2007

Algorithms in supertree inference and phylogenetic data mining

Duhong Chen
Iowa State University

Algorithms in supertree inference and phylogenetic data mining

by

Duhong Chen

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Iowa State University

Ames, Iowa

2007

DEDICATION

I would like to dedicate this dissertation to my wife Jing Wang, to my daughter Vivian F. Chen, and to my son Arthur Y. Chen without whose support I would not have been able to complete this work. I would also like to thank my friends and family for their loving guidance and assistance during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation.

First and foremost, I am honored to express my deepest gratitude to my supervisor Dr. David Fernández-Baca for his guidance, patience and support throughout this research and the writing of this dissertation. He has offered invaluable ideas, suggestions and criticisms with his profound knowledge in computer science and rich research experience. His patience and kindness are greatly appreciated. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Oliver Eulenstein, Dr. Shashi K. Gadia, Dr. Leslie Miller, and Dr. Giora Slutzki.

I wish to extend my thanks to Dr. Gordon Burleigh and Dr. Mike Sanderson for their help in research, paper writing, and invaluable suggestions.

Thanks are also due to my office colleagues, who never failed to give me great encouragement, valuable discussion and suggestions. Special thanks should go to Mukul Bansal, Wen-Chieh Chang, Jianrong Dong, and Andre Wehe.

ABSTRACT

Science and society would benefit enormously from comprehensive phylogenetic knowledge of the Tree of Life (ToL), a framework that includes above 1.7 million species on Earth. ToL shows how living things have evolved since the origins of life billions of years ago. With existing computational approaches, we cannot produce a global estimate of evolutionary history from molecular data of all these species. Therefore, the development of computer algorithms and methods is critical to the field of phyloinformatics. The goal of this dissertation is to design, analyze, and implement algorithms to solve three specific problems in the the field of phyloinformatics to support ToL studies.

The first part of this dissertation is about algorithms for the matrix representation with flipping (MRF) method to construct large subtrees of the ToL. The utility of the MRF supertree method has been limited by the speed of its heuristic algorithms. We describe a new heuristic algorithm for MRF supertree construction that improves upon the speed of the previous heuristic by a factor of n (the number of taxa in the supertree). This new heuristic makes MRF tractable for large-scale supertree analyses and allows the first comparisons of MRF with other supertree methods using large empirical data sets. Analyses of three published supertree data sets indicate that MRF supertrees are equally or more similar to the input trees on average than matrix representation with parsimony (MRP) and modified mincut supertrees. The results also show that large differences may exist between MRF and MRP supertrees, and demonstrate that the MRF supertree method is a practical and potentially more accurate alternative to the nearly ubiquitous MRP supertree method.

The second part of this dissertation is dedicated to new algorithms for partitioning phylogenetic data sets. We describe two new methods to partition phylogenetic data sets of discrete characters based on pairwise compatibility. The partitioning methods make no assumptions regarding the phylogeny, model of evolution, or characteristics of the data. The methods first build a compatibility graph, in which each node represents a character in the data set. Edges in the compatibility graph may represent strict compatibility of characters or they may be weighted based on a fractional compatibility scoring procedure that measures how close the characters are to being compatible. Given the desired number of partitions, the partitioning methods then seek to cluster together the characters with the highest average pairwise compatibility, so that characters in each cluster are more compatible with each other than they are with characters in the other cluster(s). We demonstrate that the spectral partitioning effectively identifies characters with different evolutionary histories in simulated data sets, and it is better at highlighting phylogenetic conflict within empirical data sets than previously used partitioning methods.

The third part of this dissertation describes a new intelligent search engine, named PhyloFinder, for phylogenetic databases that we have implemented using trees from TreeBASE. PhyloFinder enables taxonomic queries, in which it identifies trees in the database containing the exact name of the query taxon and/or any synonymous taxon names and provides spelling suggestions for the query when there is no match. Additionally, PhyloFinder can identify trees containing descendants or direct ancestors the query taxon by making use of the ontology guide from NCBI taxonomy database. PhyloFinder also executes phylogenetic queries, in which it identifies trees that contain the query tree or topologies that are similar to the query tree. PhyloFinder can enhance the utility of any tree database by providing tools for both taxonomic and phylogenetic queries as well as visualization tools that highlight the query results and provide links to NCBI and TBMap.

GENERAL INTRODUCTION

This dissertation is in computational biology and bioinformatics field which is an interdisciplinary area that applies the techniques of computer science, information technology, applied mathematics, and statistics to address problems in biology. There are many computational challenging questions in molecular and systematic biology. Many of those questions can be modeled as combinatorial optimization or algorithmic graph problems. The goal of this research is to design, analyze, and implement algorithms for open problems in the following specific areas:

- Improved heuristic algorithms for minimum-flip supertree construction
- Spectral clustering in partitioning phylogenetic data sets
- Design and implement an intelligent search engine for phylogenetic tree databases

The remainder of the introduction is devoted to a brief overview of phylogenetic trees and an overview of the rest of the dissertation.

Introduction

Phylogenies, i.e., evolutionary histories of groups of organisms, are essential for understanding the relationships among species. Phylogenetic knowledge plays an important role in studying the history of life on earth (Maizels and Weiner 1994), gene function (Lerat et al 2003), protein-protein interaction (Plachetzki et al 2007), discovering emerging viruses (Gonzalez et al 1996; Moonan et al 2000), understanding infectious diseases (Franks 2002) and genetics (Tamura et al 2007) etc. In this section, I will introduce some basic terminology and notation that are used in this thesis.

A *tree* is a acyclic, connected graph which consists of a set of nodes (vertices) connected by a set of branches (edges). The *degree* of a vertex equals to the number of edges to which it is connected. Let $S = \{s_1, \ldots, s_n\}$ be a taxa set. A *phylogenetic tree* T over S is a tree with exactly n leaves, each of which is labelled with a distinct element of S. Such a tree is used to represent evolutionary relationships among the taxa set S. A phylogenetic tree can be rooted or unrooted. A *rooted tree* is a tree in which one of the nodes is stipulated to be the root, from which a unique path leading to every other node. An *unrooted tree*, as could be imagined, has no pre-determined root and therefore induces no hierarchy.

Suppose $T = (V, E)$ is a tree. A vertex $v \in V$ is *internal* if its degree is greater than one; otherwise, it is a *leaf*. An edge $e = (u, v) \in E$ is internal if both u and v are internal vertices; otherwise we say e is an *external* edge. We write $\mathcal{L}(T)$ to denote the leaf set of tree T. A vertex a is a *descendent* of a vertex b if the path from a to the root passes through b. Then b is called the *ancestor* of a. For a vertex $v \in V$, the set containing all leaves that are descendants of v is called a *cluster*. The set S is always a cluster of T; every other cluster is said to be *proper*.

Phylogenetic Inference

Under the natural assumption, species with similar characters (features) are genetically related. Thus, the basic idea of phylogenetic inference is to compare specific characters of the species. Phylogenetic analysis was originally developed by systematists who wanted to reconstruct genealogical relationships of species based on morphological similarities. Classic phylogenetic inference dealt mainly with morphological features such as size, color, number of legs, etc. Modern phylogeny uses information directly comes from molecular sequences. The characters used are usually DNA or protein *sites*, where a site is a single position in the sequence after alignment with several such sequences. Since phylogenetic trees play an important role in the study of molecular evolution, phylogenetic inference is one of the fundamental tasks of computational biology.

The construction of optimal evolutionary trees is a very challenging problem. Many instances of this problem have been defined and studied and most popular phylogenetic inference methods attempt to solve $\mathcal{NP}$-hard optimization problems, such as maximum parsimony (Felsenstein 1981), maximum character compatibility (Adams 1986), and maximum likelihood (Estabrook et al 1975). Polynomial time methods, such as neighbor-joining (Saitou and Nei, 1987), also exist. Even though these problems have been studied extensively, evolutionary tree construction still remains an open problem (Korostensky and Gonnet 2000).

Supertree Inference

In evolutionary biology, supertree inferences are used to combine the information from a collection of small trees into a larger tree, while preserving the phylogenetic information among those input trees. Supertree construction has been paid increasing attention in the past few years. It is perhaps a key to reconstruct the Tree of Life (TOL) . The objective of supertree methods is to preserve information from the input trees and infer novel statements of relationships not appear in any single input tree.

A *profile* is a multiset $\mathcal{T}$ of phylogenies. The elements of $\mathcal{T}$ are called *input trees*. A *supertree* for a profile $\mathcal{T}$ is a phylogeny T such that $\mathcal{L}(T) = \bigcup_{t \in \mathcal{T}} \mathcal{L}(t)$. A *character matrix* for S is an $n \times m$ matrix $M = [a_{ij}]$ over $\{0, 1, ?\}$, whose i-th row corresponds to taxon s_i. The j-th column of M is called *character* j. Let $\mathcal{T}$ be a profile such that $\bigcup_{t \in \mathcal{T}} \mathcal{L}(t) = S$. For our study, we define a *matrix representation* of $\mathcal{T}$ as a character matrix M for S obtained as follows. For each tree $t \in \mathcal{T}$ and each cluster X in t, create a column of M whose i-th entry is 1 if $s_i \in X$, 0 if $s_i \in \mathcal{L}(t) - X$, and ? if $s_i \notin \mathcal{L}(t)$.

Since the input trees often conflict in practice, the matrix representation from a collection of incompatible input trees does not correspond to a perfect phylogeny. Matrix representation with parsimony (MRP) method seek the most parsimonious supertrees that is, supertrees with fewest number of steps based on this matrix representation. However, minimizing the number of steps on a tree in the context of a matrix representation of input trees does not have quite this obvious justification. Homoplasy on a supertree derived via the MRP problem represents conflicts between clusters rather that between individual evolutionarily novel character states, such as a substitution in a DNA sequence. Thus, this type of homoplasy is somewhat removed from phylogenic evidence. Matrix representation with flip (MRF) method on the other hand is to determine the minimum number of flips between 0 and 1 that are required to turn the matrix representation of input trees into one that corresponds to a perfect phylogeny.

MRF solves the conflicts among input trees by maximally preserving phylogenetic information.

Thesis Organization

The thesis consists of 5 chapters. Chapter 2 describes an efficient heuristics algorithm for minimum-flip supertree reconstruction. This new heuristic makes minimum-flip supertree method tractable for large-scale supertree analyses. Chapter 3 describes two new methods to partition phylogenetic data sets of discrete characters based on pairwise compatibility. Chapter 4 describes an intelligent search engine for phylogenetic tree database. General conclusions are given in chapter 5.

Literature Review

Minimum-Flip Supertree

Most versions of the supertree problem are consensus tree problems, which apply only to phylogenies with the same taxon set (Kannan et al 1995; Kearney et al 1999; Bryant et al 2000). Only a few methods apply to phylogenies that do not necessarily share all taxa. In the absence of incompatibility among the input trees, a supertree can be built in polynomial time using the algorithms of Aho et al (1981) and Henzinger et al (1996).

In practice, the main obstacle to fast and accurate supertree construction is that input trees invariably conflict with one another, that is, errors are present. Semple and Steel (2000) modified Aho et al (1981) algorithm to handle incompatible input. Their procedure, the MinCutSupertree algorithm (MC), resolves conflicts that prevent the Aho et al. algorithm from proceeding by deleting a minimum amount of information. The approach is thus guided by a local optimization criterion, although some global properties can be shown for the output tree. More recently, an improved version of MC was proposed (Page 2002) that retains more of the uncontradicted information of the input trees and still runs in polynomial time.

For biologists, the most popular supertree method is MRP, which seeks the most parsimonious tree(s) for the matrix representation of the input trees (Baum, 1992; Ragan, 1992; Baum and Ragan, 2004). While computing a most parsimonious tree is $\mathcal{NP}$-complete (Graham and Foulds, 1982), the wide availability of reasonably fast maximum-parsimony software (eg,Nixon 1999; Goloboff 2000) has made MRP the clear choice of phylogeneticists working with real data. However, as mentioned above, although MRP is operationally similar to maximum parsimony, it cannot rest on the same kind of rationale in terms of minimizing homoplasy. Homoplasy on a supertree derived via MRP represents conflicts between clades rather than between individual evolutionarily novel character states and, therefore, although popular, MRP is, in no strong sense, any better justified than other supertree methods.

We have studied a new approach, called minimum-flip supertree method (Chen et al 2002, 2003, 2006), to construct supertrees that addresses error directly, through a notion of error correction, rather than by the analogy that parsimony inference has provided in the justification for other supertree methods, such as MRP. Our approach starts from the same matrix representation of the input as is used in MRP, but it proceeds with a different rationale. The matrix representation treats the input as a collection of incomplete binary characters in which each character represents a cluster (clade) from one of the input trees. Taxa present in the cluster are scored 1, those absent are scored 0, and those not sampled on that input tree are scored by a ?. One notion of error is the presence of an incorrect taxon in a cluster or the absence of one that should be present. Thus, errors correspond to flips from $0 \rightarrow 1$ or $1 \rightarrow 0$; these flips prevent the matrix from representing any phylogenetic tree perfectly. A natural optimization problem is to find the minimum number of flips that converts the matrix into one that does represent a phylogenetic tree; we call this the minimum-flip problem and the resulting trees minimum-flip supertrees. We showed that the minimum-flip problem is $\mathcal{NP}$-complete (Chen et al, 2006), even when restrictions are placed, such as allowing flips in just one direction or only permitting input trees with the same taxon set. On the positive side, we have proved that, when the input trees have identical taxon sets, the problem is fixed-parameter tractable, fixed-ratio approximable, and that an approximation scheme exists if the objective is to maximize the similarity between the clusters in the input tree and the clusters of the supertree.

Another relative of the minimum-flip problem is the fractional character compatibility problem (FCC) of Kearney et al (1999). Given a set of partitions of a taxon set, FCC aims to find an unrooted tree T such that the total similarity between the partitions and T is maximized. Although FCC was not intended to be used as supertree construction method, it can be viewed as the flip problem for unrooted phylogenies over the same taxon set. Kearney et al (1999) have shown that FCC is $\mathcal{NP}$-complete, but have given an approximation scheme for a special case.

In chapter 2 of this dissertation, We describe improvements to existing MRF heuristics that increase the speed of the heuristics by a factor of n, where n is the total number of taxa represented in the input trees. In comparisons of the performance of MRF with other supertree methods using empirical data sets, we demonstrate that the MRF supertree method is a practical and potentially more accurate alternative to the nearly ubiquitous MRP supertree method.

Spectral clustering of phylogenetic data set

The presence of conflict with a data set can greatly complicate phylogenetic tree reconstruction (e.g., Bull et al 1993; Chippindale and Wiens 1994; De Queiroz et al 1995; Farris et al 1994; Huelsenbeck et al 1996; Cunningham 1997; Thornton et al 2000; Wilgenbusch and De Queiroz 2000; Brandley et al 2005). Phylogenetic conflict among sites may be due to heterogeneity in the process of molecular evolution, such as the rates of evolution (Rokas et al, 2003). Alternately, characters in a data set may have different evolutionary histories due to events such as hybridization, lineage sorting, or horizontal transfer (e.g., Doyle et al 2003; Takahashi et al 2001; Lerat et al 2003). To understand variation in the processes of evolution within a data set and to effectively account for conflict in phylogenetic analyses, we first must identify and describe the conflict.

Perhaps the most common strategies to partition phylogenetic data sets and reveal phylogenetic conflict utilize known characteristics of the data. For molecular data, examples include partitioning by locus, codon position, coding and non-coding regions, or stem and loop regions (Brandley et al, 2005). These partitions require a priori knowledge regarding the structure of the data with the expectation that these partitions will correspond to notable heterogeneous patterns of evolution within the data set. These ad hoc partitioning methods may miss unexpected patterns of heterogeneity within the data, and they will not necessarily represent the most significant conflicts within the data set. Another class of methods seeks to partition sites based on their rate of evolution, often for the purpose of excluding fast-evolving sites from phylogenetic analyses (Brinkman and Philippe 1999; Hirt et al 1999; Burleigh and Mathews 2004; Pisani 2004) rather than conflicting phylogenetic signals. Furthermore, the methods of Brinkman and Philippe (1999) and Burleigh and Mathews (2004) require some knowledge of the phylogeny, and Burleigh and Mathews' (2004) method also assumes a model of evolution.

Spectral methods is known as effective methods for data clustering (Azar et al, 2001), image segmentation (Shi and Malik, 2000), speech processing (Bach and Jordan, 2006), information retrieval (Cheng et al, 2005), and dimension reduction (Brand and Huang, 2003; Higgs et al, 2006). The spectral clustering methods are actually a class of closely related techniques. These methods usually rely on the eigenstructure of a similarity matrix to partition points into disjoint clusters, with points in the same cluster having high similarity and points in different clusters having low similarity.

Spectral clustering methods origin from spectral graph partitioning (Donath and Hoffman, 1973; Fiedler, 1973), which is based on eigenvectors of adjacency matrix. Fiedler (1973) found that a graph is closely related to the the second eigenvector of the Laplacian of the graph adjacency (pairwise similarity) matrix, and he suggested to use this eigenvector to bi-split the graph. A popular spectral clustering algorithm has been applied in high performance computing (Pothen et al, 1990) which led to ratio-cut clustering (Hagen and Kahng, 1992; Chan et al, 1993). Shi and Malik (2000) proposed normalized-cut and used the eigenvector of the normalized Laplacian (Chung, 1997) and brought renewed interest in the topic. A large number of papers has subsequently published with various extensions, theoretical results, and new applications of spectral clustering. In computational biology and bioinformatics field, spectral clustering has been successfully applied to gene selection (Wolf et al, 2005) and biclustering of microarray data (Cheng and Church, 2000; Kluger et al, 2003; Higham et al, 2006), which simultaneous cluster both genes and conditions, to knowledge discovery from expression data. Paccanaro et al (2003) used spectral methods to cluster protein sequences into functionally similar families, and he also showed how this method is often able to identify correctly the superfamilies to which the sequences belong.

In chapter 3 of this dissertation, we describe a fast method to partition phylogenetic data sets that effectively highlights phylogenetic conflicts among sites within a data set. Most spectral clustering algorithms explicitly or implicitly assume a metric or a similarity structure over the space of configurations, which is then used by clustering algorithms. The success of such algorithms depends heavily on the choice of the metric. We describe a graph theoretic spectral partitioning method for discrete phylogenetic data that partitions the data based on character compatibility metric. This first requires building a compatibility graph for the data, in which each character is a node and the edges or edge weights are based on character compatibility. Next, given a specified number of partitions, the spectral analysis method estimates the partitions with the highest pairwise compatibility. We demonstrate that the spectral partitioning method effectively highlights phylogenetic conflict in simulated data sets, and it identifies greater phylogenetic conflict than other partitioning methods in selected empirical data sets.

Phylogenetic databases and phylogenetic tree search

In recent years there has been a rapidly expanding wealth of new phylogenetic information from across the tree of life (Sanderson et al 1993). This data provide unprecedented opportunities for large-scale evolutionary studies and for studying an array of biological problems in a phylogenetic context. However, the utility of the phylogenetic information is determined by its accessibility, and currently, much of the phylogenetic data is not easily accessible to biologists (Sanderson et al 1993; Piel et al 2003; Page 2007b, 2005a; Nakhleh et al 2003). Thus, the storage and retrieval of phylogenetic data are critically important challenges for bioinformatics.

TreeBASE is the largest relational database of published phylogenetic information, storing more than 4,400 trees that include over 75,000 taxa (TreeBASE 2007; Piel et al 2002). Data from TreeBASE include phylogenetic trees and the corresponding data matrices and meta-data, such as bibliographic information and details of the phylogenetic analyses. By assembling and storing phylogenetic information across the tree of life, TreeBASE provides an extremely beneficial service to the biological community, and the information within TreeBASE is potentially valuable for both synthesizing phylogenetic data and studying phylogenetic methods. Yet there is a lack of tools to fully access and exploit stored phylogenetic information in TreeBASE.

The essential information of a phylogeny includes identity of each node, topology of hierarchically nested nodes, and branch length of each edge. Storing phylogenies with all relationships between its nodes in a relational database is a challenging task. XML, a standard tree structure data model, looks perfectly suitable for storing phylogenetic tree. However, in comparison to most XML documents, phylogenetic trees are normally larger and deeper in topology, and the type of queries are structure based rather than path-oriented. Thus, the data management strategies developed for XML are not suitable for phylogenetic trees (Zheng et al, 2006).

The phylogenetic trees in TreeBASE are stored as a plain text in Newick format (Felsenstein, 2007). One drawback of using the Newick format is that the database cannot directly support queries related to the structure of phylogeny. Nakhleh et al (2003) showed that many phylogenetic queries were best supported by a normalized model where phylogenies are stored as lists of edges. They also demonstrated that many phylogenetic queries require transitive traversals of phylogenies, which could be conveniently constructed by Datalog programs. However, transitive queries are not supported by many relational database management system (DBMS) and the structure query such as least common ancestor (LCA) involves transitive closure operations in DBMS which can be very expensive for large data sets (Shi et al, 2000).

Fortunately, there is an alternative SQL technique, tree encoding (Tropashko, 2005), for querying tree structure without recursion. There are two types of tree encoding methods: *materialized path* and *nested-set representation*. In the materialized path method, each tree node is encoded to a path from itself to the root. That is, each record stores as string that encodes the whole path to the root. In this representation, each node knows everything about all other ancestors and descendants. This is easiest to query: get all the descendants, simply look for all the nodes that share the same prefix, and all ancestors are implied. However, querying trees with materialized path requires string parsing which is not efficient and elegant. Davidson et al (2005) used a *Dewey numbering scheme* (Vesper, 2007) which is similar to materialized path for representing phylogenetic trees. In the nested-set representation proposed by Celko (2004), each node requires two additional attributes: enter number and exit number. These two numbers are obtained from the preorder traversal of the tree: starting with the root node, each node is visited twice. When a node is entered or exited during the traversal, the sequence number of all visits is saved in the current nodes enter number and exit number. In this model, the hierarchical information is implied by the nested integer intervals.

The current TreeBASE provides keyword-based queries on attributes (e.g. query on a tree ID, the name of a taxon) to search the database of trees. Shan et al (2002) and Wang et al (2003, 2005) proposed methods to extend TreeBASE to support structure-based queries (e.g. finding a particular tree pattern, nearest neighbor of a tree) by parsing the Newick format of phylogenetic tree from TreeBASE. Nakhleh et al (2003) and Page (2005a) discussed the requirements of a phylogenetic tree database and some phylogenetic meaningful queries. Currently TreeBASE II (CIPRES, 2007) is under development, which supports structure-based queries by storing the tree structure explicitly using the technique of Nakhleh et al (2003).

However, these tools are still not good enough for retrieving information from TreeBASE. Page (2006) has written in his blog that "The current TreeBASE is a black hole – data disappears in and is difficult to extract again" . The complexity of taxonomy presents a major challenge for accessing phylogenetic data (Page 2005a,b, 2007a,b). Taxonomic names used in stored phylogenetic trees may be based on different and inconsistent taxonomies (Page 2007a). Consequently, species may be represented in multiple trees by different but equivalent names. Taxonomic queries are further complicated by misspellings or unique subspecies designations in stored trees. Many of these issues have been addressed by TBMap, a database that maps names of taxa found in TreeBASE to other taxonomic databases and clusters equivalent taxonomic names (Page 2007a). However, TBMap is not currently incorporated into any phylogenetic search engines.

The hierarchical nature of taxonomy presents further challenges for accessing phylogenetic data. The leaves in phylogenetic trees may represent different taxonomic levels, such as families, genera, species, or even subspecies. Ideally, a taxonomic query to a tree database could identify not only the specific taxon name used in the query, but also any taxonomic descendants of the query name (Page 2005a, 2007b). This option is also currently not available in TreeBASE.

Another important feature of a phylogenetic search engine is the ability to make phylogenetic queries in which a specified tree is compared to trees in the database (Page 2007b, 2005a; Nakhleh et al 2003). For example, with pattern-matching query, would identify all trees that contain or agree with a query tree, or the trees in the database in which the query tree is embedded (Page 2007b, 2005a). Additionally, since there is often much disagreement among trees, it is useful to identify all the trees that are similar, but not necessarily identical, to a query tree (Shan et al 2002; Wang et al 2003, 2005).

LCA computations are essential for many structural queries on phylogenetic trees. The LCA problem has been well studied in algorithms literature (e.g. Harel and Tarjan 1984; Cole and Hariharan 1999; Bender and Farach-Colton 2000; Bender et al 2001). Bender and Farach-Colton (2000) gives a linear preprocessing time, linear space, and constant query time algorithm to answer LCA. However, while this algorithm is good for processing trees in main memory, it is not well suitable for the database context.

In chapter 4 of this dissertation, we describe PhyloFinder, a new intelligent search engine for phylogenetic databases that is implemented using the data from TreeBASE. PhyloFinder provides several unique features that enable researchers to better identify and exploit phylogenetic data within TreeBASE or potentially any phylogenetic tree database. PhyloFinder uses TBMap to expand the utility of taxonomic queries, and it uses the GenBank Taxonomy database to identify trees with ancestors or descendants of the taxon query. It also provides alternate spelling suggestions for taxonomic queries. Furthermore, PhyloFinder supports phylogenetic queries, identifying identical or similar trees to the query tree, and it can identify the path length between any two taxa. A tree visualization tool highlights the taxon or tree query results, and it provides a hyperlink to the NCBI taxonomy website.

References

Adams Edward. September 1986. N-trees as nestings: Complexity, similarity, and consensus. *Journal of Classification*, 3 (2):299–317.

Aho AV, Sagiv Y, Szymanski et al. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SICOMP*, 10:405–21.

Azar Yossi, Fiat Amos, Karlin Anna R., Mcsherry Frank, Saia Jared. 2001. Spectral analysis of data. In ACM Symposium on Theory of Computing. p 619–626.

Bach Francis R., Jordan Michael I.. 2006. Learning spectral clustering, with application to speech separation. *J. Mach. Learn. Res.*, 7:1963–2001.

Baum BR. 1992. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, 41:3–10.

Baum BR, Ragan MA. 2004. The MRP method. In Bininda-Emonds ORP, ed. Phylogenetic supertrees: Combining Information to Reveal the Tree of Life. Dordrecht: Kluwer Academic, p 17–34.

Bender Michael A., Farach-Colton Martin. 2000. The lca problem revisited. In Latin American Theoretical Informatics. p 88–94.

