

Improved Heuristics for Minimum-Flip Supertree Construction

Duhong Chen¹, Oliver Eulenstein¹, David Fernández-Baca¹
and J.Gordon Burleigh²

¹Department of Computer Science, Iowa State University, Ames, IA 50011, U.S.A. ²Section of Evolution and Ecology, University of California, Davis, CA 95616, U.S.A.; Current Address, NESCent, Durham, NC 27705, U.S.A.

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

Keywords: Supertree, phylogenetic trees, matrix representation with flipping, matrix representation with parsimony, tree search heuristics.

Introduction

There is increasing interest in supertree methods for phylogenetics (see Bininda-Emonds et al. 2002; Bininda-Emonds, 2004). Supertree methods combine phylogenetic trees with incomplete taxonomic overlap into a comprehensive phylogeny that incorporates all taxa from the input trees. Since the ultimate aim of many supertree analyses is to build large phylogenies, an effective supertree method must be fast as well as accurate. Therefore, the development and implementation of fast algorithms is a critically important part of establishing useful supertree methods.

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

The matrix representation with flipping (MRF, or minimum flip) supertree method, like MRP, uses a matrix representation of the input trees (Chen et al. 2003; Chen et al. 2004; Burleigh et al. 2004; Eulenstein et al. 2004). While MRP seeks trees that minimize the parsimony score of the matrix representation of input trees, MRF seeks the minimum number of flips, character changes from 0 to 1 or 1 to 0, that will make the matrix representation of input trees consistent with a phylogenetic tree (see Chen et al. 2003; Burleigh et al. 2004; Eulenstein et al. 2004). The resulting phylogenetic trees are MRF supertrees. Like the parsimony problem, the minimum flip problem is NP-hard (Chen et al. 2006), and therefore, estimating an MRF supertree requires heuristic algorithms when the input trees contain more than approximately 20 taxa. Simulation experiments (Eulenstein et al. 2004; Piaggio-Talice et al. 2004) indicate that MRF supertrees retain more of the relationships from the input trees than MinCut (Semple and Steel, 2000), Modified MinCut (MMC; Page, 2002), and quartet supertree methods (Piaggio-Talice

Correspondence: Duhong Chen, Tel: (515) 294-2597; Fax: (515) 294-0258; Email: duhong@iastate.edu

et al. 2004). Also, MRF supertrees retain relationships from the input trees at least as well as MRP supertrees, with MRF appearing to slightly outperform MRP as the taxon overlap among input trees decreases (Chen et al. 2003; Burleigh et al. 2004; Eulenstein et al. 2004). Though these results suggest that MRF is a promising supertree method, the characteristics of the simulated data sets likely differ greatly from those of empirical data sets. Furthermore, the first MRF heuristics were slow (Eulenstein et al. 2004; Goloboff, 2005), and consequently the performance of the MRF supertree method on large empirical data sets has been largely unexamined.

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

Definitions

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

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

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

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

such that $a_{ij} = 0$ is the *0-set* of character j and is denoted by Z_j .

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

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

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

$$f_j(v) = z_j(v) + (|O_j| - o_j(v)). \quad (1)$$

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

The flip distance of character j to T is

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

The flip distance of character matrix M to T is

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

The flip distance of a profile τ to T is

$$f_\tau(T) = f_M(T), \quad (4)$$

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

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

Heuristics for MRF

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

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

heuristic, are due to changes in the implementation of the branch swapping operations.

We consider three rooted branch swapping operations.

Rooted Nearest Neighbor Interchange (rNNI)

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

Rooted Subtree Pruning and Regrafting (rSPR)

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

Rooted Tree Bisection and Reconnection (rTBR)

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

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

In the remainder of this section, we first describe a *bottom-up assignment* algorithm that is used in all our branch swapping procedures. We then describe the new rSPR and rTBR algorithms.

