Performance of Flip Supertree Construction with a Heuristic Algorithm

OLIVER EULENSTEIN,1 DUHONG CHEN,1 J. GORDON BURLEIGH,1,2 DAVID FERNÁNDEZ-BACA,1 AND MICHAEL J. SANDERSON2

1Department of Computer Science, Iowa State University, Ames, Iowa 50011, USA; E-mail: oeu@cs.iastate.edu (O.E.)
2Section of Evolution and Ecology, University of California, Davis, California 95616, USA

Abstract.—Supertree methods are used to assemble separate phylogenetic trees with shared taxa into larger trees (supertrees) in an effort to construct more comprehensive phylogenetic hypotheses. In spite of much recent interest in supertrees, there are still few methods for supertree construction. The flip supertree problem is an error correction approach that seeks to find a minimum number of changes (flips) to the matrix representation of the set of input trees to resolve their incomparabilities. A previous flip supertree algorithm was limited to finding exact solutions and was only feasible for small input trees. We developed a heuristic algorithm for the flip supertree problem suitable for much larger input trees. We used a series of 48- and 96-taxon simulations to compare supertrees constructed with the flip supertree heuristic algorithm with supertrees constructed using other approaches, including MinCut (MC), modified MC (MMC), and matrix representation with parsimony (MRP). Flip supertrees are generally far more accurate than supertrees constructed using MC or MMC algorithms and are at least as accurate as supertrees built with MRP. The flip supertree method is therefore a viable alternative to other supertree methods when the number of taxa is large. [Flipping; phylogeny; simulation study; supertree.]

A supertree is a phylogenetic tree built from a collection of input trees that share some but not necessarily all of their terminal taxa. The recent interest in supertrees among systematists (reviewed by Sanderson et al., 1998; Bininda-Emonds et al., 2002) is largely due to the potential use of these trees for reconstructing large phylogenies of major clades of the tree of life and synthesizing phylogenetic hypotheses derived from disparate types of data. Supertrees also have introduced a new set of computational problems that require algorithm development and research (Semple and Steel, 2003). Systematists have been constructing informal supertrees for many years (e.g., Donoghue, 1989; Martin and Clobert, 1996; Donoghue et al., 1998; Ortolani, 1999), but formal algorithms for solving computational problems associated with supertree construction are still in their infancy. The most widely used supertree method is matrix representation using parsimony (MRP) (e.g., Purvis, 1995a; Bininda-Emonds et al., 1999; Bininda-Emonds, 2000; Linder, 2000; Wojciechowski et al., 2000; Liu et al., 2001; Plunkett, 2001; Schwilk and Ackerly, 2001; Daubin et al., 2002; Gatesy et al., 2002; Kennedy and Page, 2002; Pisani et al., 2002; Salamin et al., 2002). Here, we discuss an alternative approach to MRP based on the same initial matrix representation of the input trees but incorporating a different optimality criterion associated with “flipping” elements of the matrix. We developed a heuristic algorithm for supertree construction in the context of this matrix representation with flipping (MRF) method and compared the performance of the MRF supertree algorithm with that of other supertree methods using simulations. A supertree is the solution to a supertree construction problem, or more succinctly a supertree problem. Informally, a supertree is constructed by a supertree algorithm that solves a supertree problem by mapping any given collection of input trees onto a collection of one or more output trees (supertrees). The usual objective of supertree problems is to preserve information from the input trees and extract novel statements of relationships not apparent in any single input tree. Two issues must be considered in all supertree problems: input trees often conflict, and the phylogenetic position of taxa unique to one tree has the potential to introduce ambiguity into the resulting supertree (Sanderson et al., 1998). All effective supertree methods must grapple with these two issues while clearly defining the properties that the supertree problems address (Steel et al., 2000). Currently, only a few supertree methods have been described (Bininda-Emonds et al., 2002), and efficient algorithms have been developed for even fewer.

