Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle

Richard Desper[1] and Olivier Gascuel[2]

Keywords: Phylogenetic inference, Distance methods, Minimum evolution, Topological accuracy, Computational speed.

[1] *National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, 45 Center Drive, Bethesda, MD, 20892, USA, Phone: (301) 594-8095, Fax: (301) 480-2918, Email:* desper@ncbi.nlm.nih.gov

[2] (address for correspondence) *Département Informatique Fondamentale et Applications, LIRMM, 161 rue Ada, 34392 Montpellier, France, Phone: +33 4 67 41 85 47, Fax: +33 4 67 41 85 00, Email:* gascuel@lirmm.fr, *Web:* http://www.lirmm.fr/~w3ifa/MAAS/

Abstract

The Minimum Evolution (ME) approach to phylogeny estimation has been shown to be statistically consistent when it is used in conjunction with ordinary least-squares (OLS) fitting of a metric to a tree structure. The traditional approach to using ME has been to start with the Neighbor Joining (NJ) topology for a given matrix, and then do a topological search from that starting point. The first stage requires $O(n^3)$ time, where n is the number of taxa, while the current implementations of the second are in $O(pn^3)$ or more, where p is the number of swaps performed by the program. In this paper, we examine a greedy approach to Minimum Evolution which produces a starting topology in $O(n^2)$ time. Moreover, we provide an algorithm that searches for the best topology using nearest neighbor interchanges (NNIs), where the cost of doing p NNIs is $O(n^2 + pn)$, i.e. $O(n^2)$ in practice because p is always much smaller than n. The Greedy Minimum Evolution (GME) algorithm, when used in combination with NNIs, produces trees which are fairly close to NJ trees in terms of topological accuracy. We also examine ME under a balanced weighting scheme, where sibling subtrees have equal weight, as opposed to the standard "unweighted" OLS, where all taxa have the same weight so that the weight of a subtree is equal to the number of its taxa. The balanced minimum evolution scheme (BME) runs slower than the OLS version, requiring $O(n^2 * \mathrm{diam}(T))$ operations to build the starting tree and $O(pn * \mathrm{diam}(T))$ to perform the NNIs, where $\mathrm{diam}(T)$ is the topological diameter of the output tree. In the usual Yule-Harding distribution on phylogenetic trees, the diameter expectation is in $log(n)$, so our algorithms are in practice faster that NJ. Moreover, this BME scheme yields a very significant improvement over NJ and other distance-based algorithms, especially with large trees, in terms of topological accuracy.

Introduction

Minimum evolution was proposed by several authors (Kidd and Sgaramella-Zonta 1971; Saitou and Nei 1987; Rzhetsky and Nei 1993; Swofford, Olsen, Waddell, and Hillis 1996) as a basic principle for phylogenetic inference. Given the matrix of pairwise evolutionary distances between the taxa being studied, this principle involves first estimating the length of any given topology and then selecting the topology with shortest length. Minimum evolution is thus conceptually close to character-based parsimony, and complies with Occam's principle of scientific inference, which essentially maintains that simpler explanations are preferable to more complicated ones and that *ad hoc* explanations should be avoided.

Numerous variants of the minimum evolution principle exist, depending on how the branch lengths are estimated and how the tree length is calculated from these branch lengths. Several definitions of tree length have been proposed, differing from one another by the treatment of negative branch lengths. The most common solution (Saitou and Nei 1987; Rzhetsky and Nei 1993) simply defines the tree length as the sum of all branch lengths, regardless of whether they are positive or negative. Branch lengths are usually estimated within the least-squares framework. If all distance estimates can be assumed to be independent and to have the same variance, we use the ordinary least-squares (OLS) framework. The weighted least-squares framework corresponds to the case were distance estimates are independent but (possibly) with different variances, while the generalized least-squares approach does not impose any restriction and is able to take benefit from the covariances of the distance estimates. It is well-known that distance estimates obtained from sequences do not have the same variance, because the largest distances are much more variable than the shortest

ones (Fitch and Margoliash 1967), and are mutually dependent when they share a common history (or path) in the true phylogeny (Nei and Jin 1989). Therefore, to estimate branch lengths from evolutionary distances, using generalized least-squares is theoretically superior to using weighted least-squares, which is in turn more appropriate than ordinary least-squares (Bulmer 1991).

The minimum evolution principle has been shown to be statistically consistent when combined with ordinary least-squares (Rzhetsky and Nei 1993; Denis and Gascuel 2002). This important property implies that the more accurate the distance estimates, as induced by the use of long sequences when a correct sequence evolution model is chosen, the higher the probability of recovering the true phylogeny. However, ordinary least-squares poorly fits the features of evolutionary distance data, as explained above. Thus, it is tempting to combine the minimum-evolution principle with a more reliable estimation of branch lengths, using weighted least-squares or generalized least-squares. However, we recently demonstrated that such a combination is not always statistically consistent and, therefore, could represent a dead end towards obtaining better phylogenetic inference methods, especially in the case of generalized least-squares (Gascuel, Denis and Bryant 2001).

This paper further investigates the minimum evolution principle, but with a more optimistic perspective. First, we demonstrate that its usage in combination with ordinary least-squares, even when not fully optimal in terms of topological accuracy, has the great advantage of leading to very fast algorithms, much faster than the NJ algorithm (Saitou and Nei 1987), fast enough as to be able to build very large trees as envisaged in biodiversity studies. Second, we show that a new version of this principle, first introduced by Pauplin

(2000) to simplify tree length computation, is more appropriate than the OLS version. In this new version, sibling subtrees have equal weight, as opposed to the standard unweighted OLS, where all taxa have the same weight and thus the weight of a subtree is equal to the number of its taxa. This new version can be seen as weighted, just as WPGMA is the weighted version of UPGMA (Sneath and Sokal 1973), but we will prefer the term "balanced" to avoid confusion with weighted least-squares. In addition to the aforementioned fast OLS minimum evolution algorithms, we also present algorithms to deal with this new balanced version that are also faster than NJ, though not as fast as their OLS counterparts. Furthermore, the balanced algorithms produced output trees with better topological accuracy than those from NJ, BIONJ (Gascuel 1997a) and WEIGHBOR (Bruno, Socci and Halpern 2000). The rest of this paper is organized as follows: we first provide the notation and definitions, we describe the algorithms for the OLS version of the minimum evolution principle, we explain how these algorithms are modified to deal with the balanced version, we provide simulation results to illustrate the gain in topological accuracy and run times, and we then conclude by a brief discussion. The appendix provides the details of the algorithms and some mathematical proofs.

1 Notation, definitions, and formulae

A tree is made of nodes (or vertices) and of edges (or branches). Among the nodes we distinguish the internal (or ancestral) nodes and the leaves (or taxa). The leaves are denoted as i, j or k, the internal nodes as u, v or w, while an edge e is defined by a pair of nodes and a length $l(e)$. We shall be considering various length assignments of the same underlying shape. In this case, we shall use the word "topology", while "tree" will be reserved for an instance of a topology with given edge lengths associated. We use the letter $\mathcal{T}$ to refer to a topology and T to refer to a tree. A tree is also made of subtrees (or clades), typically denoted as A, B, C or D. For the sake of simplicity, we shall use the same notation for the subtrees and for the sets of taxa they contain. Accordingly, T also represents the set of taxa being studied, and n is the number of these taxa. Moreover, we shall use lowercase letters, e.g. a, b, c or d, to represent the subtree roots. If A and B are two disjoint subtrees, with roots a and b respectively, we'll say that A and B are distant-k subtrees if there are k edges in the path from a to b.

Δ is the matrix of pairwise evolutionary distance estimates, with Δ_{ij} being the distance between taxa i and j. Let A and B be two non-intersecting subtrees from a tree T. We define the average distance between A and B as:

$$\Delta_{A|B} = \frac{1}{|A||B|} \sum_{i \in A, j \in B} \Delta_{ij} \tag{1}$$

$\Delta_{A|B}$ may also be defined recursively as:

- If A and B are singleton sets, i.e. $A = \{a\}$ and $B = \{b\}$, then $\Delta_{A|B} = \Delta_{ab}$,

- Else, without loss of generality let $B = B_1 \cup B_2$ as shown in Figure 1, we then have

$$\Delta_{A|B} = \frac{|B_1|}{|B|}\Delta_{A|B_1} + \frac{|B_2|}{|B|}\Delta_{A|B_2} \tag{2}$$

It is easily seen that Equations (1) and (2) are equivalent. Equation (2) follows the notion that the weight of a subtree is proportional to the number of its taxa. So every taxon has the same weight, and the same holds for the distances as shown by Equation (1). Thus, this average is said to be unweighted (Sneath and Sokal 1973). It must be noted that the unweighted average distance between subtrees does not depend on their topologies, but only of the taxa they contain.

