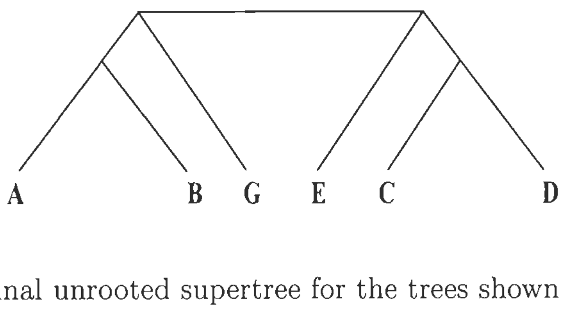


Figure 4.16: Final unrooted supertree for the trees shown in Figure 4.12.

Unlike the other rooting methods the proposed approach preserves the information of edges removed during rooting the trees. Moreover, nodes to be connected to temporary roots can be easily identified and resolved in polynomial time. This depicts the superiority and uniqueness of the proposed method.

4.7 Conclusion

The *DISTSUPERTREE* algorithm, presented in this chapter, demonstrates all desirable properties of supertree algorithms. It has a polynomial time complexity and results in a single output tree for a given set of input trees. Most of the supertree methods do not return a supertree for incompatible input trees. The supertree method presented in this chapter always returns a single supertree. The supertree methods which depends on the maximal agreement such as Jesper et al. [93], remove the edges which represents conflicting information resulting in removal of certain leaf nodes. MRP is a global optimization technique, which finds one or more optimal supertrees. These methods are computationally heavy.

The concluding comparison of *DISTSUPERTREE* with different existing supertree methods is given in table 4.1. The supertree methods are evaluated on the basic properties such as, time complexity, number of tree returned, and results of the methods when the input trees carry evolutionary conflicts. Comparisons on polytomy, where more lineages diverge from the single ancestor at the same time, and optimization criteria are also made. The result of comparisons shows that *DISTSUPERTREE* outperforms all the existing methods.

Table 4.1: A comparison of different supertree methods

Algorithm	Time Complexity	Number of Trees returned	Result of incompatible input trees	Polytomies in input trees	Optimization Technique
MRP	Exponential	Many	Many Trees	Many Trees	Global Minima
Mincut	Polynomial	One	One Tree	Unresolved Tree	Local Minima
Modified Mincut	Polynomial	One	One Tree	Resolved Tree	Local Minima
RankedTree	Polynomial	One	No Tree	No Tree	Local Minima
Strict	Polynomial	One	No Tree	No Tree	Local Minima
DISTSUPERTREE	Polynomial	One	One Tree	Highly resolved Tree	Local Minima

Chapter 5

Combining semi-labelled rooted phylogenetic trees

There are two approaches, agreement and optimization, for combining the small phylogenetic rooted trees with overlapping taxa into a single tree. Almost all the existing supertree methods, irrespective of the underlying approach, are developed based on the implicit assumption that only leaf nodes are labelled in the input trees. On the other hand, it is common that the phylogenetic trees constructed on morphology and fossils to have labels on internal nodes [94]. The existing methods to solve this problem took an agreement approach and fail to return any tree if the input trees are incompatible.

5.1 Introduction

Recently, Page [94] posted an interesting problem of combing the input trees in which the leaf nodes and some internal nodes are labelled. The phylogenies constructed based on morphological studies often contain the labelled internal nodes. This needs a more generalized supertree approach. As yet, AncestralBuild

[70] and NestedSupertree [95] are the only known algorithms for constructing supertree of the small phylogenetic input trees with labelled internal nodes.

Both the algorithms are all-or-nothing algorithms. NestedSupertree is a generalization of the AncestralBuild. Both the algorithms give either a tree compatible to all the input trees or a message indicating that the input trees are not ancestrally compatible or there is ancestor descendent conflict. The formal definitions for ancestral compatibility and ancestor descendent conflict are given in the coming sections. This chapter propose an optimization based divide and conquer method to combine semi-labelled trees. This method will return a tree even for incompatible input trees. The conflict is removed by deleting the conflicting edges based on least square error estimates. Once the conflict is resolved, the Adam's consensus algorithm [91] is used to construct the supertree. The algorithm is applied on two datasets, one consists of semi labelled phylogenetic trees of spiders, and the second data set consists of hypothetical fully labelled tress.

5.2 Background and Preliminaries

In this chapter, we follow the terminology analogous to [70] and [95]. We present some of the basic concepts which are sufficient to understand the problem as well as the proposed solution.

A rooted phylogenetic tree, on label set S, is a tree T with a function f, in which all the nodes have degree three or more except the root node and leaf node, which have degree at least two and one respectively. The function f maps from S to leaf nodes of the tree T, $f : S \rightarrow setofleafnodes$. Semi-labelled phylogenetic tree on label set S is a generalization of a simple phylogenetic tree with all the leaf nodes as well as some of the internal nodes labelled. Let T be a semi-labelled tree with vertex set V. A function assigns the labels to node with the degree one and two. Moreover, some of the internal nodes can even be labelled but the leaf nodes must be labelled.

Let T be the rooted phylogenetic tree with the label set S. Given the label set $S^{'}$, such that the topological restriction of T to $S^{'}$ is the tree obtained by deleting the nodes which are not in the path from root to any node in $S^{'}$ and then contracting the internal edges whose degree is two. The topological restriction is represented as $T' = T|_{S'}$, an example of which is shown in Figure 5.1. T' is called the induced subtree of T by S'. A rooted tree T is said to display another rooted tree T' if $T|_{S'}$ is isomorphic to T'. A set of semi-labelled trees G is said to be compatible if there exists a semi-labelled tree T, which displays every tree in G.

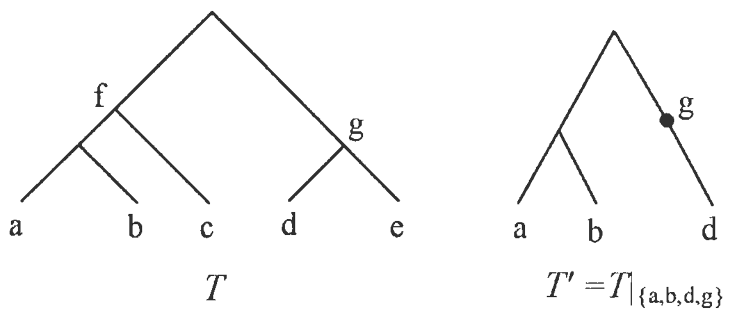


Figure 5.1: Two rooted semi-labelled trees. T' is induced subtree or restriction of T on the labels $\{a, b, d, g\}$.

A node a is said to be descendent of another node b if the path from root to a includes the node b. A rooted tree T is said to ancestrally display another rooted tree T' if $S' \subseteq S$ and $T|_{S'}$ is restricted in such a way that whenever a is strict descendent of b in T then a is also a strict descendent of b in T'. A collection of semi labelled trees Z is ancestrally compatible if a semi-labelled tree R ancestrally displays all the trees in Z. Two labels, a and b, are said to be pairwise consistent if whenever a is a strict descendent of b in some tree in Z then a is always a strict descendent of b in every tree of Z whose label contains both a and b.

5.3 The ResolvedSupertree Algorithm

In this section, we define the ResolvedSupertree algorithm for semi-labelled collection of input trees. The algorithm is based on the construction of conflict graph. The conflicts are identified and resolved using least square error criterion. Then the Adam's consensus tree method [91] is used for the construction of the semi-labelled supertrees.

As a first step, new distinct labels are assigned to the root nodes of the input trees if the root node is not labelled. The input trees are then divided into multiple trees by cutting the tree till its internal label. The internal labelled node is considered as the leaf node label for the parent tree and root node label for the child tree. The division of the semi-labelled input trees is shown in Figure 5.2.

The conflict graph consists of only weighted arcs (directed edges) and may have cycles in it. Let Z' be the collection of semi-labelled rooted trees and their child trees. The arcs from node labelled a to the node labelled b is added to the conflict graph if a is an ancestor of b in any of the tree in Z'. The number of trees that represents the same child parent relationship of the nodes are assigned as the arc weight. An example is shown in Figure 5.3.

The root nodes of the trees T_1 and T_2 are not labelled, for the construction of conflict graph we assign new distinct labels u_1 and u_2 to the root nodes of the trees T_1 and T_2 respectively. The conflict graph for the trees T_1 and T_2 (shown in Figure 5.2) are given in Figure 5.3.

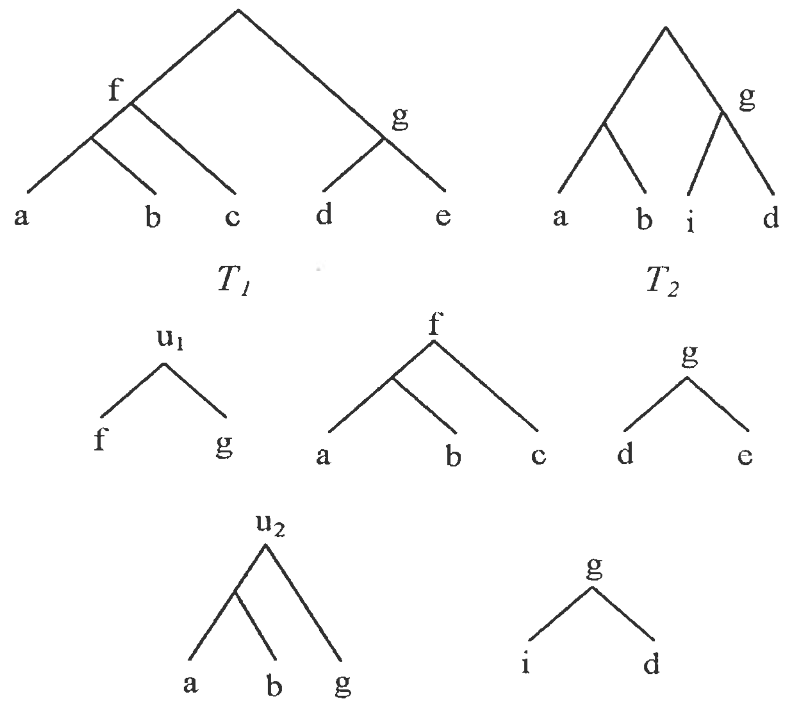


Figure 5.2: Two rooted semi-labelled trees $T1$ and $T2$ and the resulting trees after dividing from internal labels.