The first supertree algorithm likely came from computer science (Aho et al., 1981) and was taken up by scientists interested in clustering and consensus algorithms (Steel, 1992), culminating in an efficient algorithm to identify the strict consensus of all supertrees compatible with a collection of congruent input trees (Steel, 1992). Semple and Steel (2000) later extended this work and created the MinCut (MC) algorithm to handle incongruent input trees. Whenever a conflict is encountered, the MC algorithm deletes a minimal amount of information from the input trees to allow the computation to proceed (Semple and Steel, 2000; see Page, 2002). Page (2002) modified the MC algorithm by modifying the criteria for the minimal amount of deleted information. Hereinafter, we refer to Page’s (2002) algorithm as the modified MC (MMC) algorithm.

Systematists also adapted earlier notions from cladistic biogeography to construct supertrees from a matrix representation of the input trees using parsimony methods (Baum, 1992; Doyle, 1992; Ragan, 1992; Purvis, 1995b). Input trees for the MRP problem are represented as a partial binary character matrix, where each character represents a clade from an input tree. Taxa in the clade are scored 1; taxa absent from the clade are scored 0, and taxa not sampled for the clade are scored “?.” The matrix representation from a collection of incompatible input trees does not correspond to a “perfect phylogeny” (e.g., Semple and...
Steel, 2003), i.e., a tree in which every character can be mapped with no homoplasy. MRP methods treat the 0’s and 1’s in the input trees as characters and seek a tree that requires the fewest number of steps based on this matrix representation. The MRP problem outputs the most-parsimonious trees based on the input matrices. Perhaps because of the widespread acceptance of the philosophy underlying parsimony approaches and the mechanics of parsimony analysis, the MRP problem is currently the only supertree problem widely used in systematic studies. Parsimony is relatively easy to justify in the original tree-building problem in which homoplasy represents additional assumptions of parallel evolutionary changes or reversals (e.g., Sober, 1988). 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 clades rather than between individual evolutionarily novel character states, such as a substitution in a DNA sequence. Thus, this type of homoplasy is somewhat removed from phylogenetic evidence.

The MRF problem departs from the objective of MRP to solve a variant of the supertree problem. The general strategy is to determine the minimum number of flips between 1 and 0 that are required to turn the matrix representation of input trees into one that corresponds to a perfect phylogeny (a tree with no homoplasy). This is different from determining what tree, given the matrix, has the least homoplasy (i.e., MRP). The minimum flip problem for trees (Chen et al., 2003) is to find a set of compatible characters by performing a minimum number of flip operations on the original matrix representation. A flip supertree is a phylogenetic tree that corresponds to this perfect matrix. Thus, the minimum number of flips separating these two matrices is a measure of the distance between them based on our notion of error correction. The minimum flip problem therefore seeks the supertree closest to the set of input trees in this sense. The motivation for minimizing the number of flips separating input trees from a supertree is that mistaken assignments of 1’s and 0’s in a matrix representation of trees can be viewed as noise, and the process of assessing how many changes must be made to generate a phylogeny in the presence of this noise is akin to noise reduction. An incorrect cell in the matrix representation is exactly equivalent to an incorrect statement about membership of a single terminal taxon in some clade. There are obviously other kinds of phylogenetic mistakes. For example, sets of terminals might be counted as a single error, but a mistaken single terminal taxon represents an irreducible item of error.

Comparisons between proposed solutions to supertree problems must evaluate their performance in terms of time efficiency and effectiveness. The time efficiency of supertree problems makes an important statement about the size of supertrees that can be constructed in reasonable time. The MRP problem is a parsimony optimization problem that is intrinsically computationally difficult even when the input trees have identical sets of taxa (Day et al., 1986). Therefore, in all but the small-est problems heuristic algorithms are used in MRP to find answers that are hopefully close to exact solutions to the MRP problem. However, even the heuristics can be quite slow when the input tree matrix representations are large or numerous question marks force them to examine many equally optimal trees. The MC and MMC algorithms run in polynomial time (Semple and Steel, 2000; Page, 2002). Theoretical effectiveness of a supertree problem can be judged by various desirable properties. Steel et al. (2000) discussed elementary properties that are desirable for supertree problems and showed that it is not possible to satisfy all simultaneously. In addition, MRP, MRF, and MC supertrees preserve certain consensus properties of their input trees when the input trees include the same taxa. MRF, MRP, and MC supertrees contain the strict and semistrict consensus clusters but not necessarily the Adams or majority consensus clusters of their input trees (Semple and Steel, 2000; Bryant, 2003; Chen et al., 2003). This is a desirable property in the sense that consensus properties represent a limiting case for supertree methods when the taxon sets are the same.

