Rapid Evaluation of Least-Squares and Minimum-Evolution Criteria on Phylogenetic Trees

David Bryant and Peter Waddell

Biomathematics Research Centre, University of Canterbury, Christchurch, New Zealand

We present fast new algorithms for evaluating trees with respect to least squares and minimum evolution (ME), the most commonly used criteria for inferring phylogenetic trees from distance data. The new algorithms include an optimal $O(N^2)$ time algorithm for calculating the edge (branch or internode) lengths on a tree according to ordinary or unweighted least squares (OLS); an $O(N^3)$ time algorithm for edge lengths under weighted least squares (WLS) including the Fitch-Margoliash method; and an optimal $O(N^4)$ time algorithm for generalized least-squares (GLS) edge lengths (where N is the number of taxa in the tree). The ME criterion is based on the sum of edge lengths. Consequently, the edge lengths algorithms presented here lead directly to $O(N^2)$, $O(N^3)$, and $O(N^4)$ time algorithms for ME under OLS, WLS, and GLS, respectively. All of these algorithms are as fast as or faster than any of those previously published, and the algorithms for OLS and GLS are the fastest possible (with respect to order of computational complexity). A major advantage of our new methods is that they are as well adapted to multifurcating trees as they are to binary trees. An optimal algorithm for determining path lengths from a tree with given edge lengths is also developed. This leads to an optimal $O(N^2)$ algorithm for OLS sums of squares evaluation and corresponding $O(N^3)$ and $O(N^4)$ time algorithms for WLS and GLS sums of squares, respectively. The GLS algorithm is time-optimal if the covariance matrix is already inverted. The speed of each algorithm is assessed analytically—the speed increases we calculate are confirmed by the dramatic speed increases resulting from their implementation in PAUP* 4.0. The new algorithms enable far more extensive tree searches and statistical evaluations (e.g., bootstrap, parametric bootstrap, or jackknife) in the same amount of time. Hopefully, the fast algorithms for WLS and GLS will encourage the use of these criteria for evaluating trees and their edge lengths (e.g., for approximate divergence time estimates), since they should be more statistically efficient than OLS.

Introduction

Distance-based methods of evolutionary tree reconstruction are presently the default methods for the analysis of many data sets. They allow the implementation of a wide range of model-based corrections, including the very general LogDet transformation, which compensates for variable base composition (Barry and Hartigan 1987; Lake 1994; Lockhart et al. 1994), and the CTR compensating for unequal site rates (Waddell and Steel 1997). Distance-based analyses are considerably faster than other model-based criteria, such as maximum likelihood (ML), on sequences (Felsenstein 1981; Swofford et al. 1996). Furthermore, some data sets, such as those for DNA hybridization experiments, originate only as distances. Consequently, distance-based tree analyses appear in most papers on, or involving, phylogenetic evaluation.

Distance-based clustering algorithms such as UPGMA and neighbor joining have been very popular (e.g., see Saitou and Nei 1987; Swofford et al. 1996). However, it is generally more desirable to optimize the fit of data to an assumed model than to simply apply an algorithm (see Swofford et al. 1996). Examples of fit criteria applied to trees include ordinary, or unweighted, least squares (OLS); weighted least squares (WLS); and generalized least squares (GLS). This last criterion is closely related to the ML tree estimation of Felsenstein (1981), assuming that the only data available are the distances (for further discussion, see Felsenstein 1988; Waddell, Lewis, and Swofford 1998).

Another set of optimality criteria emerges when least squares is used to estimate the edge lengths of each tree, but the optimal tree is identified as that with the minimum sum of edge (internode or branch) lengths. These are called minimum-evolution (ME) methods and have a long history in phylogenetics (e.g., Kidd and Sgaramella-Zonta 1971; Saitou and Imanishi 1989; Rzhetsky and Nei 1992*a*; Swofford et al. 1996; Waddell, Lewis, and Swofford 1998). We denote these methods by ‘‘ME’’ followed by the optimality criterion used to estimate edge lengths; e.g., ME(OLS) is the ME method studied by Rzhetsky and Nei (1992*a*, 1992*b*, 1993), and included in PAUP* 4.0 (Swofford 1997).

Tree searching requires the evaluation of many trees, the total number of which grows exponentially with respect to the number of taxa N (Schröder 1870; Cavalli-Sforza and Edwards 1967). This makes speedy evaluation of the selection criteria for each tree highly desirable. The evaluation algorithms presented here are as fast as or faster than any previously published. In some cases, the speed increase is dramatic. When the OLS algorithm was implemented in PAUP and applied to a data set of 125 taxa, the new algorithm executed 75 times faster than the previous method (D. L. Swofford, personal communication; see *Discussion* for further details).

The speed of an algorithm is usually described in terms of order complexity $O(\ )$. This notation is standard in the biological literature (see, e.g., Rzhetsky and Nei 1993; Felsenstein 1997); however, those unfamiliar with the concepts can consult the introductory material in Day (1987) or Penrose (1989, pp. 140–145). Roughly,

Key words: least-squares method, minimum evolution, phylogenetic inference, tree statistics, algorithms.

Address for correspondence and reprints: David Bryant, Centre de recherches mathematiques, Université de Montréal, C.P. 6128, Succursale Centre Ville, Montreal, Quebec H3C 3J7, Canada. E-mail: bryant@CRM.UMontreal.ca.

Mol. Biol. Evol. 15(10):1346–1359. 1998
© 1998 by the Society for Molecular Biology and Evolution. ISSN: 0737-4038

if $f(N)$ is a function of N, for example, $f(N) = N^3$, then an algorithm takes $O(f(N))$ time if it takes at most $Kf(N)$ time for some fixed constant K. For example, we show that we can now estimate OLS edge lengths on a binary tree using less than $3N^2/2 + 64N - 126$ operations—therefore, the algorithm takes $O(N^2)$ time. In contrast, the edge length algorithm of Rzhetsky and Nei (1993) takes $O(N^3)$ time.

By ''time-optimal'' we mean that no algorithm can have a lower order of complexity. Fitch's (1971) algorithm was the first time-optimal algorithm for parsimony. The algorithms we describe here are the first time-optimal algorithms for least-squares and ME tree evaluation on both binary and multifurcating trees. Any future method will take, at best, $O(N^2)$ time to evaluate least-squares and ME criteria. Of course, the order notation $O(\ )$ can obscure ''hidden constants'' and massive overheads that make algorithms unattractive for realistic data sets. However, the algorithms presented here are extremely efficient, with minimal overheads. Below, these claims are proven by determining analytic upper bounds on the actual number of arithmetic operations performed by each algorithm. In many ways, this is a better measure than simply running simulations: it avoids the inaccuracies caused by selective choice of data, as well as those due to differing machine architectures and compilation efficiencies.

Fast algorithms enable the evaluation of more trees, but they also enable the analysis of larger data sets. Analysis of large numbers of sequences does not merely provide a more comprehensive evolutionary history; research indicates that larger taxa sets can lead to improved accuracy. Biases caused by long edges in trees can lead to inconsistency of distance-based methods (e.g., Jin and Nei 1990; Lockhart et al. 1996), just as with parsimony (Felsenstein 1978; Swofford et al. 1996). Trees for larger sets of taxa may have these longer edges broken up and thus are much less susceptible to biases due to the process of evolution not matching the assumed model (e.g., Swofford et al. 1996).

Presently, the most popular distance-based criteria are OLS, Fitch Margoliash least squares (FM, a form of WLS), and ME(OLS), available in packages such as Phylip 3.5 (Felsenstein 1993) and PAUP*4.0 (Swofford 1997). Some of the criteria evaluation algorithms introduced here were added to PAUP 4.0 during the time this article was under review (see *Discussion*). We hope that the fast algorithms in this paper will encourage the use of WLS and GLS, criteria which are predicted to be more accurate for tree estimation (e.g., Bulmer 1991; Kuhner and Felsenstein 1994; Swofford et al. 1996; Waddell, Lewis, and Swofford 1998). An additional advantage of WLS and GLS estimation is that they give more reliable estimates of a tree's edge lengths than OLS (e.g., Bulmer 1991; Kuhner and Felsenstein 1994). This can be useful when inferring approximate relative divergence times or determining which genes have evolved faster.

Methods and Definitions

Least-Squares Criteria

We begin with a number of definitions, all of which are standard. Throughout this paper, we adopt the vector notation used by Rzhetsky and Nei (1992*a*) and others. Let L be the set of taxa and let N be the number of taxa in L. A **distance** on L, possibly given by an evolutionary distance, is represented by a column vector with $\frac{1}{2}N(N - 1)$ entries. We use $\mathbf{d}$ to denote a general distance and $\mathbf{p}$ to denote the taxon-to-taxon distances in a tree. Each entry in either vector corresponds to a different pair of taxa. For example, when $L = \{1, 2, 3, 4\}$ we have $\mathbf{d} = (\mathbf{d}_{12}, \mathbf{d}_{13}, \mathbf{d}_{14}, \mathbf{d}_{23}, \mathbf{d}_{24}, \mathbf{d}_{34})^t$, where the superscript t denotes transpose.

The shape of a tree T can be encoded using a matrix of zeros and ones called a **topological matrix,** here denoted by the matrix $\mathbf{A}$. The columns of $\mathbf{A}$ correspond to edges of T, and the rows of $\mathbf{A}$ correspond to pairs of taxa in L. If the path connecting two taxa i and j passes though edge k, then we put a 1 in row ij, column k; otherwise, we put a 0 there. Using this notation, we can write the relationship between edge lengths and taxon-to-taxon distances:

$$\mathbf{p} = \mathbf{Ab}. \qquad (1)$$

Here, $\mathbf{p}$ is the column vector for the taxon-to-taxon distances, $\mathbf{A}$ is the topological matrix, and $\mathbf{b}$ is the column vector for the edge lengths.

Note that, following Penny, Hendy, and Steel (1992), we refer to edges in a tree rather than branches of a tree. The term ''branch'' is ambiguous because it can also refer to an entire subtree, as in Darwin's *Origin of the Species* (see introduction to Waddell and Steel 1997).

A **split $A|B$** is a partition of a set of taxa into two parts, A and B. Splits are especially useful when working with phylogenetic trees. Each edge e of a tree corresponds to a unique split, because removing that edge divides the tree and, consequently, the taxon set of the tree, into two parts. This split is called the **split corresponding to the edge *e*.** The set of splits corresponding to the edges of a tree T is called the **splits of *T*.** A tree can be constructed in linear time from its set of splits (Gusfield 1991).