The given input tree collection can have the conflicting branch information. These conflicts are detected using a graph based approach that results in a conflict graph. This conflict graph is used for resolving the conflicts. A simple strategy is applied, which removes the edges that agree (appear in) by least number of trees is to resolve the conflict. In phylogenetics it is usually assumed that the information given in most number of trees is near to true phylogeny [96].

Once the conflict graph (CG) is ready, it is searched for different conflicts and the algorithm ConflictResolve is applied to resolve the conflicts. The algorithm results in a directed tree and parent table. The directed tree, with all the nodes having indegree exactly one, except the root node with indegree zero, is also

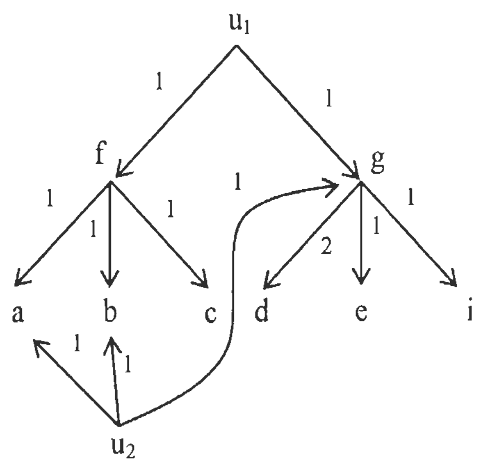


Figure 5.3: Conflict graph for semi-labelled trees T_1 and T_2 (shown in Figure 5.2).

called resolved graph. Based on the resolved graph, the subtrees are modified.

Let the node a be ancestor of nodes $\{b, c, d, e\}$ in divided collection of trees. The resolved graph shows that if the node a has child list $\{b, c, d\}$, then all the trees in the divided collection with the root node labelled a are refined for child list $\{b, c, d\}$. Therefore, the final trees having root as a, which are combined to get the supertree, have $\{b, c, d\}$ subset as its descendants. The parent table gives the information about the label of the least common ancestor of the set of labels given in the table.

```
ALGORITHM: ConflictResolve ( CG )
Input: Conflict Graph constructed for the given input semi-labelled trees.
Output: Resolved graph and parent table.
Begin
  for each node $n$ in $CG$ having indegree two or more do
    if the originating node has indegree 0 (zero) then add node $n$ to the
    child list of originating node and remove the arc between them.
    else
      Delete all minimum weight incoming edges.
      In case of tie or more than one incoming edges, the following rules
      can be applied:
      If $n$ has $O_1$ and $O_2$ as originating nodes then if $O_1$ is the descendent
      of $O_2$ then the link between $O_2$ and node $n$ is removed.
    end (if-else)
  end (for)
  for each directed cycle in CG do
    Remove the arc with least weight. In case of tie remove; each
    minimum weighted arc and calculated the error using Least Square
    error criterion. Remove the edge with minimum least square error.
  end (for)
  return Modified Conflict Graph (MCG) and parent table.
end (Algorithm)
```

The algorithm *ConflictResolve* returns the modified conflict graph and parent table. The modified conflict graph is obtained by removing the conflicting arcs which leads to minimum error. All the nodes in the resulting modified graph have indegree equal to one. The immediate descendents of each node n in modified conflict graph represents the set of labels that the trees with n as root label can have. Finally the Adams consensus tree is used for the construction of the supertree.

ALGORITHM: ResolveSupertree (Z, MCG, parent table)

Input: Divided collection of semi-labled input trees (Z), Modified
Conflict Graph and parent table.

Output: Supertree for semi-labelled input trees.

```
begin
  for each node $n$ in modified conflict graph do
    Make a set, $S$, of all immediate descendents of node $n$.
    Find the restrictions of all the trees in $Z$ with root
    $n$ on $S$.
    for all modified trees with root node $n$ do
      Find the restriction of each tree on common nodes
      Apply Adams consensus tree for the modified trees.
      Add remaining taxa to appropriate edge.
    end (inner for)
  end (outer for)
  Merge all the trees to a make single tree as if tree $T$ has a leaf
    node label $x$ same the root label of $T'$
    $T'$ is attached to $T$ at the leaf node $x$.
  for each entry $e$ in parent table do
    Add a label, $e$, to the most recent common ancestor of the set
    element in entry $e$.
  end(for)
  Remove all the new distinct labels assigned to the roots of the
  Unlabelled roots of the input trees.
end (Algorithm)
```

5.4 Experimental results

To illustrate the algorithm, ResolvedSupertree is applied to a data set consists of semi labelled phylogenetic trees of spiders. The two spider trees, shown in

Figure 5.4 are taken from [94].

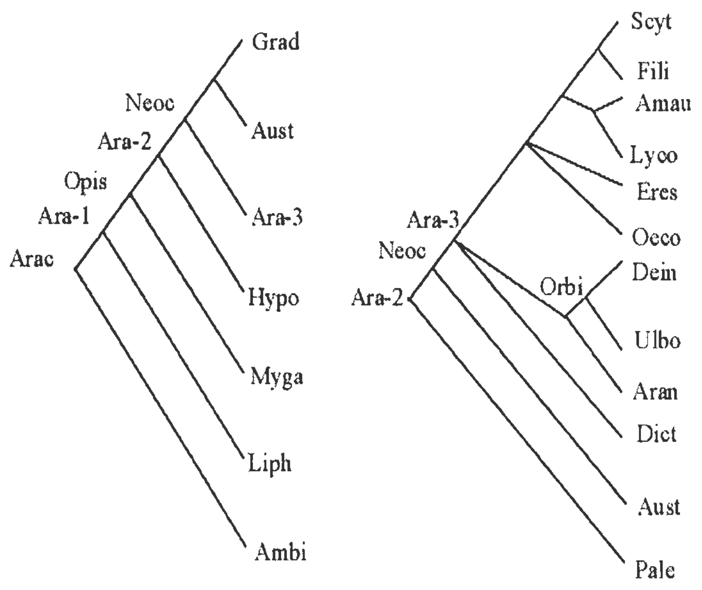


Figure 5.4: Two trees for spiders and related taxa. These can be obtained from study $S1x6x97c14c42c30$ from TreeBase.

We have taken the first four letters of the taxa in the Figure 5.4, the full names of the taxa can be obtained from study $S1x6x97c14c42c30$ from TreeBase [97] or [94]. The first step is to label the root nodes of the input trees. However, since both the trees are root labelled, therefore this step is avoided. Next, the trees are divided into child and parent tree through its internal labelled node. The internal label node, in master tree, is now leaf node in parent tree and root node label in child tree. The collection of trees divided from input trees (Figure 5.4) is shown in Figure 5.5.

The conflict graph is constructed based on the guidelines given in the algorithm. The conflict graph for input trees (Figure 5.4) is shown in Figure 5.6.

The collection of divided trees and the conflict graph is given as an input to the ConflictResolve algorithm. As the input conflict graph does not have any directed or underlying cycles, the algorithm return the same conflict graph with only Arac and relative arcs from the graph; Arac is added as the most recent common ancestor of the labels Ara-1 and Ambi, in parent table.

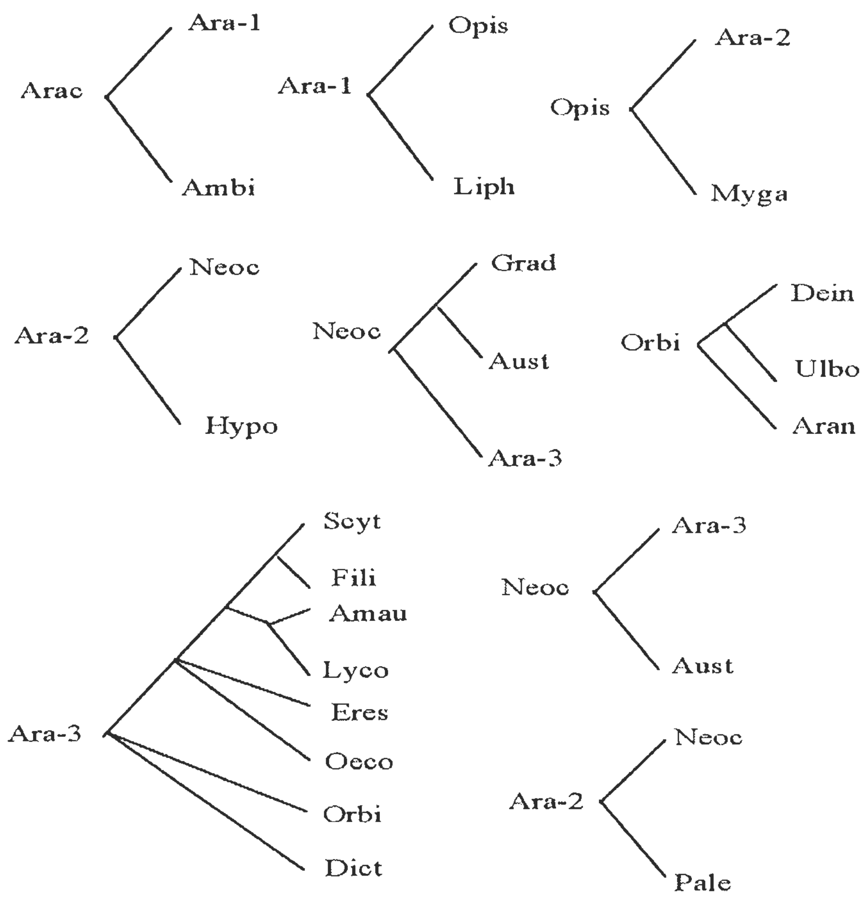


Figure 5.5: Trees obtained from dividing the input trees shown in Figure 5.4.