Δ^T is the distance induced by the tree T, i.e., Δ^T_{ij} is equal to the length of the path connecting i to j in T, for every taxon pair (i, j) . Given a topology $\mathcal{T}$ and a distance matrix Δ, the OLS branch length estimation produces the tree T with topology $\mathcal{T}$ minimizing the sum of squares:

$$\sum_{i,j \in T} (\Delta^T_{ij} - \Delta_{ij})^2.$$

Vach (1989), Rzhetsky and Nei (1993) and others showed analytical formulae for the proper OLS edge length estimation, as functions of the average distances. Suppose e is an internal edge of T, with the four subtrees A, B, C and D defined as depicted in Figure 2(a). Then, the OLS length estimate of e is equal to:

$$l(e) = \frac{1}{2}(\lambda(\Delta_{A|C} + \Delta_{B|D}) + (1-\lambda)(\Delta_{A|D} + \Delta_{B|C}) - (\Delta_{A|B} + \Delta_{C|D})), \tag{3}$$

where

$$\lambda = \frac{|A||D| + |B||C|}{(|A| + |B|)(|C| + |D|)}.$$

Suppose e is an external branch, with i, A and B as represented in Figure 2(b). Then we have:

$$l(e) = \frac{1}{2}(\Delta_{A|i} + \Delta_{B|i} - \Delta_{A|B}). \tag{4}$$

Equations (3) and (4) demonstrate an important property of OLS edge length estimation: the length estimate of any given edge does not depend of the topology of the "corner" subtrees, i.e., A, B, C and D in Equation (3) and A and B in Equation (4), but only of the taxa contained in these subtrees.

Following Saitou and Nei (1987) and Rzhetsky and Nei (1993), we define the tree length $l(T)$ of T to be the sum of the edge lengths of T. The OLS minimum evolution tree is then that tree with topology $\mathcal{T}$ minimizing $l(T)$, where T has the OLS edge length estimates for $\mathcal{T}$, and $\mathcal{T}$ ranges over all possible tree topologies for the taxa being studied.

Now, suppose that we are interested in the length of the tree T shown in Figure 2(a), depending on the configuration of the corner subtrees. We then have (proof in Appendix 1):

$$\begin{aligned} l(T) &= \frac{1}{2}(\lambda(\Delta_{A|C} + \Delta_{B|D}) + (1-\lambda)(\Delta_{A|D} + \Delta_{B|C}) + \Delta_{A|B} + \Delta_{C|D}) \\ &\quad + l(A) + l(B) + l(C) + l(D) - \Delta_{a|A} - \Delta_{b|B} - \Delta_{c|C} - \Delta_{d|D} \end{aligned} \tag{5}$$

where λ is defined as in Equation (3). The advantage of Equation (5) is that the lengths $l(A), l(B), l(C)$ and $l(D)$ of the corner subtrees as well as the average root/leaf distances, $\Delta_{a|A}, \Delta_{b|B}, \Delta_{c|C}$ and $\Delta_{d|D}$, do not depend of the configuration of A, B, C and D around

e. Exchanging B and C or B and D might change the length of the five edges shown in Figure 2(a), and then the length of T, but not the lengths of A, B, C and D. This simply comes from the fact that the edge e is within the corner subtrees associated to any of the edges of A, B, C and D. As we shall see in the next section, this property is of great help in designing fast OLS tree swapping algorithms.

Let us now turn our attention toward the balanced version of minimum evolution, as defined by Pauplin (2000). The tree length definition is the same. Formulae for edge length estimates are identical to Equations (3) and (4), with λ replaced by $1/2$ and using a different definition of the average distance between subtrees, which depends on the topology under consideration. Letting $\mathcal{T}$ be this topology, the balanced average distance between two non-intersecting subtrees A and B is then recursively defined by:

- If A and B are singleton sets, i.e. $A = \{a\}$ and $B = \{b\}$, then $\Delta^{\mathcal{T}}_{A|B} = \Delta_{ab}$,
- Else, without loss of generality let $B = B_1 \cup B_2$ as shown in Figure 1, we then have

$$\Delta^{\mathcal{T}}_{A|B} = \frac{1}{2}(\Delta^{\mathcal{T}}_{A|B_1} + \Delta^{\mathcal{T}}_{A|B_2}). \tag{6}$$

The change from Equation (2) is that the sibling subtrees B_1 and B_2 have now equal weight, regardless of the number of taxa they contain. Thus taxa do not have the same influence depending on whether they belong to a large clade or are isolated, which can be seen as consistent in the phylogenetic inference context (Sneath and Sokal 1973). Moreover, comparing variants of the NJ algorithm we showed by computer simulation (Gascuel 2000) that this "weighted" approach is more appropriate than the unweighted one for reconstructing

phylogenies with evolutionary distances estimated from sequences. Therefore, it was tempting to test the performance of this weighted (or balanced) version of the minimum evolution principle.

Unfortunately, this new version does not have all of the good properties as the OLS version: the edge length estimates given by Equations (3) and (4) now depend on the topology of the corner subtrees, simply because the weighted average distances between these subtrees depend on their topologies. As we shall see, this makes the algorithms more complicated and more expensive in computing time than with OLS. However, the same tree length formula, Equation (5), holds with Δ being replaced by $\Delta^{\mathcal{T}}$ and λ by 1/2, and, fortunately, we still have the good property that tree lengths $l(A), l(B), l(C)$ and $l(D)$ as well as average root/leaves distances $\Delta^{\mathcal{T}}_{a|A}, \Delta^{\mathcal{T}}_{b|B}, \Delta^{\mathcal{T}}_{c|C}$ and $\Delta^{\mathcal{T}}_{d|D}$ remain unchanged when B and C or B and D are swapped. Edge lengths within the corner subtrees may change when performing a swap, but their (balanced) sums remain identical (proof in Appendix 2).

2 Algorithms for the OLS version of Minimum Evolution

This section presents two algorithms for phylogenetic inference. The first constructs an initial tree by the stepwise addition of taxa to a growing tree, while the second improves this tree by performing local rearrangements (or swapping) of subtrees. Both follow a greedy approach and tend, at each step, to minimize the OLS version of the minimum evolution criterion. This approach does not guarantee that the global optimum will be reached, but only a local optimum. However, this kind of approach has proven to be effective in many optimization problems (Cormen, Leiserson and Rivest 2000, 329-356), and we shall see that further optimizing the minimum evolution criterion would not yield significant improvement in terms of topological accuracy. Moreover, such a combination of heuristic and optimization algorithms is used in numerous phylogenetic reconstruction methods, for example those from the PHYLIP package (Felsenstein 1989).

2.1 The GME greedy addition algorithm

Given an ordering on the taxa, denoted as $(1, 2, 3, \ldots, n)$, for $k = 4$ to n we create a tree T_k on the taxa set $(1, 2, 3, \ldots, k)$. We do this by testing each edge of T_{k-1} as a possible insertion point for k, and the different insertion points are compared by the minimum evolution criterion. Inserting k on any edge of T_{k-1} removes that edge, changes the length of every other already existing edge, and requires the computation of the length of the three newly created edges. Computing the new tree length for every possible insertion point would seem to be computationally expensive. However, a much simpler approach exists to determine the best insertion point.

Consider the tree T of Figure 3, where k is inserted between subtrees C and $A \cup B$, and

assume that we have the length $l(T) = L$ of this new tree. Consider now the tree T' of Figure 3, which is obtained from T by exchanging k and A. Using Equation (5) and our above remarks we have

$$l(T') = L + \frac{1}{2}\left[(\lambda - \lambda')(\Delta_{k|A} + \Delta_{B|C}) + (\lambda' - 1)(\Delta_{A|B} + \Delta_{k|C}) + (1 - \lambda)(\Delta_{A|C} + \Delta_{k|B})\right] \quad (7)$$

where

$$\lambda = \frac{|A| + |B||C|}{(|A| + |B|)(|C| + 1)},$$

and

$$\lambda' = \frac{|A| + |B||C|}{(|A| + |C|)(|B| + 1)}.$$

In other words, the length of T' can be computed from the length of T. For this computation to be done in $O(1)$ (i.e. constant) time, it is sufficient to have previously computed

1. all average distances $\Delta_{k|S}$ between k and any subtree S from T_{k-1};
2. all average distances between subtrees of T_{k-1} separated by two edges, for example A and B in Figure 3;
3. the number of leaves of every subtree.

Suppose we now consider the tree T'' formed by moving the insertion of k to the edge e, where e is a sibling edge to the insertion point of T'. The length of T'' is computed by Equation (7) as $l(T'') = L + f(e)$, where $f(e)$ depends on the computations for both T' and T''. We continue, searching every edge e of T_{k-1} by recursively moving from one edge to its neighboring edges, and we obtain the cost $c(e)$ that corresponds to the length of the tree

T_{k-1} plus k inserted on e. Moreover, $c(e)$ can be written as $L + f(e)$. Because we only seek to determine the best insertion point, we need not calculate the actual value of L, as it is sufficient to minimize $f(e)$ with $f = 0$ for the first insertion edge considered.

