Contents

1 High-Performance Phylogeny Reconstruction Under Maximum Parsimony

DAVID A. BADER[†], BERNARD M.E. MORET[‡], TIFFANI L. WILLIAMS[§], and MI YAN[†]

1.1 INTRODUCTION

The similarity of the molecular matter of the organisms on Earth suggest that they all share a common ancestor. Thus any set of species is related, and this relationship is called a phylogeny. The links (or evolutionary relationships) among a set of organisms (or taxa) form a phylogenetic tree, where modern organisms are placed at the leaves and ancestral organisms occupy internal nodes, with the edges of the tree denoting evolutionary relationships. Scientists are interested in evolutionary trees for the usual reasons of scientific curiosity. However, phylogenetic analysis is not just an academic exercise. Phylogenies are the organizing principle for most biological knowledge. As such, they are a crucial tool in identifying emerging diseases, predicting disease outbreaks, and protecting ecosystems from invasive species [Bader et al., 2001, Cracraft, 2002]. The greatest impact of phylogenetics will be reconstructing the Tree of Life, the evolutionary history of all-known organisms. The precise number of organisms that exist is not known; estimates range from 10-100 million species. Today, only about 1.7 million species are known, and a very

[†]Department of Electrical and Computer Engineering, University of New Mexico
[‡]Department of Computer Science, University of New Mexico
[§]Radcliffe Institute of Advanced Study, Texas A&M University

small fraction (i.e., on the order of 0.4%) are included in any sort of phylogenetic analysis [Yates et al., 2004].

Given the enormous implications of phylogenetic analysis, reconstructing the "true" evolutionary history of a set of tax is a very difficult problem. For n organisms, there are $(2n-5)(2n-3)\cdots(5)(3)$ distinct binary trees—each a possible hypothesis for the "true" evolutionary history. For example, there are over 13 billion possible trees for 13 taxa. Since the size of the tree space increases exponentially with the number of taxa, it is impossible to explore all of the possible trees within a reasonable time frame. Most phylogenetic methods limit themselves to exhaustive searches on extremely small datasets (< 50 sequences) and heuristic strategies for larger datasets. Another difficulty lies in accessing the accuracy of the reconstructed tree. Short of traveling back into time, there is no way of determining whether the proposed evolutionary history is 100% correct.

1.1.1 Phylogenetic Data

Early evolutionary trees were built by looking for similarities and differences of form and structure of the the organisms of interest. Such an approach relies on identifying morphological characters and classifying organisms based on the presence or absence of these features (i.e., beak, legs, wings). Species are represented by binary sequences corresponding to the morphological data. Each bit corresponds to a character (or feature). If a species has a given feature, the corresponding bit is set to one; otherwise, it is zero. Relying solely on morphological characters can be a major source of phylogenetic error. Morphological features used as a criteria of relatedness must have a common evolutionary origin (i.e., the features are homologous). Classing birds and bees together because they both have wings is clearly not appropriate. Hence, one of the major problems with using morphological characters is establishing which characters are homologous and which are not.

With the advent of molecular data, scientists hope to avoid some of the problems with morphological criteria of relatedness. Today, most trees are built exclusively from by molecular sequences. In sequence data, characters are individual positions

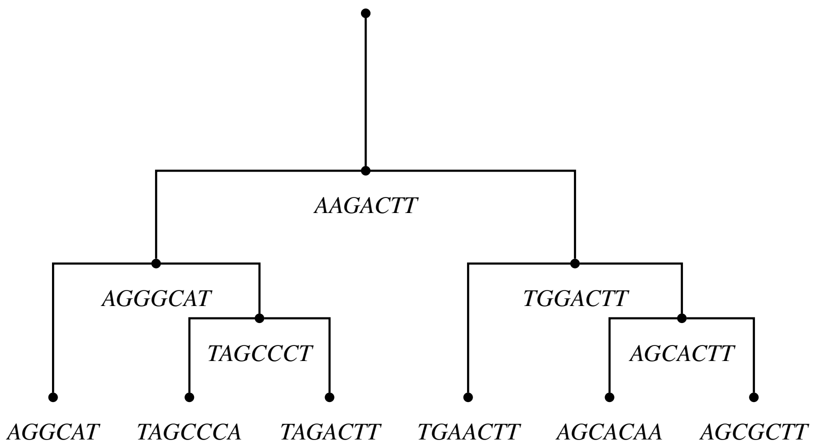


Fig. 1.1 Evolving sequences down a fixed tree.

(or sites) in the string, where characters can assume one of four states for nucleotides (A,C,G,T) or one of 20 states for amino-acids. The evolution of sequences is studied under a simplifying assumption that each site evolves independently. Data evolve through point mutations (i.e., changes in the state of a character), plus insertions (including duplications) and deletions.

Figure 1.1.1 shows a simple evolutionary history, from the ancestral sequence at the root to modern sequences at the leaves, with evolutionary events occurring on each edge. Note that this history is incomplete, as it does not detail the events that have taken place along each edge of the tree. Thus, while one might reasonably conclude that, in order to reach the leftmost leaf, labeled AGGCAT, from its parent, labeled AGGGCAT, one should infer the deletion of one nucleotide (one of the three G's in the parent), a more complex scenario may in fact have unfolded. If one were to compare the leftmost leaf with the rightmost one, labeled AGCGCTT, one could account for the difference with two changes: starting with AGGCAT, insert a C between the two G's to obtain AGCGCAT, then mutate the penultimate A into a T. Yet the tree itself indicates that the change occurred in a far more complex manner: the path between these two leaves in the tree goes through the series of sequences

$$\text{AGGCAT} \leftrightarrow \text{AGGGCAT} \leftrightarrow \text{AAGACTT} \leftrightarrow \text{TGGACTT} \leftrightarrow \text{AGCACTT} \rightarrow \text{AGCGCTT}$$

and each arrow in this series indicates at least one evolutionary event.

Large amounts of sequence data are easily attainable from databases such as GenBank, along with search tools (such as BLAST). However, "raw" sequence data must first be refined into a format suitable for use in a phylogenetic analysis. The refinement process consists of:

1. Identifying *homologous* genes (i.e., genes that have evolved from a common ancestral gene—and most likely fulfill the same function in each organism) across the organisms of interest;

2. Retrieving followed by aligning the sequences of these genes across the entire set of organism, in order to identify gaps (corresponding to insertions or deletions) and matches or mutations; and

3. Deciding whether to use all available data at once for a *combined* analysis or use each gene separately and *reconcile* the resulting trees.

4. Applying the method of choice to the aligned sequence data (see Section) 1.1.2.

Many packages requiring sequence data are available to reconstruct phylogenetic trees such as PAUP* [Swofford, 2002], MacClade, Mesquite [Maddison and Maddison,], Phylip [Felsenstein,], MrBayes [Huelsenbeck and Ronquist, 2001], and TNT [Goloboff, 1999]. These packages are available either freely or for a modest fee, are in widespread use, and have provided biologists with satisfactory results on many datasets.

Other types of phylogenetic data exist to reconstruct phylogenies. Phylogenetic inference based on gene order data provide an alternative to using sequence data. Gene order data is based on the structural arrangement of genes in an organism's entire genome. Hence, the gene tree/species tree problem (i.e., the evolution of any given of any given gene need not be identical to that of the organism) is avoided when using gene order data. Gene order data is sparsely available in comparison to sequence data. Consequently, most tree reconstruction efforts are focused on the reconstruction of phylogenetic trees based on sequence data. See (Moret and Warnow refs) for a more detailed overview of phylogeny reconstruction based on gene-order data.

1.1.2 Phylogenetic Methods

Different tree reconstructing methods commonly produce different tree topologies from the same data. Thus, there is a problem in deciding which are more correct for the given data. There is an ongoing search for tree-building methods that are the most robust (that are more likely to estimate the true topology even when the evolutionary assumptions are violated), consistent (that converge on the true topology as more data is added), and efficient (that converge on the topology most quickly). Attempts to improve the accuracy of phylogenies have resulted in hundreds of methodological variations to take account of biases that result from different evolutionary processes. Yet, the plethora of methods can be divided into two broad categories. *Distance-based methods* transform the sequence data into a numerical representation of the data. On the other hand, *Criteria-based methods* rely on optimality criteria to score the tree based on the individual contribution of the characters in the sequence data.

1.1.2.1 Distance-based methods Nucleotide sequence similarities may be converted to sequence distances in order to calculate the positions of the nodes in the phylogenetic tree. Distances are calculated for all pairs of sequences under study, thus creating a distance matrix. Hence, for n sequences, an $n \times n$ matrix of distances will be computed. One possible distance metric to compute between two sequences is their Hamming distance. However, the Hamming distance is an underestimate of the true genetic distance between a pair of sequences. If mutations occur at random, there will be a certain number of positions with *silent mutations*– changes that are subsequently reversed in the course of evolution, leaving no trace in modern organisms (see Figure 1.1.2.1). Although a sequence is evolving, a fraction of the evolutionary change will be hidden. Therefore, in distance-based methods, one estimates the number of substitutions that have actually occurred by applying a specific *evolutionary model*, which makes assumptions about the nature of evolutionary changes. For example, the Jukes-Cantor model assumes that all evolutionary changes are equally likely whereas Kimura's two parameter model provides two different rates for transversions (example) and transitions (example). (refs) When all

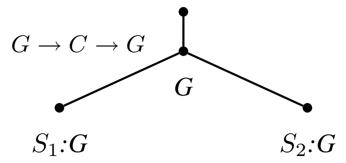


Fig. 1.2 The effect of silent mutations. The Hamming distance between sequences S_1 and S_2 is 0. However, S_1 encountered 2 mutations before reaching the state G that it shares with its parent.

the pairwise distances have been computed for a set of sequences, a tree can then be inferred.