The conflict graph shown in Figure 5.6 does not have any cycles and underlying cycles in it. The arcs, Ara-2 to Neoc, Neoc to Ara-3, Neoc to Aust, are weighted 2 as they appear in both the input trees; rest of the arcs carry unit

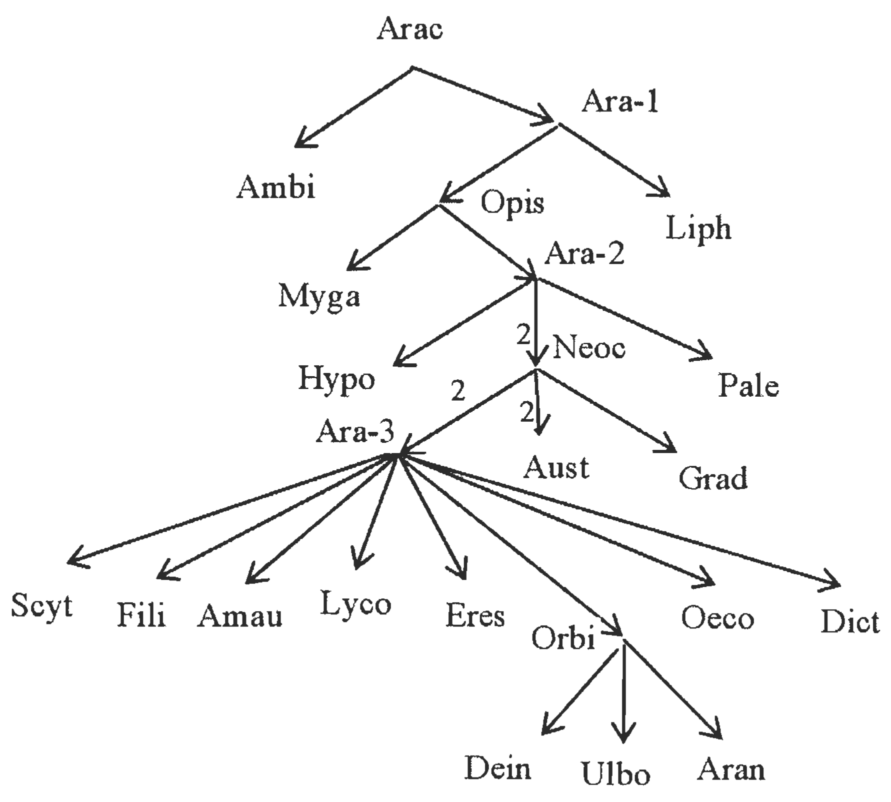


Figure 5.6: Conflict graph for the input trees shown in Figure 5.4.

weight. Now it is straightforward to costruct the supertree for each smaller tree with same root label and merge them for the final supertree. The final supertree for the input trees is shown in Figure 5.7. The supertree can be constructed for the trees shown in Figure 5.2 using the conflict graph shown in Figure 5.3.

5.5 Conclusion

The proposed algorithm addresses an extremely important problem in classification. The ResolvedSupertree outperforms the NestedSupertree and AncestralBuild as it preserves all the nesting information common to all the input trees due

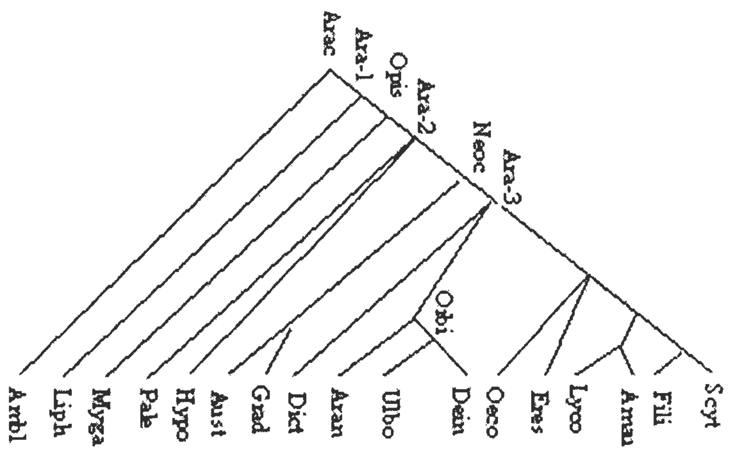


Figure 5.7: Supertree for the spider trees shown in Figure 5.4.

to underlying Adams consensus tree construction method. On the other hand, both the NestedSupertree and AncestralBuild may not display the information carried by all the input trees [95].

Above all, this is the first algorithm which resolves the conflicts or incompatibilities and returns the semi-labelled supertree even for incompatible semi-labelled input trees. The conflicts are removed based on minimum error criterion, which is a quite sensible approach for resolving the conflicts.

Chapter 6

Combining rooted phylogenetic trees with ancestral divergence time

Many supertree methods have been developed for amalgamating the small rooted phylogenetic trees with overlapping taxa into a single tree. These methods combine the input trees based on the topological information carried by each of the input tree. Other important evolutionary information such as fossil data, molecular dating data and actual divergence time estimates are usually ignored. If the available evolutionary information is considered with tree topology for combining the input collection of trees then the resulting supertree may be more accurate than the supertree constructed without using the additional information. The existing supertree methods with the ability to include additional evolutionary information are the extensions of the BUILD algorithm which have the property of all-or-nothing, i.e., if the input collection of trees is incompatible, the algorithm fails to return the supertree.

6.1 Introduction

Supertree construction methods are generally classified into two categories, optimization and agreement methods, based on their objective function. The optimization based supertree construction methods result in supertree that has the maximum fit for certain objective function when applied to a given collection of input trees. On the other hand, the supertree constructed using agreement based approaches represent the groupings which are common to all the trees. The agreement based methods show the groups or clusters present in all the input trees and was introduced by the Gorden [98]. When there are conflicts in the input collection the resulting supertree constructed using the agreement technique may have polytomies. The disadvantage of using optimization technique when the input tree collection show conflicting information is that the resulting supertree may not have all the relationships present in the input tree collection. The agreement supertree methods include MinCutSupertree [53] and its variants [12, 69, 70], and the variants of strict supertree methods [98, 99, 100]. The Matrix Representation of Parsimony (MRP) [64, 65], Bayesian supertree [101] and MinFlip supertree methods [76] optimize the objective function for constructing the supertree, therefore, they come under the optimization based approaches. A comprehensive survey on supertree methods is given in [102].

Till date, RANKEDTREE [12] is the only published supertree construction algorithm, which incorporates the additional information, such as divergence date along with the tree topologies. This is an extension of BUILD [63] and suffers with the problem of incompatible input trees. The disadvantages of the RANKEDTREE are as follows.

1. It can not process the incompatible input trees.

2. It does not represent all the nesting present in the input collection [95].

3. It assumes only topological incompatibility and neglects other conflicts, such as conflict in divergence dates information.

This chapter propose a supertree method, which incorporates relative time divergence information with tree topologies. This method returns a tree even for incompatibilities such as divergence dates conflicts and incompatibility between topology and divergence dates. We follow the least error method for removing the conflicts. Once all of the conflicts are resolved, the Adam's consensus algorithm [91] is used to construct the supertree. Finally a rank function is used to rank the internal nodes of the resulting supertree based on the divergence date information. The algorithm is applied to a data set consisting of hypothetical trees with divergence time.

In addition to the aforementioned properties, the proposed algorithm satisfies the desirable properties of the supertree construction methods, given in chapter 2.

Figure 6.1 shows the generic architecture for incorporating the divergence date information into the existing supertree methods. The process includes three modules: conflict detection, conflict resolution and supertree refinement. The conflict detection module detects and resolves the divergence date and tree incompatibilities. Three types of conflicts that are handled by the proposed method are:

1. Conflicts in relative divergence data information

2. Tree incompatibilities

3. Conflicts between trees and divergence date information

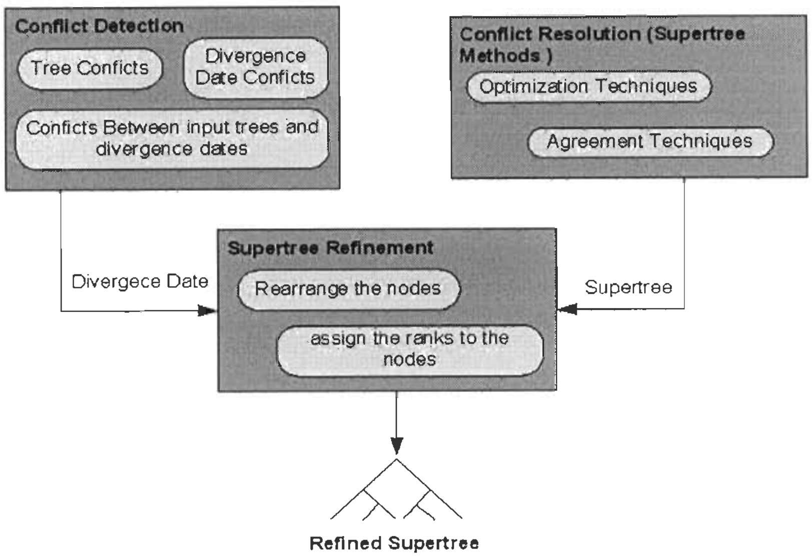


Figure 6.1: A generic architecture for constructing the supertrees with divergence date information.

The conflict resolution module is an integrated part of the optimization based supertree construction methods, whereas the methods like BUILD [63] and its variants need a collection of compatible input trees. The supertree refinement module makes use of processed divergence date information and the result of supertree module to return a refined supertree.

6.2 Background and Preliminaries

In this section, we present necessary concepts to understand the problem and solution of incorporating ancestral divergence times along with input tree topologies for constructing supertree.