Any given split has an associated **split metric**, a distance on L where two taxa are distance 1 apart if they are on different sides of the split and distance 0 apart if they are on the same side of the split. The columns of the topological matrix $\mathbf{A}$ of a tree are exactly the split metrics associated with splits in the tree. Column k of the matrix is the split metric for the split corresponding to edge e_k.

OLS

The problem: we are given an unrooted tree T with taxon set L, and we want to assign lengths to the edges of T so that the taxon-to-taxon distance of T most closely approximates a given distance $\mathbf{d}$, the measure of misfit being the sum-of-squares distance. In terms of our vector notation, we want to find $\mathbf{b}$ that minimizes the sum of squared differences

$$\mathrm{SS(OLS)} = (\mathbf{Ab} - \mathbf{d})^t(\mathbf{Ab} - \mathbf{d}), \tag{2}$$

where the elements of **b** may take on any real value, including zero or negative values.

One of the earliest references to this problem is in Cavalli-Sforza and Edwards (1967). Straightforward projection theory gives the solution

$$b = (\mathbf{A}^t\mathbf{A})^{-1}\mathbf{A}^t\mathbf{d} \tag{3}$$

(Cavalli-Sforza and Edwards 1967), but direct application of this formula leads to an inefficient algorithm with complexity $O(N^4)$. Sattath and Tversky (1977) propose a more efficient method, although they leave out the details. It seems reasonable to conclude from their description that the method they used is the same as the $O(N^3)$ method described explicitly by Vach (1989) (see also Vach and Degens 1991). A different approach was taken by Rzhetsky and Nei (1993), whose formulae for edge lengths also give an $O(N^3)$ time algorithm, although it is only applicable to binary trees. The $O(N^2)$ time algorithms for edge lengths presented here were first published in Bryant (1997). Note that Gascuel (1997) has also (and independently) developed an $O(N^2)$ algorithm for OLS edge lengths, although it is restricted to binary trees only.

WLS

The WLS method for calculating edge lengths involves the minimisation of

$$\mathrm{SS(WLS)} = (\mathbf{Ab} - \mathbf{d})^t\mathbf{W}(\mathbf{Ab} - \mathbf{d}), \tag{4}$$

where **W** is a given diagonal matrix with strictly positive entries on the diagonal, while **b** can have negative entries (e.g., Bulmer 1991; Swofford et al. 1996). The minimum is given directly by the formula

$$\mathbf{b} = (\mathbf{A}^t\mathbf{WA})^{-1}\mathbf{A}^t\mathbf{Wd}. \tag{5}$$

If we use standard matrix multiplication, this vector can be calculated in $O(N^4)$ time.

Felsenstein (1997) has recently published an algorithm for calculating edge lengths under WLS. Felsenstein's algorithm is iterative: it begins with a rough approximation of the optimal edge lengths and then progressively improves this approximation with each pass of the algorithm. Each iteration takes $O(N^3)$ time (since $O(N^2)$ time is required for each internal vertex), and there is no proven bound on the number of iterations required to achieve an acceptable solution. However, it is reported to work quite well in practice (Felsenstein 1997; D. L. Swofford, personal communication).

GLS

The function to be minimized when using GLS is

$$\mathrm{SS(GLS)} = (\mathbf{Ab} - \mathbf{d})^t\mathbf{V}^{-1}(\mathbf{Ab} - \mathbf{d}), \tag{6}$$

where **V**, and hence $\mathbf{V}^{-1}$, is a strictly positive definite symmetric matrix and, as before, **b** can have negative entries (Bulmer 1991; Swofford et al. 1996). The direct solution is

$$b = (\mathbf{A}^t\mathbf{V}^{-1}\mathbf{A})^{-1}\mathbf{A}^t\,\mathbf{V}^{-1}\mathbf{d}. \tag{7}$$

Here, **V** is an $O(N^2) \times O(N^2)$ matrix so calculating $\mathbf{V}^{-1}$ takes $O(N^6)$ time. It must be remembered that this calculation is performed only once for each data set, whereas the edge length calculation is repeated for every tree assessed. Therefore, we assume that this inverse has been computed during preprocessing, before the execution of the edge-lengths algorithm. Even without calculating the inverse, the above formula for **b** still takes $O(N^5)$ time to compute. Below, this bound is improved to $O(N^4)$.

The variance-covariance matrix of edge length estimates under GLS is given by Agresti (1990, pp. 460–462) (for any multivariate model), Hasegawa et al. (1985), and Bulmer (1991) (for trees) as

$$\mathrm{var}(\mathbf{b}) = (\mathbf{A}^t\mathbf{V}^{-1}\mathbf{A})^{-1}. \tag{8}$$

This formula takes $O(N^5)$ time using standard matrix multiplication. Below, this bound is improved to $O(N^4)$ time.

Results

The main results of this paper are as follows:

1. An algorithm to calculate $\mathbf{A}^t\mathbf{d}$ in minimal time.
2. Application of this algorithm to the calculation of OLS, WLS, and GLS edge lengths, giving a fast algorithm for WLS and a time-optimal algorithm for GLS.
3. The description of a very fast time-optimal $O(N^2)$ time algorithm for calculating OLS edge lengths, leading directly to a time-optimal algorithm for evaluating the ME(OLS) criterion on a tree.
4. A time-optimal algorithm for calculating path lengths in a tree when edge lengths are given.
5. Application of the above algorithm to give time optimal algorithms for the evaluation of least squares on trees.
6. Upper bounds on the numbers of arithmetic operations required by these algorithms and a comparison with the OLS edge-lengths algorithm of Rzhetsky and Nei (1993).

Calculating $\mathbf{A}^t\mathbf{d}$ in Minimal Time

This paper describes several new techniques and "tricks" for speeding up tree criteria evaluation. The first trick is a method for multiplying a vector by the transpose of the topological matrix of a tree in a fraction of the time taken by standard matrix multiplication. Whereas standard multiplication takes at least $\frac{1}{2}N(N - 1) \times (2N - 3)$ operations, this new method takes at most $\frac{1}{2}N(3N - 1)$ operations. Thus, the new method is faster for all $N \geq 3$, over 10 times faster for $N = 20$ and over 65 times faster for $N = 100$.

This fast multiplication result is so useful that it forms a part of every algorithm in this paper. Examine, for example, the formulae for OLS, WLS, and GLS edge lengths (eqs. 3, 5, and 7) and count the number of times a vector (or matrix) is multiplied on the left by $\mathbf{A}^t$, the transpose of the topological matrix.

The columns of **A** are the split metrics $\delta_1, \delta_2, \ldots, \delta_K$ corresponding to edges in the tree, so the elements of $\mathbf{A}^t\mathbf{d}$ are the quantities $\delta_1^t\mathbf{d}, \delta_2^t\mathbf{d}, \ldots, \delta_K^t\mathbf{d}$.

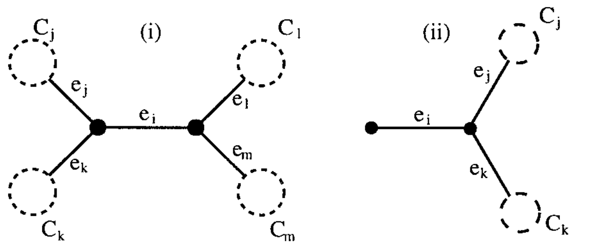

FIG. 1.—Any internal edge e_i of a binary tree can be drawn in the form of *i:* the subtrees branching off e_i are represented by dotted circles. An external edge e_i of a binary tree can be drawn in the form of *ii:* the subtrees branching off e_i are represented by dotted circles.

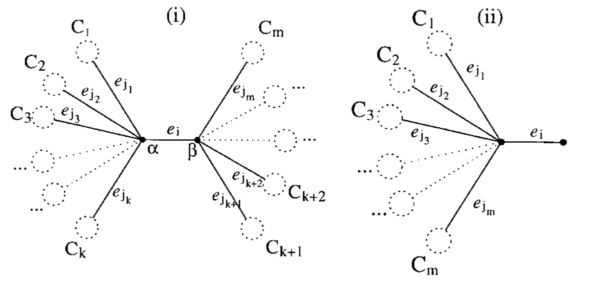

FIG. 2.—Generic cases for calculating edge lengths. *i,* Internal edge in a nonbinary tree. *ii,* External edge in a nonbinary tree.

The first step in the fast method is the calculation of $\delta_i^t\mathbf{d}$ for all of the split metrics δ_i that correspond to external edges of T. If e_i is an external edge adjacent to, say, taxon x, then

$$\delta_i^t\mathbf{d} = \sum_{y\in L-x}\mathbf{d}_{xy}. \tag{9}$$

Now for the internal edges. We consider binary trees first. The increase in speed relies on the following relationship between $\delta_i^t\mathbf{d}$ and $\delta_j^t\mathbf{d}$ for adjacent edges e_i and e_j.

Theorem 1. *Let e_i be an internal edge of a binary tree, and let e_j, e_k be edges adjacent to the same endpoint of e_i. Let C_j, C_k, and C_i be corresponding clusters* (*see fig. 1_i, noting that $C_i = C_l \cup C_m$*). *Then,*

$$\delta_i^t\mathbf{d} = \delta_j^t\mathbf{d} + \delta_k^t\mathbf{d} - 2\sum_{x\in C_j,y\in C_k}\mathbf{d}_{xy}. \tag{10}$$

Proof

We first write out $\delta_i^t\mathbf{d}$, $\delta_j^t\mathbf{d}$, and $\delta_k^t\mathbf{d}$, remembering that C_j, C_k, and C_i are disjoint:

$$\delta_i^t\mathbf{d} = \sum_{x\in C_i,y\in L-C_i}\mathbf{d}_{xy} = \sum_{x\in C_i,y\in C_j}\mathbf{d}_{xy} + \sum_{x\in C_i,y\in C_k}\mathbf{d}_{xy} \tag{11}$$

$$\delta_j^t\mathbf{d} = \sum_{x\in C_j,y\in L-C_j}\mathbf{d}_{xy} = \sum_{x\in C_j,y\in C_i}\mathbf{d}_{xy} + \sum_{x\in C_j,y\in C_k}\mathbf{d}_{xy} \tag{12}$$

$$\delta_k^t\mathbf{d} = \sum_{x\in C_k,y\in L-C_k}\mathbf{d}_{xy} = \sum_{x\in C_k,y\in C_i}\mathbf{d}_{xy} + \sum_{x\in C_k,y\in C_j}\mathbf{d}_{xy}. \tag{13}$$

We simply substitute equations (11), (12), and (13) into (10) to prove the theorem.□

The proof can easily (but tediously) be generalized to give an analogous result for multifurcating trees:

Theorem 2. *Let e_i be an internal edge of a tree T, choose either endpoint of e_i, and let $j_1, \ldots, j_k$ be the indices of all the edges adjacent to e_i at this endpoint. Let $C_{j1}, C_{j2}, \ldots, C_{jk}$ and C_i be the corresponding clusters* (*see fig.* 2). *Then,*

$$\delta_i^t\mathbf{d} = \sum_{l=1}^{m}\delta_{j_l}^t\mathbf{d} - 2\left(\sum_{1\le p<q\le k}\ \sum_{x\in C_{j_p},y\in C_{j_q}}\mathbf{d}_{xy}\right). \tag{14}$$

Of course, theorem 1 is just a special case of theorem 2.