Distance-based methods build the search for the tree into the algorithm, thus returning a unique final topology for a distance matrix associated with a given set of sequences. Neighbor-Joining (NJ) is the most popular distance-based algorithm [Saitou and Nei, 1987]. The NJ algorithm begins with each species in its own subtree. The algorithm joins the pair with the minimum distance, making a subtree whose root replaces the two chosen taxa in the matrix. Distances are recalculated based on this new node, and the joining continues until three nodes remain. These nodes are joined to form an unrooted binary tree. Appealing features of NJ are its simplicity and speed—it runs in $O(n^3)$ time. Other distance methods include UPGMA and refinements of NJ such as BioNJ [Gascuel, 1997] and Weighbor [Bruno et al., 2000].

1.1.2.2 Criteria-based methods Criteria-based methods explicitly rank the tree topologies by defining an objective function to score the trees. Tree scores allows any two or more trees to be ranked according to the chosen optimality criterion. Unlike distance-based approaches, there are many possible solutions given a set of sequences for criteria-based approaches. Hence, there is an explicit search for the "optimal" tree. Two of the most popular criteria-based methods are maximum parsimony (MP) and maximum likelihood (ML). We discuss each in turn below.

Parsimony methods An intuitive approach for ranking phylogenetic trees is counting the total number of mutations required to explain all of the observed character sequences. The maximum parsimony approach attempts to minimize this score following the philosophy of Occam's razor—the simplest explanation of the data is preferred. Under parsimony, a total cost is assigned to each tree, and the optimal tree is defined as the one with the smallest total cost. In this approach, a unit cost is given for each nucleotide substitution. A central step in the procedure is allocating sequences to the internal nodes in the tree. For any set of sequence allocations, the total cost of the tree is the sum of the costs of the various edges, where the cost of joining two internal nodes, or an internal node and a leaf, is the number of substitutions need to move from the sequence at one to the sequence at the other. Many software packages implement MP heuristics, among them the most popular are PAUP* [Swofford, 2002], Phylip [Felsenstein,], and TNT [Goloboff, 1999].

Likelihood methods Likelihood methods require the specification of a substitution model of evolution. For a given phylogenetic arrangement, the question is: what is the likelihood that evolution under the specified parameters will produce the observed nucleotide sequences? For a given evolutionary hypothesis, the likelihood of the observed change is computed for each position, and then the product of the likelihoods is expressed as a distance or branch length between the sequences. The parameters are then varied, and the combination with the highest likelihood is accepted. This procedure is then repeated for another arrangement, the two topologies compared, and the one with the highest likelihood selected. The selective process is continued (via some algorithm) until an arrangement is found with the combined maximum likelihood of both an evolutionary hypothesis and a topology. This method is the most computationally demanded of the methods discussed here.

Like MP, ML is an optimization problem. ML seeks the tree and associated model parameter values that maximizes the probability of producing the given set of sequences. ML thus depends explicitly on an assumed model of evolution. For example, the ML problem under the Jukes-Cantor mode needs to estimate one parameter (the substitution probability) for each edge of the tree, while under the General Markov model 12 parameters must be estimated on each edge. Unlike MP, scoring a fixed tree, cannot be done in polynomial time for ML [Steel, 1994], whereas it is easily accomplished in linear time for MP using Fitch's algorithm [Fitch, 1971]. Various software packages provide heuristics for ML, include PAUP* [Swofford, 2002], Phylip [Felsenstein,], fastDNAml [Olsen et al., 1994], and PHYML [Guindon and Gascuel, 2003].

1.2 MAXIMUM PARSIMONY

Maximum parsimony is an optimization problem for inferring the evolutionary history of different taxa, in which it is assumed that each of the taxa in the input is represented by a string over some alphabet. The input consists of a set S of n strings over a fixed alphabet Σ, where $\Sigma = \{A, C, G, T\}$ represents the set of four nucleotides. Σ could also represent the set of amino-acid sequences. The elements of Σ are also called "states". We assume that the sequence data has been properly prepared; particularly, the sequences are already aligned so that all sequences are of length k. Positions within the sequences are sometimes called sites.

A formal definition of the maximum parsimony problem is as follows. Given two sequences a and b of length k, the *Hamming distance* between them is defined as $H(a, b) = |\{i : a_i \neq b_i\}|$. Let T be a tree whose nodes are labeled by sequences of length k over Σ, and let $H(e)$ denote the Hamming distance of the sequences at each endpoint of e. Then the *parsimony length* of the tree T is $\sum_{e \in E(T)} H(e)$. The MP problem seeks the tree T with the minimum length; this is the same as seeking the tree with the smallest number of point mutations for the data. MP is an NP-hard problem [Foulds and Graham, 1982], but the problem of assigning sequences to internal nodes of a fixed leaf-labeled tree is polynomial [Fitch, 1971].

1.2.1 An algorithm for scoring a tree

Given the tree, and the labeling of the leaves, label the internal nodes to minimize the number of changes. The algorithm has two steps—an *upward sweep* (to compute the parsimony score and the possible states that lead to that score) and a *downward sweep* (to assign sequences to the internal nodes).

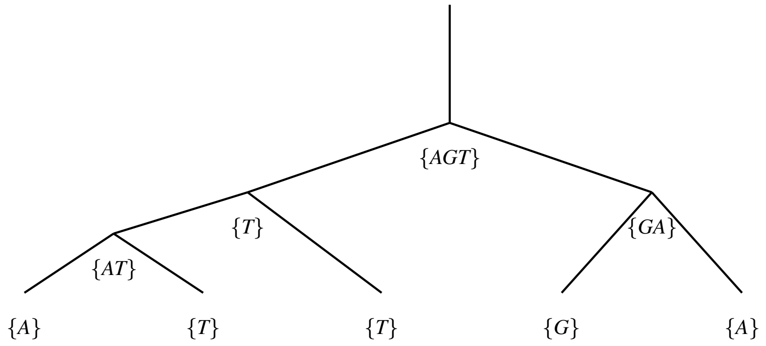


Fig. 1.3 A tree with a parsimony score of 3 based on a single site.

First, we define the possible states for each of the internal nodes that minimize the score. Let $S_v \subseteq \Sigma$ denote the set of state assignments for node v. We assume that T is binary and the children of v are x and y. If v is a leaf, then S_v is simply the state of v. If v is an internal node, then it's state is based on the state of its two children x and y (i.e., S_x and S_y). If $S_x \cap S_y \neq \oslash$, then $S_v = S_x \cap S_y$. Otherwise, $S_v = S_x \cup S_y$.

Using a postorder traversal, the above equalities allow us to compute $_v$ for every node v in T, from the bottom up. Moreover, the optimal cost (or maximum parsimony score) of T can be calculated from the bottom-up at the same time. Every time $S_u \cup S_w = \oslash$, we increment the parsimony score of the tree by one. The sum of these values over all the sites is the parsimony score of the tree. Figure 1.2.1 provides an example of the parsimony score calculation on a single site for a fixed tree.

Once states have been assigned to all of the internal nodes, we can obtain a labeling of them using a preorder traversal. Once again, we can compute the positions (sites) independently. For the root r arbitrarily assign the state for r to be any element of S_r. Then, visit the remaining nodes in turn, every time assigning a state to the node v from its set S_v. When we visit a node v we will have already set the state of its parent, p. If the selected state for p is an element of S_v, then we use the same state. Otherwise we pick a state arbitrarily from S_v.

This algorithm requires $O(nk)$ time to compute the labeling of every node in T and the optimal length (i.e., maximum parsimony score) of T, where $n = |S|$, and k is the sequence length.

1.2.2 Parsimony-Informative Characters

If two characters are observed at a site, but one of the two occurs in one taxon only, the site will require one change for any tree. In other words, if the change is only valid for one taxa, the parsimony score for that site will be the same. Such singleton sites require one change for any tree and are not informative. Parsimony-informative sites include only those sites where at least two distinct characters are observed two or more times. Consider the following sequences for four taxa.

Sequence	Site			
	1	2	3	4
S_1	G	T	A	G
S_2	G	T	A	C
S_3	G	T	G	G
S_4	G	C	G	C

```
Note: Need to draw the following unrooted trees.
Tree I = ((s1,s2),(s3,s4));
Tree II = ((s1,s3), (s2,s4));
Tree III = ((s1, s4), (s2,s3));
```

The four sequences can be related to one another in three different ways (draw figure for this). Let us now consider one site after the other in terms of the minimum changes it involves. Site 1 does not seem to have changed at all and can therefore be ignored. Site 2 must have suffered at least one change, no matter how we arrange the branches. (This site could, of course, have undergone more than one change, but this assumption is not maximally parsimonious and so is not taken into consideration). For site 2, all three trees are equally likely and so the site is regarded, like site 1, as being phylogenetically uninformative and as such is not considered any further. Site 3, on the other hand is phylogenetically informative, because it enables use to choose the most parsimonious tree of the tree. Tree 1 requires one change, whereas trees II and III each require at least two changes to explain the pattern of nucleotide distribution at this site among the four sequences. Finally, site 4 is also phylogenetically informative and identifies tree II rather than I as the most parsimonious of the tree. So, for this set of four sequences, trees I and II are the most parsimonious trees.