Bender Michael A., Pemmasani Giridhar, Skiena Steven, Sumazin Pavel. 2001. Finding least common ancestors in directed acyclic graphs. In Symposium on Discrete Algorithms. p 845–854.

Brand M., Huang K.. 2003. A unifying theorem for spectral embedding and clustering.

Brandley M. C., Schmitz A., Reeder T. W.. 2005. Partitioned bayesian analyses, partition choice, and phylogenetic relationships of scincid lizards. *Systematic Biology*, 54:373–390.

Brinkman H., Philippe H.. 1999. Archaea sister group of bacteria? indications from tree reconstruction artifacts in ancient phylogenies. *Mol. Biol. Evol*, 16:817–825.

Bryant David, Tsang John, Kearney Paul E., Li Ming. 2000. Computing the quartet distance between evolutionary trees. In Symposium on Discrete Algorithms. p 285–286.

Bull J. J., Huelsenbeck J. P., Cunningham C. W., Swofford D. L., Waddell P. J.. 1993. Partitioning and combining data in a phylogenetic analysis. *Systematic Biology*, 42:384–397.

Burleigh J. G., Mathews S.. 2004. Phylogenetic signal in nucleotide data from seed plants: implications for resolving the seed plant tree of life. *Am. J. Bot.*, 91:1599–1613.

Celko Joe. 2004. Joe Celko's SQL for Smarties: Trees and Hierarchies. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Chan Pak K., Schlag Martine D. F., Zien Jason Y.. 1993. Spectral k-way ratio-cut partitioning and clustering. In DAC '93: Proceedings of the 30th international conference on Design automation. ACM, New York, NY, USA, p 749–754.

Chen D, Diao L, Eulenstein O, et al. 2003. Flipping: A supertree construction method. In Janowitz M, Lapoint F-J, McMorris FR, Roberts FS, eds. Bioconsensus. Vol. 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, p 135–60.

Chen D., Eulenstein O., Fernández-Baca D., et al. 2002. Supertrees by flipping. In Ibarra O.H., Zhang L(eds.), eds. COCOON '02: Proceedings of the 8th Annual International Conference on Computing and Combinatorics. Springer-Verlag, New York, p 391–400.

Chen D, Eulenstein O, Fernández-Baca D, et al. 2006. Minimum flip supertrees: Complexity and algorithms. *IEEE Trans. Bioinformatics and Computational Biology*.

Cheng David, Vempala Santosh, Kannan Ravi, Wang Grant. 2005. A divide-and-merge methodology for clustering. In PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, New York, NY, USA, p 196–205.

Cheng Y., Church G. M.. 2000. Biclustering of expression data. *Proc Int Conf Intell Syst Mol Biol*, 8:93–103.

Chippindale P. T., Wiens J. J.. 1994. Weighting, partitioning, and combining characters in phylogenetic analysis. *Systematic Biology*, 43:278–287.

Chung Fan R. K.. February 1997. Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92) (Cbms Regional Conference Series in Mathematics). American Mathematical Society.

CIPRES. 2007. Treebase ii [online]. Accessed 25 September, 2007. URL: http://www.phylo.org/sub_sections/databases.

Cole Richard, Hariharan Ramesh. 1999. Dynamic lca queries on trees. In SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, p 235–244.

Cunningham C. W.. 1997. Is congruence between data partitions a reliable predictor of phylogenetic accuracy? empirically testing an iterative procedure for choosing among phylogenetic methods. *Systematic Biology*, 46:432–437.

Davidson Susan B., Kim Junhyong, Zheng Yifeng. 2005. Efficiently supporting structure queries on phylogenetic trees. In SSDBM. p 93–102.

De Queiroz K., Donoghue M. J., Kim J.. 1995. Separate versus combined analysis of phylogenetic evidence. *Annu. Rev. Ecol. Syst.*, 26:657–681.

Donath W.E., Hoffman A. J.. 1973. Lower bounds for partitioning of graphs. *IBM J. Res. Develop.*, 17:420–425.

Doyle J. J., Doyle J. L., Rausher J. T., Brown A. H. D.. 2003. Diploid and polyploid reticulate evolution throughout the history of the perennial soybeans (*Glycine* subgenus *Glycine*). *New Phytologist*, 161:121–132.

Estabrook G.F., Johnson C.S., McMorris F.R.. 1975. An idealized concept of the true cladistic character. *Mathematical Biosciences*, 23:263–272.

Farris J. S., Kållersjo M., Kluge A. G., Bult C.. 1994. Testing the significance of congruence. *Cladistics*, 10:315–319.

Felsenstein J.. 1981. Evolutionary trees from gene frequencies and quantitative characters: finding maximum likelihood estimates. *Evolution*, 35:1229–1242.

Felsenstein Joe. 2007. The newick tree format [online]. Accessed 10 September, 2007. URL: http://evolution.genetics.washington.edu/phylip/newicktree.html.

Fiedler M.. 1973. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23 (98):298–305.

Franks S A.. 2002. Immunology and evolution of infectious disease. Princeton University Press.

Goloboff PA. 2000. Techniques for analysis of large data sets. In DeSalle R, Wheeler W, Giribet G (eds.), eds. Techniques in Molecular Systematics and Evolution. Birkhauser-Verlag, Basel, p 70–9.

Gonzalez JP, Sanchez A, Rico-Hesse R. 1996. Molecular phylogeny of guanarito virus, an emerging arenavirus affecting humans. *Am J Trop Med Hyg.*, 53 (1):1–6.

Graham R.L., Foulds L.R.. 1982. Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computation time. *Math. Bioscience*, 60:133–142.

Hagen L., Kahng A.. 1992. New spectral methods for ratio cut partitioning and clustering. *IEEE. Trans. on Computed Aided Desgin*, 11:1074–1085.

Harel Dov, Tarjan Robert Endre. 1984. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.*, 13 (2):338–355.

Henzinger Monika Rauch, King Valerie, Warnow Tandy. 1996. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. In SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, p 333–340.

Higgs FBrandon W, Weller Jennifer, Solka Jeffrey L. 2006. Spectral embedding finds meaningful (relevant) structure in image and microarray data. *BMC Bioinformatics*, 7:74.

Higham Desmond J J., Kalna Gabriela, Vass J Keith K.. November 2006. Spectral analysis of two-signed microarray expression data. *Math Med Biol*.

Hirt R., Logsdon J. M. Jr., Healy B., Dorey M. W., Dolittle W. F., Embley T.M.. 1999. Microsporidia are related to fungi: evidence from the largest subunit of rna polymerase ii and other proteins. *Proc. Natl. Acad. Sci. USA*, 96:580–585.

Huelsenbeck, J. P.and Bull J. J., Dorey M. W., Dolittle W. F., Embley T.M.. 1996. A likelihood ratio test to detect conflicting phylogenetic signal. *Systematic Biology*, 45:92–98.

Kannan, Warnow, Yooseph. 1995. Computing the local consensus of trees. In SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms).

Kearney, Li, Tsang, Jiang. 1999. Recovering branches on the tree of life: An approximation algorithm. In SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms).

Kluger Y., Basri R., Chang J. T., Gerstein M.. April 2003. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res*, 13 (4):703–716.

Korostensky, Gonnet. 2000. Using traveling salesman problem algorithms for evolutionary tree construction. *Bioinformatics*, 16.

Lerat E., Daubin V., Moran N. A.. 2003. From gene trees to organismal phylogeny in prokaryotes: The case of the γ-proteobacteria. *PLoS Biol.*, 1:1–9.

Maizels N, Weiner A M. 1994. Phylogeny from function: evidence from the molecular fossil record that trna originated in replication , not translation. *Proc Natl Acad Sci*, 91 (15):6729C6734.

Moonan F., Molina J., Mirkov T.E.. 2000. Sugarcane yellow leaf virus: An emerging virus that has evolved by recombination between luteoviral and poleroviral ancestors. In Virology. Vol. 269. Elsevier, p 156–171.

Nakhleh L, Miranker D., Barbancon F, Piel W.H., Donoghue M.J.. 2003. Requirements of phylogenetic databases. In Proc. 3rd IEEE Symp. on Bioinformatics and Bioengineering. p 141–148.

Nixon K. 1999. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:39–50.

Paccanaro A., Chennubhotla C., Casbon J. A., Saqi M. A. S.. 2003. Spectral clustering of protein sequences. In International Joint Conference on Neural Networks. Vol. 4. p 3083–3088.

Page RDM. 2002. Modified mincut supertrees. In Guigó R., Gusfield D., eds. Algorithms in Bioinformatics: Second International Workshop, WABI 2002. Vol. 2452 of *Lecture Notes in Computer Science*. Springer-Verlag, p 537–51.

Page R.D.M.. 2005a. Phyloinformatics: towards a phylogenetic database. In J.T.L. Wang, Zaki M.J., Toivonen H.T.T., Shasha D.E., eds. Data Mining in Bioinformatics. Springer-Verlag, Berlin, p 219–241.

Page R.D.M.. 2005b. A taxonomic search engine: federating taxonomic databases using web services. *BMC Bioinformatics*, 6:48.

Page Rod. 2006. Treebase talk at cipres [online]. Accessed 10 September, 2007. URL: http://iphylo.blogspot.com/2006/02/treebase-talk-at-cipres.html.

Page R. D. M.. 2007a. Tbmap: A taxonomic perspective on the phylogenetic database treebase. *BMC Bioinformatics*, 8:158.

Page R.D.M.. 2007b. Towards a taxonomically intelligent phylogenetic database. *Nature Precedings*Available from Nature Precedings (2007).

Piel W, Donoghue M, Sanderson M. 2002. Treebase: a database of phylogenetic knowledge. In To the Interoperable Catalogue of Life with partners - Species 2000 Asia Oceania - Proceedings of 2nd International Workshop of Species 2000 (Research Report for the National titute of Environmental Studies, R-171-2002). Springer-Verlag, Tsukuba, Japan.

Piel W.H., Sanderson M.J., Donoghue M.J.. 2003. The small-world dynamics of tree networks and data mining in phyloinformatics. *Bioinformatics*, 19:1162–1168.

Pisani D.. 2004. Identifying and removing fast-evolving sites using compatibility analysis: an example from the arthropods. *Systematic Biology*, 53:978–989.

Plachetzki David C., Degnan Bernard M., Oakley Todd H.. Oct 2007. The origins of novel protein interactions during animal opsin evolution. *PLoS ONE*, 2 (10):e1054.

Pothen A., Simon H. D., Liou K. P.. 1990. Partitioning sparse matrices with egenvectors of graph. *SIAM Journal of Matrix Anal. Appl.*, 11:430–452.

Ragan MA. 1992. Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution*, 1:53–8.

Rokas A, Williams BL, King N., Carroll SB. 2003. Genome-scale approaches to resolving incongruence in molecular phylogenies. *Nature*, 425:798–804.

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

Sanderson M.J., Baldwin B.G., Bharathan G., Campbell C.S.. 1993. The rate of growth of phylogenetic information, and the need for a phylogenetic database. *Syst Biol*, 42:562–568.

Semple C, Steel M. 2000. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105:147–58.

Shan H, Herbert KG, Piel WH, Shasha D, Wang JTL. 2002. A structure-based search engine for phylogenetic databases. *SSDBM*:7–10.

Shi Jianbo, Malik Jitendra. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8):888–905.

Shi S. S. B., Stokes E., Byrne D., Corn C. F., Bachmann D., Jones T.. 2000. An enterprise directory solution with db2. *IBM Syst. J.*, 39 (2):360–383.

Takahashi K., Terai Y., Nishida M., Okada N.. 2001. Phylogenetic relationships and ancient incomplete lineage sorting among cichlid fishes in lake tanganyika as revealed by analysis of the insertion of retroposons. *Mol. Biol. Evol.*, 18:2057–2066.

Tamura K, Dudley J, Nei M, Kumar S. 2007. Mega4: Molecular evolutionary genetics analysis (mega) software version 4.0. *Molecular Biology and Evolution*, 24:1596–1599.

Thornton J. W., , DeSalle R.. 2000. A new method to localize and test the significance of incongruence: detecting domain shufflingin the nuclear receptor subfamily. *Syst. Biol.*, 49:183–201.

TreeBASE. 2007. Treebase [online]. Accessed 21 September 2006. URL: http://www.treebase.org.

Tropashko V. 2005. Nested intervals tree encoding in sql. *SIGMOD Record*, 34:47–52.

Vesper Virginia. 2007. Let’s do dewey [online]. Accessed 10 September, 2007. URL: http://www.mtsu.edu/ vvesper/dewey2.htm.

Wang J.T.L., Shan H., Shasha D., Piel W.H.. 2003. Treerank: a similarity measure for nearest neighbor searching in phylogenetic databases. *SSDBM*:171–180.

Wang J.T.L., Shan H., Shasha D., Piel W.H.. 2005. Fast structural search in phylogenetic databases. *Evolutionary Bioinformatics*, 1:37–46.

Wilgenbusch J., De Queiroz K.. 2000. Phylogenetic relationships among the phrynosomatid sand lizards inferred from mitochondrial dna sequences generated by heterogeneous evolutionary processes. *Syst. Biol.*, 49:592–612.

Wolf L., Shashua A., Mukherjee S.. 2005. Gene selection via a spectral approach. In CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops. IEEE Computer Society, Washington, DC, USA.

Zheng Yifeng, Fisher Stephen, Cohen Shirley, Guo Sheng, Kim Junhyong, Davidson Susan B.. 2006. Crimson: a data management system to support evaluating phylogenetic tree reconstruction algorithms. In VLDB '06: Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, p 1231–1234.

IMPROVED HEURISTICS FOR MINIMUM-FLIP SUPERTREE CONSTRUCTION

A paper published in Evolutionary Bioinformatics Online [1]

Duhong Chen [2] [3] , Oliver Eulenstein [2] , David Fernández-Baca [2] ,J. Gordon Burleigh [4]

Abstract

The utility of the matrix representation with flipping (MRF) supertree method has been limited by the speed of its heuristic algorithms. We describe a new heuristic algorithm for MRF supertree construction that improves upon the speed of the previous heuristic by a factor of *n* (the number of taxa in the supertree). This new heuristic makes MRF tractable for large-scale supertree analyses and allows the first comparisons of MRF with other supertree methods using large empirical data sets. Analyses of three published supertree data sets with between 267 to 571 taxa indicate that MRF supertrees are equally or more similar to the input trees on average than matrix representation with parsimony (MRP) and modified mincut supertrees. The results also show that large differences may exist between MRF and MRP supertrees, and demonstrate that the MRF supertree method is a practical and potentially more accurate alternative to the nearly ubiquitous MRP supertree method.

Introduction

There is increasing interest in supertree methods for phylogenetics, especially for building very large trees (see Bininda-Emonds et al 2002; Bininda-Emonds 2004). Supertree methods combine phy-

[1]Reprinted with permission of Evolutionary Bioinformatics Online, 2006,(2):401–410
[2]Department of Computer Science, Iowa State University, Ames, IA 50011, USA
[3]Author for correspondence
[4]Section of Evolution and Ecology, University of California, Davis, CA 95616, USA

logenetic trees with incomplete taxonomic overlap into a comprehensive phylogeny that incorporates all taxa from the input trees. Since the ultimate aim of many supertree analyses is to build enormous phylogenies, an effective supertree method must be fast as well as accurate. Therefore, the development and implementation of fast algorithms is a critically important part of establishing effective and useful supertree methods.

The most popular supertree method by far is matrix representation with parsimony (MRP; see Bininda-Emonds 2004). MRP performs a maximum parsimony analysis on a binary matrix representation of the set of input trees (Baum, 1992; Ragan, 1992; Baum and Ragan, 2004). Therefore, MRP analyses can use fast maximum parsimony heuristics (eg, Nixon 1999; Goloboff 2000) and popular phylogenetics programs that implement maximum parsimony like PAUP* (Swofford, 2002) or TNT (Goloboff, 2000). Still, MRP has been criticised for its performance and properties (eg, Purvis 1995; Pisani and Wilkinson 2002; Gatesy and Springer 2004; Goloboff 2005; Wilkinson et al 2005). For example, MRP may have a size bias, in size of the input trees affects how MRP resolves conflicts (Purvis 1995). There is also evidence that MRP may have a shape bias, in which input tree shape affects the resulting supertree (Wilkinson et al 2005). Furthermore, the validity of using parsimony on a matrix representation of input trees has been questioned (Slowinski and Page 1999; Eulenstein et al 2004; Gatesy and Springer 2004) . Thus, there is a need to investigate alternate supertree methods.

The matrix representation with flipping (MRF, or minimum flip) supertree method, like MRP, uses a matrix representation of the input trees (Chen et al 2003; Chen et al 2004; Burleigh et al 2004; Eulenstein et al 2004). While MRP seeks trees that minimize the parsimony score of the matrix representation of input trees, MRF seeks the minimum number of flips, character changes from 0 to 1 or 1 to 0, that will make the matrix representation of input trees consistent with a phylogenetic tree (see Chen et al 2003; Burleigh et al 2004; Eulenstein et al 2004). The resulting phylogenetic trees are MRF supertrees. Like the parsimony problem, the minimum flip problem is $\mathcal{NP}$-hard (Chen et al 2006), and therefore, estimating an MRF supertree requires heuristic algorithms when the input trees contain more than approximately 20 taxa. Simulation experiments (Eulenstein et al 2004; Piaggio-Talice et al 2004) indicate that MRF supertrees retain more of the relationships from the input trees than MinCut (Semple and Steel 2000), Modified MinCut (MMC; Page 2002), and quartet supertree methods (Piaggio-Talice et al 2004). Also, MRF supertrees retain relationships from the input trees at least as well as MRP supertrees, with MRF appearing to slightly outperform MRP as the taxon overlap among input trees decreases (Chen et al 2003; Burleigh et al 2004; Eulenstein et al 2004). Though these results suggest that MRF is a promising supertree method, the characteristics of the simulated data sets likely differ greatly from those of empirical data sets. Furthermore, the first MRF heuristics were slow (Eulenstein et al 2004; Goloboff 2005), and consequently, the performance of the MRF supertree method on large empirical data sets has been largely unexamined.

We describe improvements to existing MRF heuristics that increase the speed of the heuristics by a factor of n, where n is the total number of taxa represented in the input trees. These improvements make it feasible to estimate MRF supertrees for large data sets. Furthermore, they allow the first comparisons of the performance of MRF with other supertree methods using empirical data sets containing many more taxa than were in the simulated data sets. The results of these analyses demonstrate that MRF may perform better than MRP or MMC supertree methods. The analyses also demonstrate notable differences between results of MRF and MRP supertrees that were not observed in small simulation studies.

Definitions

Let $S = \{s_1, \ldots, s_n\}$ denote a set of n taxa and $\mathcal{L}(T)$ denote the leaf set of a rooted tree T.

A *directed phylogenetic tree*, or *phylogeny* for short, over set S is a rooted binary tree T such that every internal node of T has two children and $\mathcal{L}(T) = S$. Let v be a node of a phylogeny T. Then, T_v denotes the subtree rooted at v, and $T - T_v$ denotes the tree T with the subtree T_v removed. The set $\mathcal{L}(T_v)$ is the *cluster* of T at v.

A *profile* is a multiset $\mathcal{T}$ of phylogenies. The elements of $\mathcal{T}$ are called *input trees*. A *supertree* for a profile $\mathcal{T}$ is a phylogeny T such that $\mathcal{L}(T) = \bigcup_{t \in \mathcal{T}} \mathcal{L}(t)$.

A *character matrix* for S is an $n \times m$ matrix $M = [a_{ij}]$ over $\{0, 1, ?\}$, whose i-th row corresponds to taxon s_i. The j-th column of M is called *character j*. The set of all s_i such that $a_{ij} = 1$ is the 1-*set* of character j and is denoted by O_j; the set of all s_i such that $a_{ij} = 0$ is the 0-*set* of character j and is denoted by Z_j.

Let $\mathcal{T}$ be a profile such that $\bigcup_{t \in \mathcal{T}} \mathcal{L}(t) = S$. For our study, we define a *matrix representation* of $\mathcal{T}$ as a character matrix M for S obtained as follows. For each tree $t \in \mathcal{T}$ and each cluster X in t, create a column of M whose i-th entry is 1 if $s_i \in X$, 0 if $s_i \in \mathcal{L}(t) - X$, and ? if $s_i \notin \mathcal{L}(t)$.

The matrix representation of trees is the basis for MRP (Baum, 1992; Ragan, 1992; Purvis, 1995) and MRF (Chen et al, 2003; Burleigh et al, 2004; Eulenstein et al, 2004) supertree methods. We note that there are numerous different matrix representations of trees (eg Farris et al 1970; Purvis 1995; Wilkinson et al 2004). In this paper, we use a standard binary matrix representation (Farris et al 1970; Baum 1992; Ragan 1992) which is by far the most commonly used in supertree studies and was used in the formal definitions of the MRF method (Chen et al 2003; Burleigh et al 2004; Eulenstein et al 2004). MRF is based on the notion of *flip distance* from a character matrix M to a tree T (Chen et al, 2003; Eulenstein et al, 2004). This quantity equals the smallest number of $1 \rightarrow 0$ and $0 \rightarrow 1$ changes (*flips*) that must be made to M so that the 1-set of each character of M corresponds to some cluster in T. An *MRF supertree* for M is a tree T that has minimum flip distance to T. We now define the above notions precisely.

Let T be a phylogeny over some subset of S, v be a node of T, and M be a character matrix for S. Let $z_j(v)$ denote the number of taxa that are in the 0-set of character j and also in the cluster at v; that is, $z_j(v) = |Z_j \cap \mathcal{L}(T_v)|$. Similarly, let $o_j(v)$ denote the number of taxa that are in the 1-set of character j as well as in the cluster at v; that is, $o_j(v) = |O_j \cap \mathcal{L}(T_v)|$. The *flip distance* of character j to v is defined as

$$f_j(v) = z_j(v) + (|O_j| - o_j(v)). \tag{1}$$

Note that $f_j(v)$ is the number of changes needed to make character j correspond to the cluster at node v. The first term in the right hand side of the above equation is the number of $0 \rightarrow 1$ changes and the second term equals the number of $1 \rightarrow 0$ changes.

The flip distance of character j to T is

$$f_j(T) = \min_{v \in T} f_j(v) \tag{2}$$

The flip distance of character matrix M to T is

$$f_M(T) = \sum_{j=1}^{m} f_j(T) \tag{3}$$

The flip distance of a profile $\mathfrak{T}$ to T is

$$f_{\mathfrak{T}}(T) = f_M(T), \tag{4}$$

where M is some matrix representation of $\mathfrak{T}$. Note that $f_{\mathfrak{T}}(T)$ is well-defined, since all matrix representations of $\mathfrak{T}$ are column permutations of each other.

The *minimum-flip problem* is: Given a character matrix M over S, find a phylogeny T over S such that $f_M(T)$ is minimum. The *fixed-tree minimum flip problem* is: Given a character matrix M for S and a phylogeny T for S, compute $f_M(T)$.

Heuristics for MRF

The MRF supertree problem is defined only for rooted trees (Chen et al 2003; Burleigh et al 2004; Eulenstein et al 2004), and the rooting of a tree can affect its flip distance from a character matrix. Thus, unlike MRP, MRF supertree heuristics cannot use existing unrooted tree search algorithms. The details of the original MRF heuristic algorithm were not described by Eulenstein et al (2004), which has led to some apparent confusion in critiques of MRF (eg Goloboff 2005). Therefore, we fully describe the accelerated MRF heuristic.

Like its predecessor, the new MRF heuristic uses a hill climbing strategy that is similar to the one used for unrooted tree searches in PAUP* (Swofford 2002). The initial tree is obtained through greedy taxon addition using a randomly-chosen order (in practice, several initial trees are usually generated). After the initial tree is obtained, the search proceeds iteratively. At each step it locates the best tree (the tree with the lowest flip distance) that can be obtained from the current tree by a *branch swap*. Each tree that can be generated by a single branch swap is called a *neighbor* of the current tree. If no neighbor has a lower flip distance, the search stops and the current tree is returned as the estimate of a MRF supertree. Otherwise, the current tree is replaced by its best neighbor. The improved run times reported here, compared to the run times in the previous MRF heuristic, are due to changes in the implementation of the branch swapping operations.

We consider three rooted branch swapping operations.

Rooted Nearest Neighbor Interchange (rNNI) Choose an internal node v of T and swap one of v's children with v's sibling. Note that T has $2n-4$ rNNI neighbors.

Rooted Subtree Pruning and Regrafting (rSPR) (See also Hein 1990; Bordewich and Semple 2004.) Choose a non-root node v of T, called a *prune node*. Prune the subtree T_v by removing the edge between v and its parent, suppressing the remaining degree-two node. Next, *regraft* T_v into $T-T_v$ as follows: Pick a node u, called the *regrafting node*, in $T-T_v$. If u is the root, create a new root p and make p the parent of u and v. Otherwise, create a new vertex p that subdivides the edge between u and its parent, and make p the parent of v. Note that T has $\Theta(n^2)$ rSPR neighbors.

Rooted Tree Bisection and Reconnection (rTBR) This operation extends rSPR by allowing the pruned subtree $T' = T_v$ to be *re-rooted* before regrafting. Re-rooting is done as follows: (i) Suppress the root node of T'. (ii) Create a new root node r by subdividing an edge $\{x,y\}$ in T' into the edges $\{x,r\}$ and $\{y,r\}$. We refer to this operation as *bending* edge $\{x,y\}$. Note that T has $\Theta(n^3)$ rTBR neighbors.

The earlier MRF heuristic found the best neighbor by computing the flip distance of each such neighbor from scratch (Eulenstein et al 2004). This failed to exploit the similarities between the current tree and its neighbors, and, consequently, was quite slow. The running times to find an optimal neighbor tree of a given n-taxon tree for rNNI, rSPR, and rTBR were $O(n^2m)$, $O(n^3m)$, and $O(n^4m)$, respectively. The new algorithms reduce these times by a factor of n, giving execution times of $O(nm)$, $O(n^2m)$, and $O(n^3m)$, respectively. In all three cases, the key is to preprocess the tree to allow evaluation of the flip distance of each neighbor in $O(1)$ time per character. Our procedures share some ideas with recently described parsimony heuristics (Ganapathy et al 2003).

In the remainder of this section, we first describe a *bottom-up assignment* algorithm that is used in all our branch swapping procedures. We then describe the new rSPR and rTBR algorithms. Finally, we explain the implementation of greedy taxon addition, which relies on rSPR. We have experimentally determined that rNNI tends to produce trees with higher flip distances than rSPR and rTBR; nevertheless, for completeness, we describe the rNNI algorithm in the Appendix. Since the flip distance $f_M(T)$ can be obtained by computing $f_j(T)$ independently for each character j and adding up the results (see Equation (3)), the descriptions of all the algorithms to follow focus on the computation of $f_j(T)$ for a single character j.