A rooted phylogenetic tree, on label set S, is a tree T in which all the internal nodes have degree three or more, while the root node and leaf nodes have degree at least two and one respectively. Polytomies are referred to the nodes that give rise to three or more descendent lineages simultaneously. Mathematically, these are the internal nodes in the tree with degree more than three. This is usually the result of insufficient or conflicting information about the order or branching information.

Let G be a collection of rooted phylogenetic trees. A tree T is said to be a supertree if and only if the tree T displays all the trees in G. Let G_1 be a phylogenetic tree. For all $v_1, v_2 \in V(T), v_1 \leq_{G_1} v_2$, represents that the node v_2 is descendent of v_1 in the tree G_1. The unique vertex of G_1 that is the greatest lower bond of v_1 and v_2 under $\leq_{G_1}$ is referred to as the most common recent ancestor of v_1 and v_2 of T, represented as $MRC_{G_1}(v_1, v_2)$. The supertree constructed using the input collection G, where $G_1 \in G$, should preserve the $v_1 \leq_{G_1} v_2$ relationship found in G_1, i.e. $v_1 \leq_T v_2$. A set of phylogenetic rooted trees G is said to be topologically compatible if there exists a phylogenetic tree T, which display every tree in G. The relative divergence date is represented in the form of statements, such as "$div(x, y)$ predates $div(u, v)$", which means that the divergence of x and y predates the divergence of u and v species. A rank function, R, maps the interior nodes(V') of the tree to some positive integer such that $\forall(v_1, v_2) \in V', R(v_1) < R(v_2)$ if v_2 is a proper descendent of v_1.

Similarly, conflict may occur between topology and divergence date information. For example, consider the supertree topology given in Figure 6.2, and divergence date information is given as "$div(c, d)$ predates $div(d, f)$ ". Both the tree and divergence date represent contradictory information.

The phylogenetic supertree T with ancestral divergence time D is said to preserve the divergence time if the rank $R(a, b) < R(c, d)$ for the given divergence time information statement, "$div(a, b)$ predates $div(c, d)$".

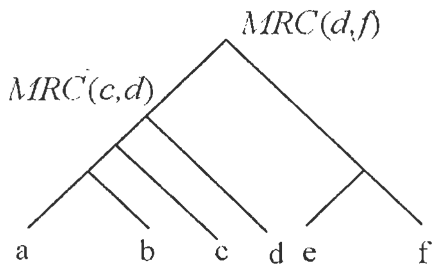


Figure 6.2: Tree topology contradicting the divergence date, "$div(c, d)$ predates $div(d, f)$".

6.3 Conditions for detecting conflicts in divergence date statements

In this section, we formulate the necessary and sufficient condition for detecting conflicts in the divergence date statements. The lemmas proved in this section represent different conditions where the divergence date conflicts may occur. Lemma 6.1 indicates an important property of the divergence date statements. It states that no subset of species can predate its superset, and if a set X predates another set Y, then any subset of Y also predates X.

Lemma 6.1. *Let T is a tree with label set $L(T)$, which satisfies the divergence date information "$div(X)$ predates $div(Y)$", where $X, Y \in L(T)$. Let x and y are subsets of X and Y, $x \subseteq Xandy \subseteq Y$, respectively then all of the following conditions should hold for compatibility:*

1. *$div(X)$ predates $div(x)$ or $MRC_T(X) = MRC_T(x)$*

2. *$div(Y)$ predates $div(y)$ or $MRC_T(Y) = MRC_T(y)$*

3. *$div(X)$ predates $div(y)$*

Proof. Let us consider the trees shown in Figure 6.3 that satisfy the predate condition, "$div(X)$ predates $div(Y)$", where $X, Y \in L(T)$. It can be easily proved that the three conditions given in lemma 6.1 should hold for compatibility.

To prove first two conditions it is sufficient to prove that the $MRC(N)$ will be the root of the restricted tree $T|_N$, where $N \subseteq L(T)$. As the root node is the first divergence event in the tree, it is ranked with the least possible positive value. Let T be a tree with label set $L(T)$, the restriction of T on a set of labels N, where $N \subseteq L(T)$, can be obtained by removing all the leaf nodes which are not in N and suppressing the nodes with degree two.

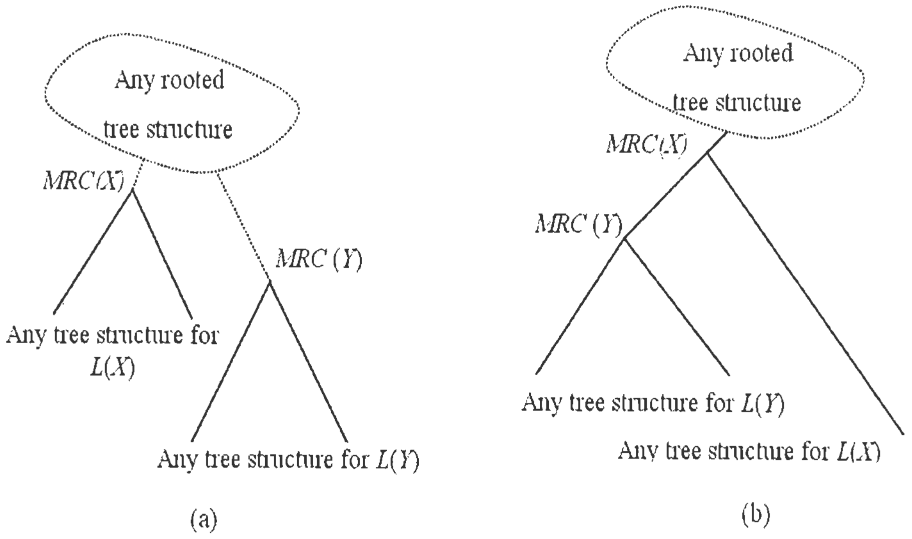

Any rooted
tree structure
MRC(X)
MRC(Y)
Any tree structure for L(X)
Any tree structure for L(Y)
(a)
Any rooted
tree structure
MRC(X)
MRC(Y)
Any tree structure for L(Y)
Any tree structure for L(X)
(b)

Figure 6.3: (a) A generalized tree satisfying the divergence date statement D, "$div(X)$ predates $div(Y)$". (b) Generalized tree structure preserving D, where $L(Y) \subset L(X)$.

Clearly, the resulting tree is $T' = T|_N$ with N as set of leaf nodes and $MRC(N)$ as root. Therefore, any subtree T' of T has the rank of its root node greater than the rank of the root node of the parent tree T. Let X be a set of labels then $MRC(X)$ predates $MRC(x)$, where $x \subset X$, and the rank of x is greater than or equal to the rank of X, $R(X) \leq R(x)$.

The ranks for most recent common ancestor of X and Y, in relative representation is $R(X) < R(Y)$. The value assigned by rank function to the most common recent ancestor of Y is always greater than the rank of most recent common ancestor of X, as "*div*(X) predates *div*(Y)". If we take a set y,$y \subseteq Y$, then $MRC(y)$ may be a descendent of $MRC(Y)$ or is same as $MRC(Y)$ and its rank will always be $R(Y) \leq R(y)$. We know that $R(X) < R(Y)$, or $MRC(X)$ predates $MRC(Y)$, hence, the value of $R(y)$ should be greater than $R(X)$. The divergence relation between X and y can be given as $R(X) < R(Y) \leq R(y)$. This proves the third condition and hence the lemma. □

Lemma 6.2. *Let the divergence date information "div(X) predates div(Y)", where $X, Y \in L(T)$, be given. There exists a tree T with $L(T)$ labels. If a tree represents $MRC(X) = MRC(Y)$ then the divergence information, which includes X and Y on different sides of divergence notation is incompatible.*

Proof. Let $X, Y \in L(T)$, they are comparable under $\leq$ if and only if the one of them is a strict descendent of other, i.e. $X \cap Y = X, Y$. If $X \cap Y = \varnothing$ then the label sets X and Y are not comparable under $\leq$, as neither X nor Y is a descendent of other. The information given in the divergence date statement "*div*(X) predates *div*(Y)" is true if and only if Y is a strict descendent of X, in other words $Y \subseteq X$, as shown in Figure 6.3(b). The information carried by the tree, $MRC(X) = MRC(Y)$, is true if they share a set of species which have the same greatest lower bond, i.e. $X \cap Y = N$, where $N \neq \{X, Y, \varnothing\}$. Let the rooted phylogenetic tree T, shown in Figure 6.4, have the labels $L(A)$ and $L(B)$ and $MRC(A)$ and $MRC(B)$ are not comparable under $\leq$ or they are on different paths from the root of the tree.

If two sets of labels X and Y are selected in such a way that they contain labels from both A and B, then $MRC(X) = MRC(Y)$, as shown in Figure 6.4. There are two conditions to achieve $MRC(X) = MRC(Y)$. They are:

1. $X \cap Y = \{X \cup Y, \varnothing\}$

2. $X \cap Y = N, where N \neq \{X, Y\}$

   The condition for preserving the divergence date statement, "$div(X)$ predates $div(Y)$", is as follows:

3. $X \cap Y = X$

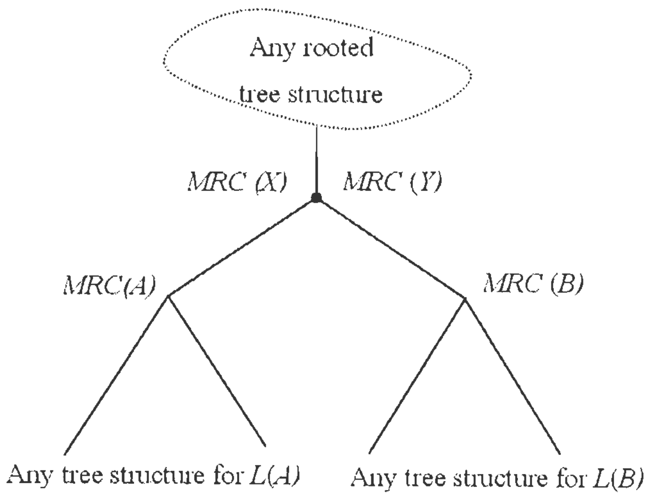


Figure 6.4: A generalized tree satisfying the $MRC(X) = MRC(Y)$, where $X, Y \subset \{A\ and\ B\}$.

