Rapid evaluation of Least Squares and Minimum Evolution Criteria on Phylogenetic Trees*

David Bryant† Peter Waddell

CRM-2580

November 1998

*A previous version of this manuscript was published in *Molecular Biology and Evolution* 15(10):1346–1359. 1998.

†Centre de recherches mathématiques, Université de Montréal, C.P. 6128, succ. Centre-ville, Montréal, Québec H3C 3J7, Canada; email: bryant@CRM.UMontreal.ca; phone: (514) 343-7501; fax: (514) 343-2254

Abstract

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. These 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 generalised least squares (GLS) edge lengths (where N is the number of taxa in the tree). The Minimum Evolution 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, or faster, than all 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 equally well adapted to multifurcating trees as 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 their use for evaluating trees and their edge lengths (e.g., for approximate divergence time estimates), since they should be more statistically efficient than OLS.

Keywords: Least squares method; Minimum Evolution; Phylogenetic inference; Tree Statistics; Algorithms.

Running Head: Rapid evaluation of Least Squares and ME on trees.

Résumé

Nous présentons de nouveaux algorithmes rapides, utlisant la méthode des moindres carrés, pour estimer les longueurs des arcs dans un arbre de phylogénie. On note N le nombre de données taxinomiques dans l'arbre. Le premier algorithme est un algorithme optimal en temps $O(N^2)$, qui calcule les longueurs des arcs dans un arbre. Cet algorithme est basé sur la méthode ordinaire des moindres carrés non pondérés. Le deuxième algorithme présenté est en temps $O(N^3)$. Il utilise la méthode des moindres carrés pondérés, ainsi que la méthode de Fitch-Margoliash. Le troisième algorithme est en temps $O(N^4)$. Il est basé sur la méthode des moindres carrés généralisés. Ces algorithmes permettent une évaluation rapide, ou optimale, de l'évolution minimale et des moindres carrés. Ces deux critères sont les plus fréquemment utilisés pour estimer les arbre phylogéniques à partir de données de distance. Ces algorithmes ont été inclus dans le programme PAUP*, largement utilisé.

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). 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 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 optimise the fit of data to an assumed model, rather than 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 generalised least squares (GLS). This last criterion is closely related to the maximum likelihood tree estimation of Felsenstein (1981), assuming that the only data available are the distances (see Felsenstein, 1988; Waddell et al., 1998, for further discussion).

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., Cavalli-Sforza and Edwards, 1967; Kidd and Sgaramella-Zonta, 1971; Saitou and Imanishi, 1989; Rzhetsky and Nei, 1992a; Swofford et al., 1996; Waddell et al., 1998). We denote these methods by 'ME' followed by the optimality criterion used to estimate edge lengths, e.g., ME(OLS) is the minimum evolution method studied by Rzhetsky and Nei (1992a,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, or faster, than all 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 (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, for example, Rzhetsky and Nei, 1993; Felsenstein, 1997) however those unfamiliar with the concepts can consult the introductory material in Day (1986) or Penrose (1989, p. 140–145). Roughly, 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 lengths 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 algorithm (1971) was the first time optimal algorithm for parsimony. The algorithms we describe here are the first 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. Later, 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 not only provides 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 1989; Lockhart et al., 1994) just as with parsimony (Felsenstein, 1978; Swofford et al., 1996). Trees on larger sets of taxa may have these longer edges broken up so 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 et al., 1998). An additional advantage of WLS and GLS estimation is that they give more reliable estimates of a trees 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 will adopt the vector notation used by Rzhetsky and Nei (1992a) 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 d to denote a general distance and 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 $\mathrm{d} = (\mathrm{d}_{12}, \mathrm{d}_{13}, \mathrm{d}_{14}, \mathrm{d}_{23}, \mathrm{d}_{24}, \mathrm{d}_{34})^{\mathrm{t}}$, 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 A. The columns of A correspond to edges of T and the rows of 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 one in row ij, column k, otherwise we put a zero. Using this notation we can write the relationship between edge lengths and taxon to taxon distances:

$$\mathrm{p} = \mathrm{Ab}. \tag{1}$$

Here p is the column vector for the taxon to taxon distances, A is the topological matrix, and b is a column vector of edge lengths.

Note that, following Lockhart et al. (1994), 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 the 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 one apart if they are on different sides of the split and distance zero apart if they are on the same side of the split. The columns of the topological matrix 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.

Ordinary Least Squares (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 d, the measure of misfit being the sum of squares distance. In terms of our vector notation, we want to find b that minimises

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

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 Cavalli-Sforza and Edwards (1967). Straightforward projection theory gives the solution

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

(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, though 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 in Rzhetsky and Nei (1993), whose formulae for edge lengths also give an $O(N^3)$ time algorithm, though 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, though it is restricted to binary trees only.