To use equation (10) or (14) to calculate $\delta_i^t\mathbf{d}$ for an edge e_i, we need only have calculated $\delta_j^t\mathbf{d}$ for all edges e_j adjacent to one endpoint of e_i. Thus, if we start by calculating $\delta_i^t\mathbf{d}$ for all external edges, we can work inward, repeatedly applying equation (10) or (14), until all values $\delta_i^t\mathbf{d}$ have been calculated.

To formalize this method, we provide a ready-to-implement algorithm, FASTMTM (short for fast multiplication by topological matrix). It is written to handle multifurcating trees but can easily be simplified for the binary case. Input is a tree T (as a list of vertices and edges) and a distance $\mathbf{d}$.

```
Algorithm FASTMTM($T$, $\mathbf{d}$)
1. For each external edge $e_i$, put
       $\delta_i^t\mathbf{d} \leftarrow \sum_{y\in L-\{x\}}\mathbf{d}_{xy}$
   where $x$ is the taxon attached to $e_i$.
2. Choose an arbitrary internal edge to
   be first in a list of edges. Add the
   edges adjacent to this edge to the end
   of the list, then the unlisted edges
   adjacent to these edges, and so on un-
   til all the internal edges are in the
   list.
3. Working backward through the list, cal-
   culate $\delta_i^t\mathbf{d}$ for each edge using equation
   (14). Note that all the edges adjacent
   to one of the endpoints will already
   have been calculated.
4. Output the vector $\mathbf{A}^t\mathbf{d}$ with entries $\delta_i^t\mathbf{d}$.
end.
```

At first glance, the algorithm appears to take $O(N^3)$ time. This is not the case:

Theorem 3. *The algorithm* FASTMTM *takes at most $3N^2/2 - N/2$ operations. The amount of memory used, in addition to the memory required to store the vector* $\mathbf{d}$, *is $O(N)$.*

Proof is in appendix B.1.

To illustrate the use of the algorithm FASTMTM and other algorithms in this paper, we estimate edge lengths of the tree T in figure 3 with respect to the distance matrix $\mathbf{d}$ in the same figure.

Consider first the calculation of $\delta_1^t\mathbf{d}$ for external edge e_1. From equation (9), we have

$$\delta_1^t\mathbf{d} = \mathbf{d}_{AB} + \mathbf{d}_{AB} + \cdots + \mathbf{d}_{AH} \tag{15}$$

$$= 80. \tag{16}$$

A	0							
B	7	0						
C	5	5	0					
D	11	8	6	0				
E	15	13	10	4	0			
F	15	11	9	11	12	0		
G	17	14	12	6	14	8	0	
H	10	12	9	11	12	18	19	0
	A	B	C	D	E	F	G	H

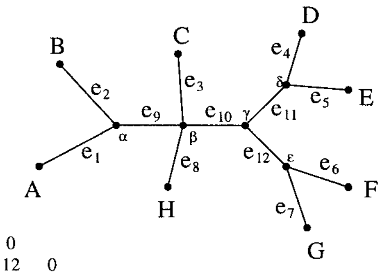

FIG. 3.—Example tree T and test distance matrix $\mathbf{d}$ for taxon set $\{A,B,C,D,E,F,G,H\}$. Internal vertices are labeled with greek characters α, β, γ, δ, and ϵ for future reference.

Similarly, $\delta_2^t\mathbf{d} = 70$, $\delta_3^t\mathbf{d} = 56$, $\delta_4^t\mathbf{d} = 57$, $\delta_5^t\mathbf{d} = 80$, $\delta_6^t\mathbf{d} = 84$, $\delta_7^t\mathbf{d} = 90$, $\delta_8^t\mathbf{d} = 91$.

The list constructed in step 2 is reflected in the numbering of the edges of T. We started with edge e_{12}, then added e_{11}, then e_{10} and e_9.

We can use equation (10) to calculate $\delta_9^t\mathbf{d}$. Put $i = 9$, $j = 1$, and $k = 2$ such that $C_j = \{A\}$ and $C_k = \{B\}$. Then,

$$\delta_9^t\mathbf{d} = \delta_1^t\mathbf{d} + \delta_2^t\mathbf{d} - 2\mathbf{d}_{AB} \tag{17}$$

$$= 136. \tag{18}$$

We use equation (14) to calculate $\delta_{10}^t\mathbf{d}$. Put $i = 10$, $k = 3$, $j_1 = 3$, $j_2 = 9$, and $j_3 = 8$ such that $C_1 = \{C\}$, $C_2 = \{A, B\}$, and $C_3 = \{H\}$. Then,

$$\delta_{10}^t\mathbf{d} = \delta_3^t\mathbf{d} + \delta_9^t\mathbf{d} + \delta_8^t\mathbf{d} - 2\left(\sum_{x\in C_1, y\in C_2} \mathbf{d}_{xy} + \sum_{x\in C_1, y\in C_3} \mathbf{d}_{xy} + \sum_{x\in C_2, y\in C_3} \mathbf{d}_{xy}\right) \tag{19}$$

$$= 283 - 2((\mathbf{d}_{AC} + \mathbf{d}_{BC}) + \mathbf{d}_{CH} + (\mathbf{d}_{AH} + \mathbf{d}_{BH})) \tag{20}$$

$$= 201. \tag{21}$$

Similarly, we have $\delta_{11}^t\mathbf{d} = 129$ and $\delta_{12}^t\mathbf{d} = 158$.

Fast Algorithms for OLS, WLS, and GLS Edge Lengths

We now apply the matrix multiplication trick of the previous section to the standard OLS, WLS, and GLS edge-lengths formulae (eqs. 3, 5, and 7). In the first two cases, the new algorithms (described below) match the speed of the fastest existing ones (e.g., Sattath and Tversky 1977), and have the the significant practical advantage of being directly applicable to multifurcating trees.

The most interesting contribution in this section, however, is the algorithm for evaluating GLS edge lengths. Given the inverse of the covariance matrix in advance, this method is time-optimal, because the number of individual entries in $\mathbf{V}^{-1}$ is $O(N^4)$, and none of these are redundant. Thus, no future algorithm for calculating GLS edge lengths when the matrix $\mathbf{V}^{-1}$ is given can have a better order of complexity.

Since OLS is the same as WLS with the weighting matrix set equal to the identity matrix, we describe the two algorithms together.

WLS and OLS

Edge lengths under WLS are given by the projection formula

$$\mathbf{b} = (\mathbf{A}^t\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^t\mathbf{W}\mathbf{d}. \tag{22}$$

The edge lengths under OLS can be obtained by putting $\mathbf{W} = \mathbf{I}$, the identity matrix. As mentioned earlier, this calculation takes $O(N^4)$ time using standard matrix multiplication, where N is the number of taxa. We apply algorithm FASTMTM to decrease the complexity. Let K be the number of edges in T. In the following, $\mathbf{w}$ is the diagonal of the weight matrix $\mathbf{W}$.

```
Algorithm WLSEDGES($T$, $\mathbf{w}$, $\mathbf{d}$)
1. For each edge $e_i$:
2. Construct $\mathbf{W}\delta_i$ using the vector $\mathbf{w}$.
3. Calculate $\mathbf{A}^t\mathbf{W}\delta_i$ using FASTMTM and
   store the resulting vector in column $i$
   of a $K \times K$ matrix $\mathbf{X}$. (The matrix $\mathbf{X}$ we
   construct here is the matrix $\mathbf{A}^t\mathbf{W}\mathbf{A}$.)
4. end (For each edge $e_i$).
5. Calculate $\mathbf{A}^t\mathbf{W}\mathbf{d}$ using $w$ and then FAST
   MTM, storing the result in a vector $\mathbf{y}$.
6. Solve $\mathbf{X}\mathbf{b} = \mathbf{y}$ for $\mathbf{b}$.
end.
```

Let $f(K)$ and $g(K)$ be, respectively, the number of operations and memory required to solve the $K \times K$ problem $\mathbf{X}\mathbf{b} = \mathbf{y}$ for $\mathbf{b}$. A simple count of the number of operations, along with repeated use of theorem 3, gives an upper bound of $(K + 1)N(2N - 1) + f(K)$ for the number of operations taken by WLSEDGES when applied to a tree with N taxa and K edges. Since only one split metric vector δ_i need be in memory at one time, the amount of memory space required is $O(N^2) + g(K)$.

There are several options for solving $\mathbf{X}\mathbf{b} = \mathbf{y}$. The most attractive is Cholesky factorization, taking $f(K) = K^3/3 + 2K^2$ operations and $O(K^2)$ memory. Consult Golub and van Loan (1996) for details and other possible solution methods. Note that the constraint of nonnegative edge lengths can be included by replacing step 6 with

```
6′. Find $\mathbf{b}$ that minimizes $(\mathbf{X}\mathbf{b} - \mathbf{y})^t(\mathbf{X}\mathbf{b} - \mathbf{y})$
    such that $\mathbf{b}_i \geq 0$ for all $i$,
```

which is a quadratic programming problem in standard form. Optimized implementations of algorithms solving these matrix problems are available in standard mathematical programming libraries.

As already observed, the algorithm of Felsenstein (1997) also takes $O(N^3)$ time and uses $O(N^2)$ memory. However, Felsenstein's algorithm applies only to binary trees and has no guaranteed bound on the number of iterations required. On the other hand, WLSEDGES works for both binary and multifurcating trees, and is guaranteed to return the exact solution in just one "iteration."

