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

No. 154 June, 1997

Keywords. Least squares method — Minimum Evolution — Phylogenetic inference — Algorithms.

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 branch (edge) 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) and the Fitch-Margoliash method; and an optimal $O(N^4)$ time algorithm for generalised least squares edge lengths. 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. These algorithms are substantially faster than all those previously published, and the algorithms for OLS and GLS are the fastest possible (with respect to order of computational complexity).

An optimal algorithm for determining path lengths in 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 considerable increases in speed enable far more extensive tree searches and statistical evaluations (e.g. bootstrap, parametric bootstrap or jackknife). 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.

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 popular LogDet transformation which compensates for variable base composition (Lake 1994, Lockhart et al. 1994). They 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 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 for 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. 1997 for further discussion).

Another set of optimality criteria emerge 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 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 1992a, Swofford et al. 1996, Waddell et al. 1997). 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, 1992b, 1993), and recently made available 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 (Felsenstein 1978, Swofford et al. 1996). This makes speedy evaluation of the selection criteria for each tree highly desirable.

The speed of an algorithm is usually described in terms of order notation $O()$. For example, the algorithm for OLS edge lengths presented here takes $O(N^2)$ time where previously developed algorithms took at best $O(N^3)$ time

(e.g. Sattath and Tversky 1977, Vach 1989, Vach and Degens 1991, Rzhetsky and Nei 1993). When N is large, say $N > 100$, an $O(N^3)$ time algorithm takes at most $C_1 N^3$ operations, for some constant C_1, while an $O(N^2)$ time algorithm takes at most $C_2 N^2$ operations, for some constant C_2. Practically, this means that $O(N^2)$ time algorithms will run much quicker on large taxa sets then an $O(N^3)$ algorithm, and faster algorithms mean larger taxa sets can be processed and more extensive tree searches completed. Note that the exact speeds, and the values for C_1 and C_2, are always dependent upon aspects of computer architecture and algorithm implementation.

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. 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. We expect them to run N times faster than existing algorithms for even small N: that means approximately 100 times faster on data sets of 100 taxa and 200 times faster on data sets of 200 taxa, and so on.

Recent advances in sequencing technology have given rise to increasingly large data sets. The analyses of large numbers of sequences not only 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 1989, Lockhart et al. 1996) just as with parsimony (Felsenstein 1978a, Swofford et al. 1996). Trees on larger sets of taxa 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). 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. 1997). 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. 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 put $N = |L|$. A **distance** on L, possibly given by an evolutionary distance, is represented by a column vector with $\frac{N(N-1)}{2}$ entries. Each entry 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.

Another useful concept is that of the **topological matrix** of a phylogenetic tree, 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 leaves in L. If the path connecting two leaves 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 branch lengths and leaf to leaf distances:

$$\mathbf{p} = \mathbf{A}\mathbf{b}. \tag{1}$$

Here $\mathbf{p}$ is the column vector for the leaf to leaf distances, $\mathbf{A}$ is the topological matrix, and $\mathbf{b}$ is a column vector of branch lengths.

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 leaf 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 $\mathbf{A}$ 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.

Ordinary Least Squares (OLS)

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

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

where the elements of $\mathbf{b}$ may take on any real value (including zero or negative values).

The problem was apparently first introduced, and solved, by Cavalli-Sforza and Edwards (1967). Straightforward projection theory gives the solution

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

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 Rzhetsky and Nei (1993). Formulae for calculating edge lengths have also been developed by Vach 1989 (see also Vach and Degens 1991).

Weighted Least Squares (WLS)

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

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

where $\mathbf{W}$ is a given diagonal matrix with strictly positive entries on the diagonal and 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 then this vector can be calculated in $O(N^4)$ time. We show later that this bound can be improved to $O(N^3)$ time.

Generalised Least Squares (GLS)

The function to be minimised when using generalised least squares is

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

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

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

Now $\mathbf{V}$ is an $\binom{N}{2} \times \binom{N}{2}$ 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 $\mathbf{b}$ still takes $O(N^5)$ time to compute. We improve this bound to $O(N^4)$.

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

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

This formula takes $O(N^5)$ time using standard matrix multiplication. We improve this bound to $O(N^4)$ time, though in practice this matrix will be calculated as part of the evaluation of GLS edge length estimates.

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 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 distances in a tree when edge lengths are given.