Weighted Least Squares (WLS)

The weighted least squares method for calculating edge lengths involves the minimisation of

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

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

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

If we use standard matrix multiplication then 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, and Swofford, personal communication).

Generalised Least Squares (GLS)

The function to be minimised when using generalised least squares is

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

where V, and hence 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

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

Now V is an $\binom{N}{2} \times \binom{N}{2}$ matrix so calculating 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 are given by Agresti (1990, pp. 460–462) (for any multi-variate model), Hasegawa et al. (1985); Bulmer (1991) (for trees) as

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

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

Results

The main results of this paper are as follows:

(1) An algorithm to calculate $\mathrm{A}^{\mathrm{t}}\mathrm{d}$ in minimal time.

(2) Application of this 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 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 $\mathrm{A^t d}$ 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 (equations 3,5 and 7) and count the number of times a vector (or matrix) is multiplied on the left by $\mathrm{A^t}$, the transpose of the topological matrix.

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

The first step in the fast method is the calculation of $\delta_\mathrm{i}^\mathrm{t}\mathrm{d}$ 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

$$\delta_\mathrm{i}^\mathrm{t}\mathrm{d} = \sum_{y \in L-x} \mathrm{d}_{xy}. \tag{9}$$

Now for the internal edges. We consider binary trees first. The speed up relies on the following relationship between $\delta_\mathrm{i}^\mathrm{t}\mathrm{d}$ and $\delta_\mathrm{j}^\mathrm{t}\mathrm{d}$ for adjacent edges e_i and e_j.

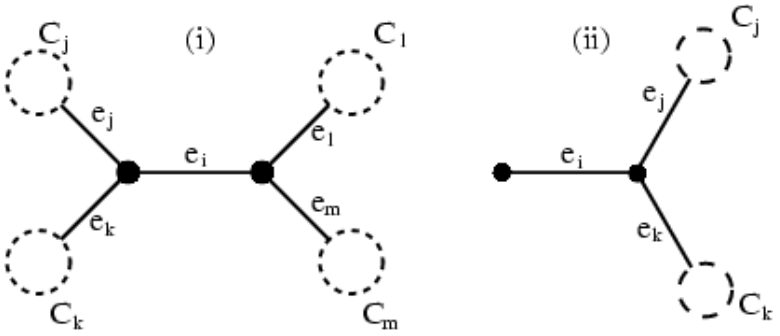


Figure 1: Any internal edge e_i of a binary tree can be drawn in the form of (i): the subtrees branching off e_i are represented by dotted circles. An external edge e_i of a binary tree can be drawn in the form of (ii): the subtrees branching off e_i are represented by dotted circles.

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 end point of e_i. Let C_j, C_k and C_i be corresponding clusters (see figure 1, noting that $C_i = C_l \cup C_m$). Then*

$$\delta_\mathrm{i}^\mathrm{t}\mathrm{d} = \delta_\mathrm{j}^\mathrm{t}\mathrm{d} + \delta_\mathrm{k}^\mathrm{t}\mathrm{d} - 2 \sum_{x \in C_j, y \in C_k} \mathrm{d}_{xy} \tag{10}$$

Proof
We first write out $\delta_\mathrm{i}^\mathrm{t}\mathrm{d}$, $\delta_\mathrm{j}^\mathrm{t}\mathrm{d}$ and $\delta_\mathrm{k}^\mathrm{t}\mathrm{d}$, remembering that C_j, C_k and C_i are disjoint:

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

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

$$\delta_\mathrm{k}^\mathrm{t}\mathrm{d} = \sum_{x \in C_k, y \in L-C_k} \mathrm{d}_{xy} = \sum_{x \in C_k, y \in C_i} \mathrm{d}_{xy} + \sum_{x \in C_k, y \in C_j} \mathrm{d}_{xy} \tag{13}$$