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

Bottom-up assignment

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

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

$$\begin{aligned} z_j(v) &= z_j(u) + z_j(w) \\ \text{and} \quad o_j(v) &= o_j(u) + o_j(w). \end{aligned} \quad (5)$$

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

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

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

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

Finding the best rSPR neighbor

The algorithm considers all possible prune nodes; for each such node, it computes the optimum regrafting node. A prune node v is processed in two steps. First, apply the bottom-up assignment algorithm to T_v and $T - T_v$. Thus, for each node w of each tree and each character j we have $z_j(w)$, $o_j(w)$, $f_j(w)$, and $f_j(T_w)$. Second, traverse the nodes of $T - T_v$ in preorder. Let the k -th node in the preorder

sequence be u_k ; thus, u_1 is the root of $T - T_v$. At step k , we compute the flip distance of the tree $T^{(k)}$ obtained by regrafting T_v at u_k . We now explain how to obtain $f_j(T^{(1)})$ in $O(1)$ time and how to compute $f_j(T^{(k)})$, $k > 1$, in $O(1)$ time using the information computed for $T^{(k-1)}$.

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

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

$$\begin{aligned} f_j(T^{(1)}) &= f_j(T_{p_1}^{(1)}) \\ &= \min\{f_j(p_1), f_j(T_v^{(1)}), f_j(T_{u_1}^{(1)})\} \end{aligned} \quad (7)$$

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

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

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

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

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

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

Note that $f_j(T_{p_k}^{(k)}) = \min\{f_j(p_k), f_j(T_v), f_j(T_A)\}$, so $f_j(T_{p_k}^{(k)})$ can be computed in $O(1)$ time given the information available at T_v and T_A from the preprocessing step. Hence, $f_j(T^{(k)})$ can be obtained from $T^{(k-1)}$ in $O(1)$ time using Equation (9). Thus, the best regrafting node for T_v can be found in $O(n)$ time per character. Since there are $O(n)$ choices for v , this leads to a time of $O(n^2)$ per character, and $O(n^2m)$ total, to find the best rSPR neighbor of T .

Finding the best rTBR neighbor

rTBR differs from rSPR in that it may re-root T_v before attaching it to $T - T_v$. To handle rerooting efficiently, we use a *three-way assignment*