As condition 3 is not possible with 1 or 2, the two statements, "$div(X)$ predates $div(Y)$" and $MRC(X) = MRC(Y)$ are incompatible and represents a conflict. $\square$

Lemma 6.2 helps in identifying the conflict between the divergence date information and tree topology. If given a tree which says $MRC(X) = MRC(Y)$ and a statement "*div*(X) predates *div*(Y)" then it leads to a conflict.

Lemma 6.3. *Let T is a tree with label set $L(T)$, which satisfies the divergence date information "div(X) predates div(Y)", where $X, Y \in L(T)$. If Z is a set of labels with the property $Z - X \neq \varnothing$ then "div(Z) predates div(X) and div(Y)".*

Proof. Let T is a tree with label set $L(T)$, which satisfies the divergence date information "*div*(X) predates *div*(Y)", indicating that $X \cap Y = X$ and $Y \subset X$. Consider a set of labels Z such that $Z - X \neq \varnothing$. Let $Z \cap X = x, MRC(x)$ can be at most $MRC(X)$, the labels $Z - x$ does not belong to X. Therefore $MRC(Z)$ is $MRC(MRC(x), MRC(Z - x))$. As the sets x and $Z - x$ belong to different clusters in the tree, they are not comparable under $\leq$. Therefore, $MRC(x, Z - x)$ is the root of these two subsets and predates both the sets. As Z predates X, it predates all its descendants. Therefore Z predates both X and Y. □

```
ALGORITHM: DivCompat( D )
Input: divergence date data $D$.
    The divergence date data is given in "div$(X)$ predates div$(Y)$" form.
Output: compatible divergence data data.
begin
  sort $D$ by the number of labels in it, in descending order
  and attach a conflict flag to each statement .
  repeat until all statements have conflict values zero
    initialize all conflicts to 0 (zero).
    begin
      for each statement, $s$, in sorted $D$ do
```

```
        for s+1 to last statement of $D$ do
          if there is a statement on $X$ and $y, y \subset Y$ then
            The statement should be "$div(X)$ predates $div(y)$".
          else
            increment the conflict by one.
          end (if)
          if there is a statement on $X$ and $Z, Z - X \neq \emptyset$ then
            the statement should be "$div(Z)$ predates $div(X)$".
          else
            increment the conflict by one.
          end (if)
        end (for) // first for
      end (for) // second for
      for each statement in $D$ do
        remove the statement with maximum conflict and modify the $D$.
      end (for)
    end (repeat)
  return $D$
  end (Algorithm)
```

Figure 6.5: DivCompat: An algorithm for finding compatible divergence dates.

6.4 Supertree with ancestral divergence time algorithm

In this section, we present the DivCompat and DdateSupertree algorithms, which incorporates the divergence date information with topological information.

6.4.1 The DivCompat Algorithm

The DivCompat algorithm takes the divergence date information as input and returns compatible divergence information. The algorithm makes use of the conditions identified in lemma 6.1, 6.2 and 6.3 and is given in Figure 6.5.

Theorem 6.1. *Let the divergence date information D applied to the algorithm DivCompat results in a compatible set of divergence date information statements.*

Proof. Let D be the set of divergence date information statements. When applied to the DivCompat algorithm, each statement is checked for conflict with other statements. Both the subsets and supersets of the labels involved in the statements are considered for conflict detection. Let the divergence date information statement be "$div(X)$ predates $div(Y)$", then X predates any subset of Y as shown in lemma 6.1. Therefore, any statement on X and y where $y \subset Y$, should be "$div(X)$ predates $div(y)$", otherwise there is conflict. On the other hand, if there a set of labels Z, such that $Z - X \neq \varnothing$, then the statement involving Z and X or Y should be of the form, "$div(Z)$ predates $div(X)$ or $div(Y)$", and Z also predates subsets of both X and Y. The algorithm DivCompat checks for all these conditions in steps 8 and 13. Therefore, DivCompat algorithm detects all the incompatibilities in divergence date statements and resolves them by removing the statements that contradict with maximum number of divergence date statements. □

The DivCompat algorithm returns a compatible set of divergence date statements in polynomial time as it is shown in proposition 6.1.

Proposition 6.1. *Let D be the collection consisting of n divergence date statements, then DivCompat results in a compatible set of divergence date information* *in time, where n is the number of divergence date statements.*

Proof. The running time of the DivCompat algorithm is dominated by steps 6 and 7, which involve comparing different statements for compatibility. Given n divergence date statements, each statement is compared with other statements. The comparisons take n iterations and the loop in step 6 runs $(n-i)$ times, where i is the statement number which is being compared with other statements. These two loops take $O(n^2)$ time. Heap sort can be used to sort the statements, which runs in $O(nlogn)$ time. The loop in step 20 takes $O(n)$ time as it process each divergence date statement ones. Therefore, the algorithm runs in $O(n^2)$ time. □

6.4.2 The DdateSupertree Algorithm

Our proposed DdateSupertree algorithm makes use of DivCompat and Adams consensus tree construction algorithms and returns a ranked supertree. As a first step the divergence data information is applied to the DivCompat algorithm, which returns the divergence date statements which are compatible to each other. A common set of labels are searched among the input tree collection and applied to the Adams consensus algorithm. The labels which are not common to all the input trees are then attached individually to the result of the Adams consensus tree. Finally the hierarchies are ranked according to the divergence date statements. The additional information usually refines the polytomies in the resulting supertree. The polytomies in the supertrees are due to incompatible or insufficient input information. The algorithm computes the ranked supertree in polynomial time and preserves all the nesting present in the input collection.

Theorem 6.2. *Let the divergence date information D and input collection of rooted phylogenetic trees G, be applied to the algorithm DdateSupertree, which* *results in a supertree that preserves all the nestings present in the input collection and also preserves the divergence date statements.*

Proof. For collection of input trees $T = \{T_1, T_2, ..., T_k\}$, if $\{a, b\}$ nest in $\{a, b, c\}$ for every tree T_i in T, then $\{a, b\}$ nest in $\{a, b, c\}$. The DdateSupertree used the Adams consensus algorithm as the base and added the nodes unique to each tree to build a supertree, the resulting supertree preserves all the nesting present in all the trees. Let the label sets for two trees T_1 and T_2 are $\{a, b, c, d, e\}$ and $\{a, b, c, d, g, h\}$ respectively. The nestings present in both the trees contain labels that are present in both of them. If all the nesting of labels present in both the trees, $T_1 \cap T_2 = \{a, b, c, d\}$, is preserved then the supertree built on this, also preserves all the nestings present in all the trees. It is proved in theorem 6.1 that the DivCompat algorithm returns only the compatible divergence date statements. The DdateSupertree algorithm ranks the internal nodes based on the compatible divergence date statements returned by DivCompat. Therefore, the resulting supertree preserves the divergence date statements. □

Proposition 6.2. *Given a set of divergence date statement D and collection of input trees G, a supertree can be constructed using DdateSupertree in $O(kn^3)$ time where k is number of input trees and n is number of unique labels in the input tree collection.*

Proof. The algorithm consists of three major steps: finding the compatible divergence date statements, constructing the supertree and ranking of the internal nodes of the tree based on the divergence date statements. Finding compatibility between the divergence date statement takes $O(m^2)$ time as shown in proposition 6.1, where m is the number of divergence date statements. Constructing the supertree with total n labels can be analyzed as follows. A straightforward algorithm for Adams consensus tree takes $O(kn^2)$ time, where k is the number of trees and n is the number of unique labels in the input tree collection [103, 104]. If the total number of leaf nodes are n, then the labels which are common to all the trees, l, is always less than or equal to the total number of labels, $l \leq n$, and the number of labels that are unique in all the trees, u, is also less than or equal to the total labels, $u \leq n$. Therefore, the total time required to build Adams consensus tree is $O(kn^2)$ and adding each unique label takes at most n more iterations, hence the total complexity for constructing the rooted supertree is $O(kn^3)$. The final step uses the compatible divergence time statements to refine the supertree, which takes at most m iterations, where m is the total number of statements divergence date statements. After the incompatibility test there can be at most m statements. Hence the total time required by DdateSupertree for constructing the supertree is $O(kn^3 + m^2 + m)$. The number of divergence date statements are always less than or equal to the unique nodes in the input tree collection, i.e. $m \leq n$, therefore the total complexity of the algorithm is $O(kn^3)$. □

6.4.3 An Example

To illustrate the working of DdateSupertree algorithm, it is applied to the tree collection shown in Figure 6.6. Divergence date information statements are:

1. "$div(d, e)$ predates $div(a, b, c)$"

2. "$div(a, b)$ predates $div(d, e)$"

3. "$div(a, c)$ predates $div(a, b)$"

As a first step in DdateSupertree, the divergence date statements are applied to DivCompat algorithm that checks the statements for the compatibility. The

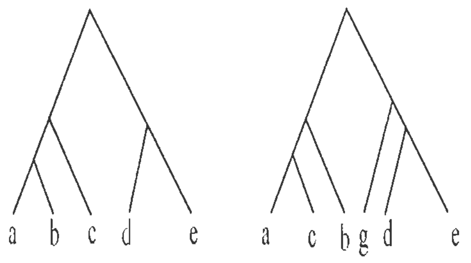


Figure 6.6: Input trees for DdateSupertree

Divcompat algorithm first sorts the statements according to the number of labels in the statement. The order of the statements remains the same as the first statement has the highest number of labels and the statement 2 and 3 have the same number of labels, i.e, 2. The first statement is compared with other two statements. The first statement is "$div(d, e)$ predates $div(a, b, c)$", and according to lemma 6.1, all the subsets of $\{a, b, c\}$ are also the predated by $div(d, e)$. But the second statement indicates that subset of $\{a, b, c\}$, i.e. $div(a, b)$, predates $div(d, e)$, which leads to a conflict and the conflict flag of second statement is incremented by one. There is no conflict between the first statement and the third statement. In second iteration, the second statement is compared with the third statement, and both are compatible with each other. Finally the statements with maximum conflicts are removed. In this case, the second statement is a victim as it is involved in highest number of conflicts. The modified list of statements are:

1. "$div(d, e)$ predates $div(a, b, c)$"

2. "$div(a, c)$ predates $div(a, b)$"

As a second step in DdateSupertree algorithm, the common leaf nodes and their restrictions are applied to Adams consensus tree algorithm. The set of common leaf nodes for the trees shown in Figure 6.6 is $\{a, b, c, d, e\}$, and the restrictions on this common set to input trees are shown in Figure 6.7.