The algorithm can be summarized as follows:

- For $k = 3$, initialize the matrix of average distances between distant-2 subtrees and the array counting the number of taxa per subtree. Form T_3 with leaf set $\{1, 2, 3\}$.
- For $k = 4$ to n,
  1. Compute all $\Delta_{k|S}$ average distances;
  2. Starting from an initial edge e_0 of T_{k-1}, set $f(e_0) = 0$, and recursively search each edge e to obtain $f(e)$ from Equation (7).
  3. Select the best edge by minimizing f, insert k on that edge to form T_k, and update the average distance between every pair of distant-2 subtrees as well as the number of taxa per subtree.
- Return T_n.

To achieve Step 1, we recursively apply Equation (2), which requires $O(k)$ computing time (see Appendix 3). Step 2 is also done in $O(k)$ time, as explained above. Finally, to update the average distance between any pair A, B of distant-2 subtrees, if k is inserted in the subtree A,

$$\Delta_{\{k\}\cup A|B} = \frac{1}{1+|A|}\Delta_{k|B} + \frac{|A|}{1+|A|}\Delta_{A|B}. \tag{8}$$

Step 3 is also done in $O(k)$ time, because there are $O(k)$ pairs of distant-2 subtrees, and all the quantities in the right-hand side of Equation (8) have already been computed. So we build T_k from T_{k-1} in $O(k)$ computational time, and thus the entire computational cost of the construction of T, as we sum over k, is $O(n^2)$. This is much faster than NJ-like algorithms which require $O(n^3)$ operations, and the FITCH (Felsenstein 1997) program which requires $O(n^4)$ operations. As we shall see, this allows trees with 4,000 taxa to be constructed in few minutes, while this task requires more than an hour for NJ, and is essentially impossible for FITCH on any existing and imaginable machine. This algorithm is called GME (Greedy Minimum Evolution algorithm) and additional details are described in Appendix 3.

2.2 The FASTNNI tree swapping algorithm

This algorithm iteratively exchanges subtrees of an initial tree, in order to minimize its OLS length estimate. There are many possible definitions of "subtree exchange". Since the number of combinations is high, we usually only consider exchanging neighboring subtrees, and, at least initially, we restrict ourselves to the exchange of subtrees separated by three edges, for example B and C in Figure 2(a). Such a procedure is called a "nearest neighbor interchange", since exchanging subtrees separated by one or two edges does not yield any modification of the initial tree. We adopted this approach because it allows a fast algorithm, and it is sufficient to reach a good topological accuracy.

Consider Figure 2(a), and assume that the swap between B and C is considered. Let T be the initial tree and T' the swapped tree. According to Equation (5) and our remarks, we

have:

$$L(T) - L(T') = \frac{1}{2}\Big[(\lambda - 1)(\Delta_{A|C} + \Delta_{B|D}) - (\lambda' - 1)(\Delta_{A|B} + \Delta_{C|D}) - (\lambda - \lambda')(\Delta_{A|D} + \Delta_{B|C})\Big], \quad (9)$$

where λ is as defined in Section 1, and

$$\lambda' = \frac{|A||D| + |B||C|}{(|A| + |C|)(|B| + |D|)}.$$

The swap has to be performed when $L(T) - L(T') > 0$, and the best among all possible swaps (2 per internal edge) corresponds to the largest difference between $L(T)$ and $L(T')$. Moreover, assuming that the average distances between the corner subtrees have already been computed, $L(T) - L(T')$ can be obtained in $O(1)$ time via Equation (9). Instead of computing the average distances between the corner subtrees (which change when swaps are realized), we compute the average distances between every pair of non-intersecting subtrees. This takes place before evaluating the swaps and requires $O(n^2)$ time, using an algorithm that is described in Appendix 4. The whole algorithm can be summarized as follows:

1. Precompute the average distances between non-intersecting subtrees;

2. Run over all internal edges and select the best swap using Equation (9);

3. If the best swap does not improve the length of the tree; i.e. if $(L(T) - L(T') \leq 0)$, stop and return the current tree, else perform the swap, compute the average distances between the newly created subtrees ($A \cup C$ and $B \cup D$ in our example above) and the

other non-intersecting subtrees using Equation (2), and go to Step 2.

Step 1 requires $O(n^2)$ time, Step 2 requires $O(n)$ time and Step 3 also requires $O(n)$ time because the total number of subtrees is $O(n)$. Thus, the total complexity of the algorithm is $O(n^2 + pn)$, where p is the number of swaps performed. In practice, p is much smaller than n, as we shall see in Section 4, so this algorithm has a practical time complexity of $O(n^2)$. It is very fast, able to improve trees with thousands of taxa, and we called it FASTNNI (Fast Nearest Neighbor Interchanges). More details are given in Appendix 4.

Rzhetsky and Nei (1993) describe a procedure that requires $O(n^2)$ to compute every branch length. In one NNI, five branch lengths are changed, so evaluating a single swap is in $O(n^2)$, searching for the best swap in $O(n^3)$, and their whole procedure in $O(pn^3)$. This can be improved using Bryant and Waddell (1998) results, but the implementation in the PAUP environment of these ideas is still in progress (David Bryant, personnal communication). In any case, our $O(n^2)$ complexity is optimal since it is identical to the data size.

Neither FASTNNI nor GME needs to explicitly compute the length of the whole tree until the final topology is reached. To obtain the length of the final tree, we use Equations (3) and (4), which requires $O(n)$ time as the average distances between corner subtrees have already been computed during the execution of FASTNNI.

3 Algorithms for the balanced version of minimum evolution

The balanced averaging scheme lends itself both for an insertion-based approach and for tree swapping from an initial topology, and the algorithms are essentially the same as with OLS. The main difference is that updating can no longer be achieved using a fast method as expressed by Equation (8), because the balanced average distance between $A \cup \{k\}$ and B now depends of the position of k within A.

3.1 The BME addition algorithm

This algorithm is similar to that for OLS. Equation (7) simplifies into:

$$L(T') = L + \frac{1}{4}\left[(\Delta^{\mathcal{T}}_{A|C} + \Delta^{\mathcal{T}}_{k|B}) - (\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{k|C})\right]. \tag{10}$$

Step 1 is identical and provides all $\Delta^{\mathcal{T}}_{k|S}$ distances by recursively applying Equation (6). The main difference is the updating performed in Step 3. Equation (10) only requires the balanced average distances between distant-2 subtrees, but to iteratively update these distances, we use (and update) a data structure that contains all distances between every pair of non-intersecting subtrees (as for FASTNNI).

How many new averages must be calculated, when k is inserted into T_{k-1}? We must calculate the average $\Delta^{\mathcal{T}_k}_{X|Y\cup\{k\}}$ for any subtree Y of T_{k-1} such that $Y \cup \{k\}$ is a subtree of T_k, and any subtree X disjoint from $Y \cup \{k\}$. We can enumerate all such pairs by considering their respective roots. Let x be the root of X and y the root of Y. Regardless of the position of k, any node of T_k could serve as the root x of X. Then, considering a fixed x, any node in the path from x to k could serve as the root y. Thus there are $O(k \times \mathrm{diam}(T_k))$ such pairs,

where diam is the tree diameter, i.e. the maximum number of edges between two leaves.

Given such a pair, X, Y, let us consider how we may quickly calculate $\Delta^{\mathcal{T}_k}_{X|Y\cup k}$ from known quantities. Consider the situation as depicted in Figure 4. Suppose k is inserted by creating a new node w which pushes the subtree Y_1 farther away from B. Suppose there are $(l-1)$ edges in the path from w to y, and the subtrees branching off this path are, in order from w to y, $Y_2, Y_3, \ldots, Y_l$. Then

$$\Delta^{\mathcal{T}_k}_{X|Y\cup\{k\}} = 2^{-l}(\Delta^{\mathcal{T}_k}_{k|X} + \Delta^{\mathcal{T}_{k-1}}_{X|Y_1}) + \sum_{i=2}^{l} 2^{-(l+1-i)}\Delta^{\mathcal{T}_{k-1}}_{X|Y_i}.$$

However, we already know the value of

$$\Delta^{\mathcal{T}_{k-1}}_{X|Y} = 2^{-(l-1)}\Delta^{\mathcal{T}_{k-1}}_{X|Y_1} + \sum_{i=2}^{l} 2^{-(l+1-i)}\Delta^{\mathcal{T}_{k-1}}_{X|Y_i}.$$

Thus

$$\Delta^{\mathcal{T}_k}_{X|Y\cup\{k\}} = \Delta^{\mathcal{T}_{k-1}}_{X|Y} + 2^{-l}(\Delta^{\mathcal{T}_k}_{k|X} - \Delta^{\mathcal{T}_{k-1}}_{X|Y_1}). \tag{11}$$

The upper bound on the number of pairs is worst when T_k is a chain, with k inserted at one end. In this case, the diameter is proportional to k and both the number of distances to update and the bound are proportional to k^2. However, the diameter is usually much lower. Assuming, as usual, a Yule-Harding speciation process (Yule 1925; Harding 1971), the expected diameter is $O(\log(k))$ (Erdös, Steel, Székély and Warnow 1999), which implies an average complexity of the updating step in $O(k\log(k))$. Other (e.g. uniform) distributions on phylogenetic trees are discussed in (Aldous 2001; McKenzie and Steel 2000), with expected diameter at most $O(\sqrt{k})$.