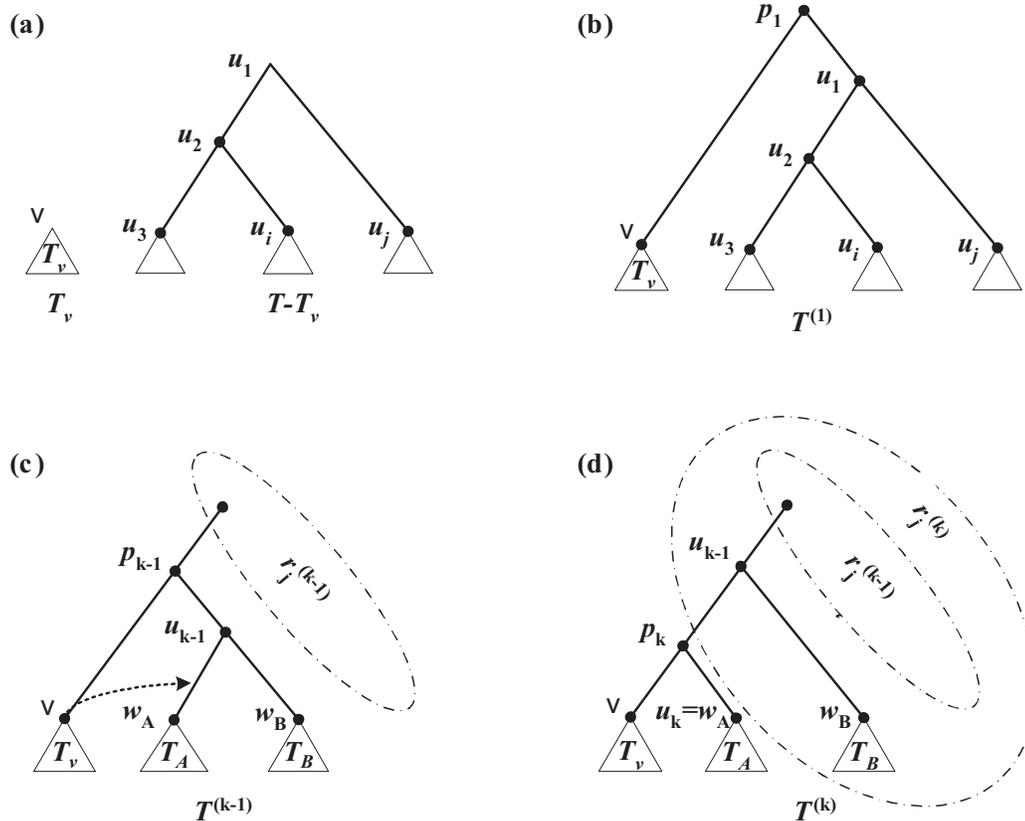


Figure 1. (a) The trees T_v and $T - T_v$ obtained after a cut at node v . (b) The first rSPR neighbor tree $T^{(1)}$ obtained by regrafting at the root. (c–d) The transformation from $T^{(k-1)}$ to $T^{(k)}$.

approach, similar to the one used for parsimony by Ganapathy et al. (2003). We now outline the main ideas of this method.

Consider an internal node u as shown in Figure 2. The subtrees of u change, depending on whether the new root is in the direction of edges 1, 2, or 3; the subtrees of u are $\{T_x, T_y\}$, $\{T_y, T_z\}$, or $\{T_x, T_z\}$, respectively. A *three-way assignment* is a labeling of each vertex u with three lists of values $\langle z_j(u), o_j(u), f_j(u), f_j(T_u) \rangle$, one for each possible rooting. Such an assignment can be obtained in $O(n)$ time per character by doing a bottomup assignment to find the assignments for the first rooting, and then doing a topdown preorder traversal to update the assignments for the other two rootings.

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

and $O(n)$ choices for v , the time required to find the best rTBR neighbor is $O(n^3)$ per character and $O(n^3 m)$ total.

Greedy taxon addition

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

To find the best place to add the k -th taxon to the $(k-1)$ -taxon tree, we use our optimum rSPR neighbor algorithm. In this case, the tree being grafted has a single node containing taxon k and there are $2k-3$ ways to add this taxon to the $(k-1)$ -taxon tree. The time per regraft is $O(km)$, yielding a total running time of $O(n^2 m)$ for the entire addition sequence. We note that the performance the MRF heuristic may be improved by

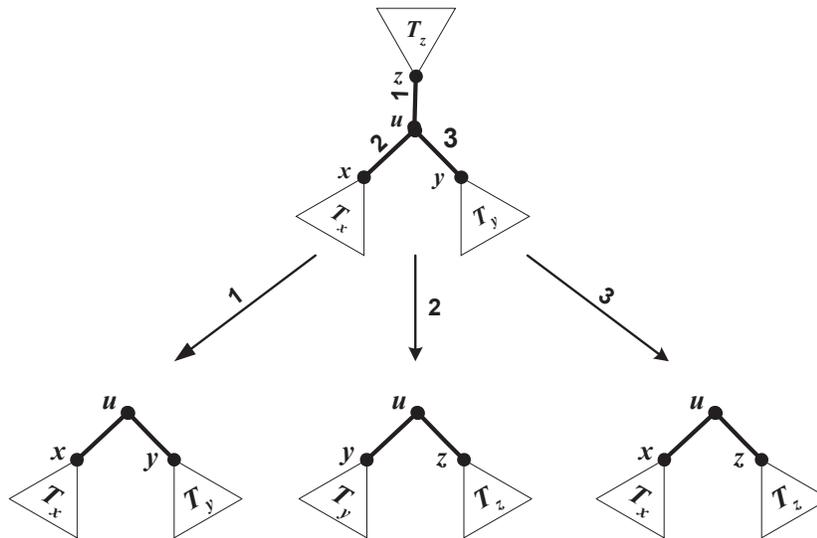


Figure 2. Internal node u and the three possible pairs of subtrees it may have, depending on the rooting. Each requires a different assignment.

repeating the greedy taxon addition using different permutations of the taxa and thus generating multiple starting trees.