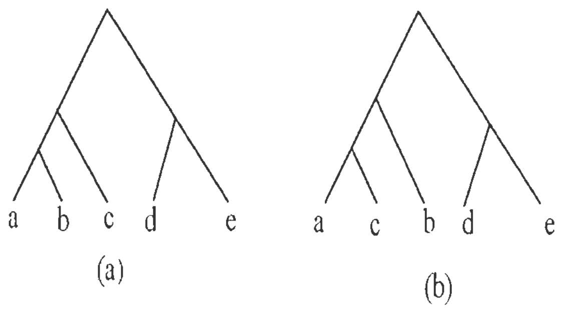


Figure 6.7: Restrictions of input trees, shown in Figure 6.6, on $\{a, b, c, d, e\}$.

The input tree collection shows the conflict due to the clusters involving species c. The clusters in tree, shown in Figure 6.7(a) are, $\{a, b\}$,$\{a, b, c\}$,$\{d, e\}$ and $\{a, b, c, d, e\}$, and the clusters in tree, shown in Figure 6.7(b), are $\{a, c\}$,$\{a, b, c\}$,$\{d, e\}$ and $\{a, b, c, d, e\}$. The two clusters $\{a, b\}$ and $\{d, e\}$ in the input collection lead to a conflict. The Adams consensus tree and the final supertree for the given input collection is shown in Figure 6.8 and Figure 6.9 respectively.

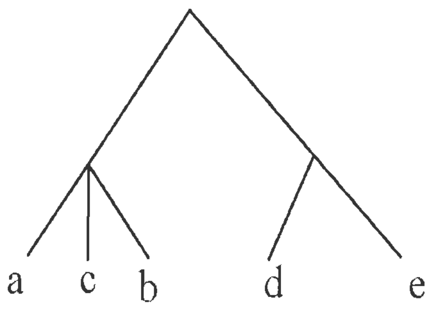


Figure 6.8: Adams consensus tree for the input tree collection shown in Figure 6.7.

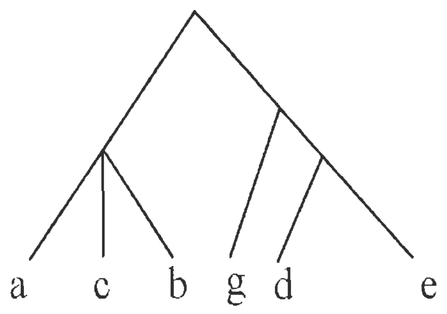


Figure 6.9: Final supertree for input tree collection

Once the final supertree is obtained the refined divergence date statements are used to rank the internal nodes of the resulting supertree. The final supertree for the input trees, given in Figure 6.6, is shown in Figure 6.9. The supertree shown in Figure 6.8, represents polytomy at the divergence node of a, b, and c due to the conflict in the input tree collection. Most of these polytomies are resolved by incorporating the divergence date statements and ranking the internal nodes in the supertree. The first divergence date statement "$div(d, e)$ predates $div(a, b, c)$" indicates that the rank of $MRC(d, e)$ is less than $MRC(a, b, c)$. Therefore, the rank of the $MRC(d, e, g)$ is always less than $MRC(d, e)$, according to lemma 6.3. It implies that the rank of $MRC(a, b, c)$ is greater than $MRC(d, e, g)$. The second statement, "$div(a, c)$ predates $div(a, b)$", indicates that the rank of $MRC(a, c)$ is less than that of $MRC(a, b)$. The final supertree after incorporating the divergence date information is shown in Figure 6.10.

6.5 Conclusion

In this chapter, we proposed an algorithm for incorporating additional information in supertree construction, which is useful in resolving polytomies and results

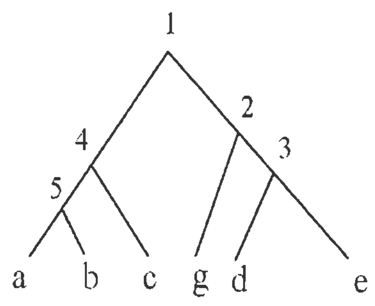


Figure 6.10: Final ranked supertree after incorporating the divergence date statements.

in more accurate and refined tree. We identified and proposed a suitable method to resolve additional conflicts such as, conflict in divergence date information, conflict between tree topology and divergence information. The proposed algorithm satisfies all the desirable properties of the supertree algorithms. DdateSupertree accomplishes the task of incorporating the divergence date information and resolves the conflicts without additional computational complexity. Both RankedTree and DdateSupertree computes a supertree in $O(kn^3)$ time, where k is the number of trees in input collection and n is the number of unique leaf nodes in the input trees. DdateSupertree shows superiority in terms of preserving all the nestings, as proved in theorem 6.2, and robustness to divergence date statement conflicts and the conflicts between divergence date and input tree topology.

Chapter 7

Conclusion and Future Work

7.1 Contributions of the thesis

The contributions made in this thesis are as follows:

- A phylogenetic network reconstruction algorithm with constrained recombination was presented in this thesis. In the previous available algorithms there is information loss due to conversion of the molecular sequences into distances. In order to avoid this, our algorithm utilizes both similarity and dissimilarity measures for the construction of the network. This gives a nearest visualization of the data to the actual representation. We proved that our algorithm results in minimum number of galls in the resulting network. Our special contribution lies in proposing a pattern recognition based node classification algorithm that considers the pattern in the similarity and dissimilarity matrices and classifies into three types: Null, Mutation, and Recombination.

  The time required to compute the phylogenetic network by popular algorithms such as SplitTree is $O(n^5)$,T-Rex is $O(n^4)$ and NeighborNet is

$O(n^3)$, where n is the number of node. On the other hand, the reconstruction algorithm proposed in this thesis computes the phylogentic network in $O(n^2)$ time. This is very significant improvement with large 'n' values.

- The amalgamation of a collection of the input trees into a single output tree known as supertree is an important task in various areas of classification, particularly evolutionary biology. We solve the two problems of supertree contruction algorithm by proposing a reasonable criterion by which to combine the input trees, and designing a polynomial time algorithm to carry this out. The least common ancestor algorithm is used for the detection of the least common ancestors of different species. The new technique for the supertree construction for rooted trees proposed by us exhibits all the desirable properties, such as preserving the branch information of the input trees, invariant to change of order of trees and so on. Our approach is based on a variant of standard UPGMA algorithm that employs a new distance measure to compute the supertree in polynomial time.

  It should be noted that the most of the supertree methods do not return a supertree for incompatible input trees. The supertree method presented in this thesis always returns a single supertree. In addition, other supertree methods generally depend on the Maximal Agreement and they remove the edges which represents conflicting information resulting in removal of certain leaf nodes. However, our method preserves all the leaf nodes. The supertree methods were evaluated on the basic properties such as, time complexity, number of trees returned, polytomy, optimization criteria and results of the methods when the input trees carry evolutionary conflicts. The comparison shows that our supertree construction algorithms outperforms all the existing methods.

- Most phylogenetic supertree construction methods utilize typically only the discrete topology of the input trees and ignore other information that might be available. In this thesis, we proposed an algorithm for incorporating additional information in supertree construction, which is useful in resolving polytomies and results in more accurate and refined tree. Till date only RankedTree [3] is a published supertree algorithm with the capability to incorporate divergence date information for supertree construction. It is an extension of BUILD [4], which could not process incompatible collection of input trees. Moreover, the supertree constructed with [4] or its variants do not represent the nestings present in all the input trees [2]. In contrast, our algorithm DdateSupertree is an extension of Adams consensus tree, which represents all the nestings represented by every input tree [5], and so does the DdateSupertree.

We identified and proposed a suitable method to resolve additional conflicts such as, conflict in divergence date information, conflict between tree topology and divergence information. The proposed algorithm satisfies all the desirable properties of the supertree algorithms. DdateSupertree accomplishes the task of incorporating the divergence date information and resolves the conflicts without additional computational complexity. Both RankedTree and DdateSupertree computes a supertree in polynomial $O(kn^3)$ time, where k is the number of trees in input collection and n is the number of unique leaf nodes in the input trees. DdateSupertree shows superiority in terms of preserving all the nestings, and robustness to divergence date statement conflicts and the conflicts between divergence date and input tree topology.

7.2 Future Work

There are a number of research issues which spring up from our work. We have worked on resolving several crucial problems of phylogenetic analysis. There are several other phylogenetic and other bioinformatics problems that could be related to the contributions in this thesis. They are as follows:

1. The reconstruction of phylogenetic network involving additional data such as ancestral divergence time.
2. The pattern recognition based algorithm proposed in thesis can be extended to implement similarity and dissimilarity measures for the additional data.
3. The supertree construction algorithms can be further modified to work with the unrooted input trees with some internal nodes labelled, than just rooted input trees. This will require developing more generalized algorithm to incorporate the labelled internal nodes.
4. An important and practical consideration will be to incorporate the weights of different attributes in the phylogenetic network reconstruction and supertree reconstruction algorithm. The weight is assigned based on the reliability of the feature, physical significance of feature, etc. The incorporation of weights will yield supertrees and phylogenetic networks that are closer to real-life .
5. Methods can be developed to find the divergence time between the species using their DNA sequences.
6. The methods can be extended to find the phylogenetic diversity for the supertrees constructed using divergence date statements and higher level taxonomic information.