Bottom-up assignment

The algorithm traverses the input tree T in postorder, computing four quantities for each node v: $z_j(v)$, $o_j(v)$, $f_j(v)$, and $f_j(T_v)$. Before the traversal starts, it computes the values of $|O_j|$ for every character j; this takes $O(nm)$ time.

Consider a node v of T. If v is a leaf, we can easily compute $z_j(v)$, $o_j(v)$, $f_j(v)$, and $f_j(T_v)$ in $O(1)$ time. Now, suppose v is an internal node with children u and w, such that $z_j(x)$, $o_j(x)$, $f_j(x)$, and $f_j(T_x)$ are known for $x = u, w$. Obviously,

$$z_j(v) = z_j(u) + z_j(w) \qquad \text{and} \qquad o_j(v) = o_j(u) + o_j(w). \tag{5}$$

Given $z_j(v)$, $o_j(v)$, and $|O_j|$, the value of $f_j(v)$ follows from Equation (1); $f_j(T_v)$ is given by

$$f_j(T_v) = \min\{f_j(T_u), f_j(T_w), f_j(v)\}. \tag{6}$$

Thus, $z_j(v)$, $o_j(v)$, $f_j(v)$, and $f_j(T_v)$ can be obtained in $O(1)$ time. Since there are $2n-1$ nodes in T, computing the four required values for every node of T takes time $O(n)$ per character, for a total of $O(nm)$ time.

When the bottom-up assignment is finished, $f_j(T) = f_j(T_v)$, where v is the root node of T, and $f_M(T)$ can be computed in $O(m)$ time via Equation (3).

Finding the best rSPR neighbor

The algorithm considers all possible prune nodes; for each such node, it computes the optimum regrafting node. A prune node v is processed in two steps. First, apply the bottom-up assignment algorithm to T_v and $T - T_v$. Thus, for each node w of each tree and each character j we have $z_j(w)$, $o_j(w)$, $f_j(w)$, and $f_j(T_w)$. Second, traverse the nodes of $T - T_v$ in preorder. Let the k-th node in the preorder sequence be u_k; thus, u_1 is the root of $T - T_v$. At step k, we compute the flip distance of the tree $T^{(k)}$ obtained by regrafting T_v at u_k. We now explain how to obtain $f_j(T^{(1)})$ in $O(1)$ time and how to compute $f_j(T^{(k)})$, $k > 1$, in $O(1)$ time using the information computed for $T^{(k-1)}$.

Let p_k denote the parent of v and u_k in the k-th rSPR neighbor tree. Let $r_j^{(k)}$ denote $f_j(T^{(k)} - T_{p_k}^{(k)})$. Define $r_j^{(1)} = +\infty$.

Note that p_1 is the root of the first rSPR neighbor tree (Figure 1(a–b)) and that

$$f_j(T^{(1)}) = f_j(T_{p_1}^{(1)}) = \min\{f_j(p_1), f_j(T_v^{(1)}), f_j(T_{u_1}^{(1)})\} \tag{7}$$

The value of $f_j(p_1)$ can be computed in $O(1)$ time using Equations (1) and (5) and the information stored at the roots of T_v and $T - T_v$. Note that $f_j(T_v^{(1)})$ equals $f_j(T_v)$, which is known, and $f_j(T_{u_1}^{(1)})$ equals $f_j((T - T_v)_{u_1})$, which is also known. Thus, $f_j(T^{(1)})$ can be obtained in $O(1)$ time.

Assume that $T^{(k-1)}$, $k > 1$, has been processed. We now describe how to process $T^{(k)}$ (Figure 1(c–d)). Let w_A and w_B denote the left and right children of u_{k-1} in $T^{(k-1)}$ and let $T_A = T_{w_A}^{(k-1)}$ and $T_B = T_{w_B}^{(k-1)}$. Without loss of generality, we assume that $u_k = w_A$.

Assume that we know $r_j^{(k-1)}$. For $T^{(k)}$ we have

$$r_j^{(k)} = \min\{r_j^{(k-1)}, f_j(u_{k-1}), f_j(T_B)\}. \tag{8}$$

Since, the cluster at u_{k-1} in $T^{(k)}$ is the same as the cluster at p_{k-1} in $T^{(k-1)}$, the above expression can be evaluated in $O(1)$ time. Now,

$$f_j(T^{(k)}) = \min\{r_j^{(k)}, f_j(T_{p_k}^{(k)})\}. \tag{9}$$

Note that $f_j(T_{p_k}^{(k)}) = \min\{f_j(p_k), f_j(T_v), f_j(T_A)\}$, so $f_j(T_{p_k}^{(k)})$ can be computed in $O(1)$ time given the information available at T_v and T_A from the preprocessing step. Hence, $f_j(T^{(k)})$ can be obtained from

$T^{(k-1)}$ in $O(1)$ time using Equation (9). Thus, the best regrafting node for T_v can be found in $O(n)$ time per character. Since there are $O(n)$ choices for v, this leads to a time of $O(n^2)$ per character, and $O(n^2m)$ total, to find the best rSPR neighbor of T.

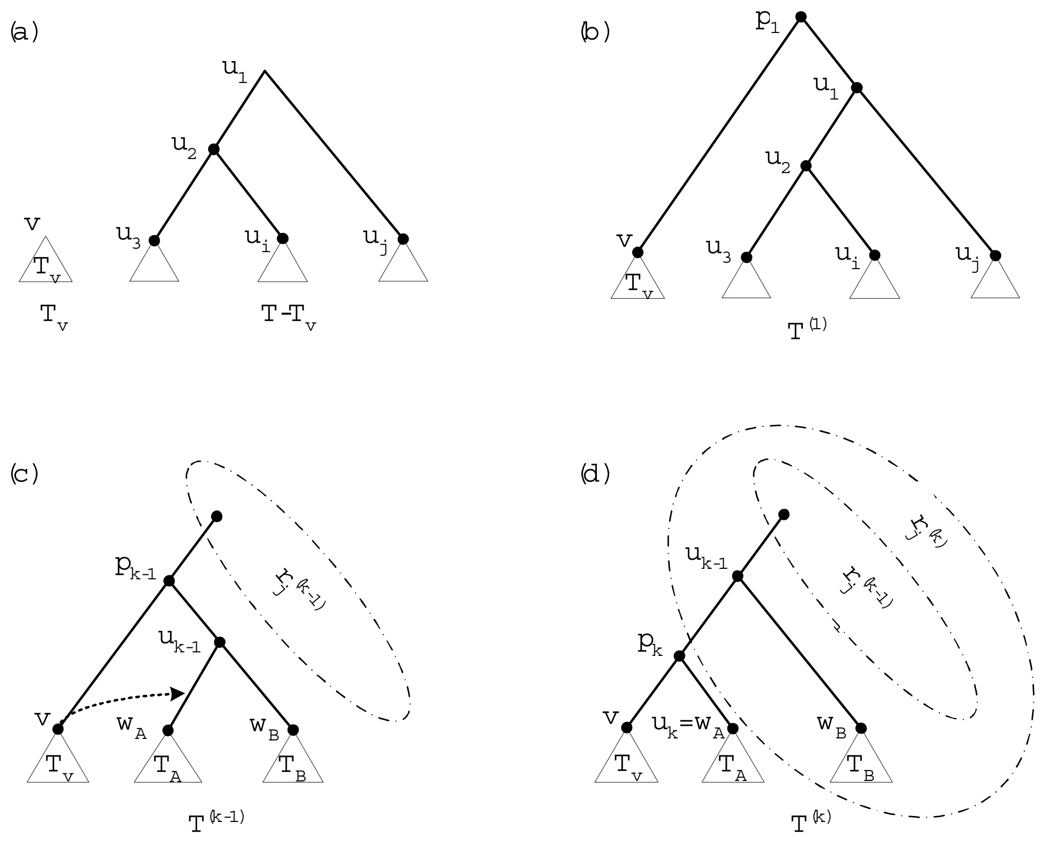


Figure 1: (a) The trees T_v and $T - T_v$ obtained after a cut at node v. (b) The first rSPR neighbor tree $T^{(1)}$ obtained by regrafting at the root. (c–d) The transformation from $T^{(k-1)}$ to $T^{(k)}$.

Finding the best rTBR neighbor

rTBR differs from rSPR in that it may re-root T_v before attaching it to $T - T_v$. To handle re-rooting efficiently, we use a *three-way assignment* approach, similar to the one used for parsimony by Ganapathy et al (2003). We now outline the main ideas of this method.

Consider an internal node u as shown in Figure 2. The subtrees of u change, depending on whether the new root is in the direction of edges 1, 2, or 3; the subtrees of u are $\{T_x, T_y\}$, $\{T_y, T_z\}$, or $\{T_x, T_z\}$, respectively. A *three-way assignment* is a labeling of each vertex u with three lists of values

$\langle z_j(u), o_j(u), f_j(u), f_j(T_u)\rangle$, one for each possible rooting. Such an assignment can be obtained in $O(n)$ time per character by doing a bottom-up assignment to find the assignments for the first rooting, and then doing a top-down preorder traversal to update the assignments for the other two rootings.

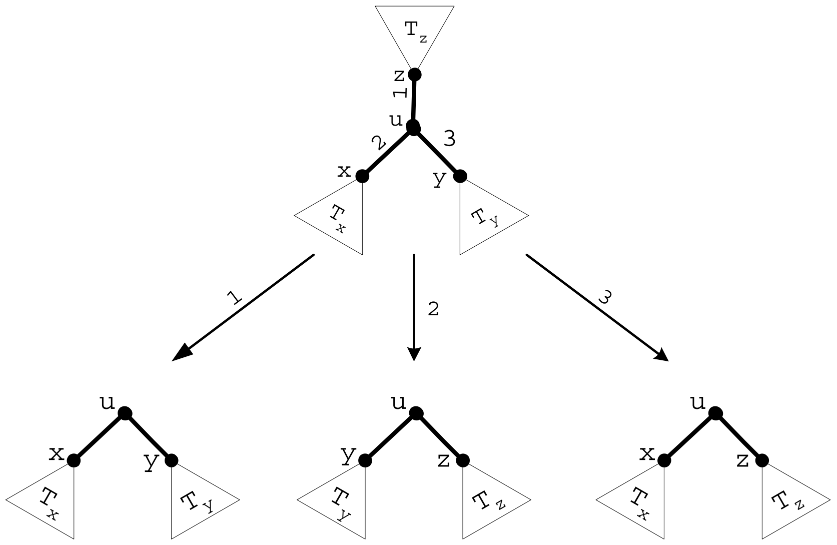


Figure 2: Internal node u and the three possible pairs of subtrees it may have, depending on the rooting. Each requires a different assignment.

After computing a three-way assignment for the pruned subtree T_v, we have, for every possible re-rooting t of T_v, the information needed to find the best possible regrafting of t into $T - T_v$, using the method earlier described for rSPR. This takes $O(n)$ time per character for each fixed re-rooting t. Since there are $O(n)$ possible re-rootings of T_v and $O(n)$ choices for v, the time required to find the best rTBR neighbor is $O(n^3)$ per character and $O(n^3 m)$ total.

Greedy taxon addition

The greedy search begins with a unique initial tree formed from the first two taxa in the input data set. The third taxon is inserted into every possible branch of the initial tree to form all possible three-taxon trees. The three-taxon tree with minimum flip distance is chosen. Each successive taxon is added in this way until a complete tree is obtained.

To find the best place to add the k-th taxon to the $(k-1)$-taxon tree, we use our optimum rSPR

Data set	Num. of input trees	Num. of taxa	Num. of characters
Marsupial	158	267	1775
Cetartiodactyla	201	290	1975
Legume	20	571	765

Table 1: Supertree data sets. The second column lists the total number of input trees in each data set, and the third column lists the number of taxa that are found in the set of all input trees. The last column lists the number of characters in the binary matrix representation of the set of input trees.

neighbor algorithm. In this case, the tree being grafted has a single node containing taxon k and there are $2k-3$ ways to add this taxon to the $(k-1)$-taxon tree. The time per regraft is $O(km)$, yielding a total running time of $O(n^2m)$ for the entire addition sequence. We note that the performance the MRF heuristic may be improved by repeating the greedy taxon addition using different permutations of the taxa and thus generating multiple starting trees.

Data sets and results

We examined the performance of MRF, and compared it to two other supertree methods, MRP and MMC, using three large, empirical data sets. MRF supertree analyses, implemented in HeuristicMFT2 (Chen, 2005), used rSPR and rTBR branch swapping on three random addition sequence replicates and saved a maximum of ten trees. MRP supertrees were constructed using PAUP* (Swofford 2002), and used TBR branch swapping on three random-addition sequence replicates and saved a maximum of 100 trees. MMC supertrees were constructed with a program supplied by Rod Page (Page, 2003). The data sets (see Table 1) were taken from large, published supertree studies of marsupials (Cardillo et al 2004), mammals (Price et al 2005), and legumes (Wojciechowski et al 2000).

The performance of each supertree method was evaluated by measuring the degree to which the supertrees agree with the input trees (eg Eulenstein et al 2004). Two measures were used for this purpose: 1) the average MAST-fit between the supertree and the input trees and 2) the average triplet-fit from the supertree to the input trees. The MAST-fit between a supertree and an input tree is the number of leaves in their maximum agreement subtree (Gordon 1980; Kubicka et al 1992) divided by the number of leaves in common between the two trees. This was calculated using PAUP* (Swofford 2002). The triplet-fit from a supertree to an input tree is $1-(d+r)/(d+r+s)$, where s is the number of rooted triplets that are identically resolved in the supertree and the input trees, d is the number of triplets resolved differently in both trees, and r is the number of triplets resolved in the input tree but not in the supertree (Page 2002). The triplet-fit, unlike the MAST-fit, is an asymmetric similarity measure. If there was more than one optimal supertree, we present the average score of all optimal supertrees to each of the input trees. In addition to measuring the quality of the supertrees, we also compared the CPU-time for each supertree method to provide a rough estimate of the computational requirements for each method. All the analyses were conducted on a Linux platform with a 3.0 GHz Pentium IV processor.

In the analyses of each of the three data sets, the MRF supertree had an equal or higher average MAST-fit and triplet-fit to the input trees than MMC or MRP supertrees (Table 2). In the marsupial and Cetartiodactyla data sets, the MAST and triplet-fit scores for the MRP supertrees were very similar to the scores for the MRF supertrees, while the scores for the MMC supertree were lower (Table 2). However, in the supertree analyses of the legume data set, the MAST and triplet-fit scores of the MRF supertree were noticeably ($\geq 6\%$) higher than for the MMC or MRP supertrees. Also in the analyses of the legume data set, the triplet-fit score for the MRP supertree was higher than that of the MMC supertree, but the MAST-fit score of the MMC supertree was higher than that of the MRP supertree (Table 2). There was little if any difference in the performance of rSPR or rTBR algorithms in MRF analyses, though rTBR analyses required more CPU time than rSPR analyses (Table 2). In the analyses of the marsupial and Cetartiodactyla data sets, MRF with rSPR still required the most CPU time of the three supertree methods, but in the analysis of the legume data set, MRF with rSPR branch swapping used less CPU time than the MRP heuristic (Table 2).

Data set	Supertree	Triplet fit	MAST fit	Pars. score	Flip dist.	CPU time (sec)
Marsupial	MMC	0.544	0.542	3891	3058	164
	MRP	0.823	0.713	2274	823	583
	MRF(rSPR)	0.823	0.717	2296	801	989
	MRF(rTBR)	0.823	0.717	2594	801	1398
Cetartiodactyla	MMC	0.489	0.508	5017	4339	144
	MRP	0.796	0.654	2510	904	805
	MRF(rSPR)	0.803	0.659	2524	893	2258
	MRF(rTBR)	0.804	0.659	2523	893	2895
Legume	MMC	0.713	0.711	1489	1567	39
	MRP	0.789	0.663	962	710	6884
	MRF(rSPR)	0.849	0.764	1043	397	4958
	MRF(rTBR)	0.856	0.764	1041	392	8099

Table 2: Results of the supertree analyses of three empirical data sets. The triplet-fit and MAST-fit columns show the average triplet-fit or MAST-fit distances of the input trees to the supertree. The Pars. score column shows the parsimony score of the supertree based on the binary matrix representation of input trees, and the Flip dist. column shows the minimum flip distance of the supertree based on the binary matrix representation of input trees. CPU time is the computational time for each supertree algorithm.

Discussion

The new heuristic algorithm makes MRF analyses feasible for large empirical data sets. The previous MRF algorithm was not only slow, its performance and implementation were questioned (Goloboff 2005). Eulenstein et al (2004) reported that the previous rSPR heuristic performed better than the rTBR heuristic for MRF. Goloboff (2005, p 289) interpreted this to mean that "SPR usually produced a better agreement with the model [true] tree". However, Eulenstein et al's (2004) statement only referred to an anecdotal observation that rSPR was faster than rTBR and that the flip distances of rSPR and rTNR trees were very similar if not identical. In this study, we again observed that the MRF heuristic with rSPR branch swapping is much faster than the heuristic with rTBR branch swapping, and that both algorithms yield trees with similar flip distances (Table 2). The similarity between the performance of rSPR and rTBR may seem intuitively surprising (eg Goloboff 2005), but it likely results from the nature of rooted branch swapping. rSPR and rTBR differ in that the latter may reroot the pruned subtree before regrafting. In most situations, it appears that rerooting does not reduce the flip distance. That is, the best rSPR and rTBR neighbors usually have the same flip distance. The implementation of the new heuristic also fixes a bug in the implementation of the previous MRF heuristic that caused the program to save suboptimal trees with rTBR and rNNI branch swapping (see Goloboff 2005). Both rSPR and rTBR heuristics generally produce supertrees with lower flip distances than supertrees produced with rNNI heuristics (not shown). Though there appears to be little benefit in using the rTBR as opposed to rSPR heuristic in a quick MRF analysis, a thorough MRF analysis should include rTBR branch swapping.

The speed of the new heuristic makes it possible to assess the performance of the MRF supertree method using data sets that would have been too computationally demanding for the previous heuristic method. In fact, these are among the first reported MRF analyses using empirical data sets (but see Burleigh et al 2004). In all three analyses, MRF appears to perform at least as well and often better than MMC and MRP (Table 2). The results also emphasize the differences that may exist between MRP and MRF supertrees. In previous simulation and empirical studies that used small input trees, the average similarity scores of MRP and MRF supertrees to the input trees were nearly identical (Chen et al 2003; Burleigh et al 2004; Eulenstein et al 2004). However, this does not necessarily mean that MRF and MRP supertrees are similar to each other. In the analyses of the marsupial and Cetartiodactyla data sets, there is a notable difference in flip distance and parsimony scores of the MRF and MRP supertrees, even though the similarity of MRF and MRP supertrees to the input trees appears nearly identical. In the analysis of the legume data set, the difference in the parsimony scores and flip distances of the MRF and MRP supertrees is much larger (Table 2). These examples also demonstrate that the parsimony score of a supertree based on its binary matrix representation may not be a good indicator of the similarity of the supertree to the input trees. MRF trees with higher (worse) parsimony scores resemble the input trees more than the optimal MRP trees (Table 2). In these cases, minimizing the flip distance appears to be a better optimality criterion than minimizing the parsimony score. The legume supertree analyses also demonstrate that the MMC supertree method, which uses no true optimality criterion, can produce supertrees that appear more similar to the input trees than MRP. Thus, it may be unwise to rely solely on an MRP supertree analysis.

A good supertree method must balance computational speed with accuracy. For example, the MMC supertree method has a fast polynomial time algorithm (Page, 2002), but it often results in low quality supertrees (Table 2; Eulenstein et al 2004). Conversely, the MRF supertree method appears to be accurate relative to other supertree methods, but previously its heuristics were too slow for large supertree studies (Eulenstein et al 2004). However, the availability of heuristics should not dictate one's choice of supertree methods. Rather the properties of a supertree method should motivate the development of useful heuristics. Though a number of supertree methods have been proposed (see Bininda-Emonds 2004), there has been much less focus on developing fast implementations of these methods. This study demonstrates that such work can benefit supertree analyses. We do not suggest that MRF is now the optimal supertree method. In some cases, MRF may exhibit undesirable properties (eg Goloboff 2005; Wilkinson et al 2005), and the speed of the new heuristics may still be a limitation for building supertrees with many thousand taxa or for implementing supertree bootstrapping replicates (eg, Creevey et al 2004; Philip et al 2005; Burleigh et al 2006). Still, with the new heuristics, MRF is, in many cases, a viable supertree method that should be considered along with other methods.

Acknowledgements

We thank Olaf Bininda-Emonds and Marty Wojciechowski for providing empirical data sets and Sagi Snir and Mike Sanderson for valuable discussions. This research was supported by National Science Foundation grants EF-0334832, CCR-9988348, and 0431154.

References

Baum BR. 1992. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, 41:3–10.

Baum BR, Ragan MA. 2004. The MRP method. In Bininda-Emonds ORP, ed. Phylogenetic supertrees: Combining Information to Reveal the Tree of Life. Dordrecht: Kluwer Academic, p 17–34.

Bininda-Emonds ORP. 2004. The evolution of supertrees. *Trends in Ecology and Evolution*, 19:315–22.

Bininda-Emonds ORP, Gittleman JL, Steel MA. 2002. The (super)tree of life: procedures, problems, and prospects. *Annual Review of Ecology and Systematics*, 33:265–89.

Bordewich M, Semple C. 2004. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, 8:409–23.

Burleigh JG, Driskell AC, Sanderson MJ. 2006. Supertree bootstrapping methods for assessing phylogenetic variation among genes in genome-scale data sets. *Systematic Biology*.

Burleigh JG, Eulenstein O, Fernández-Baca D, et al. 2004. MRF supertrees. In Bininda-Emonds ORP, ed. Phylogenetic supertrees: Combining Information to Reveal the Tree of Life. Dordrecht: Kluwer Academic, p 65–85.

Cardillo M, Bininda-Emonds ORP, Boakes E, et al. 2004. A species-level phylogenetic supertree of marsupials. *Journal of Zool., Lond.*, 264:11–33.

Chen D. 2005. HeuristicMFT2 [online]. Accessed 27 December 2005. URL: http://genome.cs.iastate.edu/CBL/download/.

Chen D, Diao L, Eulenstein O, et al. 2003. Flipping: A supertree construction method. In Janowitz M, Lapoint F-J, McMorris FR, Roberts FS, eds. Bioconsensus. Vol. 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, p 135–60.

Chen D, Eulenstein O, Fernández-Baca D. 2004. Rainbow: A toolbox for phylogenetic supertree construction and analysis. *Bioinformatics*, 20 (16):2872–2873.

Chen D, Eulenstein O, Fernández-Baca D, et al. 2006. Minimum flip supertrees: Complexity and algorithms. *IEEE Trans. Bioinformatics and Computational Biology*.

Creevey CJ, Fitzpatrick DA, Philip GK, et al. 2004. Does a tree-like phylogeny only exist at the tips in the prokaryotes? *Proceeding of the Royal Society of London B Bio.*, 271:2551–8.

Eulenstein O, Chen D, Burleigh JG, et al. 2004. Performance of flip supertrees with a heuristic algorithm. *Systematic Biology*, 53 (2):299–308.

Farris JS, Kluge AG, Eckardt MJ. 1970. A numerical approach to phylogenetic systematics. *Systematic Zool.*, 19:172–189.

Ganapathy G, Ramachandran V, Warnow T. 2003. Better hill-climbing searches for parsimony. In Benson G., Page R., eds. Algorithms in Bioinformatics: Third International Workshop, WABI 2003. Vol. 2812 of *Lecture Notes in Computer Science*. Springer-Verlag, p 245–58.

Gatesy J, Springer MS. 2004. A critique of matrix representation with parsimony supertrees. In Bininda-Emonds ORP, ed. Phylogenetic supertrees: Combining Information to Reveal the Tree of Life. Dordrecht: Kluwer Academic, p 369–88.

Goloboff PA. 2000. Techniques for analysis of large data sets. In DeSalle R, Wheeler W, Giribet G (eds.), eds. Techniques in Molecular Systematics and Evolution. Birkhauser-Verlag, Basel, p 70–9.

Goloboff PA. 2005. Minority rule supertrees? MRP, compatibility, and minimum flip may display the *least* frequent groups. *Cladistics*, 21:282–94.

Gordon AD. 1980. On the assessment and comparison of classifications. In Tomassine R, ed. Analyse de Donnees et Informatique. INRIA, Le Chesnay, p 7245–58.

Hein JJ. 1990. Reconstructing the history of sequences subject to gene conversion and recombination. *Mathematical Biosciences*, 98:185–200.

Kubicka E, Kubicki G, McMorris FR. 1992. On agreement subtrees of two binary trees. *Congressus Numerantium*, 88:217–24.

Nixon K. 1999. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:39–50.

Page RDM. 2002. Modified mincut supertrees. In Guigó R., Gusfield D., eds. Algorithms in Bioinformatics: Second International Workshop, WABI 2002. Vol. 2452 of *Lecture Notes in Computer Science*. Springer-Verlag, p 537–51.

Page RDM. 2003. Supertree [online]. Accessed 2 April 2003. URL: http://darwin.zoology.gla.ac.uk/˜rpage/supertree/.

Philip GK, Creevey CJ, McInerney JO. 2005. The opisthokonta and ecdysozoa may not be clades: Stronger support for the grouping of plant and animal than for animal and fungi and stronger support for the coelomata than ecdysozoa. *Molecular Biology and Evolution*, 22:1175–84.

Piaggio-Talice R, Burleigh JG, Eulenstein O. 2004. Quartet supertrees. In Bininda-Emonds ORP, ed. Phylogenetic supertrees: Combining Information to Reveal the Tree of Life. Dordrecht: Kluwer Academic, p 173–91.

Pisani D, Wilkinson M. 2002. MRP, taxonomic congruence and total evidence. *Systematic Biology*, 51:151–5.

Price SA, Bininda-Emonds ORP, Gittleman JL. 2005. A complete phylogeny of whales, dolphins and even-toed hoofed mammals (cetardiodactyla). *Biological Reviews*, 80:445–73.

Purvis A. 1995. A modification to Baum and Ragan's method for combining phylogenetic trees. *Systematic Biology*, 44:251–5.