So for four taxa, only three site configurations or site patterns are informative: $aabb, abab, abba$, where a and b are two different states from Σ. Informative and non-informative sites affect only parsimony. Distance and likelihood methods, all sites including the constant site affect the calculation and should be included.

1.2.3 Computing the MP score exactly

One approach to solving MP is to evaluate all possible trees. As a result, the most parsimonious tree is guaranteed to be found. Thus, an algorithm is needed that can guarantee generation of all possible trees for evaluation of all possible trees. One procedure for for generating all possible trees is as follows. The algorithm recursively adds the ith taxon in a stepwise fashion to all possible trees containing the first $i-1$ taxa until all n taxa have been joined. The above algorithm makes it clear that the number of possible trees grows by a factor that increase by two with each additional taxon. This relationship is expressed as $B(t) = \prod_{i=3}^{t}(2i-5)$, where $B(t)$ is the number of unrooted trees for t taxa. As stated earlier, for 13 taxa there are over 13 billion trees to score. Clearly, the exhaustive search method can only be used for only a relatively small number of taxa. An alternative exact procedure, the branch-and-bound method (Hendy.1982.A) operates implicitly by evaluating all possible trees, but cutting off paths of the search tree when it is determined that they cannot possibly lead to optimal trees. We provide a high-performance B&B algorithm in section 1.3.

1.2.4 Approximating the MP score

When data sets become too large to use the exact searching methods, one must resort to the use of heuristics—approximate methods that attempt to find optimal solutions but provide no guarantee of finding the optimal solution. Generally, a two-phase algorithm is used to conduct approximate searches. In the simplest case, an initial starting tree is generated using a ”greedy” algorithm that builds a tree iteratively. In general, this tree will not be optimal since decisions are made in early stages without regard for their long-term implications. After this starting tree is obtained, it is submitted to a round of perturbations in which neighboring trees are evaluated. If a perturbation yields a tree with a better MP score, it becomes the new ”best” tree and it is then submitted to a new round of perturbations. This process continues until no better tree can be found in a full round of the perturbation process.

1.3 EXACT MP: PARALLEL BRANCH & BOUND

Although it takes polynomial time to compute the tree cost by Fitch's method ([Fitch, 1971]), it is still very time-consuming to compute the exact MP by exhaustive search due to the enormous size of search space. Hence we use branch-and-bound (B&B) to prune the search space in phylogeny reconstruction. The underlying idea of the B&B algorithm is successive decomposition of the original problem into smaller disjoint subproblems and pruning those subproblems whose lower bound is greater than the upper bound until an optimal or all optimal solutions are found.

1.3.1 Basic issues in branch and bound approach

Our B&B approach have five aspects that mainly affect the performance of the algorithms: branching scheme, search strategy, lower bounding function, initial global upper bound, and the data structure. We will discuss these five aspects respectively in the following.

1.3.1.1 Branching scheme The branch scheme decides how to decompose a subproblem in the search space. In phylogeny reconstruction, each subproblem is associated with a partial tree and the objective is to find the exact MP score among those trees built from the partial tree. Our branch scheme employs the same mechanism to generate all possible unrooted binary trees for a given set of taxa. Consider the unrooted tree for three taxa. The remaining $n-3$ taxa are added to the tree in step-wise fashion as described in Section 1.3. Each new position location for taxon i of the partial tree is considered a subproblem. Thus, a subproblem associated with the original partial tree is decomposed into a set of disjoint subproblems; each associated with a new partial tree.

1.3.1.2 Search strategy The search strategy decides which of the currently open subproblems to be selected for decomposition. The two strategies most commonly used are *depth-first search (DFS)* and *best-first search (BeFS)*. DFS is space-saving and BeFS is more targeted towards a better global upper bound. In the case when the initial global upper bound obtained by heuristic approaches is exactly optimal or very close to exact optimal value, there is no significant difference in the number of examined subproblems between DFS search and BeFS search, therefore DFS has more advantage for reasons of space efficiency. Since our experiment shows that heuristics can provide a very good solution, we employ DFS as our primary B&B search strategy and adopt BeFS as the second strategy to break the tie.

1.3.1.3 Lower bounding function of the subproblems Hendy and Penny ([Hendy and Penny, 1982]) use the cost of the associated partial tree as the lower bound of a subproblem. In ([Purdom et al., 2000]), Purdom et al. used the sum of the single column discrepancy and the cost of the associated tree as the lower bound. For each column (character), the single column discrepancy is the number of states that do not occur among the taxa in the associated tree but only occur among the remaining taxa. We employ Purdom's lower bounding function since it is much tighter than than Hendy and Penny's.

1.3.1.4 Initial global upper bound We do not compute the upper bound for each subproblem. Instead, before the B&B search, a global upper bound is obtained by a fast heuristic algorithm. We investigate the performance of both the neighbor-joining (NJ) and greedy algorithms. From experiments, we found that the tree obtained from NJ is usually much worse than the one obtained from the greedy algorithm. The greedy algorithm constructs a tree in a stepwise fashion, at each step, the new taxon is added into the best position which results a partial tree with minimum score. Since adding taxa in different orders yields different trees, we use the greedy algorithm with two different addition orders and let the better score to be the initial global upper bound.

1.3.1.5 Data structure Each subproblem in the search space of the B&B phylogeny reconstruction is associated with a partial tree and the lower bound. The lower bound serves as the priority and priority queues are used to save the open problems. Since we use DFS search, it is natural to use a priority queue for each depth. Several types of heaps can be found in literature. For simplicity, a traditional D-heap ([Knuth, 1973]) is chosen to represent a priority queue. A D-heap is organized as an array, using the rule that the first location is the root of the tree, and the locations $2i$ and $2i + 1$ are the children of location i.

1.3.2 Preprocessing before B&B search

We adopt a series of preprocessing in order to conduct the B&B search efficiently.

1.3.2.1 Binary encoding of original states The basic operation of Fitch's method is to compute the intersection or union of state sets. Since most modern computers can perform efficient bitwise logical operations, we decide to use the binary encoding of state sets in order to implement intersection and union by bitwise AND and bitwise OR. Assign a one-to-one map between the bits of code and the character states. Given a species, if a state is present, then the corresponding bit is set to one otherwise it is set to zero.

1.3.2.2 Decide the addition order of the species Our experiments show that the overall execution time of B&B phylogeny reconstruction can change drastically depending on the order in which the taxa are added. This can be explained in theory. The lower bounding function we adopt heavily depends on the cost of the associated partial tree. This can also explain why the addition order decided by max-mini rule performs best in most cases. Starting with the initial core tree of three taxa, at each step, for each of the remaining taxa, we find the best inserting position which results the minimum score. Then, we choose the taxon with maximum minimum-score to be added at its best position and go onto next step until all taxa are added. This procedure is called the max_mini approach.

1.3.2.3 Reorder sites Fitch([Fitch, 1977]) made a basic classification of sequence sites (the columns of the sequence matrix). At a given site, the state that appears more than once is said to be a *non-singleton state*. A site with at most one non-singleton states is said to be a *parsimony uninformative site* since the state changes at such kind of a site can always be explained by the same number of substitutions in all topologies. At the lower levels of B&B phylogeny reconstruction, only a few sites are parsimony informative and with the addition of taxa, more and more sites turn from parsimony uninformative to parsimony informative. Hence, we may compute at which level a site turns into parsimony informative, then reorder sites so that at each level all of the parsimony informative sits are kept in a contiguous segment of memory. By reordering sites, not only the computation on parsimony uninformative sites are saved, but also the ratio of cache misses is greatly reduced.

1.3.3 Fast algorithm to compute tree length

Even in the B&B search, an enormous number of trees are required to be evaluated. Given an original tree, how can we compute the scores of each new tree generated by adding a taxon in the original one? As described in Section 1.2.1, Fitch ([Fitch, 1971]) proposed a method to score a tree. Fitch's method involves one bottom-up pass and one top-down pass of the tree. Each pass compute a set of states for each internal node by different rules, the states obtained in the first pass is called *preliminary states* and the states obtained in the second pass is called *final states*. Goloboff ([Goloboff, 1993]) proposed a method to preprocess the original tree in two passes as Fitch's method does, then it takes constant time to compute the score for each new tree. Gladstein ([Gladstein, 1997]) described an incremental algorithm based on preliminary state sets obtained from Fitch's first pass. Practically Goloboff's method works better than Gladstein's. We put forward an approach that requires preprocess the original tree in one pass and for each new tree it takes constant time to compute the score.

We do the first pass bottom-up using the rules of Fitch's first pass, then do the second pass top-down using the rules of Fitch's first pass instead of Fitch's second pass. By these two passes, we finally obtain a set of state for each edge, which is right the preliminary states of the root assuming the root to be the newly introduced node by subdividing that edge. Therefore, when inserting a new taxon in the original tree at an edge, we only compare the states of the new taxon and the states of that edge and actually obtain the same result as our bottom-up pass does on the new tree. If the result of the new tree is kept in memory, when we decompose this new tree later, only the top-down pass is required. Thus, in B&B search, our method saves one pass compared to Globoff's method. Besides the B&B search, our method can also be applied into heuristic searches such as SPR and TBR search.