Felsenstein (1997) was aware that WLS edge estimation could be performed by solving a set of linear equations (as in Cavalli-Sforza and Edwards 1967); we show that this can be done while avoiding the high complexity and memory costs that he assumed were obligatory.

GLS

Edge lengths under GLS are given by the projection formula

$$\mathbf{b} = (\mathbf{A}^t\mathbf{V}^{-1}\mathbf{A})^{-1}\,\mathbf{A}^t\mathbf{V}^{-1}\mathbf{d}. \tag{23}$$

Using standard matrix multiplication, this calculation takes $O(N^5)$ time. The next algorithm shows how to complete this calculation in $O(N^4)$ time.

```
Algorithm GLSEdges($T$, $\mathbf{V}^{-1}$, $\mathbf{d}$)
1. For each column $\mathbf{v}_j$ of $\mathbf{V}^{-1}$:
2.   Calculate $\mathbf{A}^t\mathbf{v}_j$ using FastMTM and
     store the resulting vector in column
     $j$ of a $K \times N(N-1)/2$ matrix $\mathbf{Z}$. (The ma-
     trix $\mathbf{Z}$ we construct is equal to the
     matrix $\mathbf{A}^t\mathbf{V}^{-1}$.)
3. end (For each column).
4. For each $i$ from 1 to $K$ do:
5.   Let $z_i^t$ denote row $i$ of $\mathbf{Z}$.
6.   Calculate $\mathbf{A}^t\mathbf{z}_i$ using FastMTM and
     store the resulting vector in column
     $i$ of a $K \times K$ matrix $\mathbf{X}$. (The matrix $\mathbf{X}$
     we construct here equals $\mathbf{A}^t\mathbf{V}^{-1}\mathbf{A}$.)
7. end (For each $i$).
8. Multiply $\mathbf{d}$ by $\mathbf{V}^{-1}$ and then calculate
   $\mathbf{A}^t\mathbf{V}^{-1}\mathbf{d}$ using FastMTM. Store the result
   in a vector $\mathbf{y}$.
9. Solve $\mathbf{Xb} = \mathbf{y}$.
end.
```

Once again, let $f(K)$ and $g(K)$ be, respectively, the number of operations and the memory required to solve the $K \times K$ problem $\mathbf{Xb} = \mathbf{y}$ for $\mathbf{b}$. The number of operations required by GLSEdges is bounded above by $N^4 + N(3N - 1)K/2 + f(K)$, where N is the number of taxa and K is the number of internal edges. The amount of memory space required, in addition to that required to store $\mathbf{V}^{-1}$, is $O(N^3) + g(K)$.

Note that the variance-covariance matrix for edge lengths (e.g., Agresti 1990; Hasegawa, Kishino, and Yano 1985; Bulmer 1991) can be obtained by using algorithm GLSEdges and inverting the matrix $\mathbf{X} = \mathbf{A}^t\mathbf{V}^{-1}\mathbf{A}$. This inversion takes $O(N^3)$ time, resulting in $O(N^4)$ time for the whole operation.

An Unusually Fast Algorithm for OLS Edge Lengths

We have described an $O(N^3)$ algorithm for calculating WLS edge lengths. When applied to OLS, this approach is fast, but only equal in order to several existing OLS methods. Here, we describe an even faster, $O(N^2)$ time, method. Because the number of entries in the input distance is $O(N^2)$, this method is time-optimal: the fastest possible hypothetical algorithms must take at least $O(N^2)$ time.

Returning again to the projection formula for OLS edge lengths (eq. 3), we see that the major obstacle to an $O(N^2)$ algorithm is the construction and inversion of the matrix $(\mathbf{A}^t\mathbf{A})^{-1}$. We need to explore and utilize the properties of this matrix so that we can avoid construction of the matrix altogether.

The first, and perhaps most important, observation we can make about the matrix $(\mathbf{A}^t\mathbf{A})^{-1}$ is that it consists mainly of zeros. The reason is that the length assigned to an edge e_i under OLS is not affected by the shape of a tree beyond those edges directly adjacent to e_i. Consequently, the length of an edge e_i under OLS can be written in terms of $\delta_i^t\mathbf{d}$, $\{\delta_{j_l}^t\mathbf{d}$: e_{j_l} adjacent to $e_i\}$ and the numbers of taxa in the corresponding subtrees. This observation was made in slightly different terminology by Vach (1989) and, later, independently, by Bryant (1997).

For ease of understanding, we present here only the formulas and algorithm for binary trees. Multifurcating trees are dealt with in appendix A.

We must consider two cases when presenting the OLS edge length formulas: internal edges and external edges. Let e_i be an arbitrary internal edge in a binary tree T. We can draw T in the form of figure 1*i*. The edges adjacent to e_i are denoted by e_j, e_k, e_l, and e_m, and the subtrees of T branching off these edges are represented by dotted circles. Let N_j, N_k, N_l, and N_m be the numbers of taxa in the subtrees branching off e_j, e_k, e_l, and e_m, respectively. The optimal edge length $\mathbf{b}_i$ for e_i under OLS is given by:

$$\mathbf{b}_i = \Bigg[\left(\frac{N}{N_m} + \frac{N}{N_l} + \frac{N}{N_k} + \frac{N}{N_j} - 4\right)\delta_i^t\mathbf{d} + \frac{N_j + N_k}{N_jN_k}((2N_k - N)\delta_j^t\mathbf{d} + (2N_j - N)\delta_k^t\mathbf{d}) + \frac{N_l + N_m}{N_lN_m}((2N_m - N)\delta_l^t\mathbf{d} + (2N_l - N)\delta_m^t\mathbf{d})\Bigg] \times \frac{1}{4(N_j + N_k)(N_l + N_m)}. \tag{24}$$

The formula is derived by constructing, and solving, an appropriate set of matrix equations (Bryant 1997, p. 136). The edge length formula of Rzhetsky and Nei (1993) can be recovered from equation (24) by substituting

$$\delta_j^t\mathbf{d} = \mathbf{d}_{AB} + \mathbf{d}_{AC} + \mathbf{d}_{AD} \tag{25}$$

$$\delta_k^t\mathbf{d} = \mathbf{d}_{AB} + \mathbf{d}_{BC} + \mathbf{d}_{BD} \tag{26}$$

$$\delta_l^t\mathbf{d} = \mathbf{d}_{AC} + \mathbf{d}_{BC} + \mathbf{d}_{CD} \tag{27}$$

$$\delta_m^t\mathbf{d} = \mathbf{d}_{AD} + \mathbf{d}_{BD} + \mathbf{d}_{CD} \tag{28}$$

$$\delta_i^t\mathbf{d} = \mathbf{d}_{AC} + \mathbf{d}_{BC} + \mathbf{d}_{AD} + \mathbf{d}_{BD}, \tag{29}$$

giving a substantially shorter and simpler derivation of their result.

The formula for external edges in binary trees is simpler. Let e_i be an arbitrary external edge of a binary tree T. We can draw T in the form of figure 1*ii*. Let e_j

and e_k be the adjacent edges, and let N_j and N_k be the number of taxa in the subtrees branching off these edges. The optimal edge length $\mathbf{b}_i$ for e_i under OLS is given by:

$$\mathbf{b}_i = \frac{1}{4(N_jN_k)}[(1 + N_j + N_k)\delta_i^t\mathbf{d} - (1 + N_j - N_k)\delta_j^t\mathbf{d} - (1 - N_j + N_k)\delta_k^t\mathbf{d})]. \tag{30}$$

It is now a simple matter to incorporate equations (24) and (30) into an algorithm.

```
Algorithm BinaryEdges($T$, $\mathbf{d}$)
1. Count the number of taxa on each side
   of each edge.
2. Calculate $\delta_i^t\mathbf{d}$ for each edge using Fast
   MTM.
3. For each edge $e_i$ do
4. Calculate $b_i$ using equation (24) if $e_i$
   is internal or equation (30) if $e_i$ is
   external.
5. end (For each edge $e_i$).
end.
```

This algorithm takes $O(N^2)$ time.

Theorem 4. *The algorithm* BinaryEdges *takes at most* $3N^2/2 + 127N/2 - 126$ *operations and requires* $O(N)$ *memory space in addition to that required to store the vector* $\mathbf{d}$.

Proof in appendix B.2.

Although the example tree in figure 3 is not binary, we can still use it to illustrate the application of BinaryEdges. We calculate b_i for $i = 1$ and $i = 11$. The edge e_1 is external, so we use equation (30). Put $j = 2$ and $k = 9$ such that $N_j = 1$ and $N_k = 6$. The values $\delta_1^t\mathbf{d} = 80$, $\delta_2^t\mathbf{d} = 70$, and $\delta_9^t\mathbf{d} = 136$ were calculated earlier. Substituting everything into the equation, we obtain $b_1 = 13/3$.

The edge e_{11} is internal, so we use equation (24). We put $j = 10$, $k = 12$, $l = 4$, and $m = 5$, such that $N_j = 4$, $N_k = 2$, $N_l = 1$, and $N_m = 1$. Substituting these values into the equation, we obtain $b_{11} = 1{,}325/512$.

We will have to use the algorithm FastOLS in appendix A to calculate b_i for $i = 3, 8, 9$, and 10.

Calculating Path Lengths in Minimal Time

So far, we have shown how to calculate edge lengths in $O(N^2)$ time for OLS, $O(N^3)$ time for WLS, and $O(N^4)$ time for GLS. These algorithms, followed by a linear $O(N)$ summation of the edge lengths, give fast algorithms for minimum evolution, e.g., ME(OLS), ME(FM), and ME(GLS). However, other popular tree selection criteria, namely sums-of-squares criteria, require the calculation of the taxon-to-taxon distance within the tree with these edge lengths. Given any two taxa in a tree, it takes $O(N)$ time to find the distance between them by tracing the path connecting them and, hence, $O(N^3)$ time to calculate all $O(N^2)$ pairwise distances. Alternatively, direct application of equation (1) also takes $O(N^3)$ time. While this time bound is acceptable for WLS and GLS, it is unacceptable for OLS: an $O(N^2)$ time evaluation method is thus required to make SS(OLS) time-optimal.