7. Everyday large number of phylogenetic trees are constructed, analyzed and published. Efficient methods of storing and retrieving can also be developed as an extension to this work.

Bibliography

[1] V. B. Bajic, V. Brusic, J. Li, S. Ng, and L. Wong. From informatics to bioinformatics. *In the proceedings of First Asia-Pacific Bioinformatics Conference*, pages 3–12, 2003.

[2] D. Posada and K. Crandall. Intraspecific gene genealogies: trees grafting into networks. *Trends in Ecology and Evolution*, 16:37–45, 2001.

[3] V. Makarenkov and P. Legendre. From a phylogenetic tree to a reticulated network. *J. Comput. Biol.*, 11:195–212, 2004.

[4] D. Bryant and V. Moulton. Neighbor-Net: an agglomerative method for the construction of phylogenetic networks. *Mol. Biol. Evol.*, 21:255–265, 2004.

[5] J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.*, 98:185–200, 1990.

[6] D. H. Huson. SplitsTree: A program for analyzing and visualizing evolutionary data. *Bioinformatics*, 141:68–73, 1998.

[7] D. Gusfield, E. Satish, and C. Langley. Optimal efficient reconstruction of phylogenetic network with constrained recombination. *Journal of bioinformatics and computational biology*, 2:173–213, 2004.

[8] M. A. H. Zahid, A. Mittal, and R. C. Joshi. A pattern recognition based approach for phylogenetic network construction with constrained recombination. *Pattern Recognition*, 39:2312–2322, 2006.

[9] R. R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull.*, 28:1409–1438, 1958.

[10] M. A. H. Zahid, A. Mittal, and R. C. Joshi. Least common ancestor based efficient method for constructing rooted supertrees. *Journal of Bioinformatics and Biomedical Engineering*, 1:1–6, 2005.

[11] M. A. H. Zahid, A. Mittal, and R. C. Joshi. A heuristic algorithm for optimal agreement supertrees. *In the proceedings of International Conference on Systemics, Cybernetics and Informatics (ICSCI 05), Hyderabad, India*, pages 595–599, 2005.

[12] D. Bryant, C. Semple, and M. Steel. Supertree methods for ancestral time divergence date and other applications. In O. R. P. Bininda-emonds, editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 129–150. Kluwer, 2004.

[13] M. A. H. Zahid, A. Mittal, and R. C. Joshi. A supertree method for combining rooted phylogenetic trees with ancestral divergence time. *In the proceedings of 12th International Conference on BioMedical Engineering, IFMBE Proceedings, Singapore*, 12:626:1–4, 2005.

[14] A. Khedr and R. Bhatnagar. Decomposable algorithms for minimum spanning tree. *Proceedings of the International Workshop on Distributed Computing, Springer Verlag Notes on Computer Science*, pages 33–44, 2003.

[15] M. A. H. Zahid, A. Mittal, and R. C. Joshi. An optimization based approach for combining semi-labelled rooted phylogenetic trees. *In the proceedings of 12th International Conference on BioMedical Engineering, IFMBE Proceedings, Singapore*, pages 624:1–4, 2005.

[16] T. A. Brown and K. A. Brown. Using molecular biology to explore the past. *Bioassays*, 16:719–726, 1994.

[17] http://www.ars.usda.gov/research/projects. *Last accessed 29th October 2006.*

[18] J. Felsenstein. Phylogenies and the comparative method. *Amer. Nat.*, 125:1–15, 1985.

[19] D. A. Liberles, D. R. Schreiber, S. Govindarajan, S. G. Chamberlin, and S. A. Benner. The adaptive evolution database (TAED). *Genome Biol.*, 2, 2001.

[20] W. Maddison. A method for testing the correlated evolution of two binary characters: are gains or losses concentrated on certain branches of a phylogenetic tree? *Evol.*, 44:304–314, 1990.

[21] S. B. Carroll, J. K. Grenier, and S. D. Weatherbee. From DNA to diversity. *Blackwell Science, 2001.*

[22] M. Y. Galperin and E. V. Koonin. Comparative genome analysis. *Methods Biochem. Anal.*, 43:359–392, 2001.

[23] X. Gu. Maximum-likelihood approach for gene family evolution under functional divergence. *Mol. Biol. Evol.*, 18:453–464, 2001.

[24] J. A. Eisen. Phylogenomics: Improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Research*, 8:163–167, 1998.

[25] F. Pazos and A. Valencia. Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Engineering*, 14:609–614, 2001.

[26] J. K. Chambers et al. A G protein-coupled receptor for UDP-glucose. *J. Biol. Chem.*, 275:10767–10771, 2000.

[27] J. R. Brown and P. V. Warren. Antibiotic discovery: Is it in the genes? *Drug Discovery Today*, 3:564–566, 1998.

[28] R. Overbeek et al. WIT: integrated system for high throughput genome sequence analysis and metabolicre construction. *Nucleic Acids Research*, 28:123–125, 2000.

[29] F. G. Candelas, M. A. Bracho, and A. Moya. Molecular epidemiology and forensic genetics: Application to a hepatitis c virus transmission event at a hemodialysis unit. *Journal of Infectious Diseases*, 187:352–358, 2003.

[30] B. Kolaczkowski et al. Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature*, 431:980, 2004.

[31] B. Snel. Genome trees and the nature of genome evolution. *Annual Review of Microbiology*, 59:191, 2004.

[32] P. Legendre and V. Makarenkov. Reconstruction of biogeographic and evolutionary networks using reticulograms. *Sys. Biol.*, 51:199–216, 2002.

[33] J. Louwagie, F. E. McCutchan, M. Peeters, and T. P. Brennan. Phylogenetic analysis of gag genes from 70 international HIV-1 isolates provides evidence for multiple genotypes. *AIDS*, 7:769–780, 1993.

[34] D. L. Robertson, B. H. Hahn, and P. M. Sharp. Recombination in AIDS viruses. *Journal of Molecular Evoluation*, 40:249–259, 1995.

[35] Chag-Won Lee et al. Effect of vaccine use in the evolution of mexican lineage h5n2 avian influenza virus. *Journal of Virology*, 78:8372, 2004.

[36] W. F. Doolittle. Phylogenetic classification and the universal tree. *Science*, 284:2124–2129, 1999.

[37] H-J. Bandelt and A. W. M. Dress. A canonical decomposition theory for metrics on a finite set. *Adv Math.*, 92:47, 1992.

[38] J. P. Huelsenbeck and J. J. Bull. A likelihood ratio test to detect conflicting phylogenetic signal. *Systematic Biology*, 45:92, 1996.

[39] J. Finnerty S. B. Carroll A. Rokas, N. King. Conflicting phylogenetic signals at the base of the metazoan tree evolution. *Evolution & Development*, 5:346, 2003.

[40] K. Strimmer, C. Wiuf, and V. Moulton. Recombination analysis using directed graphical models. *Mol.Biol. Evol.*, 18:97–99, 2001.

[41] M. T. Hallet and J. lagergren. Efficeint algorithms for lateral gene transfer problems. *In proceedings of 5th international conference on computational molecular biology (RECOMB01) New York*, pages 149–156, 2001.

[42] C. R. Linder, B. M. E. Moret, L. Nakhleh, and T. warnow. Reconstructing networks part II: computational aspects. *Tutorial presented at the ninth pacific symposium on Biocomputing*, 2004.

[43] H. J. Bandelt and A. W. M. Dress. Split decomposition: A new and useful approach to phylogenetic analysis of distance data. *Mol. Phylo. Evol.*, 1:242--252, 1992.

[44] V. Makarenkov. T-Rex: reconstructing and visualizing phylogenetic trees and reticulation networks. *Bioinformatics*, 17:664–668, 2001.

[45] H. J. Bandelt and A. W. M. Dress. A relational approach to split decomposition. In B. Lsusen O. Opitz and R. Kalar, editors, *Information and classification*, pages 123–131. Springier Berlin, 1993.

[46] N. Saitou and M. Nei. The Neighbor-Joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology Evolution*, 4:406–425, 1987.

[47] W. M. Fitch. Networks and viral evolution. *J Mol Evol.*, 44:65–75, 1997.

[48] H. J. Bandelt, P. Forster, B.C. Sykes, and M.B. Richards. Mitochondrial portraits of human populations using median networks. *Genetics*, 141:743–753, 1995.

[49] H. J. Bandelt, V. Macaulay, and M. Richards. Median networks: speedy construction and greedy reduction, one simulation, and two case studies from human mtDNA. *Mol Phylo. Evol.*, 16:8--28, 2000.

[50] L. Excoffier and P.E. Smouse. Using allele frequencies and geographic subdivision to reconstruct gene trees within a species: molecular variance parsimony. *Genetics*, 136:343359, 1994.

[51] M. Eigen. Statistical geometry in sequence space: a method of quantitative sequence analysis. *Proc. Natl. Acad. Sci. U. S. A.*, 85:5913–5917, 1988.

[52] L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8:69–78, 2001.

[53] C. Semple and M. Steel. A supertree method for rooted trees. *Discrete Appl. Math. 105*, pages 147–148, 2000.

[54] M. Steel, A. W. M. Dress, and S. Bocker. Simple but fundamental limitation on supertrees and consensus tree methods. *Sys. Biol.*, 49:363–368, 2000.

[55] F. R. McMorris. Axioms for consensus functions on undirected phylogenetic trees. *Math. Bioscie.*, 74:17–21, 1986.

[56] S. Bocker, A. W. M. Dress, and M. Steel. Patching up X-trees. *Ann. Combin.*, 3:1–12, 1999.