1.3.4 Parallel implementation on SMPs

To utilize the computation power of parallel computers, we implement the B&B phylogeny reconstruction on Cache-Coherent Uniform Memory Access (CC-UMA) SMPs. In the parallel implementation, each processor selects different active nodes then processes the computation on it. We use the SPMD asynchronous model in which each processor works at its own pace and does not have to wait at predetermined points for predetermined data to become available. Since the B&B search space tends to be highly irregular, any static distribution of search space is bound to result in significant load imbalance, and the dynamic distribution methods usually involve very complex protocols to exchange subspace between processors to obtain load balance. Compared to the distributed data structure, a single shared data structure balance is easily to maintained on SMPs and there is no problem of load balance with it. We modify the serial data structure by adding a lock for each heap to get the shared data structure. Each heap is protected by a lock and the entire heap is locked whenever it is being modified. Due to the small size of heaps in B&B phylogeny reconstruction, D-heap is used for simplicity and efficiency.

To minimize the concurrent access contention, a relaxed DFS search strategy is adopted. A heap is allowed to be accessed if all the heaps at higher levels are empty or locked by other processors. When a processor detects that all the heaps are unlocked and empty, this processor can terminate its own execution of the algorithm.

1.3.5 Experimental results

We use the benchmark collection at http://www.lirmm.fr/ ranwez/PHYLO/benchmarks24.html. Each data set consists of 24 sequence and the length of DNA sequences is 500. These tests allow comparison on trees whose internal branch lengths are not all equal, and over a wide variety of tree shapes and evolutionary rates.

We compared the running time between our serial code and PAUP* using the subcommand *bandb addseq=maxmini* on a Sun UltraSparcII workstation. Among 20 data sets randomly chosen from the benchmark, for 10 data sets our code is 1.2-7 times faster than PAUP*, for 5 data sets ours code runs as fast as PAUP*, for 5 data sets our code is 1.2-2 times slower than PAUP*. The experiment on our parallel code was carried out on Sun E4500, a uniform-memory-access (UMA) shared-memory parallel machine with 14 UltraSparcII 400MHz processors. We conducted the experiment on 200 data sets randomly chosen from the benchmark, in average we achieve speedups of 1.92, 2.78, and 4.34, on 2, 4, and 8 processors, respectively. The above experimental results show that our strategies on the B&B phylogeny reconstruction are efficient.

1.4 APPROXIMATE MP: DISK-COVERING METHODS

Disk-Covering Methods (DCMs) [Huson et al., 1999a, Huson et al., 1999b, Nakhleh et al., 2001, Roshan et al., 2004b, Warnow et al., 2001] are a family of divide-and-conquer methods designed to "boost" the performance of existing phylogenetic reconstruction methods. All DCMs proceed in four major phases: (i) decomposing the dataset, (ii) solving the subproblems, (iii) merging the subproblems, and (iv) refining the resulting tree. Variants of DCMs come from different decomposition methods—the last three phases are unaffected. The first DCM [Huson et al., 1999a], also called DCM1, was designed for use with distance-based methods and has provable theoretical guarantees about the sequence length required to reconstruct the true tree with high probability under Markov models of evolution [Warnow et al., 2001]. The second DCM [Huson et al., 1999b], also called DCM2, was designed to speed up heuristic searches for MP trees; we showed that when DCM2 was used with PAUP*-TBR search, it produced better trees faster on simulated datasets.

1.4.1 DCM3

We designed the third DCM. or *DCM3*, from the lessons learned with our first two DCMs. DCM1 can be viewed, in rough terms, as attempting to produce overlapping clusters of taxa to minimize the intracluster diameter; it produces good subproblems (small enough in size), but the structure induced by the decomposition is often poor. DCM2 computes a fixed structure (a graph separator) to overcome that drawback, but the resulting subproblems tend to be too large. Moreover, both DCM1 and DCM2 operate solely from the the matrix of estimated pairwise distances, so that they can produce only one (up to tiebreaking) decomposition. In contrast, DCM3 uses a dynamically updated *guide tree* (in practice, the current estimate of the phylogeny) to direct the decomposition—so that DCM3 will produce different decompositions for different guide trees. This feature enables us to focus the search on the best parts of the search space and is at the heart of the iterative use of the decomposition: roughly speaking, the iteration in *Rec-I-DCM3* consists of successive refinements of the guide tree. Thanks to the guide tree, DCM3 also produces smaller subproblems than DCM2: the guide tree provides the decomposition structure, but does so in a manner responsive to the phylogenetic estimation process. Finally, we designed DCM3 to be much faster than either DCM1 or DCM2 in producing the decompositions (mostly by not insisting on their optimality), since previous experiments had shown that dataset decomposition used most of the running time with DCM2.

1.4.1.1 The optimal decomposition We begin by describing the algorithm to find an *optimal* DCM3 decomposition. (As noted, it is not the decomposition used in our implementation, but we need it to establish the framework.) Consider a tree T on our set S of taxa and an edge weighting w of T, $w\colon E(T) \to \Re^+$. (A possible edge weighting is given by the Hamming distances under the MP labelling of the nodes of T.) We construct the *short subtree graph*, which is the union of cliques formed on "short subtrees" around each edge. Let e be an internal edge (not touching a leaf) in T; then removing e and its two endpoints from T breaks T into four subtrees. A *short quartet* around e is composed of four leaves, one from each of these four subtrees, where each leaf is selected to be the closest (according to the edge weights) in its tree to e. This short quartet need not be unique: several leaves in the same subtree may lie at the same shortest distance from e. Thus we define the *short subtree* around e to be the set $X(e)$ of all leaves that are part of a short quartet around e. We will use the *clique* on $X(e)$: the graph with $X(e)$ as its vertices and with every pairwise edge present, weighted according to w; denote this clique by $K(e)$. The *short subtree graph* is then the union, over all internal edges e of the guide tree, of the $K(e)$.

Theorem 1. *The short subtree graph G of an edge-weighted binary tree T is triangulated (that is, it does not contain any simple induced cycle with more than three edges).*

(Proof omitted due to space constraints.) Since the short subtree graph G is triangulated, we can find in polynomial time, as proved in [Golumbic, 1980], a maximal clique separator X that minimizes $max_i|X \cup C_i|$, where $G - X$ is the union of m components $C_1, C_2, \ldots, C_m$. Using X, we can define the *Optimal DCM3 Decomposition* to be formed of the subsets $C_i \cup X$, for $i = 1, 2, \ldots, m$. Its computation thus proceeds in two steps:

1. Construct the short subtree graph.
2. For each of the maximal cliques in the short subtree graph, determine if it is a separator. If so, then compute the maximum size of any created subproblem and choose that separator which minimizes it.

Theorem 2. *The Optimal DCM3 decomposition can be computed in $O(n^3)$ time.*

Proof. Constructing the short subtree graph takes $O(n^3)$ time. There are $O(n)$ maximal cliques [Golumbic, 1980]. For each maximal clique, determining whether it is a separator and, if so, computing the sizes of the components, takes $O(n^2)$ time. $\square$

Constructing the short subtree graph is the costliest part of this decomposition. Although it may be possible to compute the graph faster, it seems likely that obtaining an Optimal DCM3 decomposition requires $\Omega(n^2)$ time. Thus the optimal decomposition is too expensive to compute for the kind of n we have in mind—even if we can obtain one optimal decomposition relatively fast for one tree on a million taxa, we will need to iterate this computation many, many times.

1.4.1.2 A fast suboptimal decomposition We now describe a fast heuristic to obtain a good, if not optimal, decomposition based on the guide tree. Our approach is based on the notion of a *centroid edge* in T—that is, an edge that, when removed, produces the most balanced bipartition of the leaves. Let X be the leaves of the short subtree around a centroid edge e. In our experience to date, X is always a separator of the short subtree graph, so we can define the subproblems as $A_i = X \cup C_i$, where $G - X$ has m distinct connected components, $C_1, C_2, \ldots, C_m$. (Should X fail to be a separator in the short subtree graph, we would then resort to computing all maximal clique separators in G.) Since we cannot afford to compute the short subtree graph, we cannot directly verify that X is a separator, but the following theorem shows that we can proceed without knowing the short subtree graph.

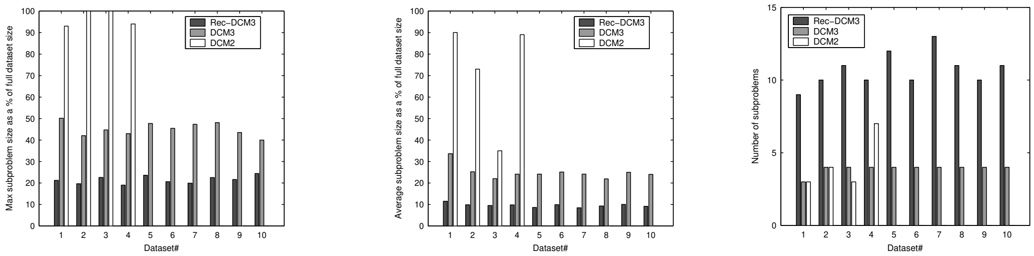
Theorem 3. *Let $e = \{a, b\}$ be an edge in the guide tree T and $X(e)$ the leaves in the short subtree around e. Let A, B, C, and D be the sets of leaves in the four subtrees obtained by deleting a and b from T. Finally, let G be the short subtree graph defined by T. Then every component of $G - X(e)$ is a subset of one of the following sets: $A \cup X(e)$, $B \cup X(e)$, $C \cup X(e)$, and $D \cup X(e)$.*

(Proof omitted due to space constraints.) By using this result, we can compute a decomposition that is not exactly that induced by the centroid edge, but that retains good characteristics (a small number of small subproblems).