Therefore, we developed the algorithm DistanceInTree, which calculates taxon-to-taxon distances in $O(N^2)$ time from a tree T and its edge lengths. Note that $\mathbf{p}_{uv}$ is used to denote the distance between vertices u and v in a tree. The algorithm is based on repeated application of a simple observation: if v is a vertex common to the path from a to c and the path from b to c, then $\mathbf{p}_{ac} = \mathbf{p}_{av} + \mathbf{p}_{cv}$ and $\mathbf{p}_{bc} = \mathbf{p}_{bv} + \mathbf{p}_{cv}$. If we were to calculate the distance from a to c and the distance from b to c separately, we would end up tracing the path from v to c twice. This observation enables us to avoid this repetition. It is more efficient to calculate all vertex-to-vertex path lengths than to calculate just the taxon-to-taxon distances.

The distance from a vertex a to a vertex b is clearly the same as the distance from b to a. Therefore, after calculating the distances from a particular vertex to all other vertices, we remove that vertex from the the tree, being careful to leave other vertex-to-vertex distances the same.

```
Algorithm DistanceInTree($T$)
1. For each taxon $a$ in $T$ do
2. $\mathbf{p}_{aa} \leftarrow 0$.
3. Repeat until $\mathbf{p}_{ax}$ has been calculated
   for all vertices $x$ in $T$.
4.   Choose a vertex $v$ such that $\mathbf{p}_{av}$ has *not*
     been calculated but $v$ is adjacent to
     a vertex $u$ for which $\mathbf{p}_{au}$ *has* been cal-
     culated.
5.   $\mathbf{p}_{av} \leftarrow \mathbf{p}_{au}$ + length of edge between $u$
     and $v$.
6. end (repeat until).
7. Remove taxon $a$ from $T$, together with
   its adjacent edge.
8. If there are vertices of degree two in
   $T$, then remove them and replace the ad-
   jacent edges with a single edge with
   length equal to the sum of the lengths
   of the two adjacent edges.
9. end (For each taxon $a$).
end.
```

A simple count of operations gives an upper bound of $N(N - 1)$ operations, while $O(N^2)$ memory space is required.

We illustrate the algorithm by applying it to the example in figure 3 with optimal edge weights

$$\mathbf{b} = \left(\frac{13}{3}, \frac{8}{3}, \frac{23}{24}, \frac{1}{12}, \frac{47}{12}, \frac{7}{2}, \frac{9}{2}, \frac{163}{24}, \frac{15}{16}, \frac{97}{32}, \frac{25}{16}, \frac{51}{16}\right)^t \tag{31}$$

calculated using BinaryEdges and FastOLS (see appendix A). The internal vertices of the tree are conveniently labeled α, β, γ, δ, and ϵ.

First of all, $\mathbf{p}_{A\alpha} = b_1 = 13/3$. Then, $\mathbf{p}_{AB} = \mathbf{p}_{A\alpha} + b_2 = 7$ and $\mathbf{p}_{A\beta} = \mathbf{p}_{A\alpha} + b_9 = 253/48$. From there, we can calculate $\mathbf{p}_{AB}$, $\mathbf{p}_{AB}$, $\mathbf{p}_{AB}$, and so on, until $\mathbf{p}_{Ax}$ has been calculated for all $x \in \{B, C, D, E, F, G, H, \alpha, \beta, \gamma, \delta, \epsilon\}$. At that point, we remove the vertices A and α and add an edge between B and β of weight $b_2 + b_9$. The process then repeats, this time starting with B.

Calculating Sums of Squares Quickly

Using the algorithm CALCULATEDISTANCE and the fast edge length calculation methods we can speed up sum-of-squares (SS) evaluations of trees.

For OLS,

$$\mathrm{SS(OLS)} = (\mathbf{Ab} - \mathbf{d})^t(\mathbf{Ab} - \mathbf{d}). \tag{32}$$

The optimal edge lengths $\mathbf{b}$ are provided by the algorithm BINARYEDGES if T is binary and by FASTOLS (appendix A) if T is multifurcating. The calculation of $\mathbf{Ab}$ can be reduced from $O(N^3)$ time to $O(N^2)$ time using the algorithm DISTANCEINTREE. Thus, the whole calculation takes $O(N^2)$ time. In both cases, a bound on the actual number of operations is obtainable: $5N^2/2 + 125N/2 - 126$ operations for a binary tree, and

$$26K^2 - 104NK + 213N^2/2 + 182K - 415N/2 - 3 \tag{33}$$

operations for a multifurcating tree with K edges.

For WLS (including FM), we can calculate edge lengths in $O(N^3)$ time. Calculating $\mathbf{Ab}$ takes $O(N^2)$ time, so the entire calculation

$$\mathrm{SS(WLS)} = (\mathbf{Ab} - \mathbf{d})^t\mathbf{W}(\mathbf{Ab} - \mathbf{d}) \tag{34}$$

takes a total of $O(N^3)$ time. The actual number of operations is bounded above by $(K + 1)N(2N - 1) + N(N - 1) + f(K)$, where K is the number of edges and $f(K)$ is the number of operations required to solve a $K \times K$ (positive definite) linear system.

By a similar method, the SS calculation

$$\mathrm{SS(GLS)} = (\mathbf{Ab} - \mathbf{d})^t\mathbf{V}^{-1}(\mathbf{Ab} - \mathbf{d}) \tag{35}$$

can be completed in $O(N^4)$ time, provided that, as above, the inverse matrix $\mathbf{V}^{-1}$ is calculated beforehand. The actual number of operations required for a tree with K edges is bounded above by $N^4 + N(N - 1) + N(3N - 1)K/2 + f(K)$, where, once again, $f(K)$ is the number of operations required to solve a $K \times K$ (positive definite) linear system.

Summary of Results

A comprehensive selection of algorithms, covering all of the main least-squares and minimum-evolution evaluation criteria, has been provided. The algorithms described are superior to all published algorithms in several respects: they are as fast as or faster than existing algorithms, many of the speed increases are dramatic, many of the algorithms are provably time-optimal, and the algorithms apply both to binary and multifurcating trees.

Here, we compare the worst-case performance of the algorithm BINARYEDGES with that of the OLS algorithm presented in appendix B of Rzhetsky and Nei (1993). Theorem 4, proved in appendix B.2, gives an upper bound of $3N^2/2 + 127N/2 - 126$ on the number

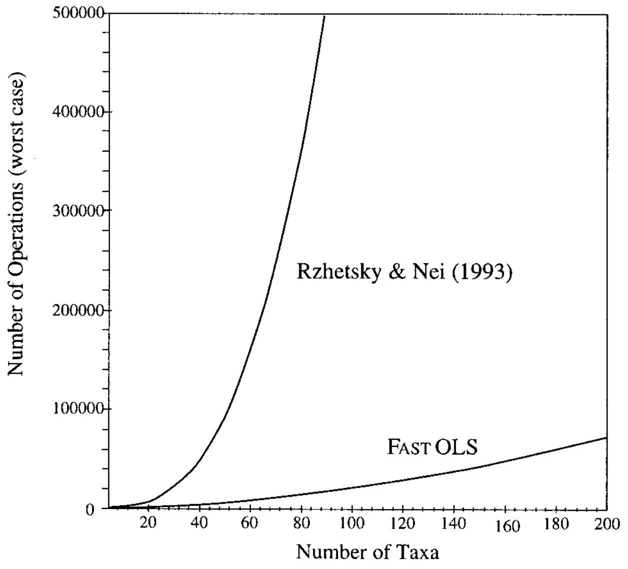

FIG. 4.—Plots of the worst-case number of operations for the FASTOLS algorithm and the worst-case number of operations for the algorithm of Rzhetsky and Nei (1993). Note that when $N = 200$, Rzhetsky and Nei's algorithm can take over 5 million operations, while FASTOLS takes under 75,000.

of operations taken by BINARYEDGES applied to any binary tree with N leaves. The number of operations required for Rzhetsky and Nei's algorithm depends on the shape of the tree. For ease of calculation, we assume that the tree is a caterpillar tree. We are not sure whether this gives the worst-case performance for Rzhetsky and Nei's algorithm; if it does not, then we will have underestimated the worst-case performance of Rzhetsky and Nei's algorithm and will have subsequently underestimated the increase in speed given by BINARYEDGES. The figure $2N^3/3 + 3N^2 + 94N/3 + 8$, derived in appendix B.4, is a lower bound on the worst-case complexity of the Rzhetsky and Nei (1993) algorithm.

The bounds for up to 200 taxa are plotted in figure 4. Note that the curve for the Rzhetsky and Nei (1993) algorithm is not complete: this would require a vertical axis stretching up to over 5 million operations. We have not differentiated between additions and multiplications, as this would be laborious and possibly confusing (furthermore, the ratio appears similar in both algorithms). Differentiating between them would also be of limited interest, since both types of calculation would be done with floating-point numbers and the scalar multibit chips now being used do both types of operation in similar time.

In all calculations, memory is an important additional parameter (especially since limited memory can be stored on the L1 or L2 on-chip cache, which runs two, four, or more times as fast as that on the mother board). Both algorithms use $O(N)$ memory, making the dominant memory term the pairwise distances ($O[N^2]$). As we mentioned in the introduction, computer architecture can have a significant impact on the speed of algorithms. In the discussion, we cite examples to show conclusively that the new algorithms are indeed sub-

Table 1
Summary of Increases in Evaluation Speed

Evaluation Criterion	Algorithm(s)	Complexity Bound	Comlexity Order	Time-Optimal?
Edge lengths (and ME)				
OLS (binary)	BinaryEdges	$3N^2/2 + O(N)$	$O(N^2)$	Yes
OLS (general)	FastOLS	$26K^2 - 104NK + 211N^2/2 + O(N)$	$O(N^2)$	Yes
WLS	WLSedges	$K^3/3 + 2N^2K + O(N^2)$	$O(N^3)$	Possibly
GLS	GLSedges	$N^4 + K^3/3 + 2N^2K + O(N^2)$	$O(N^4)$	Yes
Sum of squares				
OLS (binary)	BinaryEdges + DistanceInTree	$5N^2/2 + O(N)$	$O(N^2)$	Yes
OLS (general)	FastOLS + DistanceInTree	$26K^2 - 104NK + 213N^2/2 + O(N)$	$O(N^2)$	Yes
WLS	WLSedges + DistanceInTree	$K^3/3 + 2N^2K + O(N^2)$	$O(N^3)$	Possibly
GLS	GLSedges + DistanceInTree	$N^4 + K^3/3 + 2N^2K + O(N^2)$	$O(N^4)$	Yes
Variance-covariance matrix	GLSedges + inversion	$3N^4/2 + O(N^3)$	$O(N^4)$	Yes

stantially faster than previous ones with standard computer systems.