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

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

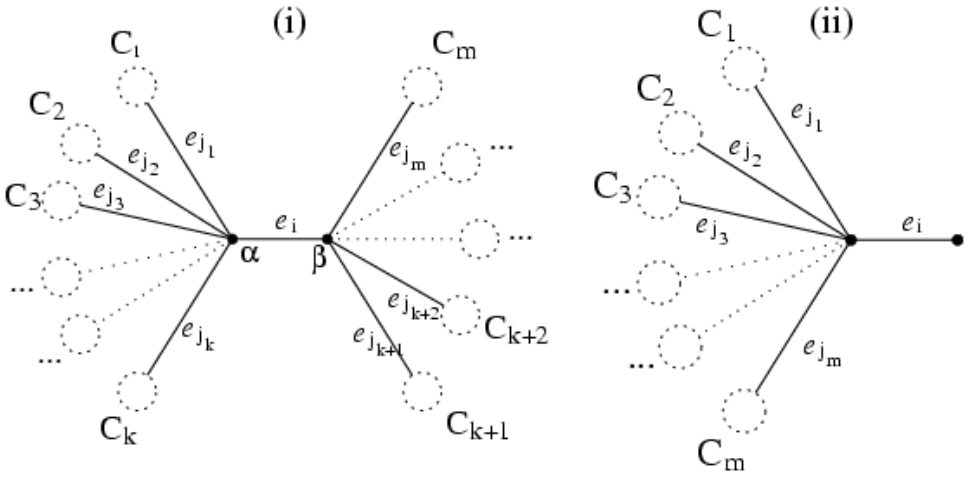


Figure 2: Generic cases for calculating edge lengths. (i) Internal edge in a non-binary tree. (ii) External edge in a non-binary tree.

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 $C_{j_1}, C_{j_2}, \ldots, C_{j_k}$ and C_i be the corresponding clusters (see figure 2). Then*

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

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

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

To formalise 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 d.

1. For each external edge e_i put

$$\delta_i^{\mathrm{t}}\mathrm{d} \leftarrow \sum_{y\in L-\{x\}} \mathrm{d}_{xy}$$

   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, and then the unlisted edges adjacent to these edges, and so on until all the internal edges are in the list.
3. Working *backwards* through the list, calculate $\delta_i^{\mathrm{t}}\mathrm{d}$ 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 $\mathrm{A}^{\mathrm{t}}\mathrm{d}$ with entries $\delta_i^{\mathrm{t}}\mathrm{d}$.

end.

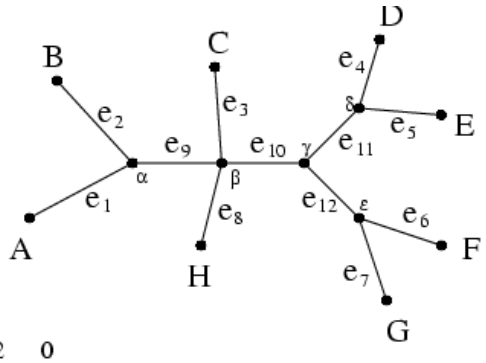
Algorithm 1: FastMTM(T,d)

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 $\frac{1}{2}N(3N-1)$ operations. The amount of memory used, in addition to the memory required to store the vector* 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 d in the same figure.



	A	B	C	D	E	F	G	H
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

Figure 3: Example tree T and test distance matrix d for taxon set $\{A, B, C, D, E, F, G, H\}$. Internal vertices are labelled with greek characters $\alpha, \beta, \gamma, \delta, \epsilon$ for future reference.

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

$$\begin{aligned} \delta_1^{\mathrm{t}}\mathrm{d} &= \mathrm{d}_{AB} + \mathrm{d}_{AC} + \cdots + \mathrm{d}_{AH} && (15)\\ &= 80. && (16)\end{aligned}$$

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

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

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

$$\begin{aligned} \delta_9^{\mathrm{t}}\mathrm{d} &= \delta_1^{\mathrm{t}}\mathrm{d} + \delta_2^{\mathrm{t}}\mathrm{d} - 2\mathrm{d}_{AB} && (17)\\ &= 136 && (18)\end{aligned}$$

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

$$\begin{aligned} \delta_{10}^{\mathrm{t}}\mathrm{d} &= \delta_3^{\mathrm{t}}\mathrm{d} + \delta_9^{\mathrm{t}}\mathrm{d} + \delta_8^{\mathrm{t}}\mathrm{d} - 2\left(\sum_{x\in C_1, y\in C_2} \mathrm{d}_{xy} + \sum_{x\in C_1, y\in C_3} \mathrm{d}_{xy} + \sum_{x\in C_2, y\in C_3} \mathrm{d}_{xy}\right) && (19)\\ &= 283 - 2((\mathrm{d}_{AC} + \mathrm{d}_{BC}) + \mathrm{d}_{CH} + (\mathrm{d}_{AH} + \mathrm{d}_{BH})) && (20)\\ &= 201. && (21)\end{aligned}$$

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

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 (equations 3, 5 and 7). In the first two cases the new algorithms (described below) match the speed of the 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 V^{-1} is $O(N^4)$ and none of these are redundant. Thus, no future algorithm for calculating GLS edge lengths when the matrix 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.