[57] F. R. McMorris, D. B. Meronk, and D. A. Neumann. A view of some consensus methods for trees. In J. Felsenstein, editor, *Numerical Taxonomy*, pages 122–125. Springer-Verlag, 1983.

[58] D. L. Swofford. When are phylogeny estimates from molecular and morphological data incongruent? In M. M. Miyamoto and J. Cracraft, editors, *Phylogenetic analysis of DNA sequences*, pages 295–333. Oxford University Press, 1991.

[59] J. P. Barthelemy and F. R. McMorris. The median procedure for n-trees. *J. Classif.*, 3:329–334, 1986.

[60] K. Bremer. Combinable component consensus. *Cladistics*, 6:369–372, 1990.

[61] A. de Queiroz. For consensus (sometimes). *Syst. Biol.*, 42:368–372, 1993.

[62] E. N. Adams. N-trees as nesting: complexity, similarity, and consensus. *J. Classif.*, 3:299–317, 1986.

[63] A. V. Aho, T. G. Sagiv, T. G. Szymanski, and J. D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput.*, 10:405–4021, 1981.

[64] B. R. Baum. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability for combining phylogenetic trees. *Taxon*, 41:3–10, 1992.

[65] M. A. Ragan. Phylogenetic inference based on matrix representation of trees. *Mol. Phylo Evol*, 1:53–58, 1992.

[66] F.-J. Lapointe and C. Levasseur. Everything you always wanted to know about the average consensus, and more. In *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 87–105. Kluwer Academic, 2004.

[67] P. Buneman. The recovery of trees from measures of dissimilarity. In *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.

[68] V. Berry and D. Bryant. Faster reliable phylogenetic analysis. *In the proceedings of 3rd international conference on computational molecular biology*, 3:59–68, 1999.

[69] R. D. M. Page. Modified mincut supertrees. *Lecture Notes in Computer Science, LNCS-2452*, pages 537–551, 2002.

[70] P. Daniel and C. Semple. A supertree algorithm for nested taxa. In *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 151–171. Kluwer Academic, 2004.

[71] J. S. Farris. On comparing the shapes of taxonomic trees. *Syst. Zool*, 22:50–54, 1973.

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

[73] O. R. P. Bininda-Emonds and H. N. Bryant. Properties of matrix representation with parsimony analyses. *Syst. Biol.*, 47:497–508, 1998.

[74] A. Purvis. A modification to Baum and Ragan's method for combining phylogenetic trees. *Syst. Biol.*, 44:251–255, 1995.

[75] C. J. Creevey and J. O. McInerney. Clann: investigating phylogenetic information through supertree analyses. *Bioinformatics*, 21:390–392, 2004.

[76] D. Chen, L. Diao, O. Eulenstein, D. Fenndez-Baca, and M. J. Sanderson. Flipping: A supertree construction method. In *Bioconsensus*, pages 135–160. DIMACS series in discrete mathematics and theoretical computer science, American Mathematical Society, Providence, 2003.

[77] O. Eulenstein, D. Chen, J. G. Burleigh, D. Fernandez-Baca, and M. J. Sanderson. Performance of flip-supertree construction with a heuristic algorithm. *Syst. Biol.*, 53:299–308, 2004.

[78] A. W. Liew, H. Yan, and M. Yang. Pattern recognition techniques for the emerging field of bioinformatics: a review. *Journal of Pattern Recognition.*, 38:2055–2073, 2005.

[79] S. V. Rao and A. Mukhopadhyay. Fast algorithms for computing beta-skeleton and their relatives. *Pattern Recognition*, 34:2163–2172, 2001.

[80] K. C. Gowda and T. V. Ravi. Divisive clustering of symbolic objects using the concept of both similarity and dissimilarity. *Journal of pattern recognition*, 28:1277–1282, 1995.

[81] J. Hein. A heuristic method to construct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36:396–405, 1993.

[82] L. R. David, P. M. Sharp, F. E. McCutchan, and B. H. Hahn. Recombination in HIV-1. *Nature*, 374:124 - 126, 2002.

[83] M. H. Schierup and J. Hein. Consequences of recombination on traditional phylogenetic analysis. *Genetics*, 156:879–891, 2000.

[84] M. H. Schierup and J. Hein. Recombination and the molecular clock. *Mol. Biol. Evol.*, 17:1578–1579, 2000.

[85] D. Posada and K. Crandall. The effect of recombination on the accuracy of phylogeny estimation. *Journal of Molecular Evolution*, 54:396–402, 2002.

[86] D. Gusfield, E. Satish, and C. Langley. The fine structure of galls in phylogenetic networks. *INFORMS Journal on computing, special issue on Computational Biology*, 16:459–469, 2004.

[87] A. Chakravarthi. It's raining SNP's hallelujah? *Nature Genetics*, 19:216–866, 1998.

[88] J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.*, 17:368–376, 1981.

[89] W. M. Fitch and E. Margoliash. A non-sequential method for constructing trees and hierarchical classifications. *Journal of Molecular Evolution*, 18:30–37, 1967.

[90] D.Bryant and M. Steel. Extension operations on leaf labelled trees. *Adv. Appl. Math.*, 16:425–453, 1995.

[91] E. N. Adams. Consensus techniques and the comparison of taxonomic trees. *Syst. Zool.*, 21:390–397, 1972.

[92] S. K. Gupta and A. Singh. On tree roots of graph. *International Journal of Mathematics*, 76:157–166, 1999.

[93] J. Jansson, H. K. N. Joseph, K. Sadakane, and W. Sung. Rooted maximum agreement supertrees. In *LATIN 2004, LNCS 2976*, pages 499–508. 2004.

[94] D. M. R. Pages. Taxonomy, supertrees, and the tree of life. In *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 1–20. Kluwer Academic, 2004.

[95] P. Daniel and C. Semple. A class of general supertree methods for nested taxa. *SIAM Journal of Discrete Maths*, 19:463–480, 2005.

[96] D. A. Neumann. Faithful consensus methods for n-trees. *Math. Biosci*, 63:271–287, 1983.

[97] W. H. Piel, M. J. Donoghue, and M. J. Sanderson. TreeBASE: a database of phylogenetic knowledge catalog of life with partners. In *Research Report from the National Institute for Environmental Studies No. 171*, pages 41–47. 2002.

[98] A. D. Gordon. Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labelled leafs. *J. Classif.*, 3:31–39, 1986.

[99] D.H. Huson, S. M. Nettles, and T. Warnow. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *J. Comput. Biol.*, 6:369–386, 1999.

[100] P. A. Goloboff and D. Pol. Semi-strict supertrees. *Cladistics*, 18:514–525, 2002.

[101] F. Ronquist, J. P. Uelsenbeck, and T. Britton. Bayesian supertrees. In *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, pages 193–224. Kluwer Academic, 2004.

[102] O. Bininda-emonds. The evolution of supertrees. *Trends in Ecol. and Evol.*, 19:315–322, 2004.

[103] T. Margush and F. R. McMorris. Consensus n-trees. *B. Math. Biol.*, 43:239–244, 1981.

[104] C. Semple and M. Steel. *Phylogentics.* Oxford University Press, 2003.

Author's Publications

International Journals

1. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "A Pattern Recognition Based Approach for Phylogenetic Network Construction with Constrained Recombination", Journal of Pattern Recognition, Elsevier, 39:2312–2322, 2006.

2. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "Least common ancestor based efficient method for constructing rooted supertrees", Journal of Bioinformatics and Biomedical Engineering, 1:1–6, 2005.

National Journals

1. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "Use of Phylogenetic network and its reconstruction Algorithms", Journal of Bioinformatics India, ISSN 0972-7655, January-March:47–58, 2005.

International Conferences

1. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "A Classification Based Approach for Root Unknown Phylogenetic Networks under Constrained Recombination", in the proceedings of the 2nd International Conference

on Distributed Computing and Internet Technology (ICDCIT05), LNCS 3816 , ISBN: 3-540-30999-3, DOI: 10.1007/11604655, Springer, December 2005, pp. 592–603.

2. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "A supertree method for combining rooted phylogenetic trees with ancestral divergence Time", in the proceedings of 12th International Conference on BioMedical Engineering, IFMBE Proceedings, Vol. 12. Singapore: ISSN 1727-1983, ISBN: 981-05-4572-X, Singapore, December 2005, pp. 626-1 to 626-4.

3. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "An optimization based approach for combining semi-labeled rooted phylogenetic trees", in the proceedings of 12th International Conference on BioMedical Engineering, IFMBE Proceedings, Vol. 12. ISSN 1727-1983, ISBN: 981-05-4572-X, Singapore, December 2005, pp. 624-1 to 624-4.

4. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "A Pattern Matching Based Approach towards Phylogenetic Networks with Constrained Recombination", in the proceedings of the 8th International Conference on Information Technology (CIT 2005), Bhubaneswar, India, December 2005, pp. 53–58.

5. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "A Heuristic Algorithm for Optimal Agreement Supertrees construction", International Conference on Systemics, Cybernetics and Informatics (ICSCI 05), Hyderabad, India, January 2005, pp. 595–599.

6. M. A. H. Farquad and M. A. H. Zahid, "Network Representation of Phylogenetic Relationship", International Conference on Systemics, Cybernetics and Informatics (ICSCI 05), Hyderabad, India, January 2005, pp. 450–455.

International Journals (under review)

1. M. A. H. Zahid, Ankush Mittal, R. C. Joshi and G. Atluri, "CLINIQA: A Machine intelligence based clinical question answering system," IEEE Transaction on Information Technology in Biomedicine, 2006.

2. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "Combining rooted phylogenetic trees with ancestral divergence time," Journal of Medical and Biological Engineering and Computing, Springer. (*Under second review*) 2006.

3. M. A. H. Zahid, Ankush Mittal and R. C. Joshi, "An automated phylogenetic question answering system (PhyloQA)", Informatica, 2007.