Bounds on the number of operations required by all of the algorithms introduced in this paper are summarized in table 1. All cited values assume that Cholesky decomposition is used to solve linear equations—if a faster algorithm becomes available, this will further speed things up. The number of taxa is N and the number of edges in T is K.

The figures in the table allow us to draw conclusions about the comparative time costs of the various evaluation criteria. For example, SS evaluation takes an additional $N(N - 1)$ operations on top of those required to calculate edge lengths.

We also note that for moderate values of N, the time taken by the GLS algorithms is only a few times that taken by the WLS algorithms. As a general rule of thumb, when there are N taxa, the GLS methods take $N/8$ times as long as WLS. For example, when $N = 50$, the GLS algorithms take between seven and eight times as long as the WLS algorithms (and about 1,200 times as long as OLS!).

Discussion

The increase in speed of least-squares tree criterion evaluation described in this paper should allow faster and more extensive tree search strategies for distance-based methods. The algorithms have already been incorporated into PAUP* for OLS, ME(OLS), and WLS (Fitch Margoliash, with $\mathbf{W}_{ij} = 1/(\mathbf{d}_{ij})^2$) tree searches. As expected, the new algorithms performed well. In the largest benchmark tried (a 125-taxon distance matrix), the new OLS algorithm evaluated 132 trees per second, compared with a rate of 1.68 trees per second for the previously implemented Felsenstein (1997) algorithm. Note that the algorithms of Sattath and Tversky (1977) and Rhzetsky and Nei (1993) were not considered practical for PAUP* because of their inability to evaluate nonbinary trees (a key feature in the program, especially when using star decomposition searches).

Rzhetsky and Nei (1992*a*) describe an alternative localized, but still useful, method of evaluating all trees within a small partition distance of a good starting tree. Use of our algorithm FastOLS makes this approach run $O(N)$ times as fast. See also the fast one-step local search method of Gascuel (1996). Bryant (1997, pp. 149–154) has developed a fast algorithm that optimizes ME(OLS) over trees within an arbitrary partition distance, without evaluating each individual tree.

It is important to note that the algorithms we describe will sometimes assign negative lengths to edges. It has been a long-running argument whether this is desirable or not (e.g., Felsenstein 1984; Farris 1985; Kuhner and Felsenstein 1994; Swofford et al. 1996; Waddell, Lewis, and Swofford 1998). To date, there is no definitive answer and a good deal of disagreement. There are some who feel that negative edge lengths should be avoided wherever possible and propose that any tree containing a negative edge be automatically rejected (e.g., Kidd and Sgaramella-Zonta 1971). Another approach is to define the ME score of a tree as

$$\sum_i |b_i| \quad \text{instead of} \quad \sum_i b_i, \tag{36}$$

thereby penalizing negative edges (Kidd and Sgaramella-Zonta 1971; Swofford et al. 1996).

A third approach is to calculate edge lengths on a tree subject to the constraint that edges lengths must be nonnegative. This is not simply a matter of collapsing negative edges to zero then reoptimizing the remaining edge lengths; we have discovered a seven-taxa tree for which this approach fails (unpublished data). Thus, when there are two or more negative edge lengths in the unconstrained optimum for a tree, a more sophisticated method is required to guarantee constrained optima. The only polynomial time methods proven to give optimal edge lengths in the constrained case are ellipsoid and interior point algorithms for convex quadratic programming (e.g., Kozlov, Tarasov, and Khachiyan 1979; Goldfarb and Liu 1991). It is possible that versions of these algorithms could be made simpler and faster when they are tailored to the specific least-squares edge length problem on trees.

Presently, a practical alternative is to employ a heuristic method such as the iterative algorithm of Felsenstein (1997) or the iterative QR2 algorithm of Gascuel and Levy (1996). One can, for example, switch from our exact algorithms to these iterative algorithms when encountering trees exhibiting the problem of negative edges, thereby optimizing overall speed. This is the approach taken by Swofford (1997) in PAUP*. When there is a constraint in place of all nonnegative edges, all negative edges are first collapsed, and the edge lengths are re-estimated by the algorithms described in this paper (applied to nonbinary trees). At this point, the version of the algorithm WLSEDGES giving nonnegative edge lengths is used—the quadratic programming problem is currently being solved using Gauss Siegel iteration.

The increases in speed offered by the algorithms presented here will hopefully encourage the use of WLS and GLS. Since these criteria come closer to ML on distances than any others currently implemented, it is reasonable to expect they will be more statistically efficient and return the correct answer more often than the computationally faster OLS methods. A useful combination of the algorithms presented here may be fast searches of tree space with SS(OLS) or ME(OLS), followed by the use of WLS or GLS to select among the better trees found.

Acknowledgments

We are grateful to David Penny, Masami Hasegawa, Hidetoshi Shimodaira, and Mike Steel for comments on the manuscript and to Dave Swofford for comments, implementing the algorithm, and checking for typographic errors. This work was supported by a University of Canterbury Doctoral Scholarship to D.B. and the Marsden Fund (96 MAU-ALS 0042 to P.W., and 95 UOC-516 to D.B.).

APPENDIX A
Calculating OLS Edge Lengths in Multifurcating Trees

We have described an $O(N^2)$ algorithm for calculating OLS edge lengths in a binary tree. The method can be extended to multifurcating trees. Although the equations are longer, the basic idea is the same: we calculate the quantities $\delta_i^t\mathbf{d}$ for each edge e_i and then apply an edge length formula, in this case equation (37). However, in order to obtain an $O(N^2)$ time algorithm we have to be careful about how we apply the formula. We show how to do this explicitly in the algorithm FASTOLS, making the algorithm code longer than necessary but easier to implement.

As with binary trees, we consider two cases when presenting the OLS edge length formulas: internal edges and external edges. Any internal edge e_i in a multifurcating tree can be drawn in the form of figure 2*i*. Let α and β be the endpoints of e_i. Let $e_{j1}, e_{j2}, \ldots, e_{j_k}$ be the remaining edges adjacent to α, and let $e_{j_{k+1}}, \ldots, e_{j_m}$ be the remaining edges adjacent to β. The dotted circles represent the subtrees branching off the respective edges. Let $C_1, \ldots, C_m$ be the taxon sets of these subtrees, and let N_l be the number of taxa in C_l, for all $l = 1, \ldots, m$. Put $N_\alpha = \sum_{l=1}^{k} N_l$ and $N_\beta = \sum_{l=k+1}^{m} N_l$. Let $\mathbf{v}$ be the vector with N_β in positions $1, \ldots, k$ and N_α in positions $k + 1, \ldots, m$.

We use similar labeling in the case of an external edge e_i, any example of which can be drawn in the form of figure 2*ii*. Let $e_{j_1}, \ldots, e_{j_m}$ be the edges adjacent to e_i, let $C_1, \ldots, C_m$ be their associated taxon sets, and let N_l be the number of taxa in C_l, for all $l = 1, \ldots, m$. Let $\mathbf{v}$ be a vector of m ones, and put $N_\alpha = N - 1$ and $N_\beta = 1$.

One formula, equation (37), suffices for both internal and external edges. What it might lack in aesthetic appeal it makes up for in usefulness.

Theorem 5. *Let e_i be an external or internal edge with adjacent edges and subtrees as described. The optimal edge length $\mathbf{b}_i$ for e_i under OLS is given by*

$$\mathbf{b}_i = \frac{\delta_i^t\mathbf{d} - \mathbf{w}^t\mathbf{N}^{-1}\mathbf{P}}{N_\alpha N_\beta - \mathbf{w}^t\mathbf{v}}, \tag{37}$$

where N is the number of taxa,

$$\mathbf{P} = (\delta_{j_1}^t\mathbf{d}, \delta_{j_2}^t\mathbf{d}, \ldots, \delta_{j_m}^t\mathbf{d})^t, \tag{38}$$

and

$$\mathbf{w} = (N\mathbf{N}^{-1} - 2\mathbf{I} + \mathbf{U})^{-1}\mathbf{v}, \tag{39}$$

where the matrix $\mathbf{U}$ is the $m \times m$ matrix of ones, $\mathbf{N}$ is the diagonal matrix with diagonal $N_1, N_2, \ldots, N_m$, and $\mathbf{I}$ is the identity matrix.

See Vach (1989) and Bryant (1997) for two independent and alternative derivations of this result, as well as the proof in Bryant (1997) that the matrix $(N\mathbf{N}^{-1} - 2\mathbf{I} + \mathbf{U})$ is invertible.

There are explicit formulas for the elements of $\mathbf{w}$, with two cases to consider.

If $N_l \neq N/2$ for all $l = 1, \ldots, m$, then

$$\mathbf{w}_l = \frac{1}{(N/N_l - 2)}(\gamma + \mathbf{v}_l), \tag{40}$$

where

$$\gamma = \frac{-1}{\kappa}\sum_{j=1}^{m}\frac{\mathbf{v}_j}{(N/N_j - 2)}$$

and

$$\kappa = 1 + \sum_{j=1}^{m}\frac{N_j}{N - 2N_j}.$$

If there is some $\lambda \in \{1, \ldots, m\}$ such that $N_\lambda = N/2$, then for all $l \neq \lambda$, we have

$$\mathbf{w}_l = \frac{\mathbf{v}_l - \mathbf{v}_\lambda}{N/N_l - 2}, \tag{41}$$

whereas

$$\mathbf{w}_\lambda = \mathbf{v}_\lambda + \sum_{j\neq\lambda}\left(\frac{N_j(\mathbf{v}_\lambda - \mathbf{v}_j)}{N - 2N_j}\right). \tag{42}$$

Note that for any external or internal edge there can be at most one adjacent subtree with $N/2$ taxa, so there can be at most one λ such that $N_\lambda = N/2$. We summarize the entire process in algorithm FASTOLS and calculate the time and memory requirement in appendix B.3.