Ragan MA. 1992. Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution*, 1:53–8.

Semple C, Steel M. 2000. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105:147–58.

Slowinski JB, Page RDM.. 1999. How should species phylogenies be inferred from sequence data? *Systematic Biology*, 48:814–25.

Swofford DL. 2002. PAUP*: Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4.0 beta. Sinauer Assoc., Sunderland, Massachusetts, USA.

Wilkinson M, Cotton JA, Creevey C, et al. 2005. The shape of supertrees to come: tree shape related properties of fourteen supertree methods. *Systematic Biology*, 54:419–31.

Wilkinson M, Cotton JA, Thorley JL. 2004. The information content of trees and their matrix representation. *Systematic Biology*, 53:989–1001.

Wojciechowski MF, Sanderson MJ, Steele KP, et al. 2000. Molecular phylogeny of the "Temperate Herbaceous Tribes" of Papilionoid legumes: a supertree approach. In Herendeen PS, Bruneau A, eds. Advances in Legume Systematics. Vol. 9. Royal Botanic Gardens, Kew, p 277–98.

Appendix: Finding the best rNNI neighbor

An rNNI operation on an internal node v of T resulting in a neighbor tree T' is depicted in Figure 3. Observe that T and T' have the same clusters, except that T' does not contain cluster $\mathcal{L}(T_v) = \mathcal{L}(T_x) \bigcup \mathcal{L}(T_y)$ and that T' has a cluster $\mathcal{L}(T'_{v'}) = \mathcal{L}(T'_z) \bigcup \mathcal{L}(T'_y)$ not present in T.

Prior to starting the search for the best rNNI neighbor, we execute the bottom-up assignment algorithm. Next, we traverse T in preorder, computing $f_j(T - T_v)$ for each node v of T as follows. If v is the

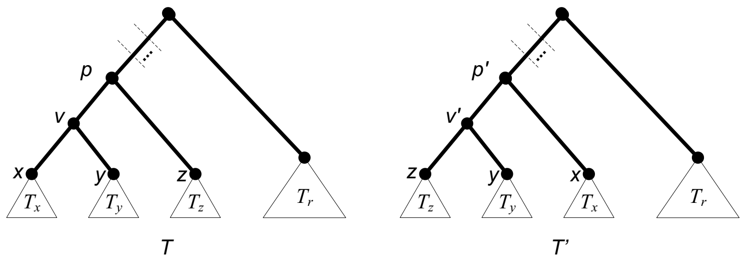


Figure 3: An rNNI operation.

root of T, then $T - T_v$ is an empty tree, denoted by $\varnothing$, and we define $f_j(\varnothing) = +\infty$. Now, suppose v has parent p and sibling z and assume that $f_j(T - T_p)$ has been correctly computed. Then,

$$f_j(T - T_v) = \min\{f_j(T - T_p), f_j(p), f_j(T_z)\}$$

Since $f_j(p)$ and $f_j(T_z)$ are known after bottom-up assignment, $f_j(T - T_v)$ can be computed in constant time. There are $2n - 1$ nodes in a binary tree T, and each visit of a node v takes constant time. Thus, the entire preorder traversal takes $O(n)$ time per character, and $O(nm)$ total.

Thus, we have $z_j(v)$, $o_j(v)$, $f_j(v)$, and $f_j(T_v)$, and $f_j(T - T_v)$ for each node v and every character j. We now show how this information can be used to obtain the flip distance of each neighbor tree in constant time per character.

By Equation (2),

$$f_j(T') = \min\{f_j(T'_v), f_j(T' - T'_v)\}. \tag{10}$$

We now argue that $f_j(T'_v)$ and $f_j(T' - T'_v)$ can be computed in $O(1)$ time, and, thus, so can $f_j(T')$.

Note that

$$\begin{aligned} f_j(T'_{v'}) &= \min\{f_j(v'), f_j(T'_z), f_j(T'_y)\} \\ &= \min\{f_j(v'), f_j(T_z), f_j(T_y)\}. \end{aligned} \tag{11}$$

The values of $f_j(T_z)$ and $f_j(T_y)$ are known, while $f_j(v')$ can be obtained in constant time using the pre-computed values of z_j and o_j for nodes z and y, and Equations (1) and (5). Thus, $f_j(T'_{v'})$ can be obtained in constant time.

On the other hand,

$$\begin{aligned} f_j(T' - T'_{v'}) &= \min\{f_j(T' - T'_p), f_j(p'), f_j(T'_x)\} \\ &= \min\{f_j(T - T_p), f_j(p'), f_j(T_x)\}. \end{aligned} \tag{12}$$

The values of $f_j(T - T_p)$ and $f_j(T_x)$ are known, while, like $f_j(v')$ above, $f_j(p')$ can be obtained in $O(1)$ time from the pre-computed information.

Hence, for each rNNI neighbor T' of T, $f_j(T')$ can be computed in $O(1)$ time and $f_M(T')$ can be computed in $O(m)$ time for a matrix of m characters. Thus, the best rNNI neighbor can be found in time $O(nm)$.

SPECTRAL PARTITIONING OF PHYLOGENETIC DATA SETS BASED ON COMPATIBILITY

A paper published in Systematic Biology, [1]

Duhong Chen [2 3] , J. Gordon Burleigh [4 5] and David Fernández-Baca [2]

Abstract

We describe two new methods to partition phylogenetic data sets of discrete characters based on pairwise compatibility. The partitioning methods make no assumptions regarding the phylogeny, model of evolution, or characteristics of the data. The methods first build a compatibility graph, in which each node represents a character in the data set. Edges in the compatibility graph may represent strict compatibility of characters or they may be weighted based on a fractional compatibility scoring procedure that measures how close the characters are to being compatible. Given the desired number of partitions, the partitioning methods then seek to cluster together the characters with the highest average pairwise compatibility, so that characters in each cluster are more compatible with each other than they are with characters in the other cluster(s). Partitioning according to these criteria is computationally intractable ($\mathcal{NP}$-hard); however, spectral methods can quickly provide high-quality solutions. We demonstrate that the spectral partitioning effectively identifies characters with different evolutionary histories in

[1]Reprinted with permission of Systematic Biology, Volume 56, Issue 4, pages 623 – 632, August 2007
[2]Department of computer science, Iowa State University, Ames, IA, 50011, USA
[3]Primary researcher and author
[4]NESCent, Durham, NC 27705, USA
[5]Author for correspondence

simulated data sets, and it is better at highlighting phylogenetic conflict within empirical data sets than previously used partitioning methods.

Introduction

The treatment of conflict within data sets is among the most debated questions in phylogenetics (e.g., Bull et al 1993; Chippindale and Wiens 1994; De Queiroz et al 1995; Farris et al 1994; Huelsenbeck et al 1996b; Cunningham 1997; Thornton et al 2000; Wilgenbusch and De Queiroz 2000; Brandley et al 2005), and it is an especially relevant concern with the prevalence of large data sets that combine data from many sources. For example, recent genomic-scale phylogenetic analyses have called attention to tremendous variation in the phylogenetic signal among loci (Bapteste et al 2002; Rokas et al 2003; Driskell et al 2004). Phylogenetic conflict may be caused by heterogeneity in the process of evolution, such as differing relative branch lengths among characters across the phylogeny (e.g., Rokas et al 2003). Alternately, characters in a data set may have different phylogenies due to hybridization, recombination, lineage sorting, or horizontal transfer (e.g., Doyle et al 2003; Takahashi et al 2001; Lerat et al 2003). To understand variation in the processes of evolution within a data set and to consider the effects of conflicting phylogenetic signals within a data set, it is first necessary to identify and describe the phylogenetic conflict within a data set.

Perhaps the most common strategies to reveal phylogenetic conflict within a data set compare phylogenies from non-overlapping sets of characters. For molecular data, partitioning characters by locus or genome, codon position, coding and non-coding regions, or stem and loop regions may reveal evidence of phylogenetic conflict (e.g., Brandley et al 2005). These biologically defined partitioning strategies require some knowledge regarding the characteristics of the data that are likely to reveal conflict, and they may miss unexpected patterns of conflict within the data. Other partitioning methods seek to cluster characters based on their rates of evolution (Brinkman and Philippe 1999; Hirt et al 1999; Burleigh and Mathews 2004; Pisani 2004) rather than conflicting phylogenetic signals. Furthermore, the methods of Brinkman and Philippe (1999) and Burleigh and Mathews (2004) require some knowledge of the phylogeny, and Burleigh and Mathews' (2004) method also assumes a model of evolution.

The goal of this paper is to describe a fast approach for partitioning phylogenetic data sets that explicitly seeks to highlight conflicting phylogenetic signals among characters within a data set. Additionally, we want partitioning methods that do not require assumptions regarding the phylogeny or process of evolution. We introduce two related partitioning methods that partition discrete phylogenetic characters based on pairwise compatibility. The methods first require building a compatibility graph for the data, in which each character is represented by a node and the edges or edge weights connecting nodes are based on pairwise compatibility. Next, given a specified number of partitions, the methods estimate the clusters of characters with the highest pairwise compatibility (or average edge weight), so that on average, characters in each cluster are more compatible with each other than with characters in the other cluster(s). Partitioning according to these criteria is computationally intractable ($\mathcal{NP}$-hard; Shi and Malik 2000) and thus requires heuristic solutions. We describe and implement spectral methods that can quickly estimate the optimal partitions based on pairwise compatibility. We demonstrate that spectral partitioning methods effectively identify characters with different phylogenies in simulated data sets, and they identify greater phylogenetic conflict than previously used partitioning methods in selected empirical data sets.

Methods

Our partitioning methods use compatibility to cluster phylogenetic characters. A set of characters C is said to be *compatible* if it admits a perfect phylogeny, or in other words, if C can evolve on a single topology without homoplasy (Le Quesne 1969). If C is compatible, all characters in C must be pairwise compatible; however, if there are more than two character states and/or missing data, sets of pairwise compatible characters are not necessarily compatible (see Felsenstein 2004). Our goal is to partition a set of discrete characters into two or more clusters so that characters in the same cluster are more compatible with each other than they are to characters in different clusters. We use strict compatibility, as well as a fractional compatibility scheme that measures the degree of compatibility between pairs of characters. The latter is useful in situations where strict pairwise compatibility is unlikely, such as when characters originate from large sets of taxa or the characters have fast rates of evolution. We cluster only parsimony-informative characters, since characters that are not parsimony informative are compatible with all other characters and will fit equally well into any cluster. The implementation of our partitioning methods removes the uninformative sites prior to partitioning and places them in a separate cluster.

The Pairwise Character Compatibility Graph

The *pairwise character compatibility graph* (PCCG) of a set of characters$C$, denoted by G_C, is a complete edge-weighted graph whose vertex set is C (see Fig. 1). We consider two versions of the PCCG. In the 0-1, or *strict compatibility*, version, the weight of an edge (c_1, c_2) in G_C is 1 if c_1 and c_2 are compatible and 0 otherwise. In the more general setting, or the *fractional compatibility* version, the weight of edge (c_1, c_2) is a number between 0 and 1 that reflects the degree of compatibility between c_1 and c_2. Note that, in practice, we remove all edges whose weight is 0 from the PCCG.

To construct the 0-1 version of PCCG, we first need to test the compatibility of every pair of characters c_1 and c_2. To this end, define the *state intersection graph* (SIG) of c_1 and c_2 as the bipartite graph $H(c_1,c_2) = (V_1,V_2,E)$, where V_1 and V_2 correspond to the states of character c_1 and c_2 respectively. More specifically, for $i = 1$ or 2, suppose c_i has character states 1 through r_i, and let S_{ij} denote the set of all taxa with state j on character i. Then, V_i consists of vertices $S_{i1},\dots,S_{ir_i}$. There is an edge between $S_{1p}(1 \leq p \leq r_1)$ and $S_{2q}(1 \leq q \leq r_2)$ if and only if S_{1p} and S_{2q} have at least one taxon in common (see Fig. 1a). It is known that two characters c_1 and $c_2$are compatible if and only if $H(c_1,c_2)$ is acyclic (Estabrook and McMorris 1977; Setubal and Meidanis 1997; Fig. 1a). This can be tested in time linear in the size of $H(c_1,c_2)$.

In the more general case of the PCCG, the weight of an edge is given by a fractional character compatibility score, denoted by $fcc(c_1,c_2)$, which measures the degree of compatibility between c_1 and c_2, based on parsimony. The value of $fcc(c_1,c_2)$ is given by

$$fcc(c_1, c2) = \frac{q-p}{q} = 1 - \frac{p}{q}$$

where p is the minimum number of edges whose removal from $H(c_1,c_2)$ yields an acyclic graph and $q = r_1r_2 - (r_1 + r_2 - 1)$, where, as before, r_1 and r_2 are the number of states in c_1 and c_2, respectively. Note that q is precisely the number of edges that need to be removed from a complete bipartite graph with r_1 and $r2$ vertices in each part in order to make it acyclic. Hence, q is an upper bound on p, and thus, $fcc(c_1, c_2)$ is a number between 0 and 1.

The following result, which is proved in the appendix, shows the relationship between $fcc(c_1, c_2)$, parsimony, and compatibility. Let $pars(c_1, c_2)$ be the parsimony score of characters c_1 and c_2, that is, the smallest number of state changes in a phylogeny for c_1 and c_2.

Theorem 1: *Let p be the number of edges that need to be removed from $H(c_1, c_2)$ to make it acyclic, for some pair of characters c_1 and c_2 with r_1 and r_2 states, respectively. Then, $p = pars(c_1, c_2) - (r_1 + r_2 - 2)$.*

Thus, if c_1 and c_2 are compatible, $p = 0$, and $fcc(c_1, c_2) = 1$. On the other hand, the highest degree of incompatibility between c_1 and c_2 occurs when these characters jointly exhibit all possible combinations of states. In this case $H(c_1,c_2)$ is a complete bipartite graph. Thus, $p = q$ and $fcc(c_1, c_2) = 0$. Theorem 1 was independently proven by Bruen and Bryant (2007), who refer to the value p as the *refined incompatibility score* of characters c_1 and c_2 (see also Bruen et al 2006; Althaus and Naujoks 2006).

Graph Partitioning

The goal of graph partitioning is to divide the vertices of a graph, which for our purposes will be a PCCG, into clusters subject to size or balance constraints while minimizing the weight of the interconnections among those clusters (see Fig. 1b). Formally, let $TRIALRESTRICTION$be an edge-weighted undirected graph, where V is the set of nodes, E is the set of edges, and w is the weight function having a number associated with each edge. A two-way partition (V_1, V_2) of V, where $V_1 \cup V_2 =$

V and $V_1 \cap V_2 = \emptyset$, is called a *cut*. The sets of vertices V_1 and V_2 can be separated by removing the edges connecting them. The *value* of *cut* (V_1, V_2) is:

$$cut(V_1, V_2) = \sum_{u \in V_1, v \in V_2} w(u, v)$$

A *minimum cut*(or min-cut for short) is a cut that has minimum value among all cuts in G. While a min-cut in G can be found in polynomial time (Hao and Orlin 1992; Karger and Stein 1996; Stoer and Wagner 1997), the min-cut tends to be extremely unbalanced, in that one of the parts may contain very few nodes (Shi and Malik 2000). These extraneous nodes may be viewed as outliers in the character partition. To make the partition more meaningful, it is useful to modify the scoring of a cut so that a low score implies some balance between the weights of the edges crossing the cut and the sizes of the resulting clusters. One way to do this was proposed by Shi and Malik (2000). They defined the *normalized* value of a cut (V_1, V_2) as

$$Ncut(V_1, V_2) = cut(V_1, V_2) \times \left(\frac{1}{c(V_1)} + \frac{1}{c(V_2)} \right)$$

where $c(V_1) = \sum_{u \in V_1, v \in V} w(u, v)$ and $c(V_2) = \sum_{u \in V_2, v \in V} w(u, v)$. The goal is to find a cut with minimum normalized value. For example, consider the 0-1 PCCG in Figure 1b, and the $cut(V_1, V_2)$ where $V_1 = \{c_1, c_2, c_3, c_4\}$ and $V_2 = \{c_5, c_6, c_7\}$. Then, $c(V_1 = 9$, $c(V_2) = 8$, and $cut(V_1, V_2) = 3$. Thus, $Ncut(V_1, V_2) = 17/24$.

The notion of a two-way partitioning of a graph can be generalized to an arbitrary number of partitions. A *multi-way cut* of a graph $G = (V, E, w)$ is a partition $\Delta = \{V_1, V_2, \ldots, V_K\}$ of V into the disjoint nonempty sets $V_1, V_2, \ldots, V_K$. The normalized value of a multi-way cut is defined as follows (Meila and Xu 2003; Yu and Shi 2003):

$$MNcut(\Delta) = \sum_{p=1}^{K} \sum_{q=p+1}^{K} Ncut(V_p, V_q)$$

The problem of finding a K-way cut with minimum normalized value is $\mathcal{NP}$-hard for $K \geq 2$ (Meila and Xu 2003). Nevertheless, spectral clustering can be used to obtain partitions with a small normalized cut value (Ding et al 2001; Shi and Malik 2000; Yu and Shi 2003). We outline the spectral method next.

Spectral Clustering

The "spectral method" for clustering is actually a class of closely related techniques. These methods usually involve taking the top eigenvectors of some matrix based on similarity between objects and then using them to cluster objects. By "top" eigenvectors, we mean the ones corresponding to either the biggest or smallest eigenvalues, depending on the formulation. In our case, they correspond to the smallest eigenvalues.

The starting point is a matrix W that represents the PCCG G of a set of characters as follows. Let n be the number of vertices G. Then W is an n by n matrix, where, for each pair of nodes u, v in G, W_{uv} is 0 if there is no edge between u and v or if $u = v$; otherwise W_{uv} is the weight of edge (u, v). Let $d_v = \sum_{u \in V} W_{uv}$ denote the *volume* of the vertex v, and D be the diagonal matrix with the (v, v)-th element having value d_v. The *Laplacian matrix* of G is $L = I - D^{-1/2} W D^{-1/2}$, where I is the identity matrix . Shi and Malik (2000) showed that the eigenvector corresponding to the second smallest eigenvalue of L can be used to approximate the cut of $G$with minimum normalized value.

Having the pairwise compatibility matrix of characters, we can apply a general multi-way spectral clustering algorithm to partition $\mathcal{C}$. Here we use the algorithm described by Ng et al (2001), whose robustness to noise has been demonstrated experimentally by Verma and Meila (2003).

Algorithm for Character Partitioning

Input: A set of characters $\mathcal{C}$, and an integer K.

Output: A partition $\Delta = \{C_1, C_2, \ldots, C_K\}$ of $\mathcal{C}$.

(1) Construct the PCCG of $\mathcal{C}$ and its corresponding matrix W.

(2) Compute the Laplacian matrix $L = I - D^{-1/2} W D^{-1/2}$

(3) Let $x_1, x_2, \ldots, x_K$ be eigenvectors corresponding to the K smallest eigenvalues. Form the matrix $X = \{x_1, x_2, \ldots, x_K\}$ by stacking the eigenvectors in columns.

(4) Form the matrix Y from X by renormalizing each of X's rows to have unit length (i.e. $Y_{ij} = \frac{X_{ij}}{\sqrt{\sum_j X_{ij}^2}}$).

(5) Treating each row of Y as a point in K dimensions, cluster those points using the K-means

algorithm to obtain K clusters (Macqueen 1967).

A program that implements spectral partitioning for sets of characters is available at the following web site: http://pilin.cs.iastate.edu/public/spectral. Our implementation uses the Lanczos algorithm (Cullum and Willoughby 2002) to obtain the eigenvectors corresponding to the K smallest eigenvalues of the Laplacian matrix of G.

Simulation Experiments

We first tested the ability of the spectral partitioning methods to detect sites with different phylogenies using simulated data sets. In each simulation, we obtained two random trees using the default parameters of the YULE_C simulate procedure from r8s, which produced random topologies with a root-tip length of 1.0 (Sanderson 2003). Alignments of 500 base pairs (bp) in length were generated based on each random tree using the Monte Carlo simulation method in Seq-Gen (Rambaut 1997). Thus, each simulation generates a 1000 bp alignment; the first 500 bp evolving from the first tree, and the second 500 bp evolving from the second tree. The simulations used a Jukes and Cantor (1969) model of sequence evolution, which assumes equal nucleotide frequencies and equal frequencies of all substitutions, and rate variation among sites was modeled by a discrete gamma distribution with four rate categories and a α shape parameter value of 0.5 (Yang 2003). We performed ten simulation replicates under two different protocols that were designed to produce different overall levels of pairwise compatibility. In the first simulation protocol, ten replicates were generated using 25-taxon trees with the branch lengths from r8s scaled by a factor of 0.1. In the second protocol, another ten replicates were generated using 100-taxon trees using the r8s branch lengths. A program that creates simulated data sets following our protocols is available at http://pilin.cs.iastate.edu/public/spectral.

For each simulation replicate, we partitioned the simulated characters into two clusters using both the strict and fractional compatibility spectral partitioning methods. We examined how accurately each partitioning method clusters nucleotides based on their evolutionary history (the tree used to simulate the data). We also calculated the incongruence length difference (ILD) for each partitioned data set. The ILD is the difference in the lengths of the most parsimonious tree for the entire data set minus the sum of the lengths of the most parsimonious trees from each partition (Mickevich and Farris 1981; Farris et al 1994). The ILD measures the extent to which two data sets result in different trees, with a higher score indicating greater incongruence, or more conflict, among the data partitions.

Empirical Examples

We also examined the performance of the spectral partitioning methods using two empirical data sets in which partitioning has revealed evidence of conflicting phylogenetic signals. The first example is a seed plant data set from Sanderson et al (2000) containing sequences from two plastid genes, *psaA* and *psbB,* from 19 taxa. This alignment contains 1165 parsimony informative characters. In the original study, Sanderson et al (2000) partitioned sites from the first and second codon positions from the third codon position sites, and separate parsimony analyses of the two partitions resulted in very different phylogenetic trees. The second example is a 13-locus, 25-taxon seed plant data set from Burleigh and Mathews (2004), which contains 5600 parsimony informative characters. This study partitioned the data according to rate class assignments based on a discrete gamma distribution with eight categories. The sites in the two fastest rate classes (the *RC78* sites) compose one partition, and the other sites (the *RCslow* sites) compose the other partition. Again, parsimony analyses of each partition support conflicting seed plant hypotheses (Burleigh and Mathews 2004). We used both spectral partition methods on the parsimony informative characters from the two empirical data sets. We compared the ILD values from the spectral partitioning and the previously used partitioning methods. We also performed a partition homogeneity (or ILD) test to examine the significance of conflict among the partitions (Farris et al 1994). For the ILD test, we examined only the parsimony informative sites, using 1000 randomly sampled partitions and a maximum parsimony heuristic with 10 random sequence addition replicates and TBR branch swapping as implemented in PAUP* (Swofford 2002). We also compared the *density*, or the average pairwise compatibility among sites, of the different partitions. The density of a set of characters may be defined using either strict or fractional compatibility. The *0-1 density* is simply the average pairwise compatibility from the 0-1, or strict compatibility, version of the PCCG. If the 0-1 density is 0.5, then on average each character is compatible with half of the other characters in the partition; if the 0-1 density is 1, then all characters are pairwise compatible, and the PCCG is a clique. For the similarity scoring version of the PCCG, *the fractional compatibility density* is the sum of all edge weights in a partition divided by the maximum possible total weight of the edges, which is $n \times (n-1)/2$, where n is the number of nodes (characters) in the partition. The fractional density is the average edge weight among characters in a partition.

Results

Simulation Experiments

The simulation experiments demonstrate the ability of spectral partitioning methods to cluster sites based on their evolutionary history. The partitions largely corresponded to the characters' simulated topology (Tables 1, 2). In the 25-taxon simulations, the spectral partitioning using strict and fractional compatibility give similar results (Tables 1a, 2a). On average across all ten replicates, 91.7% of the informative characters in a single strict compatibility partition come from one of the simulated topologies, and 92.1% of the informative characters in a single fractional compatibility partition come from one of the simulated topologies (Table 1a). The estimated parsimony scores obtained by analyzing each partition separately exceed the estimated parsimony scores obtained by analyzing the true partitions separately by an average of 0.7% and 0.4% in the strict and fractional compatibility partitions respectively (Table 2a). The ILD for strict compatibility ranges from 186 to 324 (ave. = 262.5) and from 186 to 331 (ave. = 264.7) for the fractional compatibility partitions (Table 2a).

In the 100-taxon simulations with no branch length scaling, overall spectral partitioning using fractional compatibility outperforms spectral partitioning using strict compatibility (Tables 1b, 2b). On average across all ten replicates, 85.3% of the informative characters in a single strict compatibility partition come from one simulated topology, and 97.3% of the informative characters in a single fractional compatibility partition come from one simulated topology (Table 1b). Also on average, the estimated parsimony scores obtained by analyzing each partition separately exceed the estimated parsimony scores obtained by analyzing the true partitions separately by 4.3% and 0.3% in the strict and fractional compatibility partitions respectively (Table 2b). The ILD ranges for strict compatibility ranges from 88 to 4081 (ave. = 2922.5) and from 3384 to 4130 (ave. = 3703.6) for the fractional compatibility partitions (Table 2b). The spectral partitioning analyses of the simulated data sets took approximately 2 seconds CPU time for the 25-taxon data sets and 25 seconds CPU time for 100-taxon data sets under Linux on an Intel Pentium 4 3.0 GHz processor with 1 GB memory.

Empirical Examples

In both the 2-locus data set of Sanderson et al (2000) and the 13-locus data set of Burleigh and Mathews (2004), both spectral partitioning methods highlight more conflict within the data set than prior partitioning methods. In the *psaA* and *psbB* data set of Sanderson et al (2000), the ILD of the parsimony informative sites when partitioning first and second versus third codon positions is 34, and an ILD (or partition homogeneity) test did not reject the null hypothesis of no heterogeneity ($p = 0.213$). The ILD using spectral partitioning with strict compatibility is 98, and it is 86 using fractional compatibility. Both PCCG versions of the spectral partitioning produce partitions that reject the null hypothesis of no heterogeneity in the partition homogeneity test ($p \leq 0.001$ for both). The average 0-1 density and fractional density of characters using the two spectral partitioning methods are nearly equal, and they exceed the average densities from the codon partitioning method (Table 3a).