Chen et al. (2003) described an exact branch-and-bound algorithm for solving the MRF problem for small input trees. Here, we present a heuristic algorithm for the MRF supertree problem that can build supertrees with large numbers of taxa. We used a series of simulations to compare the accuracy of supertrees built with our MRF algorithm with those constructed by MRP, MC, and MMC algorithms.

**The MRF Problem**

A flip in a binary character matrix consists of changing a 1 entry into a 0 entry, or vice versa. We measure the distance between two matrices by the flip distance that is defined canonically based on distances between columns of the matrices. This distance measures the minimal number of flips needed to transform one matrix into a subset of the columns of another matrix. The various notions of flip distance we need are defined as follows. Let A and B be matrices over the same taxa set.

1. The flip distance \( d(a, b) \) for column \( a \) in \( A \) and column \( b \) in \( B \) is the minimum number of flips of 1’s and 0’s needed to convert \( a \) into \( b \). We do not count any position where \( a \) or \( b \) is a question mark (Fig. 1i).
2. The flip distance \( d(a, B) \) for column \( a \) in \( A \) and the matrix \( B \) is the minimum flip-distance from \( a \) to any column \( b \) in \( B \) (Fig. 1ii).
3. The flip distance \( d(A, B) \) is the sum of the flip distances \( d(a, B) \) over all columns \( a \) in \( A \) (Fig. 1iii). The flip distance between two matrices is not symmetric.
4. Let \( T \) be a tree of the taxon set of \( A \) and \( H \) be a matrix representation of \( T \), then the flip distance \( d(A, T) \) is defined as \( d(A, H) \) (Fig. 1iv).
5. The flip distance \( d(A) \) is defined as the minimal flip distance \( d(A, T) \) over all trees \( T \) over the taxon set of \( A \).
The example shown, these vector-to-vector distances are, from left to right, 4, 2, 5, and 4. (ii) The flip distance between a vector and a matrix is obtained from the distances between the vector and each of the columns in the matrix. In the example shown, these vector-to-vector distances are, from left to right, 4, 2, 5, and 4. The flip distance is the minimum of these values, 2. (iii) The flip distance between the two matrices shown is obtained from the distances between each column of the first matrix and the entire second matrix. In the example shown, these vector-to-matrix distances are, from left to right, 1, 0, and 1. The distance is the sum of these values, 2. (iv) The flip distance between the matrix and the rooted tree equals the flip distance between the matrix and a matrix representation of the tree. The figure shows one such matrix representation, for which the distance is 6. The same value is obtained regardless of the representation used.

The MRF problem is to find all binary trees $\mathcal{T}$, called MRF supertrees, such that $d(\mathcal{A}, \mathcal{T}) = d(\mathcal{A})$ for a given matrix $\mathcal{A}$. There may be nonbinary trees with the same minimum flip distance. However, none can have smaller flip distances, and for algorithmic reasons we prefer to restrict the problem definition to binary trees.

**Heuristic Algorithm for the MRF Problem**

There is strong evidence from computational complexity theory that the MRF problem cannot be solved efficiently (Chen et al., 2003). Thus, a heuristic algorithm is needed to reduce the computational time in larger data sets. The MRF heuristic uses a “hill-climbing” approach to search the set of possible rooted binary trees for an optimal MRF supertree. Hill climbing starts at an initial tree and follows paths of steepest descent. The heuristic MRF algorithm uses a hill-climbing approach similar to the one used by heuristic parsimony algorithms (Swofford et al., 1996). The MRF algorithm has two phases.