Theorem 4. *The heuristic Centroid-Edge DCM3 decomposition can be computed in $O(n)$ time.*

Proof. We begin by finding a centroid edge e through a simple tree traversal, then compute $X(e)$, also in linear time. We can then compute the subproblems $A \cup X(e)$, $B \cup X(e)$, $C \cup X(e)$, and $D \cup X(e)$ in linear time. □

1.4.1.3 Comparison of DCM decompositions We designed DCM3 in part to avoid producing large subsets, as DCM2 is prone to do. Yet, of course, the subproblems produced from a very large dataset remain too large for immediate solution by a base method. Hence we used the approach successfully pioneered by Tang and Moret with DCM-GRAPPA [Tang and Moret, 2003] and used DCM3 recursively, producing smaller and smaller subproblems until every subproblem was small enough to be solved directly. Figure 1.4 shows that DCM3 produces subproblems of sizes bounded by about half the initial subproblem size and much smaller than those produced by DCM2. (Rec-DCM3 in this series of tests was set up to recurse until each subproblem was of size at most one eighth of the original size.)



(a) max. relative subproblem size (b) mean relative subproblem size (c) number of subproblems

Fig. 1.4 Comparison of DCM2, DCM3 and Recursive-DCM3 decompositions. DCM2 decompositions on datasets 5–10 could not be computed due to memory limitations.

1.4.1.4 Subtree construction and assembly Once the dataset is decomposed into overlapping subsets $A_1, A_2, \ldots, A_m$ (for us, $m \leq 4$ is typical), subtrees are constructed for each subset, A_i, using the chosen "base method," and then combined using the Strict Consensus Merger [Huson et al., 1999a, Huson et al., 1999b] to produce a tree on the combined dataset. The proof that the resulting tree is accurate (i.e., agrees, with high probability and in the limit, with the unknown underlying "true tree") follows from the following structural theorem (we omit the proof which is along the same lines as in [Huson et al., 1999a]).

Theorem 5. *Let T be the true tree and let $A_1, A_2, \ldots, A_m$ be the subproblems obtained in some DCM3 decomposition. If every short quartet in T is a four-clique in some A_i and if the base method applied to A_i returns the true tree for that subset (i.e., $T_i = T|A_i$), then the strict consensus merger of trees $T_1, T_2, \ldots, T_k$ yields the true tree T.*

Our *Rec-I-DCM3* algorithm takes as input the set $S = \{s_1, \ldots, s_n\}$ of n aligned biomolecular sequences, the chosen base method, and a starting tree T. In our experiments, we have used TNT (with default settings) as our base method, since it is the hardest to improve (in comparison, the PAUP* implementation of the parsimony ratchet [Bininda-Emonds, 2003] is easier to improve). Our algorithm produces smaller subproblems by recursively applying the centroid-edge decomposition until each subproblem is of size at most k. The subtrees are then computed, merged, and resolved (from the bottom-up, using random resolution) to obtain a binary tree on the full dataset. These steps are repeated for a specified number of iterations.

1.4.2 Experimental design

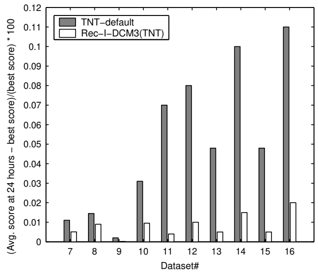
Having designed a new algorithm and having preliminary evidence (on small simulations) that it outperforms existing techniques, we now need to evaluate its performance in practice. The experimental evaluation of algorithms for phylogenetic reconstruction is a difficult endeavor (see [Moret, 2002, Moret and Warnow, 2002] for details). Because credible simulations of evolution remain lacking at the scale of 10,000 or more taxa, we chose to use biological datasets in our study. This choice ensures biological relevance of our results, but it prevents us from evaluating the accuracy of reconstructed trees, since the "true" tree is not available. However, other work from our group [Williams et al., 2004] tells us that we need to achieve excellent approximation of the parsimony score (tree length) in order to have any chance at reconstructing the true topology. Thus, we focused our testing on the quality of approximation in terms of the parsimony score.

Parameters and measurements: We chose to test performance during the first 24 hours of computation on each dataset for each method, taking hourly "snapshots" along the way in order to evaluate the progress of each method. We asked the following two questions: (i) how much of an improvement is gained by using *Rec-I-DCM3* versus TNT, if any? and (ii) how long does the best TNT trial (out of five runs) take to attain the average MP score obtained at 24 hours by *Rec-I-DCM3*? To answer these questions, we ran TNT and *Rec-I-DCM3(TNT)*, which uses TNT as its base method, on our ten biological datasets, using five independent runs, all on the same platform, with computed variances for all measurements.

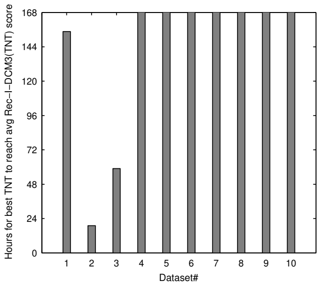
Implementation and platform: Our DCM implementations are a combination of LEDA, C++, and Perl scripts. The TNT Linux executable was obtained from Pablo Goloboff, one of the authors of TNT. We ran our experiments on three sets of processors, all running Linux: the `Phylofarm` cluster of 9 dual 500MHz Pentium III processors; a part of the 132-processor `SCOUT` cluster, consisting of 16 dual 733MHz Pentium III processors, and the `Phylocluster` of 24 dual 1.5GHz AMD Athlon processors, all at the University of Texas at Austin. For each dataset all the methods were executed on the same cluster; larger datasets were run on the faster machines.

1.4.3 Results

We defined the "optimal" MP score on each dataset to be the best score found over all five runs among all methods in the 24-hour period we allowed; on our datasets, this optimal score was always obtained by *Rec-I-DCM3(TNT)*. On each dataset and for each method, we computed the average MP score at hourly intervals and reported this value as a percentage of deviation from optimality. In our experiments, on every dataset and at every point in time (within these 24 hours), the best performance was obtained by *Rec-I-DCM3(TNT)*. Since only error rates less than 0.01% are tolerable, *Rec-I-DCM3*'s performance is very impressive; all trees are at least 99.99% correct.



(a) deviation from optimal



(b) TNT time to optimal

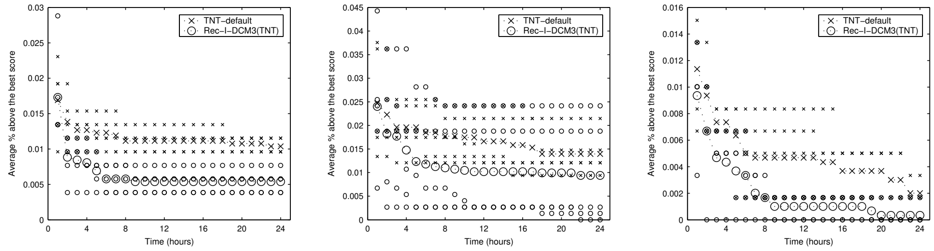
Fig. 1.5 Part (a) shows the average deviation above optimal after 24 hours by TNT and *Rec-I-DCM3(TNT)*; Part (b) shows the time taken by the single best TNT trial, extended to run for up to a week, to match the average *Rec-I-DCM3(TNT)* score at 24 hours—bars that reach the top indicate that TNT could not reach a match after a week of computation.

TNT, on the other hand, failed to reach this level of accuracy consistently—especially on datasets with more than 4,500 sequences.

Figure 1.5(a) shows the performance of *Rec-I-DCM3(TNT)* and of TNT at 24 hours. As the size of the dataset increases, the relative error in MP scores increases, but at a much faster rate for TNT than for *Rec-I-DCM3(TNT)*, so that the accuracy gap between the two increases quite rapidly. Figure 1.5(b) indicates how long it took TNT, in the best of five runs, to match the average scores obtained by *Rec-I-DCM3(TNT)* after 24 hours—we stopped the clock after one week of computation if the TNT run had not achieved a match by then, something that happened on the seven largest datasets. (The standard deviations of the MP scores at 24 hours for all the methods on all the datasets were very low, at most 0.035%.)