Theorem 6. *The algorithm* FASTOLS *takes at most*

$$26K^2 - 104NK + \frac{211N^2}{2} + 182K - \frac{413N}{2} - 3 \quad (43)$$

operations when applied to an N-taxon distance matrix and a tree with K edges. It requires O(N) memory space in addition to that required to store the vector **d**.

```
Algorithm FastOLS($T$, $\mathbf{d}$)
 1. Count the number of taxa on each side
    of each edge.
 2. Calculate $\delta_i^t\mathbf{d}$ for each edge using
    FastMTM.
 3. For each edge $e_i$ do
 4.   If $e_i$ is internal then
 5.     Label the endpoints, adjacent
        edges, and adjacent clusters of
        $e_i$ as in figure 2$i$. For each $l$, let
        $N_l$ be the number of taxa in $C_l$. Let
        $N_\alpha$ and $N_\beta$ be the number of taxa on
        each side of $e_i$.
 6.   else
 7.     Label the endpoints, adjacent
        edges, and adjacent clusters of
        $e_i$ as in figure 2$ii$. For each $l$, let
        $N_l$ be the number of taxa in $C_l$. Put
        $N_\alpha = N - 1$, $N_\beta = 1$, and $k = m$.
 8.   end (if-else).
 9.   Let $\mathbf{v}$ be the vector with $N_\beta$ in posi-
      tions $1, \ldots, k$ and $N_\alpha$ in positions $k +
      1, \ldots, m$.
10.   If there is $\lambda$ such that $N_\lambda = N/2$ then
11.     For $l = 1, \ldots, m$ except $\lambda$ do
12.       $\mathbf{w}_l \leftarrow (\mathbf{v}_l - \mathbf{v}_\lambda)/(N/N_l - 2)$.
13.     end (for).
14.     $\mathbf{w}_\lambda \leftarrow \mathbf{v}_\lambda + \Sigma_{j\neq\lambda} \{[N_j(\mathbf{v}_\lambda - \mathbf{v}_j)]/(N - 2N_j)\}$.
15.   else
16.     $\kappa \leftarrow 1 + \Sigma_{j=1}^m [N_j/(N - 2N_j)]$.
17.     $\lambda \leftarrow (-1/\kappa) \Sigma_{j=1}^m [\mathbf{v}_j/(N/N_j - 2)]$.
18.     For $l = 1, \ldots, m$ do
19.       $\mathbf{w}_l \leftarrow [1/(N/N_l - 2)](\gamma + \mathbf{v}_l)$.
20.     end (for).
21.   end (if-else).
22.   $\mathbf{b}_i \leftarrow \dfrac{\delta_i^t\mathbf{d} - \sum_{l=1}^{m} \dfrac{w_l\delta_{j_l}^t\mathbf{d}}{N_l}}{N_\alpha N_\beta - \sum_{j=1}^{m} \mathbf{w}_j\mathbf{v}_j}$.
23. end (For each edge $e_i$).
end.
```

The algorithm may appear daunting, but it is little more than an explicit implementation of equations (37)–(42). We demonstrate the use of this algorithm by calculating the optimal edge lengths $\mathbf{b}_8$, $\mathbf{b}_9$, and $\mathbf{b}_{10}$ for the example tree and distance in figure 3. We will use the values for $\delta_i^t\mathbf{d}$ calculated earlier.

First, we consider the external edge e_8. Putting the tree in the form of figure 2*ii*, we have $j_1 = 9$, $j_2 = 3$, and $j_3 = 10$. Thus, $N_1 = 2$, $N_2 = 1$, $N_3 = 4$, $N_\alpha = 7$, $N_\beta = 1$, and $k = m = 3$. The vector **v** in step 9 becomes $\mathbf{v} = (1, 1, 1)^t$.

There does exist a λ such that $N_\lambda = N/2$: $\lambda = 3$. Therefore, we use steps 11–14 to calculate **w**. We have $\mathbf{w}_1 = \mathbf{w}_2 = 0$ and $\mathbf{w}_3 = 1$, since all elements of **v** are equal.

We now substitute everything into the equation in step 22, obtaining:

$$\mathbf{b}_8 = \frac{\delta_8^t\mathbf{d} - \dfrac{\delta_{10}^t\mathbf{d}}{N_3}}{7 - 1} \quad (44)$$

$$= 163/24. \quad (45)$$

Now we consider the internal edge e_9, so $i = 9$. Putting the tree in the form of figure 2*i*, we have $j_1 = 2$, $j_2 = 1$, $j_3 = 8$, $j_4 = 10$, and $j_5 = 3$. Thus, $k = 2$ and $m = 5$, N_1, N_2, N_3, N_4, and N_5 equal 1, 1, 1, 4, and 1, respectively, $N_\alpha = 2$, and $N_\beta = 6$. The vector **v** in step 9 becomes $\mathbf{v} = (6, 6, 2, 2, 2)^t$.

Once again, there does exist a λ such that $N_\lambda = N/2$: $\lambda = 4$. We therefore use steps 11–14 to calculate **w**, obtaining $\mathbf{w} = (4/6, 4/6, 0, 4/6, 0)^t$. Substituting into the equation in step 22, we obtain

$$\mathbf{b}_9 = \frac{\delta_9^t\mathbf{d} - \left(\dfrac{4}{6}\cdot\dfrac{\delta_2^t\mathbf{d}}{N_1}\dfrac{4}{6}\cdot\dfrac{\delta_1^t\mathbf{d}}{N_2} + 0 + \dfrac{4}{6}\cdot\dfrac{\delta_{10}^t\mathbf{d}}{N_4} + 0\right)}{12 - \dfrac{28}{3}} \quad (46)$$

$$= \frac{15}{16}. \quad (47)$$

Finally, we consider the internal edge e_{10}: $i = 10$. Again putting the tree in the form of figure 2*i*, we have $j_1 = 3$, $j_2 = 9$, $j_3 = 8$, $j_4 = 12$, $j_5 = 11$, $k = 2$, and $m = 5$. Thus, N_1, N_2, N_3, N_4, and N_5 equal 1, 2, 1, 2, and 2, respectively, and $N_\alpha = 4$ and $N_\beta = 4$. The vector **v** in step 9 becomes $\mathbf{v} = (4, 4, 4, 4, 4)^t$.

This time, there is no λ such that $N_\lambda = N/2$. Therefore, we use steps 16–20 to calculate **w**. The formula in step 16 gives $\kappa = 10/3$, and the formula in step 17 gives $\gamma = -2$. Steps 18–20 give $\mathbf{w} = (1/3, 1, 1/3, 1, 1)^t$. Substituting into the equation in step 22 gives $\mathbf{b}_{10} = 97/32$.

Repeating the above calculations for all $i = 1, \ldots, 12$ gives an edge length vector of

$$\mathbf{b} = \left(\frac{13}{3}, \frac{8}{3}, \frac{23}{24}, \frac{1}{12}, \frac{47}{12}, \frac{7}{2}, \frac{9}{2}, \frac{163}{24}, \frac{15}{16}, \frac{97}{32}, \frac{25}{16}, \frac{51}{16}\right)^t \quad (48)$$

APPENDIX B

Speed and Memory Usage of the Algorithms

B.1. FastMTM

We now prove theorem 3, which we restate here:

Theorem 3. *The algorithm* FastMTM *takes at most* $3N^2/2 - N/2$ *operations. The amount of memory used,* *in addition to the memory required to store the vector* $\mathbf{d}$, *is* $O(N)$.

Proof

We discuss each step of the algorithm in turn.

Step 1. We can calculate $\delta_i^t\mathbf{d}$ for an external edge e_i in just $N - 2$ operations by using equation (9). Repeating for all N external edges takes $N(N - 2)$ operations.

Step 2. The tree T is input as an adjacency list, so we can construct the required ordering using at most $2N$ operations.

Step 3. First, observe that for each pair of taxa x,y, the length d_{xy} is used to calculate $\delta_i^t\mathbf{d}$ for at most one edge e_i. Similarly, each value $\delta_i^t\mathbf{d}$, once calculated, is only used once when calculating the other values $\delta_j^t\mathbf{d}$. The calculation of $\delta_i^t\mathbf{d}$ also requires one additional multiplication (by 2) and a subtraction. Thus, the total number of operations required in step 3 is $(N/2) + N + 2N$, and the total number of operations is at most

$$N(N - 2) + 2N + \binom{N}{2} + 3N = 3N^2/2 - N/2, \qquad (49)$$

proving the speed bound.

To prove that only $O(N)$ memory is used, we note that the only memory used by the algorithm is that used to store the tree T, the ordering of the edges, and the values $\delta_i^t\mathbf{d}$. □

B.2. BinaryEdges

Theorem 4. *The algorithm* BinaryEdges *takes at most* $3N^2/2 + 127N/2 - 126$ *operations and requires* $O(N)$ *memory space in addition to that required to store the vector* $\mathbf{d}$.

Proof

Step 1 can be completed in at most $2(2N - 3)$ operations by rooting T at some internal vertex and then recursively calculating the taxa in every cluster of this rooted tree.

Step 2 takes ½$N(3N - 1)$ operations by theorem 3.

Step 4 takes at most 40 operations for internal edges and 20 operations for external edges. Therefore steps 3–5 take $20N + 40(N - 3) = 60N - 120$ operations.

Counting all steps together, we have that the algorithm takes at most

$$4N - 6 + \frac{N(3N-1)}{2} + 60N - 120$$
$$= \frac{3N^2}{2} + \frac{127N}{2} - 126 \qquad (50)$$

operations.

The memory required is that needed to store the values $\delta_i^t\mathbf{d}$, the numbers of taxa in each subtree, and the edge length calculated. □

B.3. FastOLS

Theorem 6. *The algorithm* FastOLS *takes at most* $26K^2 - 104NK + 211N^2/2 + 182K - 413N/2 - 3$ *operations when applied to an N-taxon distance matrix and a tree with K edges. It requires* $O(N)$ *memory space in addition to that required to store the vector* $\mathbf{d}$.

Proof

Step 1 can be completed in at most $2(2N - 3)$ operations by rooting T at some internal vertex and then recursively calculating the taxa in every cluster of this rooted tree.

Step 2 takes ½$N(3N - 1)$ operations by theorem 3.

Fix some e_i. Let $m = m(i)$ be the number of edges adjacent to e_i. A simple count gives an upper bound of $20m$ operations to construct $\mathbf{w}$. A further $5m + 1$ operations are required to calculate $\mathbf{b}_i$. Thus, each edge length takes at most $26m$ operations.

Summing over all edges, we see that the number of operations required depends on the number of ordered pairs of edges (e_i, e_j) such that e_i is adjacent to e_j. The number of such edges in a tree with N leaves and K edges is bounded above by $6(K - N) + (2N - K)(2N - K - 1)$. Taking steps 1 and 2 into account we have an upper bound of $26K^2 - 104NK + 211N^2/2 + 182K - 413N/2 - 3$.