In phase I the algorithm builds a suboptimal MRF supertree (a seed tree) by stepwise addition of taxa to an initial tree containing three taxa until the tree contains all taxa. The initial tree is chosen by enumerating all three-taxon, rooted, binary trees to find one with the minimal flip distance to the input matrix constrained to the current three taxa. This approach resembles the dynamic addition sequence procedure “closest” (Swofford et al., 1996). New trees are built in subsequent steps by adding one of the remaining taxa to each of the branches in the current tree. The flip distances of the resulting trees are evaluated, and the tree with the minimal flip distance to the input matrix constrained to the current taxon set is chosen. If two trees have the same flip distance, the tie is broken arbitrarily.

In phase II the seed tree is optimized through a series of tree modifications. A tree modification can be applied to any node of a given tree. Beginning with the seed tree, a tree $\mathcal{T}$ is explored by examining each of its nodes for a possible tree modification in a random order. A node $v$ is examined by computing the flip distance of the tree $\mathcal{T}'$ to the input matrix $\mathcal{A}$, where $\mathcal{T}'$ is the tree $\mathcal{T}$ modified by the tree modification applied to node $v$. If this flip distance is smaller then the flip distance of $\mathcal{T}$ to $\mathcal{A}$, the tree exploration of $\mathcal{T}$ is terminated and tree $\mathcal{T}'$ is explored. Otherwise, the next node in the random order is examined. When all nodes have been examined, tree $\mathcal{T}$ is returned and the process is terminated.

The tree modifications considered in phase II are chosen from among nearest-neighbor interchange (NNI), subtree pruning and regrafting (SPR), and tree bisection and reconnection (TBR) modifications (Swofford et al., 1996; Steel, 2001). These modifications were originally developed for unrooted trees. The basic change in going from unrooted to rooted trees is that in unrooted trees modifications are applied to tree branches, whereas in rooted trees modifications are applied to tree nodes. Although our search heuristic can work with NNI, SPR, or TBR modifications, we found that SPR often performs best (data not shown).
Simulations

We conducted a series of simulation experiments to evaluate the performance of MRF, MRP, MC, and MMC algorithms. The general format of the experiments is described in Figure 2. An alignment of DNA sequences was simulated using a 48-taxon or 96-taxon model tree, and the alignment was partitioned into equal-size blocks of smaller data sets. Each of the smaller data sets was used to make an input tree, and the number of input tree data sets in both the 48- and 96-taxon simulations varied from 2 to 20 in increments of 2. We randomly deleted 25%, 50%, or 75% of the taxa from each of the input tree data sets and then constructed parsimony trees from each input tree data set. We performed 100 simulation replicates for all combinations of deletion frequencies and number of input trees. The input trees were always constructed from 1,000-base pair (bp) data sets, and the total length of the alignment was 1,000-bp times the number of input trees. The strict consensus of all most-parsimonious trees from each input tree data set was 1,000-bp times the number of input trees. The strict consensus of all most-parsimonious trees constructed from the input tree data set was used as the input tree for MRF, MRP, MC, or MMC supertree analyses. The accuracy of each of the resulting supertrees was determined based on its similarity to the original model tree.

Generating Model Trees and DNA Sequences

We generated a model tree according to a conditional Yule birth process with either 48 or 96 taxa using the default parameters of the YULE_C procedure from the program r8s (Sanderson, 2003). The conditional Yule birth process produces model trees with a fixed time between the root of the tree and the present, a fixed number of terminal taxa, and an age distribution of nodes that is the same on trees regardless of the number of taxa (Ross, 2000). We created a new tree for every simulation replicate. Sequences were generated along each model tree using the Monte Carlo simulation method implemented in Seq-Gen (Rambaut and Grassly, 1997). The simulations assumed a Kimura two-parameter (K2P) model of sequence evolution, which assumes equal nucleotide frequencies and a transition: transversion ratio of 2 (Kimura, 1980).