In the 13-locus data set of Burleigh and Mathews (2004), the ILD of the parsimony informative sites when partitioning the RC78 sites versus the RCslow sites is 142. The ILD using spectral partitioning with strict compatibility is 226, and it is 225 using fractional compatibility. All three partitioning methods produce partitions that reject the null hypothesis of no heterogeneity using the partition heterogeneity test ($p \leq 0.001$ for all). Again, the average 0-1 density and fractional density of the clusters using the two spectral partitioning methods are nearly equal (Table 3b). Though the 0-1 and fractional density values are highest for the RCslow partition than for any other partition, the average densities from the spectral methods exceed those from the rate class partitioning (Table 3b). All spectral partitioning analyses of empirical data were done under Linux on an Intel Pentium 4 3.0 GHz processor with 1 GB memory. The spectral partitioning of the *psaA* and *psbB* data set of Sanderson et al (2000) with 1165 parsimony informative characters used approximately 33 seconds CPU-time, and the spectral partitioning of the 13-locus data set with 5600 parsimony informative characters took just over 14 minutes (848 seconds) CPU-time.

Discussion

This study describes how a fast and powerful partitioning method from computer science (e.g., Ding et al 2001; Shi and Malik 2000; Yu and Shi 2003) can be used to cluster phylogenetic characters based on compatibility. Our simulation experiment demonstrates that partitioning characters based on our compatibility criterion performs well in identifying conflicting evolutionary histories within data sets. Furthermore, the empirical examples show that partitioning based on compatibility may help reveal previously undetected levels of phylogenetic conflict within data sets.

Partitioning and Compatibility

Compatibility and compatibility graphs have a long history in phylogenetics (see Meacham and Estabrook 1985; Semple and Steel 2003; Felsenstein 2004). While today compatibility criteria are seldom used to estimate phylogenies, compatibility has been used to weight characters for parsimony analyses (Penny and Hendy 1985, 1986; Sharkey 1989) and, more recently, to identify anomalous and fast-evolving characters in data sets (Meacham 1994; Pisani 2004). These methods are based on the notion that fast evolving characters experience more homoplasy than slowly evolving characters and thus are pairwise compatible with fewer characters than slowly evolving characters.

The lack of pairwise compatibility among fast evolving characters suggests a potential weakness for partitioning data based on compatibility. If data sets contain characters from many taxa or only fast evolving characters, there may be little pairwise compatibility among characters, and the strict, or 0-1, compatibility graph may have a very low density. In such cases it may be difficult to partition characters based on strict compatibility. Spectral partitioning using fractional compatibility addresses this concern by relaxing the strict compatibility criterion. Characters are evaluated based on how close to they are to being compatible, and in theory, no two characters need be compatible for this method to work. If there is little pairwise compatibility among characters, we might expect spectral partitioning using fractional compatibility to perform better than spectral partitioning based on strict compatibility. We see this in the simulation experiment when spectral partitioning with fractional compatibility generally outperforms strict compatibility partitioning in the replicates with larger (100-taxon) trees and longer (unscaled) branch lengths (Table 1).

Unlike previous methods, the spectral partitioning methods use pairwise compatibility to estimate shared phylogenetic signal, not rates of evolution. They implicitly assume that incompatible characters are more likely to have conflicting phylogenetic signals than compatible characters. Perhaps most similar to our methods, Pisani (2004) used compatibility to identify the fastest evolving sites using a randomization test that compares a character's observed overall pairwise compatibility with the overall pairwise compatibility that would be expected if the states of the character states were random. A justification of Pisani's (2004) method is the idea that the fastest evolving sites are most likely to contribute to long-branch attraction effects, suggesting that differences in rate of evolution may produce major conflicts within the data set. Removing or otherwise down-weighting fast-evolving sites appears to be effective in some phylogenetic analyses (e.g., Philippe et al 2000; Burleigh and Mathews 2004; Dutilh et al 2004; Pisani 2004), but it is easy to imagine cases in which notable phylogenetic conflict exists among fast or slowly evolving sites. The spectral partitioning methods attempt to directly partition sites based on phylogenetic conflict without assumptions regarding the source of the conflict. However, this goal is not necessarily incompatible with Pisani's (2004) method, and Pisani's (2004) method may even enhance the performance of spectral partitioning. If a fast evolving character is phylogeneticly uninformative or random, then it cannot be clustered in an informed manner based on compatibility. Thus, it may be useful to identify and remove such characters prior to spectral partitioning.

Performance of Spectral Partitioning

Our goal in partitioning phylogenetic data sets is to reveal conflicting evolutionary histories within the data set, whether due to different substitution processes or phylogenies. The spectral methods make it possible to estimate the optimal partitioning of a data set based on pairwise compatibility in a reasonable amount of time. Yet the effectiveness of our spectral methods in partitioning depends on how well the compatibility criterion reveals conflicting phylogenetic signals. In the simulated data sets, spectral partitioning using the compatibility criterion performs well in predicting which characters evolved from the same or different topologies (Table 1). Furthermore, if the ILD scores are indicative of the degree of conflict among partitions, spectral partitioning performs nearly as well at revealing conflict as knowing the actual underlying topologies. The combined parsimony score of the best spectral partitions never exceeds the estimate of the parsimony score of the true (tree) partition by more than 1.5%, and in one case, the estimate of parsimony score of the best spectral partition is better than the true (tree) partition (Table 1b). Since both topologies in the simulation study were generated randomly, the simulations represent an extreme case of phylogenetic conflict among sites in the alignment. We expect little pairwise compatibility among characters evolving in the different random trees, and therefore, we might expect spectral partitioning to perform well. Still, with any two topologies, at least some sites likely will be compatible with both topologies, and thus it may be extremely difficult to accurately partition every single character. We also caution against interpreting seemingly significant partitions as evidence of multiple phylogenies underlying the data. For example, in some data sets simulated from a single phylogeny with among-site rate variation, spectral partitioning may yield partitions that reject the ILD test (data not shown). This is consistent with the observation that rate variation among sites can produce apparent heterogeneity in the phylogenetic signal that is detected by the ILD test (e.g., Dolphin et al 2000; Barker and Lutzoni 2002).

In the empirical data sets, compatibility also appears to reveal higher levels of phylogenetic conflict than previously used partitioning methods. The much higher ILD scores from spectral partitioning coincide with higher average compatibility among sites in the spectral partitions. Though a phylogenetic analysis of the different codon position partitions in the *psaA* and *psbB* data set produces different phylogenetic trees (Sanderson et al 2000), an ILD test does not detect significant heterogeneity in the codon partitions. However, the ILD test detects significant heterogeneity among the spectral partitions. A χ^2 test uncovers no significant correspondence between the codon partition and the spectral partitions (Snedecor and Cochran 1989; Table 4). In the 13-locus data set, all partitions produce significant ILD tests, and a χ^2 test demonstrates a significant correspondence between the rate class and spectral partitions (Table 4; $\chi^2 = 6.50$ for strict compatibility; $\chi^2 = 18.23$ fractional compatibility; $p < 0.01$ for both). The empirical examples demonstrate that previously used partitions will not necessarily fail to find conflict; however, they may not relate to the major conflicts in the data set.

Implementation of Spectral Partitioning

While spectral partitioning based on compatibility appears to effectively highlight phylogenetic conflict within a data set, there are still questions regarding its performance and implementation. For example, the spectral partitioning methods do not require complete data sets (e.g., the data set of Burleigh and Mathews (2004) has ~35% missing data), but it is difficult to assess character compatibility with incomplete data. Thus, incomplete data may compromise the performance of the spectral partitioning methods. Also, spectral partitioning seeks to balance the size of partitions, and consequently, it may not perform well if phylogenetic conflicts are caused by a small percentage of the total characters.

An important issue in using spectral partitioning is to determine the optimal number of partitions. For simplicity, our examples use only two partitions; however, one can specify any number of partitions. There are numerous tests and methods to determine if data from different partitions represent significant phylogenetic conflict, some of which also can address the optimal number of partitions (e.g., Farris et al 1994; Huelsenbeck et al 1996a; Mason-Gamer and Kellogg 1996; Dolphin et al 2000; Kauff and F. 2002; Vogl et al 2003; Ané et al 2005; Brandley et al 2005). In one example, Ané et al (2005) suggest an information theoretic approach to determine the optimal number of partitions. In this approach, the optimal partitioning scheme is the one that minimizes the descriptive complexity, or the compressed file size, of the data and the trees resulting from the partitions. The optimal number of partitions is approximately the largest number of partitions for which $\Delta L \geq (l - 1) \times n$, where ΔL is the change in the sum of the lengths of the most parsimonious trees from each partition, l is the number of binary trees (or partitions), and n is the number of leaves in the binary tree (see Ané et al 2005). In other words, one should add an additional partition if it reduces the sum of the lengths of the most parsimonious trees from each partition by a number that is at least as high as the number of leaves in the tree. This approach also demonstrates that spectral partitioning may provide good solutions for estimating the partition that globally minimizes the compression among all possible partitions.

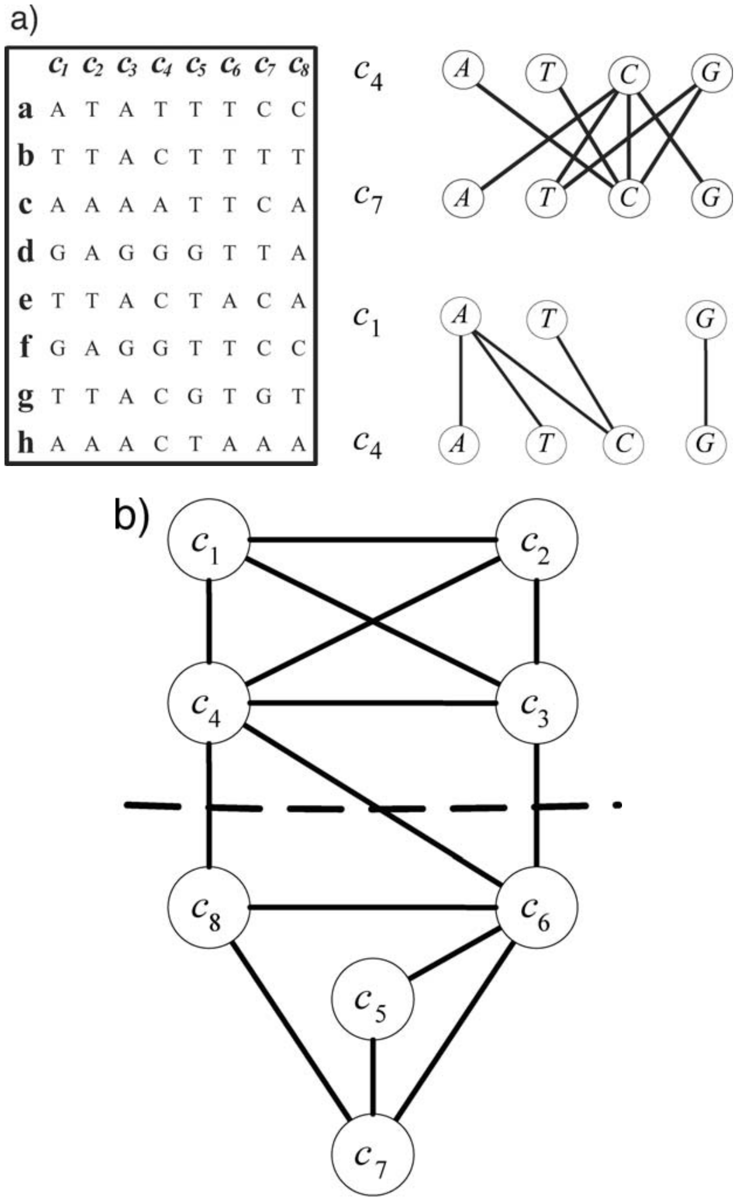
There are many opinions on how one should treat conflict in phylogenetic analyses (e.g., Kluge 1989; Bull et al 1993; Chippindale and Wiens 1994; De Queiroz et al 1995; Huelsenbeck et al 1996b; Nixon and Carpenter 1996; Brandley et al 2005), and we do not attempt to resolve this issue. However, the ability of spectral partitioning to reveal previously undetected levels of conflict within a data set make this discussion especially important. If the conflicting characters are interspersed within a linked region, it is highly unlikely that the conflict reflects different phylogenies. In such cases, it is popular to apply different substitution models to each partition (e.g., Nylander et al 2004; Brandley et al 2005). However, the spectral partitions are based on compatibility, not conflicts in substitution models or parameters. Therefore, while characters in different partitions may have different branch lengths, there is no reason to assume they will have different optimal substitution models or substitution parameter values.

The spectral partitioning method may be extended in several directions. While we use spectral partitioning to cluster characters based on compatibility, the data could be clustered based on other characteristics, such as shared rates of evolution. Furthermore, in some cases it may be useful to incorporate biological constraints on the partitioning strategy. For example, we might not wish to combine characters from different loci into the same partition. Our examples demonstrate that spectral partitioning performs well in the absence of assumptions or regarding the characteristics of the data, and it likely can be improved by incorporating additional biological information.

Acknowledgements

We thank M. J. Sanderson and O. Eulenstein for discussion of spectral partitioning, and Davide Pisani, Barbara Holland, and Mike Steel for helpful comments on the manuscript. This study was funded by NSF grants EF-0334832 (DC and DF-B) and 0431154 (JGB) and by NESCent, NSF EF-

0423641 (JGB).



	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
a	A	T	A	T	T	T	C	C
b	T	T	A	C	T	T	T	T
c	A	A	A	A	T	T	C	A
d	G	A	G	G	G	T	T	A
e	T	T	A	C	T	A	C	A
f	G	A	G	G	T	T	C	C
g	T	T	A	C	G	T	G	T
h	A	A	A	C	T	A	A	A

Figure 1: An example of the spectral partitioning method. a) The method for determining if characters are compatible. The alignment on the left is a set of characters *C*. On the right are examples of state intersection graphs (SIG) used to determine character compatibility. The bipartite graph H(c_4,c_7) on top contains a cycle, indicating that characters c_4 and c_7 are incompatible. Below, the bipartite graph H(c_1,c_4) is acyclic, indicating characters c_1 and c_4 are compatible. b) This demonstrates spectral partitioning of a compatibility graph. The graph is a pairwise character compatibility graph (PCCG) of *C*. Edges (solid lines) exist between nodes if the characters are compatible. The dotted line is the *Ncut* that defines the partitions. Characters within each partition are more compatible with each other than they are with characters in the other partition.

a. 25 taxon trees; branch lengths from r8s scaled by 0.1.

	Strict Compatibility				Fractional Compatibility			
	Partition 1		Partition 2		Partition 1		Partition 2	
Replicate	Tree 1	Tree 2	Tree 1	Tree 2	Tree 1	Tree 2	Tree 1	Tree 2
1	124	6	19	112	124	6	19	112
2	151	13	16	149	152	11	15	151
3	145	10	12	142	151	6	6	146
4	146	6	16	151	141	7	21	150
5	104	15	17	130	108	19	13	126
6	135	16	16	123	135	14	16	125
7	144	14	7	134	146	13	5	135
8	145	10	9	144	143	8	11	146
9	129	5	6	146	129	6	6	145
10	134	24	8	128	134	24	8	128

b. 100 taxon trees; branch lengths from r8s unscaled.

	Strict Compatibility				Fractional Compatibility			
	Partition 1		Partition 2		Partition 1		Partition 2	
Replicate	Tree 1	Tree 2	Tree 1	Tree 2	Tree 1	Tree 2	Tree 1	Tree 2
1	411	27	15	388	404	4	22	411
2	358	25	72	385	427	11	3	399
3	417	5	9	417	418	1	8	421
4	392	42	35	371	417	4	10	409
5	398	12	21	418	419	12	0	418
6	251	126	176	284	376	9	51	401
7	324	88	79	329	398	18	5	399
8	386	10	28	407	382	2	32	415
9	276	254	137	160	404	7	9	407
10	402	48	28	376	409	3	21	421

Table 1: Results from partitioning simulated data sets. Each simulation replicate generated a 500 bp alignment for two randomly generated topologies. The results show the distribution of parsimony informative characters from each tree among partitions generated using spectral partitioning with strict compatibility and fractional compatibility.

a. 25-taxon Simulations

Replicate	Tree Partition	Fractional Compatibility	Strict Compatibility	No Partition
1	697	701	704	919
2	1066	1071	1078	1402
3	858	858	862	1176
4	863	871	866	1126
5	739	746	746	932
6	829	829	834	1093
7	862	867	870	1148
8	907	909	913	1144
9	736	737	737	1003
10	829	832	833	1125

b. 100-taxon Simulations

Replicate	Tree Partition	Fractional Compatibility	Strict Compatibility	No Partition
1	19795	19844	19954	23437
2	19010	19038	19449	22688
3	18948	18942	18981	22835
4	17075	17080	17475	21084
5	18699	18716	18765	22846
6	20368	20669	22607	23851
7	18499	18518	19667	22232
8	18487	18564	18618	22128
9	19554	19561	22857	22945
10	19038	19102	19472	23024

Table 2: Parsimony scores resulting from different partitioning procedures. For each simulated data set, we calculated the cumulative parsimony scores that result from separate parsimony analyses of data from each partition. We compare three partitioning schemes: the true partition separating simulated data sets from the two random trees ("Tree Partition"), spectral partitioning using fractional compatibility, and spectral partitioning using strict compatibility. The parsimony scores in the column marked "No Partition" are based on a parsimony analysis of the unpartitioned data set.

a. Sanderson et al. (2001) data.

Partition	# of chars	0-1 dens.	frac. dens.
Str. Comp.1	682	0.386	0.639
Str. Comp.2	483	0.577	0.766
Average		0.450	0.693
Frac. Comp.1	630	0.365	0.624
Frac. Comp.2	535	0.581	0.771
Average		0.455	0.703
codon12	289	0.389	0.621
codon3	876	0.409	0.660
Average		0.407	0.656

b. Burleigh and Mathews (2004) data.

Partition	# of chars	0-1 dens.	frac. dens.
Str. Comp.1	2140	0.672	0.823
Str. Comp.2	3460	0.711	0.845
Average		0.700	0.839
Frac. Comp.1	2011	0.693	0.841
Frac. Comp.2	3589	0.700	0.837
Average		0.699	0.838
RCslow	2478	0.839	0.862
RC78	3122	0.475	0.769
Average		0.615	0.818

Table 3: Density of partitions from empirical data sets. The 0-1 density ("0-1 dens.") is the average pairwise compatibility of sites in a partition, and the fractional density ("frac. dens.") is the average edge weight based on the fractional compatibility. We also present the average density of the clusters ("Average"), which is the sum of the density of each cluster multiplied by the proportion of edges in that cluster, [(density1 *edges1)+(density2*edges2)] / (edges1 + edges2). For each data set, the density is shown for the spectral partitions using strict compatibility ("Str. Comp.") and fractional compatibility ("Frac. Comp.") as well as for the previously used partitioning scheme. Table 3a shows the density for the *psaA* and *psbB* data set from Sanderson et al. (2001), and table 3b shows the density for the 13-locus data set of Burleigh and Mathews (2004).

a. Sanderson et al. (2001) data.

	Str. Comp.1	Str. Comp.2	Frac. Comp.1	Frac. Comp.2
Frac. Comp.1	578	52		
Frac. Comp.2	104	431		
Codon12	174	115	148	141
Codon3	508	368	482	394

b. Burleigh and Mathews (2004) data.

	Str. Comp.1	Str. Comp.2	Frac. Comp.1	Frac. Comp.2
Frac. Comp.1	1912	99		
Frac. Comp.2	228	3361		
RCslow	993	1485	966	1512
RC78	1147	1975	1045	2077

Table 4: Similarity of different partitioning schemes in empirical data sets. These tables show the overlapping number of sites in the different partitions. Table 4a shows the overlap of partitions for the *psaA* and *psbB* data set from Sanderson et al. (2001), and table 4b shows the overlap of partitions for the 13-locus data set of Burleigh and Mathews (2004). In each data set, there are three partitioning schemes: spectral partitioning based on strict compatibility ("Str. Comp.") and fractional compatibility ("Frac. Comp.") as well as the previously used partitioning method.

References

Althaus Ernst, Naujoks Rouven. 2006. Computing steiner minimum trees in hamming metric. In SODA ’06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. ACM Press, New York, NY, USA, p 172–181.

Ané C, , Sanderson MJ. 2005. Missing the forest for the trees: phylogenetic compression and its implications for inferring complex evolutionary histories. *Systematic Biology*, 54:146–157.

Bapteste E., Brinkmann H, Lee J. A., Moore D. V., Sensen C. W., Gordon P., L. Duruflé, Gaasterland T., Lopez P., Müller M., H. Phillipe. 2002. The analysis of 100 genes supports the grouping of three highly divergent amoeba: *Dictyostelium*, *Entamoeba*, and *Mastigamoeba*. In Proc. Natl. Acad. Sci. p 1414–1419.

Barker F. K., Lutzoni F. M.. 2002. The utility of the incongruence length difference test. *Systematic Biology*, 51:625–637.

Brandley M. C., Schmitz A., Reeder T. W.. 2005. Partitioned bayesian analyses, partition choice, and phylogenetic relationships of scincid lizards. *Systematic Biology*, 54:373–390.

Brinkman H., Philippe H.. 1999. Archaea sister group of bacteria? indications from tree reconstruction artifacts in ancient phylogenies. *Mol. Biol. Evol*, 16:817–825.

Bruen T., Bryant D.. 2007. A subdivision approach to maximum parsimony. *Annals of Combinatorics*.

Bruen T., Philippe H., Bryant D.. 2006. A quick and robust statistical test to detect the presence of recombination. *Genetics*, 172:2665–2681.

Bull J. J., Huelsenbeck J. P., Cunningham C. W., Swofford D. L., Waddell P. J.. 1993. Partitioning and combining data in a phylogenetic analysis. *Systematic Biology*, 42:384–397.

Burleigh J. G., Mathews S.. 2004. Phylogenetic signal in nucleotide data from seed plants: implications for resolving the seed plant tree of life. *Am. J. Bot.*, 91:1599–1613.

Chippindale P. T., Wiens J. J.. 1994. Weighting, partitioning, and combining characters in phylogenetic analysis. *Systematic Biology*, 43:278–287.

Cullum J. K., Willoughby R. A.. 2002. Lanczos algorithms for large symmetric eigenvalue computations. In Society for Industrial and Applied Mathematics. Philadelphia, PA, USA.

Cunningham C. W.. 1997. Is congruence between data partitions a reliable predictor of phylogenetic accuracy? empirically testing an iterative procedure for choosing among phylogenetic methods. *Systematic Biology*, 46:432–437.

De Queiroz K., Donoghue M. J., Kim J.. 1995. Separate versus combined analysis of phylogenetic evidence. *Annu. Rev. Ecol. Syst.*, 26:657–681.

Ding Chris H. Q., He Xiaofeng, Zha Hongyuan, Gu Ming, Simon Horst D.. 2001. A min-max cut algorithm for graph partitioning and data clustering. In ICDM ’01: Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, p 107–114.

Dolphin K., Belshaw R., Orme C. D. L., Quicke D. L. J.. 2000. Noise and incongruence: interpreting results of the incongruence length difference tests. *Mol. Phylogenet. Evol.*, 17:401–406.

Doyle J. J., Doyle J. L., Rausher J. T., Brown A. H. D.. 2003. Diploid and polyploid reticulate evolution throughout the history of the perennial soybeans (*Glycine* subgenus *Glycine*). *New Phytologist*, 161:121–132.

Driskell A. C., Ané C., Burleigh J. G., McMahon M. M., O’Meara B. C., Sanderson M. J.. 2004. Prospects for building the tree of life from large sequence databases. *Science*, 306:1172–1174.

Dutilh B. E., Huynen M. A., Bruno W. J., Snel B.. 2004. The consistent phylogenetic signal in genome trees revealed by reducing the impact of noise. *J. Mol. Evol.*, 58:527–539.

Estabrook G. F., McMorris F. R.. 1977. When are two qualitative taxonomic characters compatible? *J. Mathematical Biol.*, 4:195–200.

Farris J. S., Kållersjo M., Kluge A. G., Bult C.. 1994. Testing the significance of congruence. *Cladistics*, 10:315–319.

Felsenstein J.. 2004. Inferring Phylogenies. Sinauer Assoc., Sunderland, Mass.

Hao Jianxiu, Orlin James B.. 1992. A faster algorithm for finding the minimum cut in a graph. In SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, p 165–174.

Hirt R., Logsdon J. M. Jr., Healy B., Dorey M. W., Dolittle W. F., Embley T.M.. 1999. Microsporidia are related to fungi: evidence from the largest subunit of rna polymerase ii and other proteins. *Proc. Natl. Acad. Sci. USA*, 96:580–585.

Huelsenbeck, J. P.and Bull J. J., , Cunningham C. W.. 1996a. Combining data in phylogenetic analysis. *Trends Ecol. Evol.*, 11:152–157.

Huelsenbeck, J. P.and Bull J. J., Dorey M. W., Dolittle W. F., Embley T.M.. 1996b. A likelihood ratio test to detect conflicting phylogenetic signal. *Systematic Biology*, 45:92–98.

Jukes T. H., Cantor C. R.. 1969. Evolution of protein molecules in ,. Mammalian protein metabolism. Academic Press, New Youk.

Karger David R., Stein Clifford. 1996. A new approach to the minimum cut problem. *J. ACM*, 43 (4):601–640.

Kauff F., F. Lutzoni. 2002. Phylogeny of the gyalectales and ostropales (ascomycota, fungi): among and within order relationships based on nuclear ribosomal rna small and large subunits. *Mol. Phylogenet. Evol.*, 25:138–156.

Kluge A. G.. 1989. A concern for evidence and a phylogenetic hypothesis of relationships among epicrates (boidae, serpentes). *Syst. Zool.*, 38:7–25.

Le Quesne W. J.. 1969. A method of selection of characters in numerical taxonomy. *Syst. Zool.*, 18:201–205.

Lerat E., Daubin V., Moran N. A.. 2003. From gene trees to organismal phylogeny in prokaryotes: The case of the γ-proteobacteria. *PLoS Biol.*, 1:1–9.

Macqueen J. B.. 1967. Some methods of classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathemtical Statistics and Probability. p 281–297.

Mason-Gamer R. J., Kellogg E. A.. 1996. Testing for phylogenetic conflict among molecular data sets in the tribe triticeae (gramineae). *Systematic Biology*, 45:524–545.

Meacham C. A.. 1994. Phylogenetic relationships at the basal radiation of angiosperms: further study by probability of character compatibility. *Systematic Biology*, 19:506–522.