Weighted least squares and Ordinary least squares

Edge lengths under WLS are given by the projection formula

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

The edge lengths under OLS can be obtained by putting W = 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, w is the diagonal of the weight matrix W.

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

end.

Algorithm 2: WLSedges(T,w,d)

Let $f(K)$ and $g(K)$ be, respectively, the number of operations and memory required to solve the $K \times K$ problem $\mathrm{X}\mathrm{b} = \mathrm{y}$ for b. A simple count of the number of operations, and 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 $\mathrm{X}\mathrm{b} = \mathrm{y}$. The most attractive is Cholesky factorisation, 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 non-negative edge lengths can be included by replacing step 6 with

6'. Find b that minimizes $(\mathrm{X}\mathrm{b} - \mathrm{y})^{\mathrm{t}}(\mathrm{X}\mathrm{b} - \mathrm{y})$ such that $\mathrm{b}_i \geq 0$ for all i.

which is a quadratic programming problem in standard form. Optimised 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.

Generalised least squares

Edge lengths under GLS are given by the projection formula

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

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.

Once again, let $f(K)$ and $g(K)$ be, respectively, the number of operations and memory required to solve the $K \times K$ problem $\mathrm{X}\mathrm{b} = \mathrm{y}$ for 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 V^{-1}, is $O(N^3) + g(K)$.

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

end.

Algorithm 3: GLSedges(T,V^{-1},d)

Note that the variance-covariance matrix for edge lengths (e.g., Agresti, 1990; Hasegawa et al., 1985; Bulmer, 1991) can be obtained using algorithm GLSedges and inverting the matrix $X = A^t V^{-1} A$. This inversion takes $O(N^3)$ time, making $O(N^4)$ 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. 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 (equation 3), we see that the major obstacle to an $O(N^2)$ algorithm is the construction and inversion of the matrix $(A^t A)^{-1}$. We need to explore and utilise 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 $(A^t A)^{-1}$ is that it is mainly 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 $\delta_i^t d$, $\{\delta_{j_l}^t d : e_{j_l} \text{ adjacent to } e_i\}$ 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).

To ease understanding we present here only the formulae 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 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, N_m be the number of taxa in the subtrees branching off e_j, e_k, e_l, e_m respectively. The optimal edge length b_i for e_i under OLS is given by:

$$\begin{aligned} b_i &= \Big((\frac{N}{N_m} + \frac{N}{N_l} + \frac{N}{N_k} + \frac{N}{N_j} - 4)\delta_i^t d + \frac{N_j + N_k}{N_j N_k}((2N_k - N)\delta_j^t d + (2N_j - N)\delta_k^t d) \\ &\quad + \frac{N_l + N_m}{N_l N_m}((2N_m - N)\delta_l^t d + (2N_l - N)\delta_m^t d)\Big) \frac{1}{4(N_j + N_k)(N_l + N_m)}. \end{aligned} \tag{24}$$

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

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

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

$$\delta_l^{\mathrm{t}}\mathrm{d} = \mathrm{d}_{AC} + \mathrm{d}_{BC} + \mathrm{d}_{CD} \quad (27)$$
$$\delta_m^{\mathrm{t}}\mathrm{d} = \mathrm{d}_{AD} + \mathrm{d}_{BD} + \mathrm{d}_{CD} \quad (28)$$
$$\delta_i^{\mathrm{t}}\mathrm{d} = \mathrm{d}_{AC} + \mathrm{d}_{BC} + \mathrm{d}_{AD} + \mathrm{d}_{BD} \quad (29)$$

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 b_i for e_i under OLS is given by:

$$\mathrm{b}_i = \frac{1}{4(N_j N_k)}\left((1 + N_j + N_k)\delta_i^{\mathrm{t}}\mathrm{d} - (1 + N_j - N_k)\delta_j^{\mathrm{t}}\mathrm{d} - (1 - N_j + N_k)\delta_k^{\mathrm{t}}\mathrm{d}\right). \quad (30)$$

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

```
1.    Count the number of taxa on each side of each edge.
2.    Calculate δ_i^t d for each edge using FastMTM.
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.
```

Algorithm 4: BinaryEdges(T,d)

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* d.

Proof in Appendix B.2.

Though 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 $k = 9$. Hence $N_j = 1$ and $N_k = 6$. The values $\delta_1^{\mathrm{t}}\mathrm{d} = 80$, $\delta_2^{\mathrm{t}}\mathrm{d} = 70$ and $\delta_9^{\mathrm{t}}\mathrm{d} = 136$ 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 $m = 5$, so $N_j = 4$, $N_k = 2$, $N_l = 1$ and $N_m = 1$. Substituting these values into the equation we obtain $b_{11} = 1325/512$.