Creating the Input Trees

Each site in the total data set simulated by Seq-Gen is independent and identically distributed. Therefore, contiguous blocks of sequence represent randomly subdivided data sets. We divided the simulated data sets and deleted 25%, 50%, or 75% of randomly chosen taxa from each subdivided data set. We then found most-parsimonious trees from each data set in the partition using heuristic searches in PAUP* (Swofford, 2002) with TBR branch swapping. The strict consensus tree of all most-parsimonious trees was used as an input tree for supertree construction. All supertree algorithms in the simulation used the same input trees. Therefore, differences in the performance of each supertree algorithm are due to the algorithm itself and not to variation in the quality of input trees.

Supertree Construction

We generated the matrix representation of each input tree. Because the trees are simulated using a model tree, the root of each input tree is known. MRF supertrees were estimated by our MRF heuristic with SPR branch swapping. The MRF heuristic is implemented in the program Rainbow (http://genome.cs.iastate.edu/supertree/index.html). MRP supertrees were estimated using PAUP* Preliminary tests showed that TBR branch swapping resulted in most accurate MRP supertrees, and therefore, we used TBR branch swapping for the MRP heuristic. We also computed exact MC and MMC supertrees using our own implementation for the MC algorithm and Rod Page’s implementation for the MMC algorithm (http://darwin.zoology.gla.ac.uk/cgi-bin/supertree.pl).

Comparing Supertrees

We evaluated the accuracy of each supertree algorithm by comparing the supertree to the model tree using the maximum agreement subtree (MAST) score, the triplet score, and the Robinson–Foulds score (Robinson and Foulds, 1981). The MAST score counts the number of leaves of a maximum agreement subtree (Gordon, 1980; Kubicka et al., 1992) normalized by the number of leaves of the supertree (e.g., Chen et al., 2003) and was calculated using PAUP*. The triplet score (Page, 2002) was originally adapted from the quartet metric (Day, 1986). This score includes counts the number of rooted triplets that are identically resolved in the supertree and the model tree(s) the triplets resolved differently in both trees(d), and triplets resolved in the model tree but not in the supertree(r), and equals \(1 - \frac{(d + r)}{(d + r + s)}\). The MAST and triplet scores are asymmetric similarity measures, reflecting a directed comparison from the supertree to the model tree. Because of its wide acceptance, we also computed the Robinson–Foulds metric between the supertree and the model tree constrained to the leaves of the supertree using our own implementation. This distance counts the number of clusters that belong to only one of the two trees. The results of the Robinson–Foulds distance were generally consistent with the MAST and triplet scores. However, the Robinson–Foulds distance generally does not distinguish between the effectiveness of MRP and MRF supertrees and between MC and MMC supertrees; therefore, the results are not reported here.

To understand the random expectation of MAST and triplet scores, we generated a distribution of both scores by comparing 10,000 48- and 96-taxon trees that were randomly generated using PAUP* with another randomly generated 48- or 96-taxon binary tree. The results from the 48- and 96-taxon trees are similar; therefore, only the random distributions using 48-taxon trees are reported here. The mean random expectation of the MAST score for a 48-taxon tree is roughly 0.22, and the mean random triplet score is 0.33. Random trees rarely have MAST scores that exceed 0.3 or triplet scores that exceed 0.4.
FIGURE 2. A flow chart of the simulation experiments. Each experiment begins by generating a model tree that is used to simulate gene sequence alignments. The sequence alignments are used to construct a set of input trees that are used with the four different supertree methods, MRP, MRF, MMC, and MC. The resulting supertrees are then compared with the original model tree to test the accuracy of each supertree method.
RESULTS