Therefore, the time complexity of the whole insertion algorithm is $O(n^3)$ in the worst case, and $O(n^2 \log(n))$ (or $O(n^2\sqrt{n})$) in practice. This is still much less than NJ and allows trees with thousands of taxa to be constructed within a reasonable amount of time. To wit: the BME program required less than 20 seconds on average to build 1000-taxon trees, and approximately 12 minutes on average to build 4000-taxon trees. In contrast, NJ required twice as much time for 1000-taxon trees, and five times as much time for 4000-taxon trees. This algorithm is called BME (Balanced Minimum Evolution) and additional details are given in Appendix 5.

3.2 The BNNI tree swapping algorithm

This algorithm is again very similar to that for OLS. Equation (9) simplifies into: .

$$L(T) - L(T') = \frac{1}{4}((\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{C|D}) - (\Delta^{\mathcal{T}}_{A|C} + \Delta^{\mathcal{T}}_{B|D})). \tag{12}$$

Step 1 is identical, but Equation (2) is replaced by Equation (6). Of course, the preliminary step of the tree swapping algorithm is unnecessary when the initial tree is constructed using the BME algorithm, because this latter computes (among other things) the average balanced distances between every pair of non- intersecting subtrees. Finally, the main difference is within Step 3, where average distances between subtrees are updated. The computations are almost identical to those of Section 3.1 and require $O(n \log(n))$ computations on average. Thus, the total time complexity is $O(n^2 + pn \log(n))$ where p is the number of performed swaps, or $O(pn \log(n))$ if Step 1 is unnecessary. As with the OLS algorithms, this allows very large trees to be improved. This algorithm is called BNNI (Balanced Nearest Neighbor

Interchanges), and details are given in Appendix 5.

Finally, as with the OLS algorithms, neither BME nor BNNI explicitely computes the branch lengths. This is done in $O(n)$ time by using the edge length formulae from Appendix 2

$$l(e) = \Delta^{\mathcal{T}}_{A\cup B|C\cup D} - \frac{1}{2}(\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{C|D})$$

and

$$l(e) = \frac{1}{2}(\Delta^{\mathcal{T}}_{i|A} + \Delta^{\mathcal{T}}_{i|B} - \Delta^{\mathcal{T}}_{A|B}),$$

for the external and internal branches, respectively (see Figure 2 for the notation).

4 Results

4.1 Protocol

We used simulations based on random trees with parameter values chosen so as to be representative of the data sets commonly encountered in evolutionary studies. Parameter values were chosen so as to cover the features of most real data sets, as revealed by the compilation of the phylogenies published in the journal *Systematic Biology* during the last few years. This approach induces much smaller contrasts between the tested methods than those based on model trees (e.g., Gascuel 1997a; Bruno, Socci and Halpern. 2000). Indeed, model trees are generally used to emphasize a given property of the studied methods, for example their performance when the molecular clock is strongly violated. Thus, model trees are often extreme and their use tends to produce strong and possibly misleading differences between the tested methods. On the other hand, random trees allow comparisons with a large variety of tree shapes and evolutionary rates, and provide a synthetic and more realistic view of the average performances.

We used 24- and 96-taxon trees, and 2000 trees per size. For each of these trees, a true phylogeny, denoted as T, was first generated using the stochastic speciation process described by Kuhner and Felsenstein (1994), which corresponds to the usual Yule-Harding distribution on trees (Yule 1925; Harding 1971). Using this generating process makes T ultrametric (or molecular clock-like). This hypothesis does not hold in most biological data sets, so we created a deviation from the molecular clock, using a method similar to that of (Guindon and Gascuel 2002). Every branch length of T was multiplied by $1.0 + \mu X$, where X followed the standard exponential distribution ($P(X > \eta) = e^{-\eta}$) and μ was a tuning

factor to adjust the deviation from molecular clock; μ was set to 0.8 with 24 taxa and to 0.6 with 96 taxa. The average ratio between the mutation rate in the fastest evolving lineage and the rate in the slowest evolving lineage was then equal to about 2.0 with both tree sizes. With 24 taxa, the smallest value (among 2000) of this ratio was equal to about 1.2 and the largest to 5.0 (1.0 corresponds to the strict molecular clock), while the standard deviation was approximately 0.5. With 96 taxa, the extreme values became 1.3 and 3.5, while the standard deviation was 0.33.

These (2×2000) trees were then rescaled to obtain "slow", "moderate" and "fast" evoutionary rates. With 24 taxa, the branch length expectation was set to 0.03, 0.06 and 0.15 mutations per site, for the slow, moderate and fast conditions, respectively. With 96 taxa, we had 0.02, 0.04 and 0.10 mutations per site, respectively. For both tree sizes, the average maximum pairwise divergence was of about 0.2, 0.4 and 1.0 substitutions per site, with a standard deviation of about 0.07, 0.14 and 0.35 for 24 taxa, and of about 0.04, 0.08 and 0.20 for 96 taxa. These values are in good accordance with real data sets. The maximum pairwise divergence is rarely above 1.0 due to the fact that multiple alignement from highly divergent sequences is simply impossible. Moreover, with such distance value, any correction formula, e.g. Kimura's (1980), becomes very doubtful, due to our ignorance of the real substitution process and to the fact that the larger the distance the higher the gap between estimates obtained from different formulae. The medium condition ($\sim$0.4) corresponds to the most favorable practical setting, while in the slow condition (from $\sim$0.1 to $\sim$0.3) the phylogenetic signal is only slightly perturbed by multiple substitutions, but it can be too low, with some short branches being not supported by any substitution.

SeqGen (Rambaut and Grassly 1997) was used to generate the sequences. For each tree T (among $3 \times 2 \times 2000$), these sequences were obtained by simulating an evolving process along T according to the Kimura (1980) two parameter model with a transition/transversion ratio of 2.0. The sequence length was set to 500 sites. Finally, DNADIST from the PHYLIP package (Felsenstein 1989) was used to compute the pairwise distance matrices, assuming the Kimura model with known transition/transversion ratio. The data files are available on our web page (see title page).

Every inferred tree, denoted as $\widehat{T}$, was compared to the true phylogeny T (i.e., that used to generate the sequences and then the distance matrix) with a topological distance equivalent to Robinson and Foulds' (1981). This distance is defined by the proportion of internal branches (or bipartitions) which are found in one tree and not in the other one. This distance varies between 0.0 (both topologies are identical) and 1.0 (they do not share any internal branch). The results were averaged over the 2000 test sets for each tree size and evolutionary rate. Finally, to compare the various methods to NJ, we measured the relative error reductions $(P_M - P_{NJ})/P_{NJ}$, where M is any tested method different from NJ and P_X is the average topological distance between $\widehat{T}$ and T when using method X.

4.2 Phylogeny Estimation Algorithm Comparison

We used a variety of different algorithms to try to reconstruct the original tree, given the matrix of estimated distances. We used the NJ program from the PAUP package (Swofford 1996), with and without the BIONJ (Gascuel 1997a) flag; WEIGHBOR version 1.2, available at `http://www.t10.lanl.gov/billb/weighbor/`; the FITCH program from the PHYLIP package (Felsenstein 1989); the Harmonic Greedy Triplet (HGT/FP) program, provided by

Miklos Csűrös (Csűrös 2002); GME and BME. Also, we used output topologies from other programs as input for FASTNNI and BNNI. All of GME, BME, FASTNNI, and BNNI will be available in the near future at our web page and via ftp at

`ftp://ncbi.nlm.nih.gov/~desper/ME`.

We also measured how far from the true phylogeny one gets with NNIs. This served as a measure of the limitation of each of the minimum evolution frameworks, as well as a performance index for evaluating our algorithms. Ordinarily, the (OLS or balanced) minimum evolution criterion will not, in fact, observe a minimum value at the true phylogeny. So, starting from the true phylogeny and running FASTNNI or BNNI, we end up with a tree with a significant proportion of false branches. When this proportion is high, the corresponding criterion can be seen as poor regarding topological accuracy. Thus this proportion represents the best possible topological accuracy that can be achieved by optimizing the considered criterion, since we would not expect any algorithm optimizing this criterion in the whole tree space to find a tree closer from the true phylogeny than the tree that is obtained by "optimizing" the true phylogeny itself.

Results are displayed in Table 1 and Table 2.

(Table 1 here)

The performances of the basic algorithms are strongly correlated with the number of computations that they perform. Both $O(n^2)$ algorithms are clearly worse than NJ. BME (in $O(n^2 \log(n))$) is still worse than NJ, but becomes very close with 96 taxa, which indicates the strength of the balanced minimum evolution framework. BIONJ, in $O(n^3)$ like NJ and having identical computing times, is slightly better than NJ in all conditions, while WEIGHBOR,

also in $O(n^3)$ but requiring complex numerical calculations, is better than BIONJ. Finally, FITCH, which is in $O(n^4)$, is the best with 24 taxa, but was simply impossible to evaluate with 96 taxa (see the computing times below).