Data Sets and Results

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

The performance of each supertree method was evaluated by measuring the degree to which the supertrees agree with the input trees (e.g. Eulenstein et al. 2004). Two measures were used for this purpose: 1) the average MAST-fit between the supertree and the input trees and 2) the average triplet-fit from the supertree to the input trees. The MAST-fit between a supertree and an input tree is the number of leaves in their maximum agreement subtree (Gordon, 1980; Kubicka et al. 1992) divided by the number of leaves in common between the two trees. This was calculated using PAUP* (Swofford, 2002). The triplet-fit from a supertree to an input tree is $1 - (d + r) / (d + r + s)$, where s is the number of rooted triplets that are identically resolved in the supertree and the input trees, d is the number of triplets resolved differently in both trees, and r is the number of triplets resolved in the input tree but not in the supertree (Page, 2002). The triplet-fit, unlike the MAST-fit, is an asymmetric similarity measure. If there was more than one optimal supertree, we present the average

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

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

score of all optimal supertrees to each of the input trees. In addition to measuring the quality of the supertrees, we also compared the CPU-time for each supertree method to provide a rough estimate of the computational requirements for each method. All the analyses were conducted on a Linux platform with a 3.0 GHz Pentium IV processor.

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

MRF with rSPR branch swapping used less CPU time than the MRP heuristic (Table 2).

Discussion

The new heuristic algorithm makes MRF analyses feasible for large empirical data sets. The previous MRF algorithm was not only slow, its performance and implementation were questioned (Goloboff, 2005). Eulenstein et al. (2004) reported that the previous rSPR heuristic performed better than the rTBR heuristic for MRF. Goloboff (2005, p 289) interpreted this to mean that "SPR usually produced a better agreement with the model [true] tree." However, Eulenstein et al.'s (2004) statement only referred to an anecdotal observation that rSPR was faster than rTBR and that the flip distances of rSPR and rTNR trees were very similar if not identical. In this study, we again observed that the MRF heuristic with rSPR branch swapping is much faster than the heuristic with rTBR branch swapping, and that both algorithms yield trees with similar flip distances (Table 2). The similarity between the performance of rSPR and rTBR may seem intuitively surprising (e.g. Goloboff, 2005), but it likely results from the nature of rooted branch swapping. rSPR and rTBR differ in that the latter may reroot the pruned subtree before regrafting. In most situations, it appears that rerooting does not reduce the flip distance. That is, the best rSPR and rTBR neighbors usually have the same flip distance. The

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

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

implementation of the new heuristic also fixes a bug in the implementation of the previous MRF heuristic that caused the program to save suboptimal trees with rTBR and rNNI branch swapping (see Goloboff, 2005). Both rSPR and rTBR heuristics generally produce supertrees with lower flip distances than supertrees produced with rNNI heuristics (not shown). Though there appears to be little benefit in using the rTBR as opposed to rSPR heuristic in a quick MRF analysis, a thorough MRF analysis should include rTBR branch swapping.

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

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

Acknowledgements

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

References

- Baum, B.R. 1992. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, 41:3–10.
- Baum, B.R. and Ragan, M.A. 2004. The MRP method. In Bininda-Emonds, O.R.P., ed. *Phylogenetic supertrees: Combining Information to Reveal the Tree of Life*. Dordrecht: Kluwer Academic, p 17–34.
- Bininda-Emonds, O.R.P. 2004. The evolution of supertrees. *Trends in Ecology and Evolution*, 19:315–22.
- Bininda-Emonds, O.R.P., Gittleman, J.L. and Steel, M.A. 2002. The (super)tree of life: procedures, problems, and prospects. *Annual Review of Ecology and Systematics*, 33:265–89.
- Bordewich, M. and Semple, C. 2004. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, 8:409–23.
- Burleigh, J.G., Driskell, A.C. and Sanderson, M.J. 2006. Supertree bootstrapping methods for assessing phylogenetic variation among genes in genome-scale data sets. *Systematic Biology*, 55:426–440.