In the simulations, the MRF supertrees were generally at least as accurate as any other kinds of supertrees we considered. In both the 48- and 96-taxon simulations, the average MAST and triplet scores of the MRF and MRP supertrees always exceeded those the MC or MMC supertrees (Figs. 3, 4). The average MAST and triplet scores of the MRF and MRP supertrees tend to increase with more input trees and are higher when the input trees contain more taxa. The MMC algorithm performs best relative to the MRF and MRP algorithm when the...
deletion probability is 25%, especially when the number of input trees is high. The average MAST and triplet scores from the MC supertrees are usually by far the lowest among the tested algorithms. MC supertrees resemble the model tree more closely than do MMC supertrees only in some cases when there are only two input trees. However, the average triplet and MAST scores of the MC supertrees always exceed the random expectation. The MC algorithm performs best with fewer input trees and fewer taxa deleted from data sets. The average MAST and triplet scores of the MRP and MRF supertrees are similar in all simulations. The average MAST and triplet scores of MRF supertrees slightly exceed those of MRP supertrees, especially as taxon deletion increases.
The effect of input tree number varies depending on the algorithm and level of taxon deletion. In the MRF and MRP algorithms, the triplet and MAST scores generally increase as the number of input trees increases, and the decrease is more evident as the taxon deletion increases (Figs. 3, 4). Thus, although MRF and MRP supertrees appear to be generally accurate whenever the input trees have at least 50% of their taxa, when the input trees have only 25% of the total taxa, increasing the number of input trees can greatly increase the accuracy of the supertree. The number of input trees has the greatest effect on the MC algorithm. This algorithm always performs best with only two input trees, and its performance drops off rapidly as the number of input trees increases. The effect of input tree number on the MMC algorithm depends on the taxon deletion probability. At 25% deletion, the average MAST and triplet scores from the MMC algorithm generally increase as the number of input trees increases, but at 50% and 75% deletion the scores generally are either stable or they decrease.

The average triplet and MAST scores are generally slightly lower in the 96-taxon simulations compared with the 48-taxon simulations (Figs. 3, 4). The size of the difference is greatest for the MC and MMC algorithm. However, the relative performance of each of the algorithms is similar in both the 48- and 96-taxon simulations.

**Discussion**

Although the MRF supertrees found by the heuristic are not necessarily optimal, they are more accurate than MC or MMC supertrees and at least as accurate as MRP supertrees under the conditions of our simulation. Previous simulation results demonstrated that the MRF supertrees built with our exact branch-and-bound algorithm had higher average MAST scores compared with the model tree than did MRP and MC supertrees (Chen et al., 2003). Although this result was encouraging, its generality is questionable because the algorithm is limited to small supertrees (~20 taxa). In the previous simulation experiment, Chen et al. (2003) also relied on a single measure, the MAST score, to compare the supertrees and the model trees. In this experiment, we used three different measures to compare supertrees with the model tree, demonstrating that the quality of MRF supertrees compared with other supertrees appears robust. The present results indicate that our MRF algorithm is a viable alternative to other supertree algorithms for building much larger supertrees.

The clearest result from the simulations is that both MRF and MRP algorithms always provide more accurate supertrees than do either the MC or the MMC algorithms (Figs. 3, 4). When the input trees disagree, the MRF algorithm deletes and inserts information based on a global objective, whereas the MC and MMC algorithms delete information based on a local objective (Semple and Steel, 2000; Page, 2002). The simulations indicate that the more directed approach of MRF to resolving incongruence may lead to supertrees that are more accurate. Most troubling for the MMC and MC algorithms is that they appear to work best relative to the MRP and MRF heuristics when the deletion probability is low, a situation in which supertree methods are less interesting. Also troubling is that MC and MMC supertrees do not always improve with more input trees, as MRF and MRP supertrees do. The relatively high MAST scores with two input trees and 75% deletion (Figs. 3b, 4b) may be an effect of the relatively small number of taxa in these trees. Thus, although MRF and MRP supertrees can be improved by adding more data, this is not necessarily so with MMC or MC supertrees. The MC algorithm performs best when there are only two input trees, and its effectiveness rapidly decreases with more input trees. The MMC algorithm appears to be a large improvement over the MC algorithm, as previously suggested by Page (2002). Although the MMC algorithm may not perform as well as the MRF or MRP heuristics, the large improvement of the MMC algorithm over the original MC algorithm is encouraging for further improvements.