Meacham C. A., Estabrook G. F.. 1985. Compatibility methods in systematics. *Annu. Rev. Ecol. Syst.*, 16:431–466.

Meila M., Xu L.. 2003. Multiway cuts and spectral clustering. Tech. rep., U. Washington.

Mickevich M. F., Farris J. S.. 1981. The implications of congruence in *Menidia*. *Syst. Zool.*, 30:351–370.

Ng A., Jordan M., Weiss Y.. 2001. On spectral clustering: Analysis and an algorithm.

Nixon K. C., Carpenter J. M.. 1996. On simultaneous analysis. *Cladistics*, 12:221–241.

Nylander J. A. A., Ronquist F., Huelsenbeck J. P., Nieves-Aldrey J. L.. 2004. Bayesian phylogenetic analysis of combined data. *Systematic Biology*, 53:47–67.

Penny D., Hendy M.. 1986. Estimating the reliability of evolutionary trees. *Mol. Biol. Evol.*, 3:403–417.

Penny D., Hendy M. D.. 1985. Testing methods of evolutionary tree construction. *Cladistics*, 1:266–272.

Philippe H., Lopez P., Brinkmann H., Budin K., Germor A., Laurent J., Moreira D., Müller M., Le Guyader H.. 2000. Early-branching or fast-evolving eukaryotes? an answer based on slowly evolving positions. *Proc. R. Soc. Lond.*, 267:1213–1221.

Pisani D.. 2004. Identifying and removing fast-evolving sites using compatibility analysis: an example from the arthropods. *Systematic Biology*, 53:978–989.

Rambaut, A.and Grassly N. C.. 1997. Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Comput. Appl. Biosci*, 13:235–238.

Rokas A, Williams BL, King N., Carroll SB. 2003. Genome-scale approaches to resolving incongruence in molecular phylogenies. *Nature*, 425:798–804.

Sanderson M. J. 2003. r8s: inferring absolute rates of molecular evolution and divergence times in the absence of a molecular clock. *Bioinformatics*, 19:301–302.

Sanderson M. J., Wojciechowski M. F., Hu J.-M., Khan T. S., Brady S. G.. 2000. Error, bias, and long-branch attraction in data for two chloroplast photosystem genes in seed plants. *Mol. Biol. Evol.*, 17:782–797.

Semple C., Steel M.. 2003. Phylogenetics. Oxford University Press, London.

Setubal J., Meidanis J.. 1997. Introduction to Computational Molecular Biology. PWS Publishing Company, Boston USA.

Sharkey M. J.. 1989. A hypothesis-independent method of character weighting for cladistic analysis. *Cladisitics*, 5:63–86.

Shi Jianbo, Malik Jitendra. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8):888–905.

Snedecor G. W., Cochran W. G.. 1989. Statistical methods. Iowa State University Press, Ames, IA, USA.

Stoer Mechthild, Wagner Frank. 1997. A simple min-cut algorithm. *J. ACM*, 44 (4):585–591.

Swofford DL. 2002. PAUP*: Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4.0 beta. Sinauer Assoc., Sunderland, Massachusetts, USA.

Takahashi K., Terai Y., Nishida M., Okada N.. 2001. Phylogenetic relationships and ancient incomplete lineage sorting among cichlid fishes in lake tanganyika as revealed by analysis of the insertion of retroposons. *Mol. Biol. Evol.*, 18:2057–2066.

Thornton J. W., , DeSalle R.. 2000. A new method to localize and test the significance of incongruence: detecting domain shufflingin the nuclear receptor subfamily. *Syst. Biol.*, 49:183–201.

Verma D., Meila M.. 2003. A comparison of spectral clustering algorithms. Tech. rep., U. Washington, technical report uw-cse-03-05-01.

Vogl C., Badger J, Kearney P, Li M., Clegg M, Jiang T.. 2003. Probabilistic analysis indicates discordant gene trees in chloroplast evolution. *J. Mol. Evol.*, 56:330–340.

Wilgenbusch J., De Queiroz K.. 2000. Phylogenetic relationships among the phrynosomatid sand lizards inferred from mitochondrial dna sequences generated by heterogeneous evolutionary processes. *Syst. Biol.*, 49:592–612.

Yang Z. 2003. Maximizing likelihood phylogenetic estimation from dna sequences with variable rates over sites: approximate methods. *J. Mol. Evol.*, 39:306–314.

Yu S., Shi J.. 2003. Multiclass spectral clustering.

Appendix

This section contains the proof of Theorem 1, which is restated below.

Theorem 1: *Let p be the minimum number of edges that need to be removed from $H(c_1,c_2)$ to make it acyclic, for some pair of characters c_1 and c_2 with r_1 and r_2 states, respectively. Then, $p = pars(c_1,c_2) - (r_1 + r_2 - 2)$.*

To prove this result, we need two lemmas. We use the following notation. Let G be a graph. Then, $V(G)$ and $E(G)$ denote the vertex and edge sets of G, and $v(G)$ and $e(G)$ denote the number of vertices and edges of G.

Lemma 1: *If $H(c_1,c_2)$ is connected, $pars(c_1,c_2) = e(H(c_1,c_2)) - 1$.*

Proof: We first show that $pars(c_1,c_2) \geq e(H(c_1,c_2)) - 1$. Let P be any phylogeny for characters c_1 and c_2. Since P contains $e(H(c_1,c_2))$ distinct state pairs, and each state pair induces a unique subtree, there must be a total of at least $e(H(c_1,c_2)) - 1$ character state changes for characters c_1 and $c_2$in P, that is, $pars(c_1,c_2) \geq e(H(c_1,c_2)) - 1$.

Next, we argue that $pars(c_1,c_2) \leq e(H(c_1,c_2)) - 1$. Since $H(c_1,c_2)$ is connected, it has a spanning tree T. Let $E' = E(H(c_1,c_2)) - E(T)$. The edges in T correspond to a subset of the taxa S' such that characters c_1 and c_2 restricted to S' are compatible. These characters therefore have a phylogenetic tree $P'$with $v(T) - 2 = e(T) - 1$ state changes.

We can extend subtree P' to a tree P containing all of the taxa including those appearing in E' as follows. For each edge $<x,y> \in E'$, we find a set of taxa in P' such that either all of them have state pair $<x,z>$ on c_1 and c_2 or all of them have state pair $<z,y>$. These taxa must all be connected to some node v of P', whose parent is, say, u. We then replace edge $\{u,v\}$ in T by edges $\{u,w\}$ and $\{w,v\}$, add an edge $\{w,w'\}$, and connect all taxa with states $<x,y>$ to w'. Continuing in this fashion until all edges in E' have been considered, we obtain a tree P that requires at most $|E(T)| - 1 + |E'| = e(H(c_1,c_2)) - 1$ state changes. Hence, $pars(c_1,c_2) \leq e(H(c_1,c_2)) - 1$.

Lemma 2: *For any pair of characters c_1 and c_2, $pars(c_1,c_2) = e(H(c_1,c_2)) + k - 2$, where $k \geq 1$ is the number of connected components in $H(c_1,c_2)$.*

Proof: Let $C_1, C_2, \ldots, C_k$ be the connected components of $H(c_1,c_2)$. By Lemma 1, each C_i has a phylogeny with $|E(C_i)| - 1$ state changes and no fewer changes are possible.

Consider any two components C_i and C_j where $1 \leq i < j \leq k$. Any state pair $<x_1,y_1>$ in C_i and

$< x_2, y_2 >$ in C_j must satisfy $x_1 \neq x_2$ and $y_1 \neq y_2$ (otherwise, C_i and C_j are connected). Thus, two state changes are required in order to connect any two phylogenies from different connected components; and totally $2(k-1)$ state changes are required for connecting the k phylogenies to form a tree for all the taxa. Hence,

$$\begin{aligned} pars(c_1,c_2) \geq & \sum_{i=1}^{k} (|E(C_i)|-1)+2(k-1) \\ = & \sum_{i=1}^{k} |E(C_i)|-k+2k-2 \\ = & e(H(c_1,c_2))+k-2 \end{aligned}$$

On the other hand, we can construct a phylogeny P of c_1 and c_2 with $e(H(c_1,c_2))+k-2$ state changes. First, we construct phylogenies for each of the C_is. Then, we create a new root r that links to each root r_i of each such phylogeny, and assign r the same states as r_1. Any edge $\{r, r_i\}$ where $2 \leq i \leq k$, counts for 2 changes. Thus, the parsimony score of P is $\sum_{i=1}^{k}(|E(C_i)|-1)+2(k-1) = \sum_{i=1}^{k}|E(C_i)|-k+2k-2 = e(H(c_1,c_2)+k-2$.

Proof of Theorem 1: Let E' be a minimum set of edges whose removal from $H(c_1,c_2)$ yields an acyclic subgraph, i.e. a spanning forest R of $H(c_1,c_2)$. Then,

$$e(R) = v(R)-k = r_1+r_2-k$$

where the first equality follows from the properties of forests, and the second is because $v(R) = v(H(c_1,c_2)) = r_1+r_2$. Now,

$$\begin{aligned} p = & |E'| \\ = & e(H(c_1,c_2))-e(R) \\ = & pars(c_1,c_2)-k+2-(r_1+r_2-k) \quad \text{(by Lemma 2)} \\ = & pars(c_1,c_2)-(r_1+r_2-2) \end{aligned}$$

PHYLOFINDER: AN INTELLIGENT SEARCH ENGINE FOR PHYLOGENETIC TREE DATABASE

A paper submitted to BMC Bioinformatics

Duhong Chen [1 2 3] , J. Gordon Burleigh [4] , Mukul S. Bansal [1] , David Fernández-Baca [1]

Abstract

The rapid growth of phylogenetic information necessitates the development of tools to store and access phylogenetic data. These tools should enable users to search for phylogenetic trees containing a specified taxon or set of taxa and to compare a specified phylogenetic hypothesis to existing phylogenetic trees.

PhyloFinder is a new intelligent search engine for phylogenetic databases that we have implemented using trees from TreeBASE. It enables taxonomic queries, in which it identifies trees in the database containing the exact name of the query taxon and/or any synonymous taxon names and provides spelling suggestions for the query when there is no match. Additionally, PhyloFinder can identify trees containing descendants or direct ancestors of the query taxon. PhyloFinder also executes phylogenetic queries, in which it identifies trees that contain the query tree or topologies that are similar to the query tree.

PhyloFinder can enhance the utility of any tree database by providing tools for both taxonomic and phylogenetic queries as well as visualization tools that highlight the query results and provide links

[1]Department of computer science, Iowa State University, Ames, IA, 50011, USA
[2]Primary researcher and authoer
[3]Author for correspondence
[4]NESCent, Durham, NC 27705, USA

to NCBI and TBMap. An implementation of PhyloFinder using trees from TreeBASE is available at `http://pilin.cs.iastate.edu/phylofinder/`.

Background

The rapidly expanding wealth of phylogenetic information from across the tree of life offers unprecedented opportunities for large-scale evolutionary studies and for examining an array of biological questions in a phylogenetic context (Sanderson et al, 1993). However, at present much of the published phylogenetic data is not easily accessible. Therefore, the storage and retrieval of phylogenetic data are important challenges for bioinformatics (Sanderson et al, 1993; Piel et al, 2003; Page, 2007b, 2005a; Nakhleh et al, 2003). TreeBASE is the largest relational database of published phylogenetic information, storing more than 4,400 trees that include over 75,000 taxa, as well as data matrices used to infer the trees, and additional meta-data, such as bibliographic information and details of the phylogenetic analyses (Page, 2005b, 2007a). Though TreeBASE is a valuable repository for phylogenetic data, it is often difficult to identify and access relevant phylogenetic data from within TreeBASE. In this paper, we present PhyloFinder, a new phylogenetic tree search engine that greatly expands upon the current search features in TreeBASE and thus can enhance the utility of TreeBASE, or any phylogenetic database.

In general, to effectively utilize the existing phylogenetic data, we need tools that can quickly identify and access phylogenetic trees containing a specified taxon or set of taxa and that can compare a specified phylogenetic hypothesis to existing phylogenetic trees. The complexity of taxonomy presents a first major challenge for accessing phylogenetic data (Page, 2005a,b, 2007a,b). Taxonomic names used in stored phylogenetic trees often are based on different and inconsistent taxonomies (Page, 2007a). Furthermore, taxonomic classifications and names frequently change, and these changes may not be reflected in database trees. Consequently, repositories such as TreeBASE contain many species that are represented in multiple trees by different but equivalent names. Taxonomic queries are further complicated by misspellings or unique subspecies designations in stored trees, both of which are also common in TreeBASE (Page, 2007a). Many of these taxonomic issues have been addressed by TBMap, a database that maps names of taxa found in TreeBASE to other taxonomic databases and clusters equivalent taxonomic names (Page, 2007a). However, TBMap is not incorporated in TreeBASE or in any other phylogenetic search engines.

The hierarchical nature of taxonomic classifications presents further challenges for accessing phylogenetic data. The leaves in stored phylogenetic trees may represent different taxonomic levels, such as families, genera, species, or subspecies. It should be possible to query a tree database to identify trees containing not only the specific taxon name used in the query, but also trees containing descendants or ancestors of the query taxon (Page, 2007b, 2005a). For example, a query on the plant family name “Pinaceae” ideally would identify not only trees that contain the exact name “Pinaceae”, but also trees containing Pinaceae genera such as *“Pinus”* or *“Abies”* or species such as *“Pinus thunbergii”* or *“Abies alba”*. It also would be useful to identify trees containing direct ancestors (the internal nodes on the path from the root of a taxonomy tree to the query taxon) of the query taxon. Thus, a query on the species name *“Pinus thunbergii”* would identify trees that contain the genus name *“Pinus”* or the family name “Pinaceae” as leaves. Currently, TreeBASE does not directly utilize information from taxonomic classifications to allow the user to find trees containing ancestors or descendants of the query taxon (Page, 2007b, 2005a). Instead, the user can find all the taxa matching a partial name taxon query. For example, querying “Pinus@” or even “Pinu@” in TreeBASE will identify all trees containing *“Pinus”* in their species name. However, querying using “Pinaceae@” will not identify trees with *“Pinus”* or *“Abies”* species, because they do not contain “Pinaceae” in the species name. Alternately, the user can identify trees with related taxa through “tree surfing”, in which the user identifies neighboring trees (trees with shared taxa) of a specified tree(s). Tree surfing can be time consuming, and it is difficult if not impossible for the user to determine if s/he has found all the trees containing the related taxa.

Another important feature of an effective phylogenetic search engine is the ability to make phylogenetic queries in which the user can assess a specified tree by comparing it to the trees in the database (Page, 2007b; Nakhleh et al, 2003). Tree mining queries must first be able to identify all trees that contain or agree with a query tree, or the trees in the database in which the query tree is embedded (Page, 2005a, 2007b). Additionally, since there is often much disagreement among trees, it is very useful to be able to identify all the trees that are similar, but not necessarily identical, to a query tree. Some tree mining features are implemented with TreeBASE (Shan et al, 2002; Wang et al, 2003, 2005).

In this paper, we introduce PhyloFinder, a search engine for phylogenetic databases that addresses the issues described above. PhyloFinder uses TBMap (Page, 2007a) to address the problem of taxonomic inconsistency, thereby expanding the power of taxonomic queries by recognizing synonymous taxon names. It also offers alternate spelling suggestions for taxonomic queries that do not find a match in the tree database. PhyloFinder further increases the querying power by using the hierarchical structure of the NCBI taxonomy (NCBI, 2007) to search for trees containing descendants or ancestors of a query taxon, and it also enables a wide range of tree mining queries. PhyloFinder has a tree visualization tool that displays the query results, highlighting relevant taxa and branches, and provides hyperlinks to the NCBI taxonomy and TBMap websites. The implementation of PhyloFinder uses simple but powerful information retrieval techniques. These include the use of an inverted index that maps taxa to trees, which allows the system to filter out many trees that are irrelevant to a given query, and a representation of trees that allows fast least common ancestor queries directly on the database. We have tested PhyloFinder using trees from TreeBASE (Piel et al, 2002; TreeBASE, 2007). However, PhyloFinder can, in principle, be used with any phylogenetic database.

Implementation

Before describing the implementation of PhyloFinder, we outline the features that it supports.

Taxonomic Queries and Phylogenetic queries

PhyloFinder's web-based interface allows the user to make *taxonomic* and *phylogenetic* queries. Taxonomic queries involve a single taxon or set of taxa. Phylogenetic queries take as input a phylogenetic tree and attempt to locate trees in the database that match it in some specified way.

Taxonomic Queries

PhyloFinder supports three different types of taxonomic queries: contains, related, and pathlength.

1. **Contains:** The input for this command is a set of taxon names, given as a comma-separated list. The output is a list of the tree IDs of all trees from the database that, depending on the user's choice, contain *all* or *any* of the taxon names in the set. Note: in our implementation of PhyloFinder using TreeBASE trees, the output is a list of the TreeBASE tree IDs and corresponding study IDs.

2. **Related:** The input for this query is a taxon name. The search engine finds all trees in the database involving any taxon that, depending on the user's choice, is a descendant or a direct ancestor of the query taxon in the NCBI taxonomy tree. For example (see Figure 1), if the query taxon is *"birds"*, and the user chooses the descendant option, the *related* command will identify all the trees in the database that contain any bird taxa (Page, 2005a).

3. **Pathlength:** The input for this query is a pair of taxon names. The output is a list of the tree IDs of all trees containing the two species, along with the distance (path length) between the two taxa in each tree. Note: in our implementation of PhyloFinder using TreeBASE trees, the output is a list of the TreeBASE tree IDs, the corresponding study IDs, and the distance (path length) between the two taxa in each tree.

Phylogenetic Queries

PhyloFinder supports two different types of phylogenetic queries: *tree mining* and *tree similarity search*. To describe these commands, we need some definitions.

Definitions

Let T be a phylogenetic tree, and A be a subset of the leaves of T. Following standard terminology (Semple and Steel, 2003), we write $T(A)$ to denote the minimal subtree of T that contains the leaves in

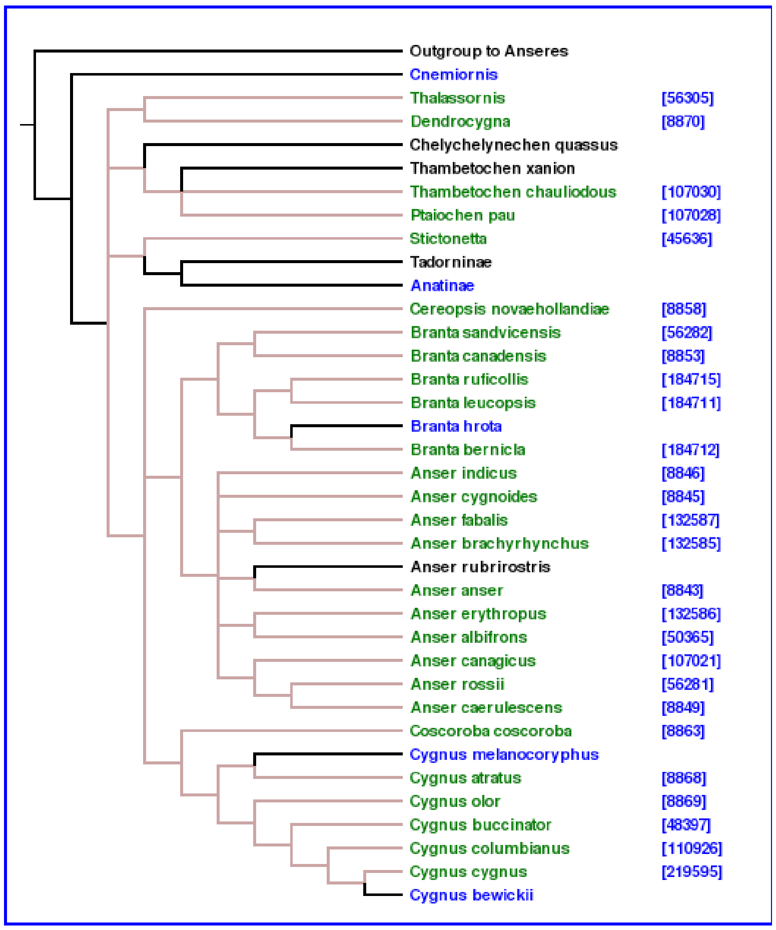


Figure 1: Query result visualization: one of the trees returned by querying for "birds" with the *related* command. The version of TreeBASE used by PhyloFinder contains around 88 such trees. Taxa in dark green are bird species that are found in the NCBI taxonomy database. The blue numbers are the NCBI taxon ID numbers, and indicate hyperlinks to the NCBI taxonomy web site. The taxon names displayed in color (other than black) indicate hyperlinks to TBMap.

A, and $T|A$ to denote the tree obtained from $T(A)$ by suppressing all internal nodes that have only one child. An example is shown in Figure 2.

Let Q be a query tree and T be a candidate tree (that is, a tree from the database). Let A denote the set of leaves of Q. Tree Q is a *pruned subtree* of T if and only if either (i) $T = Q$, or (ii) there exists an edge in T which, when pruned, produces Q as the cut-out (i.e. pruned) subtree. This is illustrated in Figure 3.

Tree Q is an *embedded subtree* of T if and only if it is identical to $T|A$ (see Figure 3). Informally, this means that Q shows the same evolutionary relationships implied by T. Note that if Q is a pruned

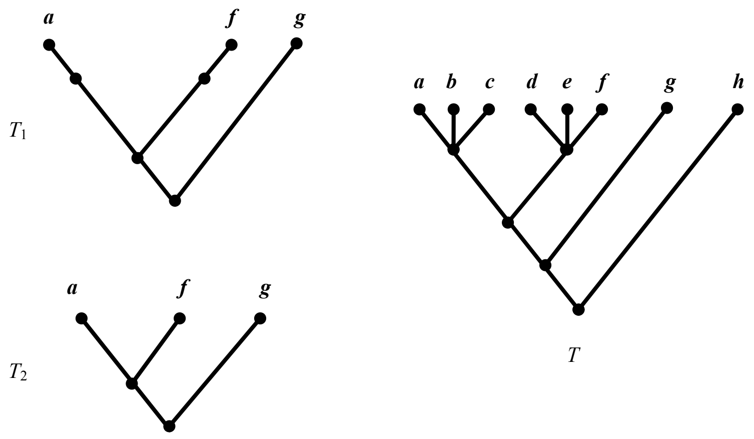


Figure 2: $T(A)$ and $T|A$: for $A = \{a, f, g\}$, T_1 is the tree $T(A)$ and T_2 is the tree $T|A$

subtree of T, it must also be an embedded subtree, but the converse is not true (Figure 3). If Q is an embedded subtree of T, the tree $T(A)$ is called the *embedding* of Q in T. We say that tree Q is *refined* by tree T (or that T refines Q) if the set of clusters of Q is a subset of the set of clusters of $T|A$; i.e., $T|A$ is a refinement of Q. Note that if Q is an embedded subtree of T, it must also be refined by T, but not vice versa (Figure 3).

A *similarity measure* is a function that, given a *query tree* and a *candidate tree* returns a percentage score between 0 and 100%, called a *similarity score*, reflecting how similar the query tree is to the candidate tree. PhyloFinder uses two similarity measures, one based on the Robinson-Foulds (RF) distance (Robinson and Foulds, 1981) and the other based on least common ancestors (LCAs). If the LCA-based score is 100%, the query tree is refined by the candidate tree, while if the RF similarity score is 100%, the query tree can be embedded in the candidate tree. These measures are described in the Appendix.

Commands

1. **Tree mining:** The input is a query tree Q in Newick format. The output is a list of tree IDs (TreeBASE tree IDs and study IDs in our current implementation) of all trees that exhibit the query tree Q in some way. There are three options for this command, which are listed below:

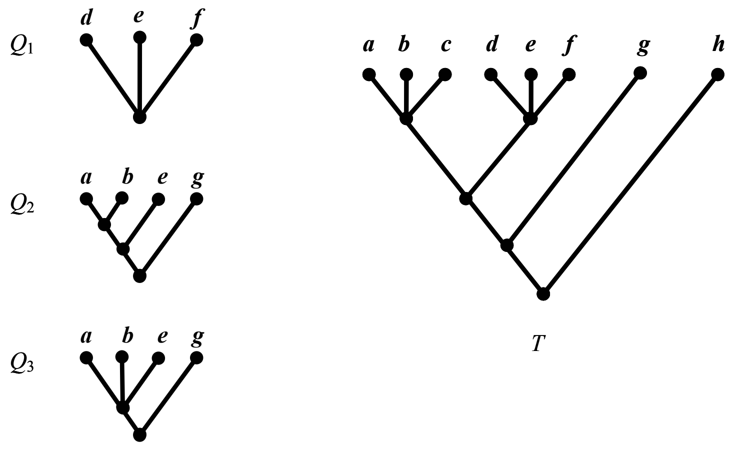


Figure 3: Pruned induced, embedded of tree mining: in this example, the query tree Q_1 is an induced subtree and an embedded subtree of tree T; Q_1 is also refined within T. The query tree Q_2 is an embedded subtree of T and also refined in T; however, it is not an induced subtree of T. The query tree Q_3 is refined by T but is neither an embedded nor an induced subtree of tree T.

- Pruned: The output is a set of trees that contain Q as a pruned subtree.
- Embedded: The output is a set of trees that have Q as an embedded subtree. For the same query tree Q, the result of embedded subtree mining will be a superset of the output returned by the pruned subtree mining command.
- Refined: The output is all trees that refine Q. The result will be a superset of the output set returned by the embedded subtree mining command on the same query tree Q.

2. **Similar:** The input is a tree in Newick format. The output is a list of the IDs of all trees that share at least three taxa with the query tree, ranked according to their similarity scores. Two options are provided, depending on whether the similarity score is computed with the RF-based or the LCA-based measure.