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), 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 $\binom{N}{2}$ 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 get OLS least squares 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, 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 $\mathrm{p}_{ac} = \mathrm{p}_{av} + \mathrm{p}_{cv}$ and $\mathrm{p}_{bc} = \mathrm{p}_{bv} + \mathrm{p}_{cv}$.

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. The observation enables us to avoid this repetition. It is more efficient to calculate all vertex to vertex path lengths rather than 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.

```
1.   For each taxon a in T do
2.   p_aa ← 0
3.   Repeat until p_ax has been calculated for all vertices x in T
4.     Choose a vertex v such that p_av has not been calculated but v is
       adjacent to a vertex u for which p_au has been calculated.
5.     p_av ← 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 adjacent 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.
```

Algorithm 5: DistanceInTree(T)

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

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

calculated using BinaryEdges and FastOLS (see Appendix A). The internal vertices of the tree are conveniently labelled by greek characters $\alpha, \beta, \gamma, \delta, \epsilon$.

First of all, $\mathrm{p}_{A\alpha} = b_1 = 13/3$. Then $\mathrm{p}_{AB} = \mathrm{p}_{A\alpha} + b_2 = 7$ and $\mathrm{p}_{A\beta} = \mathrm{p}_{A\alpha} + b_9 = 253/48$. From there we can calculate $\mathrm{p}_{AB}, \mathrm{p}_{AB}$ and p_{AB}, and so on until 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,

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

The optimal edge lengths b are provided by the algorithm BinaryEdges if T is binary and FastOLS (Appendix A) if T is multifurcating. The calculation of 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 Ab takes $O(N^2)$ time and so the entire calculation

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

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 by K (positive definite) linear system.

By a similar method the sum of squares calculation

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

can be completed in $O(N^4)$ time, provided that, as above, the inverse matrix 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 by K (positive definite) linear system.

Table 1: Summary of evaluation speed increases

Evaluation criterion	Algorithm(s)	Complexity bound	Complexity 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

Summary of Results

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

We compare the worst case performance of the algorithm BinaryEdges to 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 of operations taken by BinaryEdges applied to any binary tree with N leaves. The number of operations taken by 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 subsequently *underestimated* the increase in speed given by BinaryEdges. The figure of $2N^3/3 + 3N^2 + 94N/3 + 8$ 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 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 is laborious and possibly confusing (further, 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 multi-bit 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 faster than 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 the new algorithms are indeed substantially 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 summarised 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, sum of squares evaluation takes an additional $N(N-1)$ operations on top of that required to calculate edge lengths. In the case of OLS this is a significant increase—as much as 60case of WLS and GLS the extra time required to calculate sums of squares is negligible, relative to the time taken to calculate edge lengths.