(Table 2 here)

After FASTNNI, we observe that the output topology does not depend much on the input topology. Even the deeply flawed HGT becomes close to NJ, which indicates the strength of NNIs. However, except for the true phylogeny in some cases, this output is worse than NJ. This confirms our previous results (Gascuel 2000) , which indicated that the OLS version of minimum evolution is reasonable but not excellent for phylogenetic inference. But this phenomenon is much more visible with 24 than with 96 taxa, so we expect the very fast GME+FASTNNI $O(n^2)$ combination to be equivalent to NJ with large n.

After BNNI, all initial topologies become better than NJ. With 24 taxa, the performance is equivalent to that of FITCH, while with 96 taxa the results are far better than those of WEIGHBOR, especially in the Fast condition where BNNI improves NJ by 21%, against 10% for WEIGHBOR. These results are somewhat unexpected, since BNNI has a low $O(n^2 + pn\log(n))$ average time complexity. Moreover, as explained above, we do not expect very high contrast between any inference method and NJ, due to the fact that our data sets represent a large variety of realistic trees and conditions, but do not contain extreme trees, notably concerning the maximum pairwise divergence. First experiments indicate that maximum parsimony is close to BNNI in the Slow and Moderate conditions, but worse in the Fast condition, the contrast between these methods being in the same range as those reported in Table 1 and 2. The combination of BNNI with BME or GME can thus be seen as

remarkably efficient and accurate. Moreover, regarding the true phylogeny after BNNI, it appears that little gain could be expected by further optimizing this criterion, since our simple optimization approach is already close to these upper values.

4.3 Computing Times

To compare the actual running speeds of these algorithms, we tested them on a Sun Enterprise E4500/E5500, with ten 400 MHz processors and 7 GB of memory, running the Solaris 8 operating system. Table 3 summarizes the average computational times (in hours:minutes:seconds) required by the various programs to build phylogenetic trees. The leftmost two columns were averaged over two thousand 24- and 96-taxon trees, the third over ten 1000-taxon trees, and the final column over four 4000-taxon trees. Stars indicate entries where the algorithm was deemed to be too slow to bother with that test.

(Table 3 here)

FITCH would take approximately 25 minutes to make a 96-taxon tree, which makes it impractical for real applications, because they often include bootstrapping which requires the construction of a large number of trees. As WEIGHBOR's running time increased more than 60-fold when moving from 24-taxon trees to 96-taxon trees, we judged it infeasible to run WEIGHBOR on even one 1000-taxon tree. The same reasoning held for BIONJ when we reached the 4000-taxon level.

The PAUP version of NJ started quite slowly, but was tolerably fast up through the 1000-taxon level. A closer examination of how PAUP operates leads us to believe that the program takes an inordinately long amount of time to load the data matrices, which yield a time cost function along the lines of $c_1 n^2 + c_2 n^3$, where c_1 was relatively large. In contrast, BIONJ is

twice as fast as PAUP NJ for 24-taxon trees, but this advantage disappears quickly. This result is due to the different implementations of the two algorithms: BIONJ only requires an additional $O(n^2)$ calculations beyond those used by NJ, so the implementation refinements in PAUP's NJ could be used to speed up BIONJ considerably.

The fastest programs in Table 3 were the ME, ME + FASTNNI, and HGT/FP algorithms. The HGT/FP algorithm was the only one which was able to maintain the fast speed of the ME + FASTNNI combination, but it did so at a serious cost in terms of topological accuracy. The BME and BME + BNNI combinations lagged behind their OLS-based counterparts, but were still significantly faster than NJ and BIONJ. Of particular interest is the GME + BNNI combination, which was not only markedly faster than NJ at the 96-, 1000-, and 4000-taxon levels, but also produced superior topologies. Table 3 confirms that all of our algorithms are faster than NJ, with the improvement becoming notably impressive when thousands of taxa are dealt with. Furthemore, we suspect that implementation refinements such as those used in PAUP's NJ could be used to make our algorithms still much faster.

Table 4 contains the number of NNIs performed by each of the three combinations which appear in Table 3. Not surprisingly, the largest number of NNIs was consistently required when the intial topology was made to minimize the OLS ME criterion, but NNIs were chosen to minimize the balanced ME criterion (i.e., GME + BNNI). This table shows the overall superiority of the BME tree over the GME tree, when combined with BNNI. In all of the cases considered, the average number of NNIs considered for each value of n was considerably less than n itself.

(Table 4 here)

5 Discussion

We have presented a new greedy implemention of minimum evolution tree topology searching that is considerably faster than most distance algorithms currently in use. The current most popular fast algorithm is Neighbor-Joining, an $O(n^3)$ algorithm. Our greedy ordinary least-squares minimum evolution tree construction algorithm (GME) runs at $O(n^2)$, the size of the input matrix. Although the GME tree is not quite as accurate as the NJ tree, it is a good starting point for nearest neighbor interchanges (NNIs). The combination of GME and FASTNNI, which achieves NNIs according to the ordinary least-squares criterion, also in $O(n^2)$ time, has a topological accuracy very close to that of NJ, especially with large numbers of taxa.

However, the balanced minimum evolution framework appears much more appropriate for phylogenetic inference than the ordinary least-squares version. This is likely due to the fact that it gives less weight to the topologically long distances, i.e. those containing numerous edges, while the ordinary least-squares method puts the same confidence on each distance, regardless of its length. Even when the usual and topological lengths are different, they are strongly correlated. The balanced minimum evolution framework is thus conceptually closer to weighted least-squares (Fitch and Margoliash, 1967), which is more appropriate than ordinary least-squares for evolutionary distances estimated from sequences. Studying the formal relationship between weighted least-squares and balanced minimum evolution, as well as the consistency of this latter approach, are important directions for further research.

The balanced NNI algorithm (BNNI) allows reaching outstanding performance, superior to those of NJ, BIONJ and WEIGHBOR. BNNI is an $O(n^2 + np\,\mathrm{diam}(T))$ algorithm, where

$\mathrm{diam}(T)$ is the diameter of the inferred tree, and p the number of swaps performed. With $p\,\mathrm{diam}(T) = O(n)$ for most data sets, the combination of GME and BNNI effectively gives us an $O(n^2)$ algorithm with high topological accuracy.

Acknowledgments

Special thanks go to Stéphane Guindon, who generated the data sets used in Section 4.

Appendix

1 Derivation of Tree Length Formula

In this section, we'll derive Equation (5), which we use throughout our NNI analysis. First, we need the following lemma: (Vach 1989; Gascuel 1997b)

Lemma 1.1 *If T is the OLS tree for its topology for the matrix Δ, and u is a node in T which separates the three subtrees X, Y, and Z, we then have $\Delta_{X|Y} = \Delta^T_{X|Y}$. (And by symmetry this identity also holds for $X|Z$, and $Y|Z$.)*

Now, let's reconsider Equation (5) and Figure 2(a),

$$\begin{aligned} l(T) &= \frac{1}{2}(\lambda(\Delta_{A|C} + \Delta_{B|D}) + (1-\lambda)(\Delta_{A|D} + \Delta_{B|C}) + \Delta_{A|B} + \Delta_{C|D}) \\ &\quad + l(A) + l(B) + l(C) + l(D) - \Delta_{a|A} - \Delta_{b|B} - \Delta_{c|C} - \Delta_{d|D}, \end{aligned}$$

where

$$\lambda = \frac{|B||C| + |A||D|}{(|A| + |B|)(|C| + |D|)}.$$

It is clear that

$$\begin{aligned} l(T) &= l(A) + l(B) + l(C) + l(D) + \\ &\quad l(v,a) + l(v,b) + l(w,c) + l(w,d) + l(v,w) \end{aligned} \tag{13}$$

and from Lemma 1.1,

$$\Delta_{A|B} = \Delta_{A|B}^T = \Delta_{a|A} + l(v,a) + l(v,b) + \Delta_{b|B},$$

and similarly

$$\Delta_{C|D} = \Delta_{C|D}^T = \Delta_{c|C} + l(w,c) + l(w,d) + \Delta_{d|D}.$$

Thus,

$$l(v,a) + l(v,b) + l(w,c) + l(w,d) = \Delta_{A|B} + \Delta_{C|D} - (\Delta_{a|A} + \Delta_{b|B} + \Delta_{c|C} + \Delta_{d|D}). \quad (14)$$

We substitute the right-hand side of Equation (14) and the OLS value for $l(v,w)$ from Equation (3) into Equation (13) to achieve the desired Equation (5).