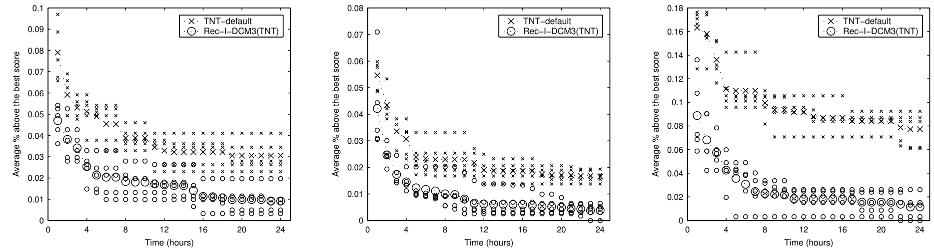
Figure 1.6 compares the time-dependent behaviors of TNT and *Rec-I-DCM3(TNT)* on our three smallest datasets (1, 2, and 3), while Figure 1.7 shows the same for three medium datasets (4, 5, and 6). and Figure 1.8 shows the same for our three largest datasets (8, 9, and 10). (It should be noted that the 24-hour time limit was perhaps overly limiting for the largest dataset: a quick look at the curves appears to indicate that even *Rec-I-DCM3(TNT)* has not yet reached a plateau at that point.) The improvement achieved by boosting TNT with *Rec-I-DCM3* is significant on all datasets as well as at all time intervals. In particular, note that the boosted version of TNT shows much stronger decreases in MP scores in the first several hours than the unboosted version.

Figure 1.9 shows how the error rate (deviation above the optimal MP score) of *Rec-I-DCM3(TNT)* decreases with computation time on each of the ten datasets. While the initial trees computed for the large datasets tend to exhibit large error (as large as 0.35%), the error drops very rapidly—even more rapidly for the large datasets than for the smaller ones. Thus, not only do the error rates of *Rec-I-DCM3(TNT)* fall more rapidly than those of TNT alone, but they have a positive second derivative: the larger they are, the faster they fall.



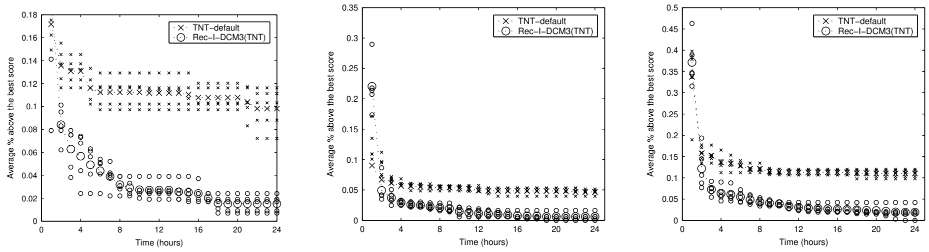
(a) Dataset #1 *(b) Dataset #2* *(c) Dataset #3*

Fig. 1.6 Average MP scores of TNT and *Rec-I-DCM3(TNT)* on datasets 1, 2, and 3, given as the percentage above the optimal score. Note: the vertical range varies across the datasets.



(a) Dataset #4 (b) Dataset #5 (c) Dataset #6

Fig. 1.7 Average MP scores of TNT and *Rec-I-DCM3(TNT)* on datasets 4, 5, and 6, given as the percentage above the optimal score. Note: the vertical range varies across the datasets.



(a) Dataset #8 (b) Dataset #9 (c) Dataset #10

Fig. 1.8 Average MP scores of TNT and *Rec-I-DCM3(TNT)* on datasets 8, 9, and 10, given as the percentage above the optimal score. Note: the vertical range varies across the datasets.

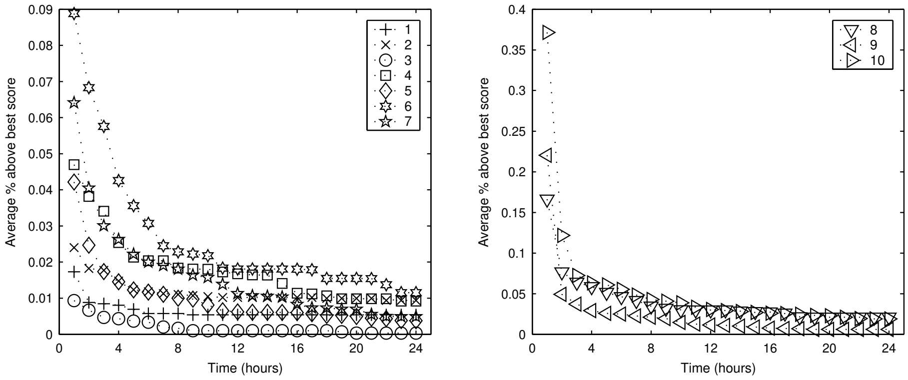


Fig. 1.9 Decrease in error rates with time on all datasets for *Rec-I-DCM3(TNT)*

1.4.4 Related Work and Discussion

Because of the importance of MP analyses in phylogeny reconstruction, systematists and algorithms researchers in phylogeny have studied the existing methods (specifically, implementations of heuristics in different software packages) to see which performed the best. The main criterion by which these methods have been studied is the time needed to get to the optimal score (or, more accurately, the best known score) on various real datasets, preferably of at least one hundred sequences. These studies [Goloboff, 1999, Nixon, 1999, Quicke et al., 2001, Roshan et al., 2004a, Soltis et al., 2000] have suggested that the "parsimony ratchet" [Nixon, 1999] was more effective than Tree-Bisection and Reconnection (TBR) hill-climbing [Maddison, 1991], and that TNT (which uses the parsimony ratchet) was more efficient than PAUP*'s implementation of the ratchet.

Our previous studies of our DCM variants included a study of the performance of boosted PAUP* MP heuristics using DCM2, which showed good results [Huson et al., 1999b]. However, our own analyses of phylogenetic reconstruction methods showed convincingly that the default PAUP* heuristic was not as powerful as the default TNT heuristics; they also showed that DCM2 failed to boost TNT. DCM3 was our attempt to remedy this situation, but it, too, failed to boost TNT reliably. The two key ideas were to use (i) iteration in order to refine the guide tree (since, obviously, the initial guide tree is perforce poor), and (ii) recursion, which had been used with spectacular success by Tang and Moret [Tang and Moret, 2003] in boosting the GRAPPA reconstruction software (for gene-order data) over three orders of magnitude.

Looking over our experience with both *Rec-I-DCM3* and DCM-GRAPPA, we can make reasonable inferences as to the reasons for the success of both. Recursion is necessary to bridge the gap between datasets that can be analyzed today by the base method (2,000–3,000 sequences for TNT, 14–15 genomes for GRAPPA) and large datasets. DCM-GRAPPA bridged three orders of magnitude in simulation studies (from 14 to 1,200 taxa); our *Rec-I-DCM3* bridges a least one order, but may be capable of more—biological datasets with 100,000 aligned sequences do not yet exist! But speed is only one component of the equation: the other is accuracy. In the case of DCM-GRAPPA, which uses no iteration and a simple DCM1 decomposition, the accuracy derives in good part from the excellent properties of gene-order data in phylogenetic reconstruction—and only simulations were used. In our case, recursion alone (a version we could call *Rec-DCM3*, which we tested), does not suffice: we need the ability, through iteration, to modify the guide tree at every level of the recursion. The combination of the two has synergistic effects: the decomposition is dictated by the guide tree, and thus by the successive iterations, and localizes the work for the recursive stages and the base method; and recursion enables the same process to take place at finer and finer levels of resolution, while communicating the results back to the coarser levels. Our results suggest that DCM-GRAPPA could be significantly enhanced by using a version of *Rec-I-DCM3*, but also that a more discriminating use of the guide tree, yielding more than just four subsets (e.g., by using a type of DCM1 decomposition guided by the guide tree), would probably work better for us, especially when tackling truly large datasets (on the order of millions of sequences) where several orders of magnitude will have to be bridged.

1.5 TREE ACCURACY

Several interesting and striking facts are uncovered from our study. First, topological accuracy correlates significantly with MP scores, but only within a certain range. When MP scores are within 1% of optimal the inferred trees tend to have poor topological accuracy. However, for MP scores within 0.01% of the optimal score, negligible improvement in topological accuracy is gained by better scores. Secondly, our experiments demonstrate that the “parsimony ratchet” [Nixon, 1999] can find trees within 0.01% of optimal an order of magnitude faster than searching for the optimal tree. Moreover, these near-optimal solutions have topologies that are very good approximations of the optimal solution. Since we cannot know the optimal score in advance, we consider a candidate stopping rule for terminating a search early. Our stopping rule allows us to find reasonable trees in a fraction of the time; a remarkably positive consequence for large-scale phylogenetic analysis. Thus, the development of stopping criteria for reconstructing phylogenies deserves serious consideration as an optimization technique for finding good solutions quickly.

Performance studies evaluating the performance of MP heuristics have generally focused on two issues: speed and topological accuracy. Studies that explore speed have examined how quickly each heuristic can solve MP (or reach the current best known score) for specific real biological datasets (see [Goloboff, 1999, Roshan et al., 2004a] for examples of such studies). However, the relative scarcity of large datasets (containing more than 200 or so sequences) has made it hard to investigate the performance of heuristics on datasets for which MP might be quite hard to solve.

Assessing the topological accuracy of an inferred tree on a real dataset is difficult because the true tree cannot be known precisely. As a result, estimations of topological accuracy have required simulations. Here, a DNA sequence is evolved down a known model tree, producing a set of sequences at the leaves of the model tree; the phylogeny reconstruction method is then applied to the set of sequences, and a tree is returned. Comparisons between the model and inferred tree provide a precise measure of topological accuracy. Simulation studies have been highly influential, and have suggested that good MP heuristics can produce reasonable estimates of trees, with respect to having acceptably low RF error rates (bounded by 10% or so) with respect to the true tree (see [Hillis, 1996]). On the other hand, we (along with other phylogenetics researchers) have noted that MP seems easier to solve on simulated data than on real data. We conjecture that the models used to represent sequence evolution are too "well-behaved" resulting in data that are not sufficiently realistic.