The MRF method, which is based on error correction in the input trees, and the MRP method, which is based on finding the supertree with the minimal number of character changes with respect to the tree matrices, represent different philosophical approaches to supertree construction. In some cases, MRP and MRF supertrees may be similar, if not identical. For example, when the set of input trees contains no inconsistencies, MRF and MRP supertrees will be identical. Therefore, if the input trees are constructed with adequate data and taxon sampling, MRF and MRP supertrees will likely be similar. Our simulation results suggest that MRF supertrees are similar to MRP supertrees. Both MRF and MRP algorithms perform relatively well across all simulation conditions. The average triplet score from the MRF or MRP supertrees compared with the model tree is almost never less than 0.8 when the input trees have at least 50% of the total taxa.

MRP and MRF supertrees should be most different when the input trees conflict. Differences in the performance of MRF and MRP supertrees are more evident as the deletion probability increases and most obvious as the deletion probability reaches 75% (Figs. 3, 4). The promise of supertree algorithms is ultimately their ability to construct large phylogenetic trees from collections of much smaller trees; therefore, an effective supertree algorithm must perform well when the taxon sampling of the input trees and the taxon overlap between input trees is limited. In such cases, our simulations predict that an MRF supertree analysis would be more accurate than that using MRP, MC, or MMC supertrees.

A supertree method must be reasonably time efficient to be of practical use. In all simulations, MC, MMC, and MRP algorithms were faster than MRF algorithms. For example, it took approximately 5 hr to calculate a 96-taxon MRF supertree from 10 input trees with 50% taxon deletion, but the MC, MMC, and MRP algorithms took <1 min. Though the MRF algorithm is slower than the MRP algorithm, it is feasible to compute any 96-taxon supertree with the MRF algorithm, and we are working on increasing its computational efficiency. In our
simulations, the exact MC and MMC supertrees are far worse than the heuristically computed MRF and MRP supertrees. Thus, even though the MRP and MRF algorithms are not guaranteed to find an optimal supertree, they may be much more accurate than the MC or MMC algorithms for building large supertrees. Although it is desirable to find the solution to a supertree problem in polynomial time (Steel et al., 2000; Page, 2002), our primary goal is to find supertree algorithms that recover trees that most accurately reflect the true relationships of the taxa.

There is abundant room for the development of new accurate and tractable algorithms for inferring supertrees. Our heuristic makes the MRF problem a viable alternative to other supertree algorithms. Although our simulation results strongly suggest that MC and MMC supertrees should be interpreted cautiously, our MRF algorithm appears to perform as well as or better than the MRP algorithm under a variety of conditions. The properties of the MRF heuristic appear similar to those of the MRP algorithm, but the MRF algorithm may produce more accurate supertrees than the MRP algorithm would in cases in which there is little taxon overlap between input trees. Even though MRF and MRP algorithms tend to construct similar supertrees, they generate different sets of solutions. Both methods together may give a more complete picture of supertree solutions than either does alone.

ACKNOWLEDGMENTS

We thank Rod Page for generously providing us with his programs for calculating MMC supertrees and triplet scores and for his helpful comments. We also thank Wen-Chieh Chang for assistance with the simulations and Lisa Thurston for help with the UC–Davis cluster. David Bryant, Junhyong Kim, and Claus Wilke all made valuable comments on the manuscript. This research was supported in part by NSF grant 1053164 awarded to O.E. and M.S. and by NSF grant CCR-9502836 comments on the manuscript. This research was supported in part by NSF grant 1053164 awarded to O.E. and M.S. and by NSF grant CCR-9502836.

REFERENCES


First submitted 9 May 2003; reviews returned 12 August 2003; final acceptance 23 November 2003

Associate Editor: Junhyong Kim