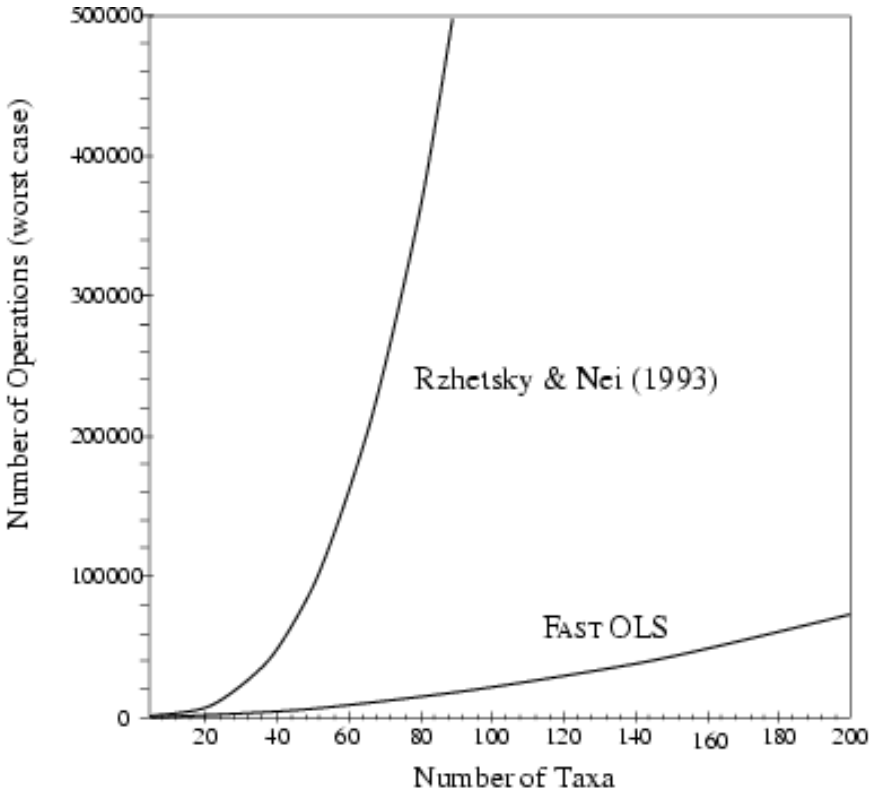


Figure 4: Plots of the worst case number of operations for the FastOLS algorithm and the worst case number of operations for the algorithm of Rzhetsky and Nei (1993). Note that when $N = 200$ Rzhetsky and Nei's algorithm can take at over 5 million operations while FastOLS takes under 75,000.

We also note that for moderate values of N the time taken by the GLS algorithms is only a few times more than that taken by the WLS algorithms. As a general rule of thumb, when there are N taxa the GLS methods take $N/8$ times longer than WLS. For example when $N = 50$ the GLS algorithm take between 7 and 8 times longer than the WLS algorithms (and about 1200 times longer than OLS!).

Discussion

The speed up 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 $W_{ij} = 1/(d_{ij})^2$) 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 rate of 1.68 trees per second by the previously implemented Felsenstein (1997) algorithm. Note that the algorithms of Sattath and Tversky (1977) and Rzhetsky and Nei (1993) were not considered practical for PAUP*, due to their inabiliy to evaluate non-binary trees (a key feature in the program, especially when using star decomposition searches).

Rzhetsky and Nei (1992a) describe an alternative localised, 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 faster. See also the fast 1-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 et al., 1998). To date there is no definitive answer and a good deal of disagreement. There are some who feel negative edge lengths should be avoided wherever possible, and propose that any tree containing a negative edge is automatically rejected Kidd and Sgaramella-Zonta (1971, e.g.,). Another

approach is to define the ME score of a tree as

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

thereby penalising 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 non-negative. This is not simply a matter of contracting negative edges to zero then re-optimising the remaining edge lengths: we have discovered a seven taxa tree for which this approach fails (Bryant and Waddell, unpublished results). 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 et al., 1979; Goldfarb and Liu, 1991). Recently, Bryant and Swofford (1998) have reported that the constrained OLS problem can be quickly and accurately solved by combining the FastOLS algorithm with a modified active set method. On 128 taxa trees their algorithm runs over 100 times faster that the iterative algorithm of Felsenstein (1997) and produces more accurate solutions. The new method has been incorporated into PAUP*.

The speed increases 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 other 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 the 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 algorithms, and checking for typographic errors. This work was supported by a University of Canterbury Doctoral Scholarship (D.B.) and the Marsden Fund (96 MAU-ALS 0042 (P.W.), and 95 UOC-516 (D.B.)).

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. Though the equations are longer, the basic idea is the same: we calculate the quantities $\delta_i^{\mathrm{t}}\mathrm{d}$ 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 $e_{j_1}, e_{j_2}, \ldots e_{j_k}$ 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 v be the vector with N_β in positions $1, \ldots, k$ and N_α in positions $k+1, \ldots, m$.

We use similar labelling 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 v be a vector of m ones, 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 b_i for e_i under OLS is given by:*

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

where N is the number of taxa,

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

and

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

where the matrix U *is the* $m \times m$ *matrix of ones,* N *is the diagonal matrix with diagonal* $N_1, N_2, \ldots, N_m$ *and* 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 $(N\mathrm{N}^{-1} - 2\mathrm{I} + \mathrm{U})$ is invertible.

There are explicit formulae for the elements of w, with two cases to consider.

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

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

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

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

whereas

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

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 summarise 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*

$$26K^2 - 104NK + 211N^2/2 + 182K - 413N/2 - 3 \tag{43}$$

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

The algorithm may appear daunting, but it is little more than an explicit implementation of equations (37) to (42). We demonstrate the use of this algorithm by calculating the optimal edge lengths b_8, b_9 and b_{10} for the example tree and distance in figure 3. We will use the values for $\delta_i^{\mathrm{t}}\mathrm{d}$ calculated earlier.

First 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 $N_1 = 2$, $N_2 = 1$ and $N_3 = 4$; $N_\alpha = 7$, $N_\beta = 1$ and $k = m = 3$. The vector v in step 9 becomes $\mathrm{v} = (1, 1, 1)^{\mathrm{t}}$.