Most studies attempt to evaluate the accuracy of a given phylogenetic reconstruction with respect to the true, unknown tree, which is typically limited to simulation studies. Instead, we answer a different, but highly related question. Here, we wish to determine the topological accuracy of near-optimal trees with respect to the optimal solution. Since MP is NP-hard, we cannot know the true optima without resorting to exhaustive techniques, which is not possible on dataset sizes of interest to us (≥ 100 sequences). However, by establishing a set of "optimal" trees (i.e., trees with the best known score for a particular dataset), topological accuracy can be examined in two ways: (i) by measuring the topological distance between the majority consensus tree obtained for the set of suboptimal trees and the majority consensus tree for the set of optimal trees; and (ii) by measuring the topological distance between a random suboptimal tree and a random optimal tree. Both of the above measures provide us with a mechanism for characterizing the error rate in an estimation of the evolutionary history—especially in the context of terminating an MP analysis before it reaches the optimal solution.

1.5.1 Results

As stated earlier, the objective of our experimental analysis is to investigate the relationship of MP scores and tree topologies. The following questions are of particular interest to us.

1. Does topological accuracy correlate with improved parsimony scores? If so, for what range of parsimony scores does it correlate well?
2. What are the ramifications to phylogeny reconstruction of stopping a maximum parsimony search early?
3. How can we tell when it is safe to stop?

We consider each of the above questions, in turn, in the subsections below.

1.5.1.1 Topological Accuracy and MP Scores We examined the relationship between topological error (with respect to estimating the maximum parsimony trees) and the error in the MP analysis, using two criteria. The first criterion examines the RF distance between the two majority consensus trees (one of the "optimal" trees, and the other of the suboptimal trees), and the second criterion examines the average RF distance between a random tree having the optimal score, and a random tree having a score that is i steps worse, for small values of i.

1.5.1.2 Comparison between majority consensus trees Figure 1.10 plots the relationship between the topological distance between the two majority consensus trees, and the MP scores obtained, as a function of the number of iterations on each dataset.

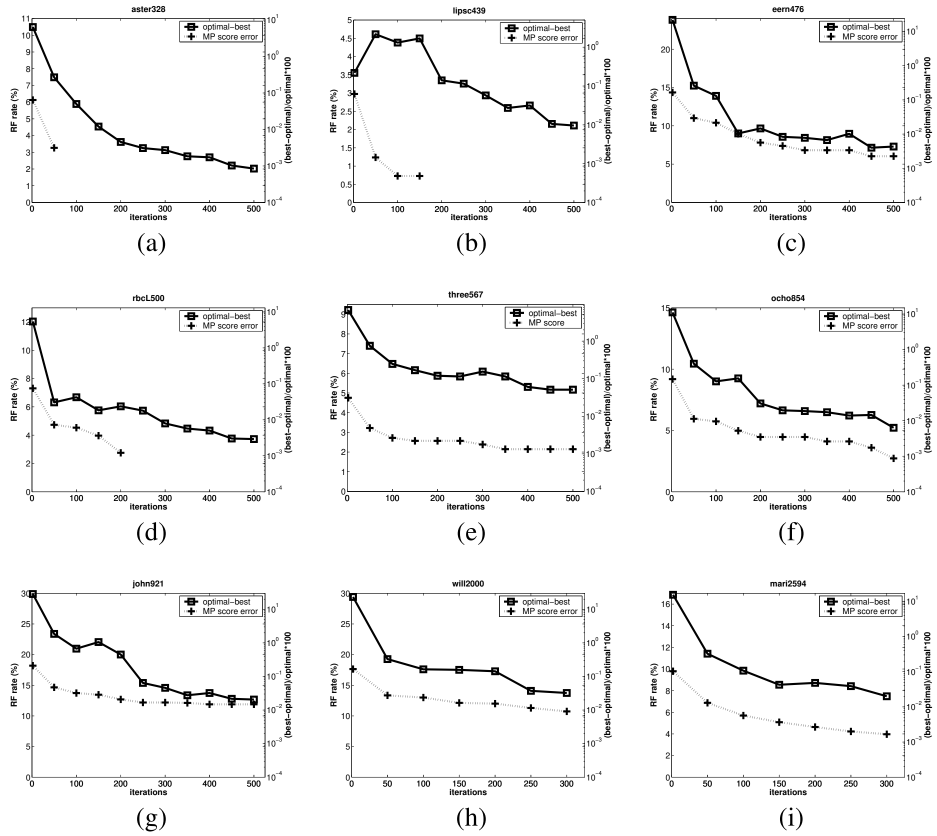


Fig. 1.10 Plots (a)–(i) show the behavior of RF rates and MP scores on the aster328, lipsc439, eern476, rbcL500, three567, ocho854, john921, will2000, and mari2594 datasets with average runtime requirements of 1.80, 33.58, 5.75, 2.99, 11.96, 31.82, 30.06, 307.17, and 164.41 hours, respectively. Here, the left and right y-axis correspond to the optimal-best and MP-score error curves, respectively. The optimal-best curve plots the RF rate between the majority tree constructed from optimally-scoring trees and the majority tree constructed from the best-scoring trees found by the end of iteration i. The MP score error curve is the percentage the best score is above the optimal score. Since this curve is plotted on a log scale, an MP score error of 0% is undefined.

RF distances below 5% are desirable but distances below 10% are relatively acceptable, hence we are interested in solving parsimony so that the resultant majority trees are within 10% of each other. In this figure, the initial majority trees can be further than 10% from the majority tree of the set of optimal trees, but the RF distance drops fairly quickly to below 10%; john921 and will2000 are the two notable exceptions (see Figures 1.10(g) and 1.10(h)). Although the MP score error decreases monotonically, the RF distance may not decrease monotonically. However, the general decrease in RF distance is consistent with the MP score error curve.

Additionally, note that on these datasets, whenever the RF distance is above 10%, the MP error is also above .01% (i.e., one hundredth of a percent), and that almost all the time if the MP error is below .01% the RF error will be below 10%. This suggests that a sufficient condition (for these datasets) for acceptable ($\leq$10% RF distance) topological accuracy is to have the MP analysis be within one hundredth of a percentage of optimal.

However, the above trends are not always present on the three largest datasets (see Figures 1.10(g), 1.10(h), and 1.10(i)). john921 and will2000 never reach an RF error of 10% even though their MP score error is near 0.01%. Moreover, when the mari2594 achieves 0.01% MP score error at 50 iterations, the corresponding RF rate is above 10%. For this dataset, the tolerable MP score error appears to be closer to 0.001% to achieve a good approximation of the optimal solution. Since larger datasets potentially have more optimal solutions and less resolved majority trees, trends seen for small datasets may only hold for datasets with features comparable to these datasets. Therefore, the relevant topological error rates will be greater on larger datasets, for a correspondingly accurate MP analysis.

1.5.1.3 Proximity of optimal and near-optimal trees Our other criterion measures the topological distance between two randomly selected trees, one which has an optimal score, and the other which has a score which is i steps worse, but only for small values of i. This study evaluates what additional information might be kept if we did not only return a single consensus tree.

Table 1.1 shows the average RF distance between optimally-scoring trees (OPT_0) and trees i steps from optimal (OPT_i). Since the number of trees in OPT_i is large, we randomly select a tree t_0 from OPT_0 and a tree t_i from OPT_i and compute the RF rate between them. The average RF rate is computed by repeating the above

dataset	OPT_0	OPT_1	OPT_2	OPT_3	OPT_4
aster328	8.94	9.82	10.68	11.16	11.72
lipsc459	4.69	5.44	6.08	6.53	7.14
eern476	6.45	10.33	12.44	13.88	14.40
rbcL500	8.61	9.40	10.19	11.04	11.72
three567	3.49	6.86	7.44	8.23	8.68
ocho854	8.82	9.90	10.46	11.10	11.59
john921	0.00	4.48	7.41	12.59	10.64
will2000	0.00	5.56	N/A†	9.32	15.41
mari2594	10.44	11.92	11.51	11.67	11.69

Table 1.1 Average RF distance between optimal trees and phylogenetic trees i steps from optimal (OPT_i), where $i = 0, 1, 2, 3, 4$. † denotes no OPT_2 trees were found for the will2000 dataset.

step $4 \cdot |\mathrm{OPT}_i|$ times. Notice that the RF distance between two random optimal trees ranges from 0% (only one optimal tree found for the john921 and will2000 datasets) to almost 11% (the mari2594 dataset). Thus, the set of optimal trees is itself not a highly specific estimator of the unknown true topology.

Note the following trends – typically (but not always) the RF distance increases as the parsimony score worsens, but this value does not grow very quickly. Thus, the RF distance between a random suboptimal tree and a random optimal tree is larger than the RF distance between two random optimal trees, but not by a substantial amount. For example, two randomly selected optimal trees for the lipsc459 dataset will differ by 4.69%, but if we increase the parsimony score by 4 steps, a randomly selected tree will differ from a randomly selected optimal tree by only 7.14%. Similar trends can be seen for the other datasets. This suggests that near-optimal solutions to MP—provided they are not too suboptimal—can yield estimates of the MP trees that are not considerably different.