2 Constant Subtree Lengths under Tree Swapping in the Balanced Scheme

First, we consider Equation (3) for internal edge length estimation when using balanced weights. Since $\lambda = 1/2$, the equation simplifies to:

$$\begin{aligned} l^{\mathcal{T}}(e) &= \frac{1}{2}(\frac{1}{2}(\Delta^{\mathcal{T}}_{A|C} + \Delta^{\mathcal{T}}_{B|D} + \Delta^{\mathcal{T}}_{A|D} + \Delta^{\mathcal{T}}_{B|C}) - (\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{C|D})) \\ &= \Delta^{\mathcal{T}}_{A\cup B|C\cup D} - \frac{1}{2}(\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{C|D}) \end{aligned} \tag{15}$$

The balanced edge length formula for external edges is essentially the same: Equation (4) simplifies to

$$l^{\mathcal{T}}(e) = \Delta^{\mathcal{T}}_{i|A\cup B} - \frac{1}{2}\Delta^{\mathcal{T}}_{A|B}. \tag{16}$$

Now let's consider a topology $\mathcal{T}$ with three subtrees A, B, and C, which meet at the vertex v. Let a, b, and c be the roots of these three subtrees, with b_1 and b_2 the children of b and c_1, c_2 the children of c, determining leaf subsets B_1, B_2, C_1 and C_2, respectively. By Equation (15),

$$\begin{aligned} l(v,b) &= \Delta^{\mathcal{T}}_{A\cup C|B} - \frac{1}{2} * (\Delta^{\mathcal{T}}_{B_1|B_2} + \Delta^{\mathcal{T}}_{A|C}) \\ &= \frac{1}{2} * (\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{B|C} - \Delta^{\mathcal{T}}_{B_1|B_2} - \Delta^{\mathcal{T}}_{A|C}). \end{aligned}$$

Similarly,

$$l(v,c) = \frac{1}{2} * (\Delta^{\mathcal{T}}_{A|C} + \Delta^{\mathcal{T}}_{B|C} - \Delta^{\mathcal{T}}_{C_1|C_2} - \Delta^{\mathcal{T}}_{A|B}),$$

and thus

$$l(v,b) + l(v,c) = \Delta^{\mathcal{T}}_{B|C} - \frac{1}{2} * (\Delta^{\mathcal{T}}_{B_1|B_2} + \Delta^{\mathcal{T}}_{C_1|C_2}). \tag{17}$$

In particular, the right-hand side of Equation (17) is completely independent of the internal structure of A. Thus if we perform a tree swap internal to A, the sum $l(v,b) + l(v,c)$ will remain constant. Analagous arguments will show the same result if either b or c is a leaf. This indicates that neither the length of a subtree (here $B \cup C$) nor its average root/leaves distance (here, from v to any leaves of $B \cup C$) is changed when a swap is performed within a disjoint subtree (here, A).

3 Details of GME Algorithm

In this section, we provide the details of the Greedy ME algorithm. Recall that we iteratively form the tree T_k with leaf set $[k] = \{1, \ldots, k\}$ from T_{k-1} by selecting an insertion point. Step 1 of the insertion process is to calculate $\Delta_{k|S}$ for every subtree S of T_{k-1}. We'll use the following notation:

- We root T_{k-1} at any taxon r, and let d be its unique direct descendant.
- Let DFS-POST (Tarjan 1983, 14–19) be the depth-first post-order of the vertices of T_{k-1}.
- Let DFS-PRE (Tarjan 1983, 14–19) be the depth-first pre-order of the vertices of T_{k-1}.
- For any non-leaf node v (or w), let v_1 and v_2 (or w_1 and w_2, respectively) denote its two children.
- For any node v of T_{k-1}, if v is a leaf, let $L(v) = \{v\}$, otherwise let $L(v)$ be the set of all nodes of T_{k-1} which are descendants of v, including v itself. Let $U(v)$ the complement to $L(v)$ among the nodes of T_{k-1}.

The details of Step 1 are:

1. To calculate $\Delta_{k|S}$ for all S which are subtrees of T_{k-1}:

   (a) Loop for w from DFS-POST - $\{r\}$. IF $w \in [k-1]$, $\Delta_{k|L(w)} = \Delta_{kw}$, ELSE

$$\Delta_{k|L(w)} = \frac{|L(w_1)|\Delta_{k|L(w_1)} + |L(w_2)|\Delta_{k|L(w_2)}}{|L(w)|}.$$

(b) Set $\Delta_{k|U(d)} = \Delta_{kr}$. Loop over w in DFS-PRE - $\{r, d\}$. Let s be the sibling of w and p be the parent of s and w. Compute

$$\Delta_{k|U(w)} = \frac{|U(p)|\Delta_{k|U(p)} + |L(s)|\Delta_{k|L(s)}}{|U(w)|}.$$

This achieves the computation of all $\Delta_{k|S}$ average distances. All other steps of the algorithm are described in Section 2.1.

4 Details of FASTNNI algorithm

In this appendix, we fill in the details for the OLS tree swapping algorithm. Recall that the first step is to compute the average distances between non-intersecting subtrees. We use the same notation as in the previous section.

1. Pre-calculating average distances:

   (a) We first calculate all the average distances of the form $\Delta_{L(v)|L(w)}$. Loop over v in DFS-POST - $\{r, d\}$; loop over w in DFS-POST including all nodes not equal to or below v in this order. For any w which is not an ancestor of v:

      i. IF $v, w \in [n]$, then $\Delta_{L(v)|L(w)} = \Delta_{vw}$.

      ii. ELSE IF $v \notin [n]$, set

$$\Delta_{L(v)|L(w)} = \frac{|L(v_1)|\Delta_{L(v_1)|L(w)} + |L(v_2)|\Delta_{L(v_2)|L(w)}}{|L(v)|}.$$

      iii. ELSE

$$\Delta_{L(v)|L(w)} = \frac{|L(w_1)|\Delta_{L(v)|L(w_1)} + |L(w_2)|\Delta_{L(v)|L(w_2)}}{|L(w)|}.$$

   (b) To calculate all $\Delta_{L(v)|U(d)}$ distances, loop v over DFS-POST - $\{r, d\}$. IF $v \in [n]$, then $\Delta_{L(v)|U(d)} = \Delta_{vr}$, ELSE

$$\Delta_{L(v)|U(d)} = \frac{|L(v_1)|\Delta_{L(v_1)|U(d)} + |L(v_2)|\Delta_{L(v_2)|U(d)}}{|L(v)|}.$$

   (c) We now calculate all distances of the form $\Delta_{L(v)|U(w)}$, where w is an ancestor of v. Loop w over DFS-PRE - $\{r, d\}$. Let s be the sibling of w and p be the parent

of s and w. Loop over v from $L(w)$ via any manner. For each v, set

$$\Delta_{L(v)|U(w)} = \frac{|L(s)|\Delta_{L(v)|L(s)} + |U(p)|\Delta_{L(v)|U(p)}}{|U(w)|}.$$

It is easily seen that every formula above uses already computed terms, and thus requires $O(1)$ time, due to the computing orchestration based on the DFS-POST and DFS-PRE orders. Each pair of vertices v, w is the subject of exactly one $\Delta_{L(v)|L(w)}, \Delta_{L(v)|U(w)}$ or $\Delta_{L(w)|U(v)}$ calculation, thus the computational time is $O(n^2)$, and we can store all of these average distances unambigously in a matrix.

Now to Steps 2 and 3.

2. Create the heap of possible swaps. Loop e over the internal edges of T via any method.

   (a) Using Equation (9) determine the change in total lengths $s_1(e), s_2(e)$ for each of the two possible tree swaps across e. Let $s(e) = \max(0, s_1(e), s_2(e))$.

   (b) Form a heap containing all the values of $s(e)$ which are positive.

3. Achieve the best swap and update the data.

   (a) Assuming the heap is non-empty, let $e = (v, w)$ be the best edge on the heap. Let A, B, C and D denote the four subtrees which meet at e, with roots a, b, c and d, as in Figure 2(a), such that a and b are incident to v, while c and d are incident to w. Suppose $B \leftrightarrow C$ is the indicated swap. Remove edges (v, b) and (w, c) from the topology and add edges (v, c) and (w, b).

(b) Loop over the subtrees S of $A \cup C$ via any manner. Compute $\Delta_{S|B\cup D}$ by averaging $\Delta_{S|B}$ and $\Delta_{S|D}$ using Equation (2). Achieve the same for $B \cup D$.

(c) Set $s(e) = 0$, and remove it from the heap. Let f range over the four edges incident to e, recalculate $s(f)$ by testing the two possible tree swaps across f, and, if $s(f) > 0$, insert $s(f)$ into the heap.

(d) If the heap is non-empty, return to Step 3(a). Otherwise, use the matrix of average distances and Equations (3) and (4) to assign branch lengths to all of the edges of the final tree.

5 Balanced Minimum Evolution Algorithms

The structure of the BME and BNNI algorithms are identical to their OLS counterparts, except in the step updating the matrix of average distances. Also, the BME algorithm requires that the full matrix of average distances be kept throughout tree building, as opposed to the OLS version, which requires only the distant-2 subtree average distances.