System Architecture

Figure 4 shows the system architecture of PhyloFinder. The search engine is built on top of MySQL, an open-source relational database management system (RDBMS). PhyloFinder stores the phylogenetic

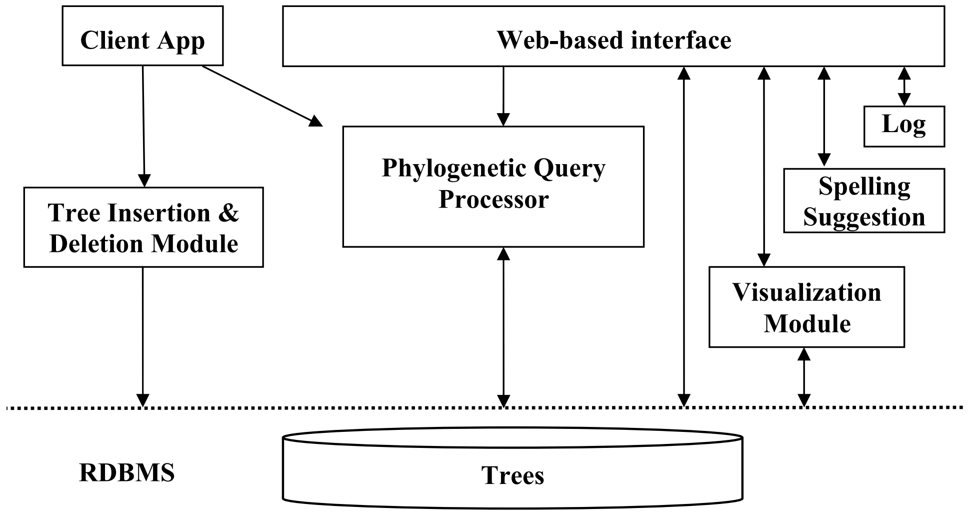


Figure 4: PhyloFinder's system architecture

trees, which in our test implementation are from TreeBASE (Piel et al, 2002), and the NCBI taxonomy tree (NCBI, 2007) in MySQL using a slight modification (Zaki, 2005) of *nested-set representation* (Celko, 2004; Tropashko, 2005). Under this scheme, each node of a given tree is represented by an ordered pair of numbers as follows: the first number, called NodeID, is defined by a pre-order traversal (Felsenstein, 2004) of the tree, and the second number is the NodeID of the descendant with the largest NodeID (see Figure 5). This approach provides an efficient and elegant mechanism that allows quick identification of the ancestors and descendants of any node in a tree. In particular, it allows a simple and elegant implementation of LCA queries directly on the database. Given any two nodes in a tree, their LCA is simply that ancestor of both these nodes that has the largest NodeID. More efficient algorithms are available to solve the LCA problem (Bender and Farach-Colton, 2000), but our approach has the advantage of being directly supported by the database, and is fast in practice. (Alternately, we could have stored trees as lists of edges (Nakhleh et al, 2003); however, phylogenetic queries would have required recursive SQL extensions, which are not supported by many RDBMSs.)

The RDBMS also stores the NCBI taxon names, and in order to automatically translate between synonymous names for the same species, the RDBMS stores a collection of *taxon clusters,* where each

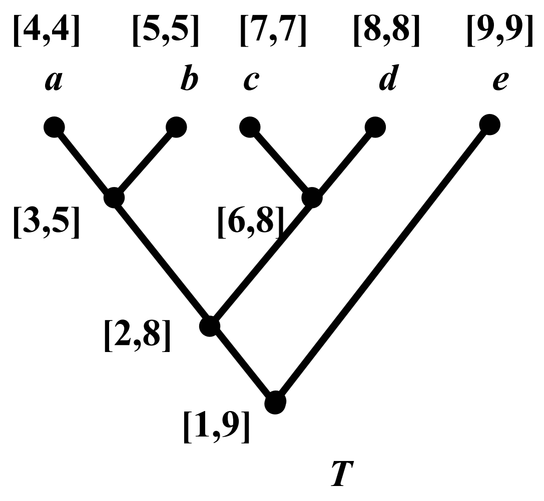


Figure 5: A slight modification of nested-set representation of a rooted tree

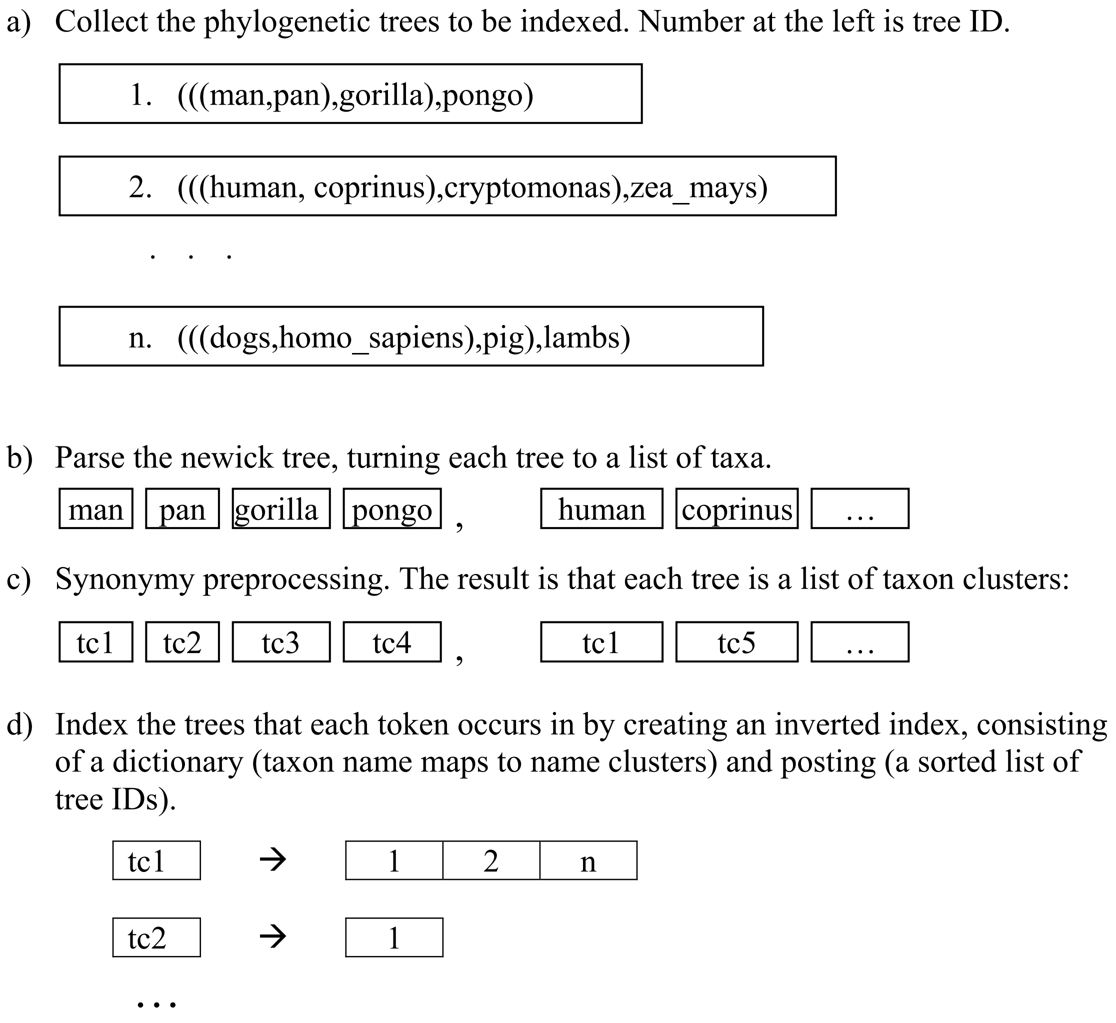
cluster contains a set of synonymous taxa. Taxon clusters are generated using TBMap (Page, 2007a) and the NCBI taxonomy database.

PhyloFinder uses an *inverted index* to achieve fast querying. In text retrieval and web mining, such indices are used as mappings from words to sets of documents that contain them (Zobel and Moffat, 2006). PhyloFinder's inverted index treats taxon clusters as words and phylogenetic trees as documents (Figure 6).

The *phylogenetic query processor* is the kernel of the search engine. It provides functions to parse user commands, to perform taxonomic and phylogenetic queries, and to coordinate other modules. It relies heavily on the inverted index. The query processor uses the NCBI tree as a classification guide for species in queries such as the *related* command.

The *spelling suggestion module* checks spellings in the query and provides suggestions based on taxon names in TreeBASE or in the NCBI taxonomy database. This is implemented using the GNU Aspell c library (Aspell, 2006), with some modifications in order to handle special alphabet characters (e.g., '-', '&', '.') and compound words in taxon names.

The *Visualization module* creates HTML files with image map, which show phylogenetic trees in a dendrogram format, highlight the embedded query tree or species, and provide link-outs for species



a) Collect the phylogenetic trees to be indexed. Number at the left is tree ID.

b) Parse the newick tree, turning each tree to a list of taxa.

c) Synonymy preprocessing. The result is that each tree is a list of taxon clusters:

d) Index the trees that each token occurs in by creating an inverted index, consisting of a dictionary (taxon name maps to name clusters) and posting (a sorted list of tree IDs).

1. (((man,pan),gorilla),pongo)
2. (((human, coprinus),cryptomonas),zea_mays)
. . .
n. (((dogs,homo_sapiens),pig),lambs)
man pan gorilla pongo , human coprinus …
tc1 tc2 tc3 tc4 , tc1 tc5 …
tc1 → 1 2 n
tc2 → 1
…

Figure 6: Construction of the inverted Index

in NCBI taxonomy database. While there are many available tree visualization tools (Zmasek and Eddy, 2001; Hillis et al, 2005; Sanderson, 2006), we could not find one that could easily be adapted to highlight query results and provide outlinks. Therefore, we developed a new tree visualization tool for PhyloFinder.

The *client application* is an administration tool that provides an interface to handle tree insertions and deletions (via the *insertion and deletion module*). It also provides commands for updating the inverted index and performing low-level queries (SQL commands or specified query commands with unformatted results) for testing and debugging.

The *web-based interface* is an interactive web application that uses the AJAX technique. It is written using the Google Web Toolkit (GWT, 2007) and the GWT Window Manager (GWM, 2007).

The system maintains a *log*, where it records user queries and timestamps. The statistics can be used to analyze user needs and help optimize the search engine performance.

Taxonomic name consistency

As mentioned above, the search engine relies on taxon clusters provided by TBMap (Page, 2007a) to identify synonymous taxonomic names. Ideally, TBMap and PhyloFinder should include the same set of taxon names as TreeBASE. However, TBMap is based on a 2004 snapshot of TreeBASE (Page, 2007a). Thus, some new taxon names in TreeBASE are not included in TBMap. Whenever PhyloFinder encounters an inconsistency, it ignores the classification provided by TBMap, and relies instead on the NCBI taxonomy. We do this because the "Related" query is based on the NCBI taxonomy, and using the NCBI taxonomy maximizes the utility of this feature. Another complication is that TBMap uses four external taxonomic databases – ITIS (ITIS, 2007), IPNI (IPNI, 2004), uBIO (uBio, 2004) and NCBI (NCBI, 2007) – among which there are conflicts. For example, "*Antennaria solitaria*" and "*Antennaria monocephala*" are treated as synonyms in IPNI but not in NCBI (see Figure 7). In such cases, PhyloFinder again uses the NCBI classification.

Query processing

When a user submits a query from a web browser, the web CGI program parses the user query into query command (e.g., *contains*) and query contents (e.g., the taxa in the list). The results for some queries (such as obtaining the NCBI taxon ID for a given taxon name) can be retrieved directly from the database. Other queries (such as the *tree mining* command) go through the query processor, which first identifies candidate trees using the inverted index and then performs further computations to get the final results (see Figure 8). Identifying the candidate trees using the inverted index allows us to filter a large fraction of the database trees from further consideration, significantly accelerating the query processing.

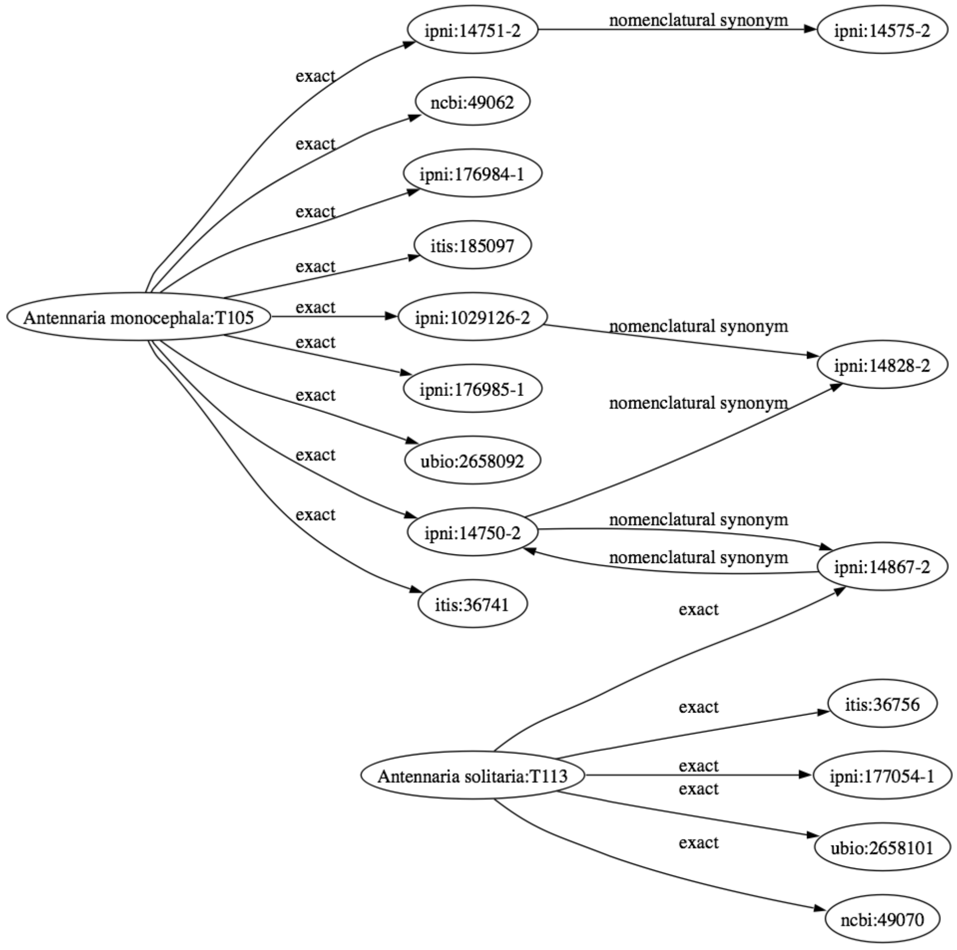


Figure 7: Synonym conflict in IPNI and NCBI: name cluster containing "*Antennaria solitaria*" and "*Antennaria monocephala*" produced by TBMap. "*Anternnaria solitaria*" and "*Antennaria monocephala*" are synonyms in IPNI, but are treated as distinct taxa by NCBI. In such cases, PhyloFinder uses the NCBI classification.

Phylogenetic queries (other than RF-based similarity search) are implemented by using LCA queries to compare ancestor-descendant relationships in the query tree and in the database tree. Figure 9 shows a simple example where the goal is to determine if query tree Q can be embedded in database tree T. The two internal nodes x and y in Q map to nodes $M(x)$ and $M(y)$ in T in the sense that the LCAs of the descendants of x and y are $M(x)$ and $M(y)$, respectively. Q can be embedded in T since $M(x)$ and $M(y)$ have the same ancestor-descendant relationship in T as x and y have in Q. The use of nested-set representation for trees in the RDBMS allows LCA queries to be directly computed by the RDBMS.

Hierarchical queries (e.g., *related*), which require a classification (or ontology) guide (Page, 2007b),

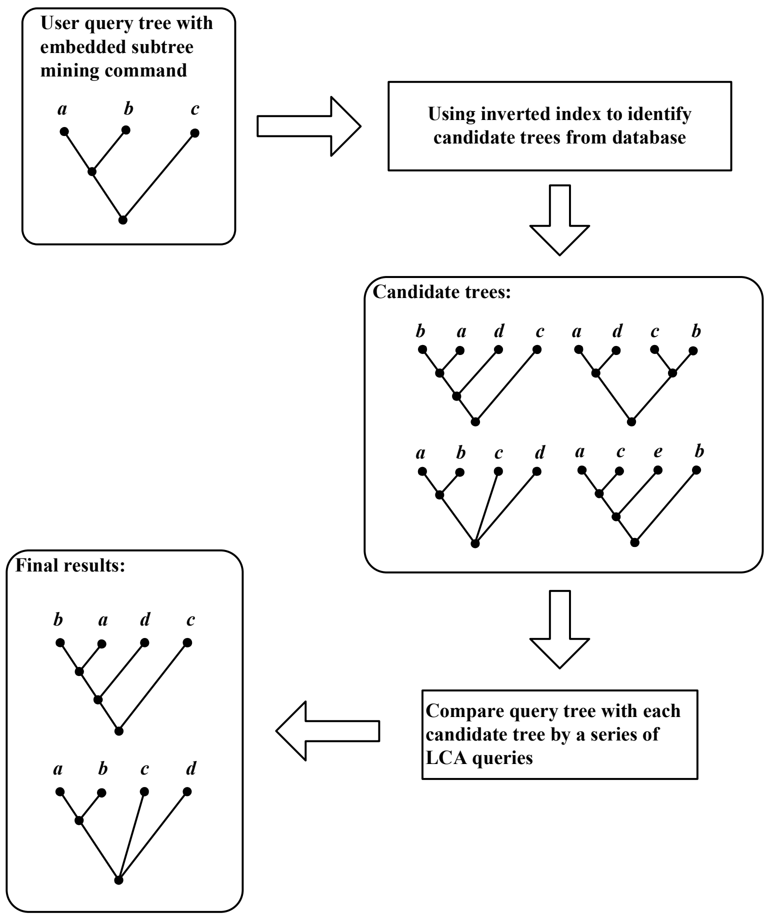

User query tree with embedded subtree mining command
a
b
c
Using inverted index to identify candidate trees from database
Candidate trees:
b a d c
a d c b
a b c d
a c e b
Compare query tree with each candidate tree by a series of LCA queries
Final results:
b a d c
a b c d

Figure 8: Processing queries through the search engine

are addressed by Boolean operations on the inverted index. For example, when a user query asks for trees that contain “birds” using the *related/descendant* command, the search engine first looks for all bird species in the stored phylogenetic trees using the NCBI taxonomy tree to identify all bird species in the trees. It then retrieves the tree ID lists corresponding to each bird species and returns their union (equivalent to Boolean “OR”).

Initial setup

While not fully automated, PhyloFinder’s setup is reasonably straightforward and can, in principle, be used on any given data set of phylogenetic trees. The first step in the setup is to assemble all the phylogenetic trees in Newick format together in a file. A program then reads this file and converts the

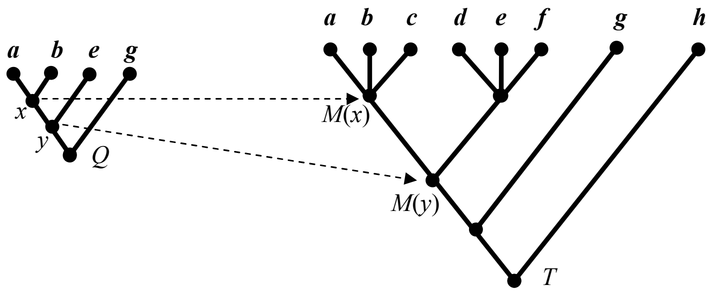


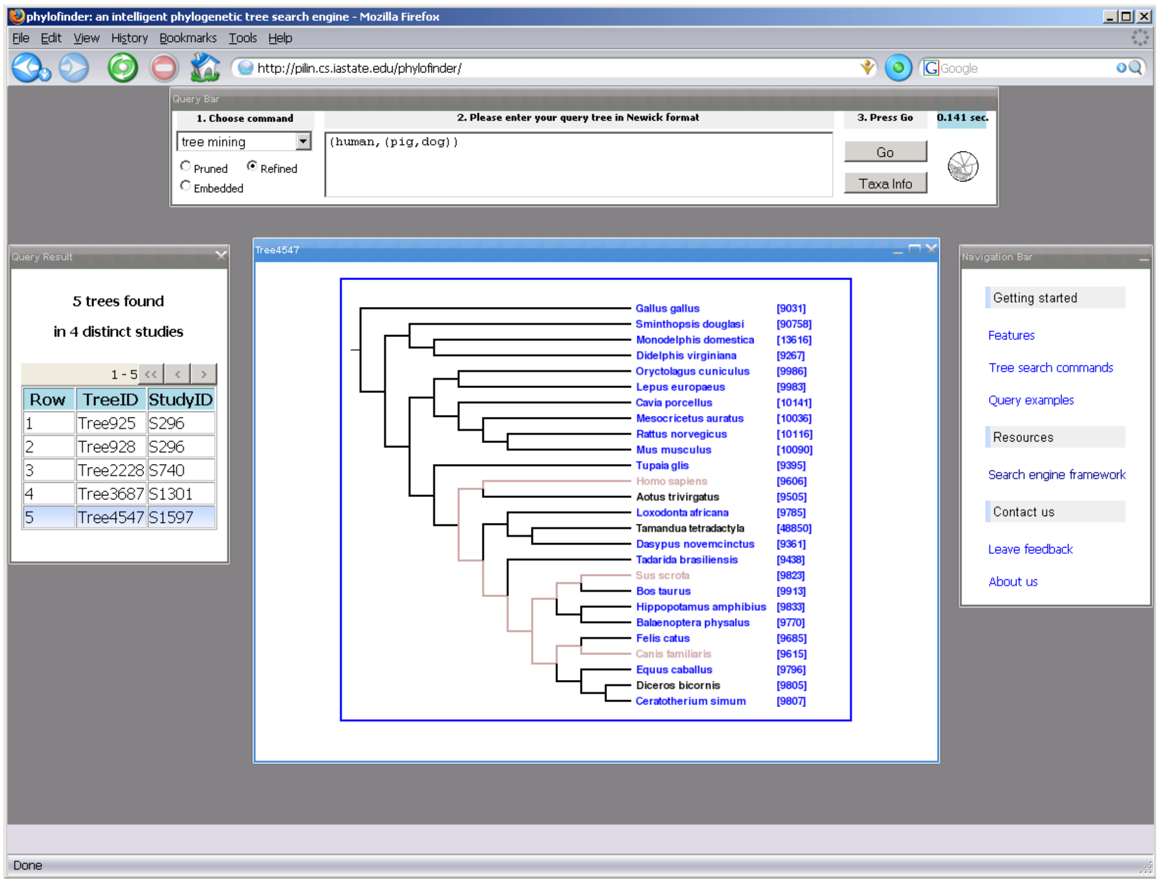
Figure 9: Tree mining using LCA mappings

trees into the nested set representation used by PhyloFinder. The output of this program consists of several MySQL tables. A second program then reads in these tables, and uses data from additional sources, like TBMap and NCBI, to create some additional MySQL tables which help to improve query processing. All these MySQL tables must then be loaded into the MySQL database. Once this is done, the main program can be run. The main program first reads some tables from the MySQL database in order to create the inverted index, and then waits for commands from the client application. Once the server is set up, the system is ready for use. A detailed diagram of the database scheme is available at: http://pilin.cs.iastate.edu/phylofinder/phylofinder-schema.pdf

In practice, we have found that setting up the system for TreeBASE data is more complicated. This is because TreeBASE data does not always conform to the Newick standard, and some of the trees contain errors. In addition, several of the trees contain special international characters. Dealing with this effectively requires some manual work and it makes it difficult to automatically update the local copy of TreeBASE.

Results

We tested PhyloFinder using the trees from TreeBASE. PhyloFinder's interactive web interface is shown in Figure 10. At the top center of the main window is a query panel that contains widgets for choosing commands and entering queries. At the right is a navigation bar that includes help links and links to information on the PhyloFinder framework. A result panel is displayed when the search engine


phylofinder: an intelligent phylogenetic tree search engine - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://pilin.cs.iastate.edu/phylofinder/
Google
Query Bar
1. Choose command
2. Please enter your query tree in Newick format
3. Press Go
0.141 sec.
Go
Taxa Info
Query Result
Tree4547
Navigation Bar
Getting started
Features
Tree search commands
Query examples
Resources
Search engine framework
Contact us
Leave feedback
About us
Gallus gallus [9031]
Sminthopsis douglasi [90758]
Monodelphis domestica [13616]
Didelphis virginiana [9267]
Oryctolagus cuniculus [9986]
Lepus europaeus [9983]
Cavia porcellus [10141]
Mesocricetus auratus [10036]
Rattus norvegicus [10116]
Mus musculus [10090]
Tupaia glis [9395]
Homo sapiens [9606]
Aotus trivirgatus [9505]
Loxodonta africana [9785]
Tamandua tetradactyla [48850]
Dasypus novemcinctus [9361]
Tadarida brasiliensis [9438]
Sus scrofa [9823]
Bos taurus [9913]
Hippopotamus amphibius [9833]
Balaenoptera physalus [9770]
Felis catus [9685]
Canis familiaris [9615]
Equus caballus [9796]
Diceros bicornis [9805]
Ceratotherium simum [9807]
Done

1. Choose command: tree mining

( ) Pruned (●) Refined ( ) Embedded

2. Please enter your query tree in Newick format:

```
(human,(pig,dog))
```

5 trees found

in 4 distinct studies

1 - 5

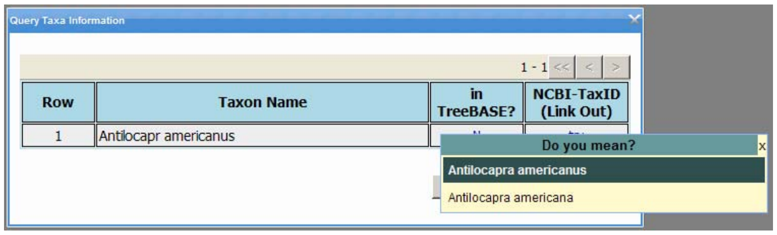
Row	TreeID	StudyID
1	Tree925	S296
2	Tree928	S296
3	Tree2228	S740
4	Tree3687	S1301
5	Tree4547	S1597

Figure 10: Screenshot of PhyloFinder's web interface

finds results matching a query. Trees are displayed graphically when the user clicks the records in the result panel. Phylogenetic query results are visualized by highlighting the set C of taxa that the result tree and the query tree have in common, as well as edges in the result tree that connect C. If a taxon in the result tree is in TBMap, its name is hyperlinked to TBMap, and if the taxon is in the NCBI taxonomy database, an NCBI taxon ID number with a hyperlink to the NCBI taxonomy browser is appended to the taxon name.

Figures 1 and 10, illustrate various features of PhyloFinder. Note the use of color by the visualization module. Embedded query trees and taxon names are highlighted using various colors. Brown indicates an exact match (*i.e.* same taxon name or NCBI taxon ID). For the *related* command, ancestors of the query taxon are highlighted in orange, and descendants are highlighted in green.