- Burleigh, J.G., Eulenstein, O. and Fernández-Baca, D. et al. 2004. MRF supertrees. In Bininda-Emonds, O.R.P., ed. *Phylogenetic supertrees: Combining Information to Reveal the Tree of Life*. Dordrecht: Kluwer Academic, p. 65–85.
- Cardillo, M., Bininda-Emonds, O.R.P. and Boakes, E. et al. 2004. A species-level phylogenetic supertree of marsupials. *Journal of Zool., Lond.*, 264:11–33.
- Chen, D. 2005. HeuristicMFT2 [online]. Accessed 27 December 2005. URL: <http://genome.cs.iastate.edu/CBL/download/>.
- Chen, D., Diao, L. and Eulenstein, O. et al. 2003. Flipping: A supertree construction method. In Janowitz, M., Lapoint, F.J., McMorris, F.R., Roberts, F.S., eds. *Bioconsensus*. Vol. 61 of *DI-MACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, p. 135–60.
- Chen, D., Eulenstein, O. and Fernández-Baca, D. 2004. Rainbow: A toolbox for phylogenetic supertree construction and analysis. *Bioinformatics*, 20(16):2872–2873.
- Chen, D., Eulenstein, O. and Fernández-Baca, D. et al. 2006. Minimum flip supertrees: Complexity and algorithms. *IEEE Trans. Bioinformatics and Computational Biology*.
- Creevey, C.J., Fitzpatrick, D.A. and Philip G.K. et al. 2004. Does a tree-like phylogeny only exist at the tips in the prokaryotes? *Proceeding of the Royal Society of London B. Bio.*, 271:2551–8.
- Eulenstein, O., Chen, D. and Burleigh J.G. et al. 2004. Performance of flip supertrees with a heuristic algorithm. *Systematic Biology*, 53 (2):299–308.
- Farris, J.S., Kluge, A.G. and Eckardt, M.J. 1970. A numerical approach to phylogenetic systematics. *Systematic Zool.*, 19:172–189.
- Ganapathy, G., Ramachandran, V. and Warnow, T. 2003. Better hill-climbing searches for parsimony. In Benson, G., Page R., eds. *Algorithms in Bioinformatics: Third International Workshop, WABI 2003*. Vol. 2812 of *Lecture Notes in Computer Science*. Springer-Verlag, p. 245–58.
- Gatesy, J. and Springer, M.S. 2004. A critique of matrix representation with parsimony supertrees. In Bininda-Emonds, O.R.P., ed. *Phylogenetic supertrees: Combining Information to Reveal the Tree of Life*. Dordrecht: Kluwer Academic, p. 369–88.
- Goloboff, P.A. 2000. Techniques for analysis of large data sets. In DeSalle, R., Wheeler, W., Giribet, G., (eds.), eds. *Techniques in Molecular Systematics and Evolution*. Birkhauser Verlag, Basel, p. 70–9.
- Goloboff, P.A. 2005. Minority rule supertrees? MRP, compatibility, and minimum flip may display the *least* frequent groups. *Cladistics*, 21:282–94.
- Gordon, A.D. 1980. On the assessment and comparison of classifications. In Tomassine, R., ed. *Analyse de Donnees et Informatique*. INRIA, Le Chesnay, p. 7245–58.
- Hein, J.J. 1990. Reconstructing the history of sequences subject to gene conversion and recombination. *Mathematical Biosciences*, 98:185–200.
- Kubicka, E., Kubicki, G. and McMorris, F.R. 1992. On agreement subtrees of two binary trees. *Congressus Numerantium*, 88:217–24.
- Nixon, K. 1999. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:39–50.
- Page, R.D.M. 2002. Modified mincut supertrees. In Guigó, R., Gusfield, D., eds. *Algorithms in Bioinformatics: Second International Workshop, WABI 2002*. Vol. 2452 of *Lecture Notes in Computer Science*. Springer-Verlag, p. 537–51.
- Page, R.D.M. 2003. Supertree [online]. Accessed 2 April 2003. URL: <http://darwin.zoology.gla.ac.uk/~rpage/supertree/>.
- Philip, G.K., Creevey, C.J. and McInerney J.O. 2005. The opisthokonta and ecdysozoa may not be clades: Stronger support for the grouping of plant and animal than for animal and fungi and stronger support for the coelomata than ecdysozoa. *Molecular Biology and Evolution*, 22:1175–84.
- Piaggio-Talice, R., Burleigh, J.G. and Eulenstein, O. 2004. Quartet supertrees. In Bininda-Emonds, O.R.P., ed. *Phylogenetic supertrees: Combining Information to Reveal the Tree of Life*. Dordrecht: Kluwer Academic, p. 173–91.
- Pisani, D. and Wilkinson, M. 2002. MRP, taxonomic congruence and total evidence. *Systematic Biology*, 51:151–5.
- Price, S.A., Bininda-Emonds, O.R.P. and Gittleman J.L. 2005. A complete phylogeny of whales, dolphins and eventooed hoofed mammals (cetartiodactyla). *Biological Reviews*, 80:445–73.
- Purvis, A. 1995. A modification to Baum and Ragan's method for combining phylogenetic trees. *Systematic Biology*, 44:251–5.
- Ragan, M.A. 1992. Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution*, 1:53–8.
- Semple, C. and Steel, M. 2000. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105:147–58.
- Slowinski, J.B. and Page, R.D.M. 1999. How should species phylogenies be inferred from sequence data? *Systematic Biology*, 48:814–25.
- Swofford, D.L. 2002. PAUP*: Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4.0 beta. Sinauer Assoc., Sunderland, Massachusetts, U.S.A.
- Wilkinson, M., Cotton, J.A. and Creevey, C. et al. 2005. The shape of supertrees to come: tree shape related properties of fourteen supertree methods. *Systematic Biology*, 54:419–31.
- Wilkinson, M., Cotton, J.A. and Thorley, J.L. 2004. The information content of trees and their matrix representation. *Systematic Biology*, 53:989–1001.
- Wojciechowski, M.F., Sanderson, M.J. and Steele, K.P. et al. 2000. Molecular phylogeny of the “Temperate Herbaceous Tribes” of Papilionoid legumes: a supertree approach. In Herendeen, P.S., Bruneau, A., eds. *Advances in Legume Systematics*. Vol. 9. Royal Botanic Gardens, Kew, p. 277–98.