Maximum parsimony is the most popular optimization criterion for analyzing large datasets. However, such datasets require an enormous amount of computational effort to solve optimally. Given that the actual objective of MP heuristics is to obtain a reasonable estimation of the underlying evolutionary history, we investigate whether it is necessary to solve MP exactly. Our results demonstrate that once an MP search is within 0.01% of the optimal score, little topological accuracy is gained by continuing the search further. Although the search will terminate at trees with suboptimal parsimony scores, our study demonstrates that topological differences between optimal and near-optimal trees are not significantly different. Hence, the analysis of large-scale datasets will benefit from new MP heuristics designed to quickly locate near-optimal solutions.

In the absence of knowing the optimal solution, our results provide strong evidence for the study of stopping criteria for MP heuristics. Currently, it seems that what is considered to be a "appropriate" or "sufficient" running time for an MP search depends mostly upon a researcher's patience: some systematists are willing to wait weeks or months for an MP search to terminate (even though termination does not ensure proximity to the optimal solution). However, our results demonstrate that such search times may be unnecessary. For most of the datasets analyzed here, we were able to shave 50% off of the execution time required to find an optimal tree, without a significant loss in the quality of the outcome with respect to the topological estimation. For such searches, the resulting majority tree is a good approximation of the set of optimal trees found using a more thorough search. Although the stopping rule described in this paper may only apply to our nine datasets, the implications are clear: very good estimates of the set of optimal MP trees can be obtained without doing a particularly thorough search. Stopping rules that work well will enable highly accurate estimations of the optimal trees of extremely large datasets ($\geq 100,000$). Thus, heuristics that utilize stopping criteria provide a fruitful approach to reconstructing the "Tree of Life"—the "holy grail" of phylogenetics.

1.6 SUMMARY AND OPEN PROBLEMS

REFERENCES

Bader et al., 2001. Bader, D., Moret, B. M., and Vawter, L. (2001). Industrial applications of high-performance computing for phylogeny reconstruction. In Siegel, H., editor, *Proceedings of SPIE Commercial Applications for High-Performance Computing*, volume 4528, pages 159–168, Denver, CO. SPIE.

Bininda-Emonds, 2003. Bininda-Emonds, O. (2003). Ratchet implementation in PAUP*4.0b10. available from http://www.tierzucht.tum.de:8080/WWW/Homepages/Bininda-Emonds.

Bruno et al., 2000. Bruno, W., Socci, N., and Halpern, A. (2000). Weighted neighbor joining: A likelihood-based approach to distance-based phylogeny reconstruction. *Mol. Biol. Evol.*, 17(1):189–197.

Cracraft, 2002. Cracraft, J. (2002). The seven great questions of systematic biology: An essential foundation for conservation and the sustainable use of biodiversity. *Ann. Missouri Bot. Gard.*, 89:127–144.

Felsenstein, . Felsenstein, J. Phylogenetic inference package (phylip), version 3.5. University of Washington, Seattle.

Fitch, 1977. Fitch, W. (1977). On the problem of discovering the most parsimonious tree. *The American Naturalist*, 111(978):223–257.

Fitch, 1971. Fitch, W. M. (1971). Toward defining the course of evolution: minimal change for a specific tree topology. *Syst. Zool.*, 20:406–416.

Foulds and Graham, 1982. Foulds, L. R. and Graham, R. L. (1982). The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49.

Gascuel, 1997. Gascuel, O. (1997). BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, 14:685–695.

Gladstein, 1997. Gladstein, D. S. (1997). Efficient incremental character optimization. *Cladistics*, 13:21–26.

Goloboff, 1999. Goloboff, P. (1999). Analyzing large data sets in reasonable times: solution for composite optima. *Cladistics*, 15:415–428.

Goloboff, 1993. Goloboff, P. A. (1993). Character optimization and calculation of tree lengths. *Cladistics*, 9:433–436.

Golumbic, 1980. Golumbic, M. (1980). *Algorithmic graph theory and perfect graphs*. Academic Press, Inc.

Guindon and Gascuel, 2003. Guindon, S. and Gascuel, O. (2003). A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.*, 52(5):696–704.

Hendy and Penny, 1982. Hendy, M. and Penny, D. (1982). Branch and bound algorithms to determine minimal evolutionary trees. *Math. Biosci.*, 59:277–290.

Hillis, 1996. Hillis, D. (1996). Inferring complex phylogenies. *Nature*, 383:130–131.

Huelsenbeck and Ronquist, 2001. Huelsenbeck, J. P. and Ronquist, F. (2001). MR-BAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755.

Huson et al., 1999a. Huson, D., Nettles, S., and Warnow, T. (1999a). Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *Journal of Computational Biology*, 6:369–386.

Huson et al., 1999b. Huson, D., Vawter, L., and Warnow, T. (1999b). Solving large scale phylogenetic problems using DCM2. In *Proc. 7th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 118–129. AAAI Press.

Knuth, 1973. Knuth, D. (1973). *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley Publishing Company, Reading, MA.

Maddison, 1991. Maddison, D. (1991). The discovery and importance of multiple island of most parsimonous trees. *Syst. Bio.*, 42(2):200–210.

Maddison and Maddison, . Maddison, W. and Maddison, D. Mesquite: a modular system for evolutionary analyses, version 0.98. mesquiteproject.org.

Moret, 2002. Moret, B. (2002). Towards a discipline of experimental algorithmics. In Goldwasser, M., Johnson, D., and McGeoch, C., editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, volume 59 of *DIMACS Monographs*. American Mathematical Society.

Moret and Warnow, 2002. Moret, B. and Warnow, T. (2002). Reconstructing optimal phylogenetic trees: A challenge in experimental algorithmics. In Fleischer, R., Moret, B., and Schmidt, E., editors, *Experimental Algorithmics*, volume 2547 of *Lecture Notes in Computer Science*, pages 163–180. Springer-Verlag.

Nakhleh et al., 2001. Nakhleh, L., Roshan, U., St. John, K., Sun, J., and Warnow, T. (2001). Designing fast converging phylogenetic methods. In *Proc. 9th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB'01)*, volume 17 of *Bioinformatics*, pages S190–S198. Oxford U. Press.

Nixon, 1999. Nixon, K. C. (1999). The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:407–414.

Olsen et al., 1994. Olsen, G., Matsuda, H., Hagstrom, R., and Overbeek, R. (1994). fastdnaml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *Comput. Appl. Biosci.*, 10:41–48.

Purdom et al., 2000. Purdom, P. J., Bradford, P., Tamura, K., and Kumar, S. (2000). Single column discrepancy and dynamic max-mini optimization for quickly finding the most parsimonious evolutionary trees. *Bioinfomatics*, 2(16):140–151.

Quicke et al., 2001. Quicke, D. L., Taylor, J., and Purvis, A. (2001). Changing the landscape: A new strategy for estimating large phylogenies. *Syst. Biol.*, 50(1):60–66.

Roshan et al., 2004a. Roshan, U., Moret, B. M., Warnow, T., and Williams, T. L. (2004a). Performance of supertree methods on various dataset decompositions. In Bininda-Emonds, O., editor, *Phylogenetic Supertrees: Combining information to reveal the Tree of Life,*, pages 301–328. Kluwer Acad. Publ., Dordrecht.

Roshan et al., 2004b. Roshan, U., Moret, B. M. E., Williams, T. L., and Warnow, T. (2004b). Rec-I-DCM3: a fast algorithmic technique for reconstructing large phylogenetic trees. In *Proc. IEEE Computer Society Bioinformatics Conference (CSB 2004)*, pages 98–109. IEEE Press.

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

Soltis et al., 2000. Soltis, D. E., Soltis, P. S., Chase, M. W., Mort, M. E., Albach, D. C., Zanis, M., Savolainen, V., Hahn, W. H., Hoot, S. B., Fay, M. F., Axtell, M., Swensen, S. M., Prince, L. M., Kress, W. J., Nixon, K. C., and Farris, J. S. (2000). Angiosperm phylogeny inferred from 18s rDNA, *rbcL*, and *atpB* sequences. *Botanical Journal of the Linnean Society*, 133:381–461.

Steel, 1994. Steel, M. (1994). The maximum likelihood point for a phylogenetic tree is not unique. *Systematic Biology*, 43(4):560–564.

Swofford, 2002. Swofford, D. L. (2002). PAUP*: Phylogenetic analysis using parsimony (and other methods). Sinauer Associates, Underland, Massachusetts, Version 4.0.

Tang and Moret, 2003. Tang, J. and Moret, B. (2003). Scaling up accurate phylogenetic reconstruction from gene-order data. In *Proc. 11th Int'l Conf. on Intelligent Systems for Molecular Biology ISMB'03*, volume 19 (Suppl. 1) of *Bioinformatics*, pages i305–i312.

Warnow et al., 2001. Warnow, T., Moret, B., and St. John, K. (2001). Absolute convergence: True trees from short sequences. In *Proc. 12th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA'01)*, pages 186–195. SIAM Press.

Williams et al., 2004. Williams, T., T. Berger-Wolf, B. M., Roshan, U., and Warnow, T. (2004). The relationship between maximum parsimony scores and phylogenetic tree topologies. Technical Report TR-CS-2004-04, Department of Computer Science, The University of New Mexico.

Yates et al., 2004. Yates, T. L., Salazar-Bravo, J., and Dragoo, J. W. (2004). *Assembling the Tree of Life*, chapter The Importance of the Tree of Life to Society. Oxford University Press.