Consider the insertion of the node k, in tree T_{k-1}. Let X and Y be a pair of disjoint subtrees of T_{k-1}, such that k is inserted in Y. We then use Equation (11) to calculate $\Delta^{\mathcal{T}_k}_{X|Y\cup\{k\}}$. The updating step of BME is as follows:

3. Let (s,u) be the edge where k is inserted, with S and U the subtrees having roots s and u, respectively and $T_{k-1} = S \cup U$ and $S \cap U = \emptyset$.

   (a) Loop over the subtrees Z of S.

      i. Let Y be the complement of Z in T_{k-1}.

      ii. Loop over $X \subseteq Z$, and use Equation (11) to calculate $\Delta^{T_k}_{X|Y\cup\{k\}}$.

   (b) Repeat (a) with U in the place of S.

A similar adjustment is done for BNNI. Suppose we start with the topology $\mathcal{T}$, as shown in Figure 2(a), and swap subtrees B and C to form the topology $\mathcal{T}'$. Suppose x and y are nodes in $A\cup\{v\}$, with y on the path from x to v, and l edges between y and v. Let X and Y be the nonintersectiong subtrees with roots x and y respectively. We allow for the possibility that $l = 0$, in which case we choose $X \subseteq A$. (See Figure 5.) Our updating equation is:

$$\Delta^{\mathcal{T}'}_{X|Y} = \Delta^{\mathcal{T}}_{X|Y} - 2^{-(l+2)}\Delta^{\mathcal{T}}_{X|B} + 2^{-(l+2)}\Delta^{\mathcal{T}}_{X|C} \tag{18}$$

The modified step in BNNI is:

3. Update the matrix of average distances

   (a) Loop over pairs of nodes x, y where x is any node in the subtree A and y is any node in the path from x to v. Let X and Y denote the non-intersecting subtrees (see Figure 5) with roots x and y, respectively. Use Equation (18) to calculate $\Delta^{\mathcal{T}'}_{X|Y}$. Repeat the same steps for all analogous pairs x, y in each of B, C, and D.

   (b) For any subtree $X \subset A \cup C$, compute

$$\Delta^{\mathcal{T}'}_{X|B\cup D} = \frac{1}{2}(\Delta^{\mathcal{T}}_{X|B} + \Delta^{\mathcal{T}}_{X|D}).$$

   Perform analogous computations to compute $\Delta^{\mathcal{T}'}_{Y|A\cup C}$ for all subtrees $Y \subset B \cup D$.

   (c) Calculate

$$\Delta^{\mathcal{T}'}_{A\cup C|B\cup D} = \frac{1}{4}(\Delta^{\mathcal{T}}_{A|B} + \Delta^{\mathcal{T}}_{A|D} + \Delta^{\mathcal{T}}_{C|B} + \Delta^{\mathcal{T}}_{C|D})$$

References

Aldous, D. J. 2001. Stochastic models and descriptive statistics for phylogenetic trees from Yule to today. *Statist. Sci.*, 16, 23–34.

Bruno, W. J., Socci, N. D., and Halpern, A. L. 2000. Weighted neighbor joining: A likelihood-based approach to distance-based phylogeny reconstruction. *Mol. Biol. Evol.*, 17, 189–197.

Bryant, D. and Waddell, P. 1998. Rapid evaluation of least-squares and minimum-evolution criteria on phylogenetic trees. *Mol. Biol. Evol.*, 15, 1346–1359.

Bulmer, M. 1991. Use of the method of generalized least squares in reconstructing phylogenies from sequence data. *Mol. Biol. Evol.*, 8, 868–883.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. 2000. *Introduction to algorithms*. MIT Press, Cambridge, MA.

Csűrós, M. 2002. Fast recovery of evolutionary trees with thousands of nodes. *J. Comp. Biol.*, 9, 277–297.

Denis, F. and Gascuel, O. 2002. On the consistency of the minimum evolution principle of phylogenetic inference. *Discr. Appl. Math.*, In press.

Erdös, P. L., Steel, M., Székély, L., and Warnow, T. 1999. A few logs suffice to build (almost) all trees: Part II. *Theo. Comp. Sci.*, 221, 77–118.

Felsenstein, J. 1989. PHYLIP — Phylogeny Inference Package (Version 3.2). *Cladistics*, 5, 164–166.

Felsenstein, J. 1997. An alternating least-squares approach to inferring phylogenies from pairwise distances. *Syst. Biol.*, 46, 101–111.

Fitch, W. M. and Margoliash, E. 1967. Construction of phylogenetic trees. *Science*, 155, 279–284.

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

Gascuel, O. 1997b. Concerning the NJ algorithm and its unweighted version, UNJ. Pages 149–170, *In* Mirkin, B., McMorris, F., Roberts, F., and Rzetsky, A., eds. *Mathematical Hierarchies and Biology*. American Mathematical Society, Providence.

Gascuel, O. 2000. On the optimization principle in phylogenetic analysis and the minimum-evolution criterion. *Mol. Biol. Evol.*, 17, 401–405.

Gascuel, O., Bryant, D., and Denis, F. 2001. Strengths and limitations of the minimum evolution principle. *Syst. Biol.*, 50, 621–627.

Guindon, S. and Gascuel, O. 2002. Efficient biased estimation of evolutionary distances when substitution rates vary across sites. *Mol. Biol. Evol.*, 19, 534–543.

Harding, E. 1971. The probabilities of rooted tree-shapes generated by random bifurcation. *Adv. Appl. Probab.*, 3, 44–77.

Kidd, K. and Sgaramella-Zonta, L. 1971. Phylogenetic analysis: concepts and methods. *Am. J. Human Genet.*, 23, 235–252.

Kimura, M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16, 111,120.

Kuhner, M. K. and Felsenstein, J. 1994. A simulation comparison of phylogeny algorithms under equal and unequal rates. *Mol. Biol. Evol.*, 11, 459–468.

McKenzie, A. and Steel, M. 2000. Distributions of cherries for two models of trees. *Math. Biosci.*, 164, 81–92.

Nei, M. and Jin, L. 1989. Variances of the average numbers of nucleotide substitutions within and between populations. *Mol. Biol. Evol.*, 6, 290–300.

Pauplin, Y. 2000. Direct calculation of a tree length using a distance matrix. *J. Mol. Evol.*, 51, 41–47.

Rambaut, A. and Grassly, N. C. 1997. Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput. Appl. Biosci.*, 13, 235–238.

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

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

Sneath, P. H. A. and Sokal, R. R. 1973. *Numerical Taxonomy*, Pages 230–234. W.K. Freeman and Company, San Francisco.

Swofford, D. 1996. PAUP—Phylogenetic Analysis Using Parsimony (and other methods), Version 4.0.

Swofford, D. L., Olsen, G. J., Waddell, P. J., and Hillis, D. M. 1996. Phylogenetic inference. chapter 11, Pages 407–514, *In* Hillis, D., Moritz, C., and Mable, B., eds. *Molecular Systematics*. Sinauer, Sunderland, MA.

Tarjan, R. E. 1983. *Data Structures and Network Algorithms.* SIAM, Philadelphia.

Vach, W. 1989. Least squares approximation of addititve trees. Pages 230–238, *In* Opitz, O., ed. *Conceptual and numerical analysis of data.* Springer-Verlag, Berlin.

Yule, G. 1925. A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis. *Philos. Trans. Roy. Soc. London Ser. B, Biological Sciences*, 213, 21–87.

Table 1: Topological accuracy for 24-taxon trees at various rates of evolution

slow rate	w/o NNIs		+ FASTNNI		+ BNNI	
True Tree			.109	-1.6%	.104	-6.2 %
FITCH	.109	-1.9%	.113	2.0%	.107	-3.4%
WEIGHBOR	.109	-1.8%	.112	1.7%	.107	-3.0%
BIONJ	.111	-0.3%	.113	2.0%	.107	-3.6%
NJ	.111	0%	.113	2.0%	.107	-3.5%
BME	.118	7.1%	.113	1.9%	.107	-2.8%
GME	.122	10%	.113	2.1%	.107	-3.4%
HGT/FP	.334	202%	.112	1.1%	.107	-2.9%
moderate rate	w/o NNIs		+ FASTNNI		+ BNNI	
True Tree			.092	3.7%	.083	-5.8%
FITCH	.085	-4.9%	.094	6.0%	.085	-4.0%
WEIGHBOR	.085	-4.3%	.094	6.2%	.085	-4.0%
BIONJ	.087	-2.0%	.094	6.5%	.085	-4.2%
NJ	.088	0%	.094	6.6%	.085	-4.0%
BME	.100	13%	.094	6.3%	.084	-4.9%
GME	.107	21%	.095	7.1%	.084	-4.8%
HGT/FP	.326	268%	.095	7.5%	.088	-0.2%
fast rate	w/o NNIs		+ FASTNNI		+ BNNI	
True Tree			.088	6.5%	.076	-8.3%
FITCH	.076	-7.8%	.090	8.8%	.077	-7.0%
WEIGHBOR	.077	-6.8%	.089	8.2%	.077	-6.9%
BIONJ	.079	-3.6%	.090	9.0%	.077	-6.6%
NJ	.082	0%	.090	9.1%	.077	-6.9%
BME	.098	19%	.090	9.1%	.076	-7.1%
GME	.105	28%	.090	9.8%	.076	-7.6%
HGT/FP	.329	300%	.090	9.8%	.083	0.8%

Note: The first number indicates the average topological distance between the inferred tree and the true phylogeny. The second number (in parentheses) provides the relative difference in topological distance between the method considered and NJ; the more negative this value, the better the method was relative to NJ.