Appendix: Finding the Best rNNI Neighbor

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

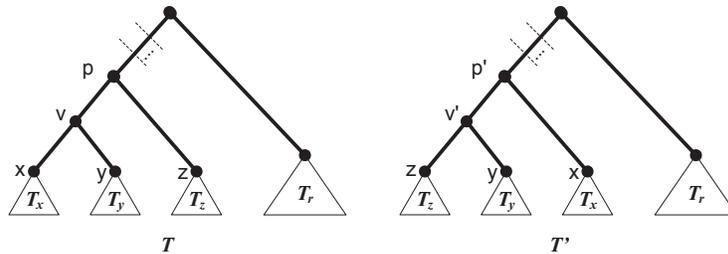


Figure 3. A rNNI operation on an internal node v of T .

Prior to starting the search for the best rNNI neighbor, we execute the bottom-up assignment algorithm. Next, we traverse T in preorder, computing $f_j(T - T_v)$ for each node v of T as follows. If v is the root of T , then $T - T_v$ is an empty tree, denoted by \emptyset , and we define $f_j(\emptyset) = +\infty$. Now, suppose v has parent p and sibling z and assume that $f_j(T - T_p)$ has been correctly computed. Then,

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

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

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

By Equation (2),

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

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

Note that

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

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

On the other hand,

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

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

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