The system provides spelling suggestions — all from TreeBASE and NCBI — for misspelled taxon names. For example, Figure 11 shows the spelling suggestions offered by PhyloFinder for the query taxon “*Antilocapr americanus*”, which is neither in any of the TreeBASE trees nor in the NCBI database. When a user chooses a name from the suggestion list, PhyloFinder automatically updates the user query and the NCBI taxonomy out-link.



Row	Taxon Name	in TreeBASE?	NCBI-TaxID (Link Out)
1	Antilocapr americanus	[illegible]	[illegible]

Figure 11: Spelling suggestions for the query taxon “*Antilocapr americanus*”

PhyloFinder effectively uses the NCBI taxonomy and the TBMap database to translate between different names for the same species (see Figure 10). For example, if one queries for trees that are similar to “(human,(pig,dog))”, PhyloFinder will look for trees that contain scientific names for human, pig, and dog. TBMap allows us to find many synonyms that would be missed by using only the NCBI names.

There are 64,529 leaf taxa in the TreeBASE trees used by PhyloFinder. By itself, NCBI allows us to map 30,007 of these taxa to 29,669 distinct NCBI TaxonIDs, leaving 34,522 isolated taxa, each of which is in its own cluster, for a total of 64,191 clusters. In contrast, using TBMap we are able to map 36,864 leaf taxa in TreeBASE to NCBI taxa. The total number of taxon clusters is 52,198, of which 29,480 are mapped to NCBI taxa.

PhyloFinder uses the NCBI classification among species in user queries. Figure 1 shows an example of a taxon query looking for trees that contain species in “*birds”*. The descendants of “*birds*” are highlighted in dark green in the displayed tree.

For the same query term, PhyloFinder normally returns more trees than TreeBASE even though PhyloFinder uses data from an older version of TreeBASE (containing fewer trees). For example, for a query on “angiosperms”, TreeBASE returns 8 studies with 17 trees while PhyloFinder retrieves 543 studies and 1,550 trees using the “related” command; When querying on “Fungi”, PhyloFinder returns

487 studies with 1,054 trees, as compared to the 6 studies with 15 trees that are retrieved using TreeBASE searching tools. This remarkable improvement over TreeBASE is achieved due to a combination of the classification guide provided by the NCBI taxonomy tree, and taxon cluster information from TBMap.

The time required to process a query depends on network speed, server load, client computer performance, among other factors. In most of our test queries, the results were received within a matter of seconds. In particular, elapsed times and CPU times for a taxonomic query on “angiosperms” using the “related” command were 0.266s and 0.076s respectively; and for “Fungi” the corresponding times were 0.172s and 0.06s respectively.

Discussion

We have demonstrated the utility of PhyloFinder using the trees from TreeBASE. We have shown that PhyloFinder can enhance the utility of TreeBASE by making it easier to assess and obtain the phylogenetic data contained within it. PhyloFinder adds several new taxonomic querying capabilities to TreeBASE, including spelling suggestions, searches for synonymous taxonomic names, and the ability to identify trees with ancestors or descendants of the query taxon and to identify path lengths between taxa in the database trees. PhyloFinder also expands the power of phylogenetic queries, offering more precise options for identifying different types of subtrees and more metrics for identifying similar trees than TreeBASE. Additionally, PhyloFinder provides a tree visualization tool that highlights query taxa in an informative manner and gives useful outlinks to GenBank and TBMap. Furthermore, PhyloFinder provides nearly immediate results for most queries. Still, PhyloFinder is a search engine, not a database, and is not meant to be a substitute for tree repositories such as TreeBASE.

PhyloFinder is not limited to TreeBASE and can be incorporated into any phylogenetic tree database. For example, we have incorporated PhyloFinder into the PhyLoTA browser for gene trees (Sanderson, 2007; Sanderson et al, 2007). Future development of PhyloFinder will include a desktop version in which users can input their own sets of phylogenetic trees. A number of other extensions to the search engine are also under development. These include handling unrooted trees, providing an interface for retrieving trees in Newick format and more options for drawing trees, displaying details of the phylogenetic study in the query results, and ranking trees returned by the *embedded subtree mining* command (e.g., the rank of a tree T could be the number of edge contractions of T that are required to get the query tree). The search engine can also be linked to tools for building supertrees, which will allow users to assemble large phylogenies by combining phylogenetic trees with incomplete taxon overlap.

The usefulness of any phylogenetic search engine is limited by the amount of phylogenetic information it can search. Although TreeBASE is the largest relational database of published phylogenetic trees, few journals require that trees be submitted to TreeBASE, and thus it contains only a small percentage of published phylogenetic trees. We hope that the development of more effective methods to access and utilize phylogenetic data will further motivate efforts to collect and store phylogenetic information.

Conclusions

While there has been great progress in understanding organismal relationships across the tree of life, this phylogenetic data is often not easily accessible to scientists. PhyloFinder enhances the utility of phylogenetic databases, by enabling scientists to identify and assess available phylogenetic data. The taxonomic search tools in PhyloFinder allow researchers to identify all trees containing taxa of interest without knowing the names of all taxa in the available trees, possible synonymous taxon names, or even the correct spelling of a taxon name. The phylogenetic search tools allow one to evaluate phylogenetic hypotheses by comparing it to existing phylogenies and identifying trees that agree with or are similar to the query tree. We have tested the utility of PhyloFinder using trees from TreeBASE, the largest relational database of published phylogenetic information, but the search engine can be used with any other tree database.

Availability and requirements

A PhyloFinder server for TreeBASE trees is set up in our lab at Iowa State University. The web client application is available at http://pilin.cs.iastate.edu/phylofinder/. All the features and functions described in this manuscript are freely accessible from our web site.

System requirements

PhyloFinder's web application has been tested with Microsoft Internet Explorer version 6.0 and above, and with Mozilla Firefox version 1.5 and above. Some minor issues with the web interface occur when using Apple Safari.

Authors contributions

DC was the primary designer and implementer of PhyloFinder, and wrote major parts of the paper. JGB was involved in the system design and the writing of the manuscript. MSB contributed in the initial stages of the system design, conceived the LCA-based similarity measure used by the system, helped with the design and implementation of phylogenetic queries, and contributed to the writing of the paper. DFB contributed to the system design and the writing of the paper. All authors read and approved the final manuscript.

Acknowledgements

We thank Bill Piel for providing TreeBASE data, Rod Page for providing TBMap and answering questions about that system, and Leslie Miller for many helpful suggestions. Mike Sanderson provided valuable discussion regarding the search engine features and comments on the manuscript. We also thank the anonymous reviewers for their helpful comments, which led to significant improvements to this manuscript as well as to PhyloFinder. This work was funded by NSF grant EF-0334832 and by NESCent, NSF EF-0423641.

References

Aspell. 2006. Gnu aspell [online]. Accessed 21 September 2006. URL: http://aspell.net.

Bender M. A., Farach-Colton M.. 2000. The LCA problem revisited. In LATIN. p 88–94.

Celko Joe. 2004. Joe Celko's SQL for Smarties: Trees and Hierarchies. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Felsenstein J.. 2004. Inferring Phylogenies. Sinauer Assoc., Sunderland, Mass.

GWM. 2007. Gwt window manager (gwm) [online]. Accessed 21 September 2006. URL: http://www.gwtwindowmanager.org.

GWT. 2007. Google web toolkit - build ajax apps in the java language [online]. Accessed 21 September 2006. URL: http://code.google.com/webtoolkit/.

Hillis David, Heath Tracy, John Katherine. 2005. Analysis and visualization of tree space. *Systematic Biology*, 54:471–482.

IPNI. 2004. International plant names index [online]. Accessed 21 September, 2006. http://www.ipni.org.

ITIS. 2007. Integrated taxonomic information system [online]. Accessed 21 September, 2006. http://www.itis.usda.gov.

Nakhleh L, Miranker D., Barbancon F, Piel W.H., Donoghue M.J.. 2003. Requirements of phylogenetic databases. In Proc. 3rd IEEE Symp. on Bioinformatics and Bioengineering. p 141–148.

NCBI. 2007. Ncbi taxonomy [online]. Accessed 21 September 2006. URL: http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html.

Page R.D.M.. 2005a. Phyloinformatics: towards a phylogenetic database. In J.T.L. Wang, Zaki M.J., Toivonen H.T.T., Shasha D.E., eds. Data Mining in Bioinformatics. Springer-Verlag, Berlin, p 219–241.

Page R.D.M.. 2005b. A taxonomic search engine: federating taxonomic databases using web services. *BMC Bioinformatics*, 6:48.

Page R. D. M.. 2007a. Tbmap: A taxonomic perspective on the phylogenetic database treebase. *BMC Bioinformatics*, 8:158.

Page R.D.M.. 2007b. Towards a taxonomically intelligent phylogenetic database. *Nature Precedings*Available from Nature Precedings (2007).

Piel W, Donoghue M, Sanderson M. 2002. Treebase: a database of phylogenetic knowledge. In To the Interoperable Catalogue of Life with partners - Species 2000 Asia Oceania - Proceedings of 2nd International Workshop of Species 2000 (Research Report for the National titute of Environmental Studies, R-171-2002). Springer-Verlag, Tsukuba, Japan.

Piel W.H., Sanderson M.J., Donoghue M.J.. 2003. The small-world dynamics of tree networks and data mining in phyloinformatics. *Bioinformatics*, 19:1162–1168.

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

Sanderson M.J.. 2007. Phylota browser [online]. Accessed 2 October 2007. URL: http://loco.biosci.arizona.edu/cgi-bin/pb.cgi.

Sanderson M.J., Baldwin B.G., Bharathan G., Campbell C.S.. 1993. The rate of growth of phylogenetic information, and the need for a phylogenetic database. *Syst Biol*, 42:562–568.

Sanderson M.J., Boss Darren, Chen Duhong, Cranston K. A., Wehe Andre. 2007. The phylota browser: Processing genbank for molecular phylogenetics research, submitted.

Sanderson Michael J.. 2006. Paloverde: an opengl 3d phylogeny browser. *Bioinformatics*, 22 (8):1004–1006.

Semple C., Steel M.. 2003. Phylogenetics. Oxford University Press.

Shan H, Herbert KG, Piel WH, Shasha D, Wang JTL. 2002. A structure-based search engine for phylogenetic databases. *SSDBM*:7–10.

TreeBASE. 2007. Treebase [online]. Accessed 21 September 2006. URL: http://www.treebase.org.

Tropashko V. 2005. Nested intervals tree encoding in sql. *SIGMOD Record*, 34:47–52.

uBio. 2004. Universal biological indexer and organizer [online]. Accessed 21 September, 2006. URL: http://www.ubio.org.

Wang J.T.L., Shan H., Shasha D., Piel W.H.. 2003. Treerank: a similarity measure for nearest neighbor searching in phylogenetic databases. *SSDBM*:171–180.

Wang J.T.L., Shan H., Shasha D., Piel W.H.. 2005. Fast structural search in phylogenetic databases. *Evolutionary Bioinformatics*, 1:37–46.

Zaki M. J.. 2005. Efficiently mining frequent trees in a forest: Algorithms and applications. *Knowledge and Data Engineering, IEEE Transactions on*, 17 (8):1021–1035.

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

Zobel Justin, Moffat Alistair. 2006. Inverted files for text search engines. *ACM Comput. Surv.*, 38 (2).

Appendix: Similarity measures

Here we describe the two similarity measures used by our system. The first of these is based on the well-known Robinson-Foulds distance. The other uses the notion of a *Least Common Ancestor* (LCA).

The LCA of a set of nodes in a tree is the most recent node in the tree whose descendants include all elements of the set.

In what follows, Q denotes the query tree and T denotes the candidate (database) tree. We write $n(T)$ to denote the total number of taxa in tree T. C denotes the set of taxa that T and Q share in common and $n(C)$ is the number of elements in C.

Robinson-Foulds similarity

If T and Q have the same set of taxa, then the Robinson-Foulds distance between them is the number of clusters in T that do not appear in Q plus the number of clusters in Q that do not appear in T. In practice, one must take into account the fact that the taxon overlap C between T and Q may be only partial. Thus, we define the (Robinson-Foulds) similarity between T and Q as

$$similarity_{RF}(T,Q) = \begin{cases} 0 & \text{if} \quad r = 0 \\ \frac{n(C)}{n(T)} \times \left(1 - \frac{RF(T|C,Q|C)}{r}\right) \times 100\% & \text{otherwise} \end{cases}$$

where $RF(T|C,Q|C)$ is the Robinson-Foulds distance between $T|C$ and $Q|C$ and r is the total number of non-trivial clusters in $T|C$ and $Q|C$.

LCA-based similarity

This measure is based on an LCA-based mapping from nodes in Q to nodes in T. The mapping assigns to each node x in Q a node $M(x)$ in T. This is done as follows. Let x be the set of all leaf-descendants of x in Q. Then, $M(x)$ is the LCA of set x in T. Node x in Q is said to be a *conflicting node* (with respect to T) if it has a sibling y such that $M(x)$ is an ancestor of (or equal to) $M(y)$ in the candidate tree *T, i.e.* $M(x)$ and $M(y)$ are the same node or two nodes that have an ancestor-descendant relationship in T but such that x and y are siblings in Q. Our search engine can find conflicting nodes rapidly because the storage mechanism it employs supports quick LCA calculation and fast determination of the ancestor/descendant relationship between any two nodes.

The number of conflicting nodes can vary between 0 and n-1, where n is the number of internal nodes in the query tree. If node x is conflicting, then x has a sibling y such that the leaf clusters defined by x and y do not induce disjoint clades in the candidate tree. Thus, the percentage of conflicting nodes is a measure of the level of agreement between the query tree Q with the candidate tree T.

To define a practical similarity measure, we must take into account the degree of overlap between the query tree Q and the candidate tree T. This is done as follows. Let q denote the number of internal nodes in $Q|C$, and p denote the number of conflicting nodes in $Q|C$ with respect to tree T. Observe that the number of conflicting nodes is at most $q-1$. Then, the similarity score of query tree Q to candidate tree T is given as follows.

$$similarity_{LCA}(Q,T) = \begin{cases} \frac{n(C)}{n(Q)} \times 100\% & \text{if} \quad q = 1 \\ \frac{n(C)}{n(Q)} \times \left(1 - \frac{p}{q-1}\right) \times 100\% & \text{otherwise} \end{cases}$$

Here, the term $1 - \frac{p}{q-1}$ captures the normalized value of the conflict in tree $Q|C$ with respect to tree T.

GENERAL CONCLUSION

As stated in the introduction, this dissertation had three purposes: (1) To present an improved heuristic algorithm for minimum-flip supertree construction. (2) To propose an alternative methods that do not require assumptions regarding the phylogeny or process of evolution to partition phylogenetic data sets (3) To present and develop framework for both taxonomic and phylogenetic queries as well as visualization tools that highlight the query results and provide links to NCBI and TBMap. In this section we review whether we achieved these objectives and discuss future research.

General Discussion

An O(n) time improved minimum-flip supertree algorithm

The earlier MRF heuristic found the best neighbor by computing the flip distance of each such neighbor from scratch. This failed to exploit the similarities between the current tree and its neighbors, and, consequently, was quite slow. The running times to find an optimal neighbor tree of a given n-taxon tree for rNNI, rSPR, and rTBR were $O(n^2m)$, $O(n^3m)$, and $O(n^4m)$, respectively. The new algorithms reduce these times by a factor of n, giving execution times of $O(nm)$, $O(n^2m)$, and $O(n^3m)$, respectively. In all three cases, the key is to preprocess the tree to allow evaluation of the flip distance of each neighbor in $O(1)$ time per character.

The new heuristic algorithm makes MRF analyses feasible for large empirical data sets and makes it possible to assess the performance of the MRF supertree method using data sets that would have been too computationally demanding for the previous heuristic method. In all three analyses, MRF appears to perform at least as well and often better than MMC and MRP. The results also emphasize the differences that may exist between MRP and MRF supertrees.

A good supertree method must balance computational speed with accuracy. For example, the MMC supertree method has a fast polynomial time algorithm (Page, 2002), but it often results in low quality supertrees. Conversely, the MRF supertree method appears to be accurate relative to other supertree methods, but previously its heuristics were too slow for large supertree studies (Eulenstein et al 2004). However, the availability of heuristics should not dictate one's choice of supertree methods. Rather the properties of a supertree method should motivate the development of useful heuristics. Though a number of supertree methods have been proposed (see Bininda-Emonds 2004), there has been much less focus on developing fast implementations of these methods. We have demonstrated that such work can benefit supertree analyses. We do not suggest that MRF is now the optimal supertree method. In some cases, MRF may exhibit undesirable properties (eg Goloboff 2005; Wilkinson et al 2005), and the speed of the new heuristics may still be a limitation for building supertrees with many thousand taxa or for implementing supertree bootstrapping replicates (eg, Creevey et al 2004; Philip et al 2005; Burleigh et al 2006). Still, with the new heuristics, MRF is, in many cases, a viable supertree method that should be considered along with other methods.

Spectral partitioning of phylogenetic data sets

In chapter 3 of this dissertation, we propose a fast approach for partitioning phylogenetic data sets that explicitly seeks to highlight conflicting phylogenetic signals among characters within a data set. Unlike previous methods, the spectral partitioning methods use pairwise compatibility to estimate shared phylogenetic signal, not rates of evolution. We show that a fast and powerful partitioning method from computer science (e.g., Ding et al 2001; Shi and Malik 2000; Yu and Shi 2003) can be used to cluster phylogenetic characters based on compatibility. Our simulation experiments demonstrate that partitioning characters based on our compatibility criterion performs well in identifying conflicting evolutionary histories within data sets. If the ILD scores are indicative of the degree of conflict among partitions, spectral partitioning performs nearly as well at revealing conflict as knowing the actual ("true") underlying topologies. Furthermore, In the empirical data sets, compatibility also appears to reveal higher levels of phylogenetic conflict than previously used partitioning methods. The much higher ILD scores from spectral partitioning coincide with higher average compatibility among sites in the spectral partitions.

An important issue in using spectral partitioning is to determine the optimal number of partitions. For simplicity, our examples use only two partitions; however, one can specify any number of partitions. There are numerous tests and methods to determine if data from different partitions represent significant phylogenetic conflict, some of which also can address the optimal number of partitions (e.g., Farris et al 1994; Huelsenbeck et al 1996; Mason-Gamer and Kellogg 1996; Dolphin et al 2000; Kauff and F. 2002; Vogl et al 2003; Ané et al 2005; Brandley et al 2005).

The spectral partitioning method may be extended in several directions. While we use spectral partitioning to cluster characters based on compatibility, the data could be clustered based on other characteristics, such as shared rates of evolution. Furthermore, in some cases it may be useful to incorporate biological constraints on the partitioning strategy. For example, we might not wish to combine characters from different loci into the same partition. Our examples demonstrate that spectral partitioning performs well in the absence of assumptions or regarding the characteristics of the data, and it likely can be improved by incorporating additional biological information.

A framework for searching phylogenetic databases

In chapter 4 of this dissertation, we design and implement PhyloFinder, a framework for phylogenetic tree search from phylogenetic databases. We have tested our system using trees from TreeBASE, the largest relational database of published phylogenetic information. PhyloFinder combines some techniques widely used in information retrieval and phylogenetic data integration that can provides many new features that are not available in current TreeBASE.

PhyloFinder enhances the utility of phylogenetic databases, by enabling scientists to identify and assess available phylogenetic data. The taxonomic search tools in PhyloFinder allow researches to identify all trees containing taxa of interest without knowing the names of all taxa in the available trees, possible synonymous taxon names, or even the correct spelling of a taxon name. The phylogenetic search tools allow one to evaluate phylogenetic hypotheses by comparing it to existing phylogenies and identifying trees that agree with or are similar to the query tree. While we have tested the utility of PhyloFinder using trees from TreeBASE, it can be implemented using any tree database.

Recommendations for future research

Although the information of the Tree of Life (ToL) would be enormously useful for biologists, there are a number of limitations that must be addressed before ToL can be explored in detail. A few interesting research directions that are related to this dissertation are:

- Developing a parallel algorithm and implementation for supertree reconstruction. Supertree methods are possible a key to ToL since there are around 1.7 million species on Earth, we cannot produce a global estimate of evolutionary history from molecular data of all these species with existing computational approaches. Therefore, the development of computer algorithms and methods is critical to the field of phylogenetics. It is important to develop tools to support large-scale phylogenetic supertree analyses using distributed computing resources and parallel computing techniques.

- A graph theoretic approach to detect fast-evolving site in phylogenetic data. It would be interesting to develop a new graph theoretic based methods that can use pairwise compatibility metric and should not require assumptions regarding the phylogeny, model of evolution, or characteristics of the data sets. Such a method would be based on the notion that fast evolving characters experience more homoplasy than slowly evolving characters and thus are pairwise compatible with fewer characters than slowly evolving characters. Compatibility and compatibility graphs have a long history in phylogenetics (see Meacham and Estabrook 1985; Semple and Steel 2003; Felsenstein 2004). While today compatibility criteria are seldom used to estimate phylogenies, compatibility has been used to weight characters for parsimony analyses (Penny and Hendy 1985, 1986; Sharkey 1989) and, more recently, to identify anomalous and fast-evolving characters in data sets (Meacham 1994; Pisani 2004).

- The relationship of phylogenetic treebases to other taxonomic and genomic databases should be explored. Integrating bio-diversity information is a very hot topic. This includes the need to accommodate multiple taxonomic names and classifications, collect massive annotation and linking to global identifiers for taxonomic names, sequences, images, and publications.

- Development of a phylogenetic knowledge-based web server that optimally represents and visualizes phylogeneticknowledge. It should provide an interactive and user friendly interface that allows users to easily navigate through the phylogenetic trees and other data resources such as character/gene sequences, published papers through a DOI or a PubMed id. It would be good to provide some XML based web services such as SOAP and tools for inter-connectivity and inter-operability between a wide variety of other data resources to facilitate data mining.

References

Ané C, , Sanderson MJ. 2005. Missing the forest for the trees: phylogenetic compression and its implications for inferring complex evolutionary histories. *Systematic Biology*, 54:146–157.

Bininda-Emonds ORP. 2004. The evolution of supertrees. *Trends in Ecology and Evolution*, 19:315–22.

Brandley M. C., Schmitz A., Reeder T. W.. 2005. Partitioned bayesian analyses, partition choice, and phylogenetic relationships of scincid lizards. *Systematic Biology*, 54:373–390.

Burleigh JG, Driskell AC, Sanderson MJ. 2006. Supertree bootstrapping methods for assessing phylogenetic variation among genes in genome-scale data sets. *Systematic Biology*.

Creevey CJ, Fitzpatrick DA, Philip GK, et al. 2004. Does a tree-like phylogeny only exist at the tips in the prokaryotes? *Proceeding of the Royal Society of London B Bio.*, 271:2551–8.

Ding Chris H. Q., He Xiaofeng, Zha Hongyuan, Gu Ming, Simon Horst D.. 2001. A min-max cut algorithm for graph partitioning and data clustering. In ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, p 107–114.

Dolphin K., Belshaw R., Orme C. D. L., Quicke D. L. J.. 2000. Noise and incongruence: interpreting results of the incongruence length difference tests. *Mol. Phylogenet. Evol.*, 17:401–406.

Eulenstein O, Chen D, Burleigh JG, et al. 2004. Performance of flip supertrees with a heuristic algorithm. *Systematic Biology*, 53 (2):299–308.

Farris J. S., Kållersjo M., Kluge A. G., Bult C.. 1994. Testing the significance of congruence. *Cladistics*, 10:315–319.

Felsenstein J.. 2004. Inferring Phylogenies. Sinauer Assoc., Sunderland, Mass.

Goloboff PA. 2005. Minority rule supertrees? MRP, compatibility, and minimum flip may display the *least* frequent groups. *Cladistics*, 21:282–94.

Huelsenbeck, J. P.and Bull J. J., , Cunningham C. W.. 1996. Combining data in phylogenetic analysis. *Trends Ecol. Evol.*, 11:152–157.

Kauff F., F. Lutzoni. 2002. Phylogeny of the gyalectales and ostropales (ascomycota, fungi): among and within order relationships based on nuclear ribosomal rna small and large subunits. *Mol. Phylogenet. Evol.*, 25:138–156.

Mason-Gamer R. J., Kellogg E. A.. 1996. Testing for phylogenetic conflict among molecular data sets in the tribe triticeae (gramineae). *Systematic Biology*, 45:524–545.

Meacham C. A.. 1994. Phylogenetic relationships at the basal radiation of angiosperms: further study by probability of character compatibility. *Systematic Biology*, 19:506–522.

Meacham C. A., Estabrook G. F.. 1985. Compatibility methods in systematics. *Annu. Rev. Ecol. Syst.*, 16:431–466.

Page RDM. 2002. Modified mincut supertrees. In Guigó R., Gusfield D., eds. Algorithms in Bioinformatics: Second International Workshop, WABI 2002. Vol. 2452 of *Lecture Notes in Computer Science*. Springer-Verlag, p 537–51.

Penny D., Hendy M.. 1986. Estimating the reliability of evolutionary trees. *Mol. Biol. Evol.*, 3:403–417.

Penny D., Hendy M. D.. 1985. Testing methods of evolutionary tree construction. *Cladistics*, 1:266–272.

Philip GK, Creevey CJ, McInerney JO. 2005. The opisthokonta and ecdysozoa may not be clades: Stronger support for the grouping of plant and animal than for animal and fungi and stronger support for the coelomata than ecdysozoa. *Molecular Biology and Evolution*, 22:1175–84.

Pisani D.. 2004. Identifying and removing fast-evolving sites using compatibility analysis: an example from the arthropods. *Systematic Biology*, 53:978–989.

Semple C., Steel M.. 2003. Phylogenetics. Oxford University Press, London.

Sharkey M. J.. 1989. A hypothesis-independent method of character weighting for cladistic analysis. *Cladisitics*, 5:63–86.

Shi Jianbo, Malik Jitendra. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8):888–905.

Vogl C., Badger J, Kearney P, Li M., Clegg M, Jiang T.. 2003. Probabilistic analysis indicates discordant gene trees in chloroplast evolution. *J. Mol. Evol.*, 56:330–340.

Wilkinson M, Cotton JA, Creevey C, et al. 2005. The shape of supertrees to come: tree shape related properties of fourteen supertree methods. *Systematic Biology*, 54:419–31.

Yu S., Shi J.. 2003. Multiclass spectral clustering.