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

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

$$\begin{aligned} \mathrm{b}_8 &= \left(\delta_8^{\mathrm{t}}\mathrm{d} - (\frac{\delta_{10}^{\mathrm{t}}\mathrm{d}}{N_3}) \right) /(7 - 1) && (44) \\ &= 163/24 && (45) \end{aligned}$$

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, N_5 equal $1, 1, 1, 4$ and 1 respectively, $N_\alpha = 2$ and $N_\beta = 6$. The vector v in step 9 becomes $\mathrm{v} = (6, 6, 2, 2, 2)^{\mathrm{t}}$.

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

$$\begin{aligned} \mathrm{b}_9 &= \left(\delta_9^{\mathrm{t}}\mathrm{d} - (\frac{4}{6} \cdot \frac{\delta_2^{\mathrm{t}}\mathrm{d}}{N_1} + \frac{4}{6} \cdot \frac{\delta_1^{\mathrm{t}}\mathrm{d}}{N_2} + 0 + \frac{4}{6} \cdot \frac{\delta_{10}^{\mathrm{t}}\mathrm{d}}{N_4} + 0) \right) /(12 - 28/3) && (46) \\ &= 15/16. && (47) \end{aligned}$$

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, N_5 equal $1, 2, 1, 2$ and 2 respectively, $N_\alpha = 4$ and $N_\beta = 4$. The vector v in step 9 becomes $\mathrm{v} = (4, 4, 4, 4, 4)^{\mathrm{t}}$.

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

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

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

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* d*, is* $O(N)$*.*

Proof
We discuss each step of the algorithm in turn.

Step 1. We can calculate $\delta_i^{\mathrm{t}}\mathrm{d}$ 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 inputted 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 $\delta_i^{\mathrm{t}}\mathrm{d}$ for at most one edge e_i. Similarly, each value $\delta_i^{\mathrm{t}}\mathrm{d}$, once calculated, is only used once when calculating the other values $\delta_j^{\mathrm{t}}\mathrm{d}$. The calculation of $\delta_i^{\mathrm{t}}\mathrm{d}$ also requires one additional multiplication (by 2) and a subtraction. Thus the total number of operations required in step 3. is $\binom{N}{2} + N + 2N$ and the total number of operations is at most

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

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 $\delta_i^{\mathrm{t}}\mathrm{d}$. □

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* 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 $\frac{1}{2}N(3N-1)$ operations by Theorem 3.
Step 4. takes at most 40 operations for internal edges and 20 operations for external edges. Therefore steps 3. through 5. take $20N + 40(N-3) = 60N - 120$ operations.
Counting all steps together we have that the algorithm takes at most

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

operations.

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

B.3 FastOLS

Theorem 6 *The algorithm* FastOLS *takes at most*

$$26K^2 - 104NK + 211N^2/2 + 182K - 413N/2 - 3 \tag{51}$$

operations when applied to a N *taxa distance matrix and a tree with* K *edges. It requires* $O(N)$ *memory space in addition to that required to store the vector* 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 $\frac{1}{2}N(3N-1)$ 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 w. A further $5m+1$ operations are required to calculate 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

$$26K^2 - 104NK + 211N^2/2 + 182K - 413N/2 - 3. \tag{52}$$

The only memory required, in addition to that used by the input distance, is that used for storing the values $\delta_i^t d$, the edges, vertices, and cluster sizes of the tree, the vectors v and 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 in binary trees. We analyze the time complexity of that algorithm, as presented in Appendix B of Rzhetsky and Nei (1993).

They calculate the length of an internal edge b using the formula

$$\hat{b} = \sum_{i<j} \omega_{ij} \mathrm{d}_{ij} \tag{53}$$

where 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 that paper takes (at least) 36 operations for internal edges and 7 operations for external edges. If the multiplication and summation is carried out over all pairs of taxa i, j then the number of operations per edge is at least $36 + N(N-1)$ or $7 + N(N-1)$, making

$$(N-3)(36 + N(N-1)) + N(7 + N(N-1)) = 2N^3 - 5N^2 + 46N - 108 \tag{54}$$

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 sum over all pairs i, j we can derive the number of operations required by the algorithm

$$36(N-3) + 7N + 2\sum_{i<j}(2\rho_{ij} - 1) = 44N - N^2 - 108 + 4\sum_{i<j}\rho_{ij} \tag{55}$$

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

The quantity $\sum_{i<j}\rho_{ij}$ 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 out analysis to the case when the tree is a caterpillar tree (a single long path with external edges branching off), 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

$$\sum_{i<j}\rho_{ij} = \frac{1}{6}N^3 + N^2 - 19N/6 + 2. \tag{56}$$

Combining all factors together, we have

Theorem 10 *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)

References

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, NZ.