Table 2: Topological accuracy for 96-taxon trees at various rates of evolution

slow rate	w/o NNIs		+ FASTNNI		+ BNNI	
True Tree			.172	-5.6%	.167	-8.8%
WEIGHBOR	.178	-2.5%	.181	-0.7%	.173	-5.2%
BIONJ	.180	-0.9%	.182	-0.3%	.173	-5.1%
NJ	.183	0%	.182	-0.2%	.173	-5.2%
BME	.186	1.9%	.181	-0.6%	.173	-5.3%
GME	.199	8.8%	.183	0.3%	.173	-5.3%
HGT/FP	.512	185%	.185	1.5%	.175	-4.3%
moderate rate	w/o NNIs		+ FASTNNI		+ BNNI	
True Tree			.132	-3.0%	.115	-15.4%
WEIGHBOR	.129	-5.4%	.137	0.5%	.118	-13.0%
BIONJ	.134	-1.9%	.138	1.3%	.118	-13.0%
NJ	.136	0%	.139	1.8%	.119	-12.9%
BME	.137	1.0%	.138	1.1%	.118	-13.2%
GME	.158	16%	.140	2.7%	.118	-13.2%
HGT/FP	.480	253%	.143	5.2%	.123	-.9.3%
fast rate	w/o NNIs		+ FASTNNI		+ BNNI	
True Tree			.115	0.6%	.088	-23.4%
WEIGHBOR	.103	-10%	.119	3.8%	.091	-21.0%
BIONJ	.112	-2.5%	.121	5.1%	.090	-21.7%
NJ	.115	0%	.121	5.5%	.090	-21.3%
BME	.117	1.8%	.120	4.4%	.090	-21.4%
GME	.144	25%	.122	6.3%	.091	-21.1%
HGT/FP	.465	306%	.126	9.4%	.098	-14.7%

Note: see note to Table 1.

Table 3: Average computational time (HH:MM:SS) for reconstruction algorithms

Algorithm	24 taxa	96 taxa	1000 taxa	4000 taxa
GME	.02522	.07088	8.366	3:11.01
GME + FASTNNI	.02590	.08268	9.855	3:58.79
GME + BNNI	.02625	.08416	11.339	6:02.10
HGT/FP	.02518	.13491	13.808	3:33.14
BME	.02534	.08475	19.231	12:14.99
BME + BNNI	.02557	.08809	19.784	12:47.45
NJ	.06304	.16278	21.250	20:55.89
BIONJ	.06528	.16287	21.440	20:55.64
WEIGHBOR	.42439	26.88176	*****	*****
FITCH	4.37446	*****	*****	*****

Table 4: Average number of NNIs performed

Algorithm	24 taxa	96 taxa	1000 taxa	4000 taxa
GME + FASTNNI	1.244	8.446	44.9	336.50
GME + BNNI	1.446	11.177	59.1	343.75
BME + BNNI	1.070	6.933	29.1	116.25

Figure 1: Subtree B is composed of the two sibling trees B_1 and B_2

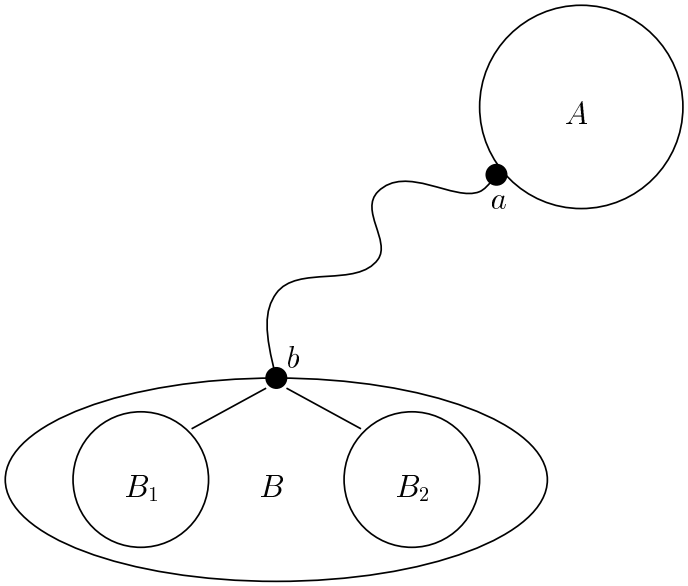

Figure 2: Corner subtrees used to estimate the length of e, (a) for e an internal edge; (b) for e an external edge

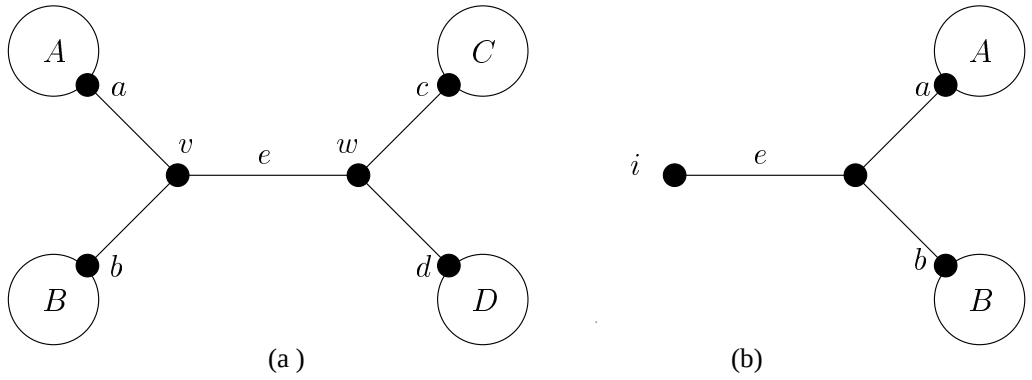

Figure 3: T' is obtained from T by swapping A and k

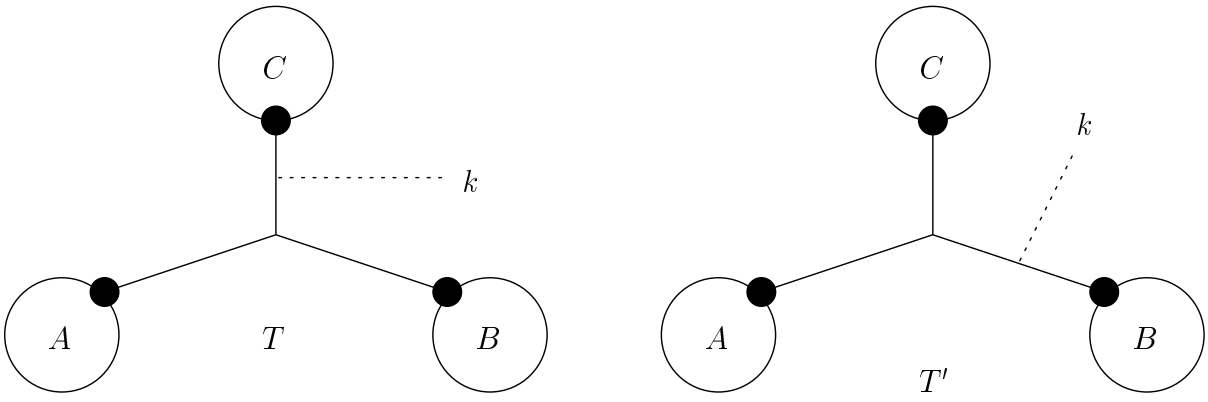

Figure 4: Calculating balanced average $\Delta^{\mathcal{T}}_{X|Y}$ when k is inserted into Y

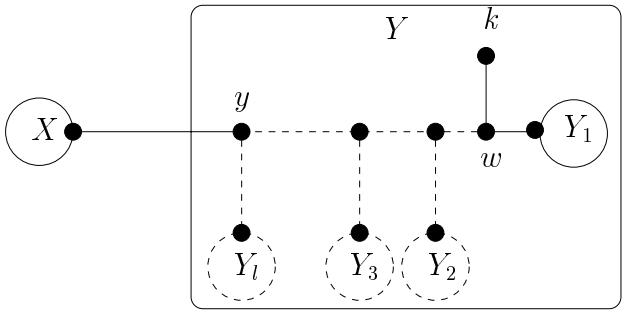

Figure 5: Recalculating balanced average $\Delta^{\mathcal{T}}_{X|Y}$ after $B \leftrightarrow C$ tree swap