The only memory required, in addition to that used for the input distance, is that used for storing the values $\delta_i^t\mathbf{d}$, the edges, vertices, and cluster sizes of the tree, the vectors $\mathbf{v}$ and $\mathbf{w}$ (which can be written over every iteration), the edge lengths calculated, and assorted placeholders in the calculation (e.g., γ and κ). Thus, the additional memory used takes a modest $O(N)$ space. □

B.4. Complexity of the Algorithm of Rzhetsky and Nei

Rzhetsky and Nei (1993) describe an algorithm for calculating OLS edge lengths for binary trees. We analyze the time complexity of that algorithm, as presented in appendix B of Rzhetsky and Nei (1993).

Rzhetsky and Nei calculate the length of an internal edge b using the formula

$$\hat{b} = \sum_{i<j} \omega_{ij}\mathbf{d}_{ij}, \qquad (51)$$

where $\mathbf{d}$ is the distance matrix and ω_{ij} takes on one of seven different quantities calculated for each separate edge.

Calculating the possible values for ω_{ij} from equations (B2) and (B3) in Rzhetsky and Nei (1993) takes (at least) 36 operations for internal edges and 7 operations for external edges. If the multiplication and summation are carried out over all pairs of taxa i,j, then the number of operations per edge is at least $36 + N(N - 1)$ or $7 + N(N - 1)$, resulting in

$$(N - 3)(36 + N(N - 1)) + N(7 + N(N - 1))$$
$$= 2N^3 - 5N^2 + 46N - 108 \qquad (52)$$

operations for the whole tree.

However, it is not necessary to count additions and multiplications for pairs i,j such that $\omega_{ij} = 0$. For any particular edge b, we have $\omega_{ij} \neq 0$ if and only if the path from i to j passes through one or both endpoints of b. The number of such paths depends on the shape of the tree and the location of b within the tree.

Suppose that we choose a particular pair of taxa i,j and then count the number of edges for which the quantity ω_{ij} is nonzero. Clearly, this includes only the edges along the path from i to j, together with one extra edge for every internal vertex on the path from i to j. Thus if we sum over all pairs i,j, we can derive the number of operations required by the algorithm,

$$\begin{aligned} &36(N - 3) + 7N + 2 \sum_{i<j} (2\rho_{ij} - 1) \\ &\quad = 44N - N^2 - 108 + 4 \sum_{i<j} \rho_{ij}, \end{aligned} \tag{53}$$

where ρ_{ij} is the number of edges on the path from i to j.

The quantity $\Sigma_{i<j}\ \rho_{ij}$ depends on the shape of the tree and is difficult to calculate for general N. However, here we are interested in worst-case complexity, so we can restrict our analysis to the case in which the tree is a caterpillar tree (a single long path with external edges branching off) and be safe in the knowledge that the worst case is at least as bad as this.

If T is a caterpillar tree with N leaves, then a recursive counting calculation gives

$$\sum_{i<j} \rho_{ij} = \frac{1}{6}N^3 + N^2 - \frac{19N}{6} + 2. \tag{54}$$

Combining all factors together, we have:

Theorem 7. *The algorithm of Rzhetsky and Nei* (1993), *appendix B, can take at least* $2N^3/3 + 3N^2 + 94N/3 + 8$ *operations.*

Thus, the complexity bound of $O(N^3)$ derived by Rzhetsky and Nei (1993) was tight.

LITERATURE CITED

AGRESTI, A. 1990. Categorical data analysis. John Wiley and Sons, New York.

BARRY, D., and J. A. HARTIGAN. 1987. Asynchronous distance between homologous DNA sequences. Biometrics **43**:261–276.

BRYANT, D. J. 1997. Building trees, hunting for trees and comparing trees—Theory and method in phylogenetic analysis. Ph.D. thesis, University of Canterbury, Christchurch, New Zealand.

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

CAVALLI-SFORZA, L., and A. EDWARDS. 1967. Phylogenetic analysis models and estimation procedures. Evolution **32**: 550–570.

DAY, W. H. E. 1987. Computational complexity of inferring phylogenies from dissimilarity matrices. Bull. Math. Biol. **49**(4):461–467.

FARRIS, J. S. 1985. Distance data revisited. Cladistics **1**:67–85.

FELSENSTEIN, J. 1978. Cases in which parsimony or compatibility methods will be positively misleading. Syst. Zool. **27**: 401–410.

———. 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**:368–376.

———. 1984. Distance methods for inferring phylogenies: a justification. Evolution **38**:16–24.

———. 1988. Phylogenies from molecular sequences: inference and reliability. Annu. Rev. Genet. **22**:521–565.

———. 1993. PHYLIP (phylogeny inference package). Version 3.5c. Distributed by the author, Department of Genetics, University of Washington, Seattle.

———. 1997. An alternating least squares approach to inferring phylogenies from pairwise distances. Syst. Biol. **46**: 101–111.

FITCH, W. M., and E. MARGOLIASH. 1967. Construction of phylogenetic trees. Science **155**:279–284.

———. 1971. Toward defining the course of evolution: minimal change for a specific tree topology. Syst. Zool. **20**:406–416.

GASCUEL, O. 1996. Algorithmes efficaces pour la construction d'arbes d'évolution minimum. Rapport LIRMM 96019, Montpellier, France.

———. 1997. Concerning the NJ algorithm and its unweighted version, UNJ. Pp. 149–170 *in* B. MIRKIN, F. R. MCMORRIS, F. S. ROBERTS, and A. RZHETSKY, eds. Mathematical hierarchies and biology. American Mathematical Society, Providence, R.I.

GASCUEL, O., and D. LEVY. 1996. A reduction algorithm for approximating a (non-metric) dissimilarity by a tree distance. J. Classif. **13**:129–155.

GOLDFARB, D., and S. LIU. 1991. An $O(n^3L)$ primal interior point algorithm for convex quadratic programming. Math. Programming **49**:325–340.

GOLUB, G. H., and C. F. VAN LOAN. 1996. Matrix Computations. 3rd edition. Johns Hopkins University Press, Baltimore.

GUSFIELD, D. 1991. Efficient algorithms for inferring evolutionary trees. Networks **21**:19–28.

HASEGAWA, M., H. KISHINO, and T. YANO. 1985. Dating of the human–ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol. **21**:160–174.

JIN, L. and M. NEI. 1990. Limitations of the evolutionary parsimony method of phylogenetic analysis. Mol. Biol. Evol. **7**:82–102.

KIDD, K. K., and L. A. SGARAMELLA-ZONTA. 1971. Phylogenetic analysis: concepts and methods. Am. J. Hum. Genet. **23**:235–252.

KOZLOV, M. K., S. P. TARASOV, and L. G. KHACHIYAN. 1979. Polynomial solvability of convex quadratic programming. Dokl. Akad. Nauk SSSR **248**. [Translated in Soviet Math. Dokl. **20**:1108–1111].

KUHNER, M. K., and J. FELSENSTEIN. 1994. A simulation study of phylogeny algorithms under equal and unequal evolutionary rates. Mol. Biol. Evol. **11**:459–468.

LAKE, J. A. 1994. Reconstructing evolutionary trees from DNA and protein sequences: paralinear distances. Proc. Natl. Acad. Sci. USA **91**:1455–1459.

LOCKHART, P. J., A. W. LARKUM, M. A. STEEL, P. J. WADDELL, and D. PENNY. 1996. Evolution of chlorophyll and bacteriochlorophyll: the problem of invariant sites in sequence analysis. Proc. Natl. Acad. Sci. **93**:1930–1934.

LOCKHART, P. J., M. A. STEEL, M. D. HENDY, and D. PENNY. 1994. Recovering evolutionary trees under a more realistic model of sequence evolution. Mol. Biol. Evol. **11**:605–612.

PENNY, D., M. D. HENDY, and M. A. STEEL. 1992. Progress with methods for constructing evolutionary trees. Trend. Ecol. Evol. **7**:73–79.

PENROSE, R. 1989. The emperor's new mind: concerning computers, minds, and the laws of physics. Oxford University Press, Oxford.

RZHETSKY, A., and M. NEI. 1992*a*. A simple method for estimating and testing minimum evolution trees. Mol. Biol. Evol. **9**:945–967.

———. 1992*b*. Statistical properties of the ordinary least-squares, generalised least-squares, and minimum-evolution methods of phylogenetic inference. J. Mol. Evol. **35**:367–375.

———. 1993. Theoretical foundation of the minimum-evolution method of phylogenetic inference. Mol. Biol. Evol. **10**: 1073–1095.

SAITOU, N., and T. IMANISHI. 1989. Relative efficiencies of the Fitch-Margoliash, maximum-parsimony, maximum-likelihood, minimum-evolution, and neighbor-joining methods of phylogenetic tree reconstruction in obtaining the correct tree. Mol. Biol. Evol. **6**:514–525.

SAITOU, N., and M. NEI. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **24**:189–204.

SATTATH, S., and A. TVERSKY. 1977. Additive similarity trees. Psychometrika **42**:319–345.

SCHRÖDER, E. 1870. Vier combinatorische Probleme. Z. Math. Phys. **15**:361–376.

SWOFFORD, D. L. 1997. Phylogenetic analysis using parsimony. Version 4.0 (PAUP* 4.0). Sinauer, Sunderland, Mass.

SWOFFORD, D. L., G. J. OLSEN, P. J. WADDELL, and D. M. HILLIS. 1996. Phylogenetic inference. Pp. 407–514 *in* D. M. HILLIS, C. MORITZ, and B. K. MABLE, eds. Molecular systematics. 2nd edition. Sinauer, Sunderland, Mass.

VACH, W. 1989. Least squares approximation of additive trees. Pp. 230–238 *in* O. OPITZ, ed. Conceptual and numerical analysis of data. Springer-Verlag, Berlin.

VACH, W., and P. O. DEGENS. 1991. Least squares approximation of additive trees to dissimilarities—characterisations and algorithms. Comput. Stat. Q. **3**:203–218.

WADDELL, P. J., P. O. LEWIS, and D. L. SWOFFORD. 1998. Distance based methods of inferring evolutionary trees. Chapter 4 *in* D. L. SWOFFORD, ed. Phylogenetic analysis using parsimony. Version 4.0 (PAUP* 4.0). Sinauer, Sunderland, Mass.

WADDELL, P. J., and M. A. STEEL. 1997. General time reversible distances with unequal rates across sites: mixing gamma and inverse Gaussian distributions with invariant sites. Mol. Phylogenet. Evol. **8**:398–414.

NARUYA SAITOU, reviewing editor

Accepted July 7, 1998