Bryant, D.J., and D. Swofford. 1998. Fast algorithms for constrained least squares branch length estimation. In preparation.

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

Cavalli-Sforza, L., and A. Edwards. 1967. Phylogenetic analysis models and estimation procedures. Evolution **32**:550–570.

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

Felsenstein, J. 1978. The number of evolutionary trees. Syst. Zool. **27**:27-33.

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

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

Felsenstein, J. 1988. Phylogenies from molecular sequences: Inference and reliability. Ann. Rev. Genet. **22**:521-565.

Felsenstein, J. 1993. PHYLIP (Phylogeny Inference Package) and manual, version 3.5c. Department of Genetics, University of Washington, Seattle.

Felsenstein, J. 1997. An alternating least squares approach to inferring phylogenies from pairwise distances. Syst. Biol. **46**(1):101–111.

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

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

Gascuel, O. 1996. Algorithmes efficaces pour la construction d'arbes d'évolution minimum. Rapport LIRMM 96019, 28 pages.

Gascuel, O. 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.

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

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

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

Golub, G.H., and C.F. Van Loan. 1996. Matrix Computations. (3rd Edition), John Hopkins University Press, Baltimore.

Hasegawa,M., and 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.

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

Kozlov, M.K., S.P. Tarasov and L.G. Khachiyan. 1979. Polynomial solvability of convex quadratic programming. Doklady Akademii Nauk SSSR **248**. [Translated in Soviet Mathematics Doklady **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., M.A. Steel, M.D. Hendy and D. Penny. 1994. Recovering evolutionary trees under a more realistic model of sequence evolution. Mol. Bio. Evol. **11**:605–612.

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

Rzhetsky, A., and M. Nei. 1992a. A simple method for estimating and testing minimum evolution trees. Mol. Bio. Evol. **9**:945–967.

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

Rzhetsky, A,. and M. Nei. 1993. Theoretical foundation of the minimum-evolution method of phylogenetic inference. Mol. Bio. 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. Molecular Biology and Evolution. **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 Under Parsimony, Version 4.0 (PAUP* 4.0). Sinaur Associates, 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, second 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.

Vach, W., and P.O. Degens. 1991. Least squares approximation of additive trees to dissimilarities—characterisations and algorithms. Computational Statistics Quarterly **3**:203–218.

Waddell, P.J., P.O. Lewis and D.L. Swofford. 1998. Distance based methods of inferring evolutionary trees (Manual chapter 4) *in* Swofford, D.L. Phylogenetic Analysis Under Parsimony, Version 4.0 (PAUP* 4.0). Sinaur Associates, 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. Molecular Phylogenetics and Evolution **8**: 398-414.

1. Count the number of taxa on each side of each edge.
2. Calculate $\delta_i^{\mathrm{t}}\mathrm{d}$ 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 diagram (i), figure 2. For each l let N_l be the number of taxa in C_l. Let N_α and N_β 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 diagram (ii), figure 2. 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 v be the vector with N_β in positions $1, \ldots, k$ and N_α in positions $k+1, \ldots, m$.
10. If there is λ such that $N_\lambda = N/2$ then
11. For $l = 1, \ldots, m$ except λ do
12. $\mathrm{w}_l \leftarrow \frac{\mathrm{v}_l - \mathrm{v}_\lambda}{N/N_l - 2}$
13. end(for)
14. $\mathrm{w}_\lambda \leftarrow \mathrm{v}_\lambda + \sum_{j \neq \lambda} \left(\frac{N_j(\mathrm{v}_\lambda - \mathrm{v}_j)}{N - 2N_j} \right)$.
15. else
16. $\kappa \leftarrow 1 + \sum_{j=1}^{m} \frac{N_j}{N - 2N_j}$
17. $\gamma \leftarrow \frac{-1}{\kappa} \sum_{j=1}^{m} \frac{\mathrm{v}_j}{(N/N_j - 2)}$
18. For $l = 1, \ldots, m$ do
19. $\mathrm{w}_l \leftarrow \frac{1}{(N/N_l - 2)}(\gamma + \mathrm{v}_l)$
20. end (for)
21. end (if-else)
22. $$\mathrm{b}_i \leftarrow \left(\delta_i^{\mathrm{t}}\mathrm{d} - \sum_{l=1}^{m} \frac{w_l \delta_{j_l}^{\mathrm{t}}\mathrm{d}}{N_l} \right) / \left(N_\alpha N_\beta - \sum_{j=1}^{m} \mathrm{w}_j \mathrm{v}_j \right)$$
23. end (For each edge e_i)

end.

Algorithm 6: FastOLS(T,d)