5. Application of the above to give a time optimal algorithm for the evaluation of least squares on trees.

Calculating $\mathbf{A}^T\mathbf{d}$ in minimal time

Notice that the equations (3), (5) and (7) all involved the multiplication of the topological matrix $\mathbf{A}^T$ with some vector, be it $\mathbf{d}$, $\mathbf{Wd}$ or $\mathbf{V}^{-1}\mathbf{d}$. Using standard matrix multiplication, this calculation takes $O(N^3)$ time, which is not surprising since $\mathbf{A}$ has $O(N^3)$ entries. However the topological matrix contains a lot of redundant information. After all, it takes only $O(N)$ information to define a tree (i.e. the vertices and edges). We show how to calculate the vector $\mathbf{A}^T\mathbf{d}$ directly from the tree T and the vector $\mathbf{d}$, completely bypassing the topological matrix.

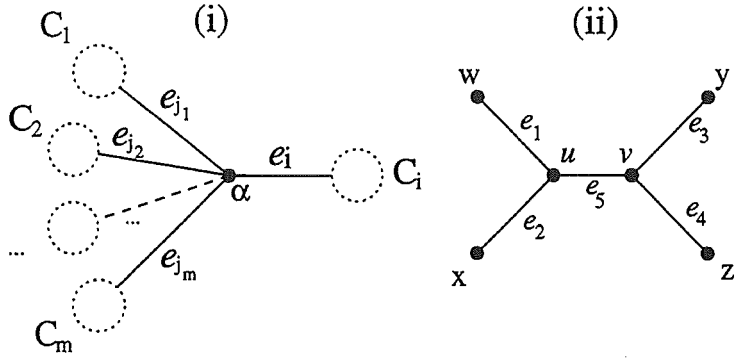

Figure 1 : (i) 'Generic' edge i with its adjacent edges and subtrees. (ii) Four taxa example tree.

We label the edges of T by $e_1, \ldots, e_K$. The matrix $\mathbf{A}$ has exactly K columns, one for every edge of T. Denote these columns by $\delta_1, \delta_2, \ldots, \delta_K$; thus δ_1 is the split metric for the split corresponding to edge e_1. The elements of $\mathbf{A}^T\mathbf{d}$ are the values $\delta_1^T\mathbf{d}, \delta_2^T\mathbf{d}, \ldots, \delta_K^T\mathbf{d}$. Algorithm CALCULATEAD calculates all of these values in $O(N^2)$ time.

Procedure CALCULATEAD(T,$\mathbf{d}$)

1. Calculate $\delta_i^T\mathbf{d}$ directly for all external edges.
2. REPEAT UNTIL $\delta_i^T\mathbf{d}$ has been calculated for all $i = 1, \ldots, K$
3. Choose an internal vertex α such that exactly one of the adjacent edges has *not* had its $\delta_i^T\mathbf{d}$ value calculated. Let i be the index of this edge and let $j_1, \ldots, j_m$ be the indices of the remaining edges adjacent to α. For each $l = i, j_1, j_2, \ldots, j_m$ let C_l be the set of leaves on the other side of edge e_l from α (figure 1 (i)).
4. $$\delta_i^T\mathbf{d} \leftarrow \sum_k \delta_{j_k}^T\mathbf{d} - 2\left(\sum_{k<l}\sum_{a\in C_{j_k}, b\in C_{j_l}} \mathbf{d}_{ab}\right)$$
5. END(REPEAT)
6. END.

Algorithm 1 : CALCULATEAD

In Appendix 1 we prove that the algorithm CALCULATEAD calculates $\mathbf{A}^T\mathbf{d}$ correctly and that it does so in $O(N^2)$ time. Since $\mathbf{d}$ has $O(N^2)$ non-redundant elements this algorithm is time optimal.

As an illustration, consider the four taxa tree (figure 1 (ii)) and the distance vector $\mathbf{d} = (\mathbf{d}_{wx}, \mathbf{d}_{wy}, \mathbf{d}_{wz}, \mathbf{d}_{xy}, \mathbf{d}_{xz}, \mathbf{d}_{yz})^T = (1, 3, 2, 5, 2, 4)^T$. We calculate $\delta_i^T\mathbf{d}$ for the external edges directly: for example $\delta_1^T\mathbf{d} = \mathbf{d}_{wx} + \mathbf{d}_{wy} + \mathbf{d}_{wz} = 6$. Similarly $\delta_2^T\mathbf{d} = 8$, $\delta_3^T\mathbf{d} = 12$ and $\delta_4^T\mathbf{d} = 8$.

The next step is to choose a vertex adjacent to exactly one edge e_i for

which $\delta_i^T \mathbf{d}$ has not been calculated. In this case there are two candidates, u and v. We will choose u. From line 4 of the algorithm we obtain

$$\begin{aligned} \delta_5^T \mathbf{d} &= \delta_1^T \mathbf{d} + \delta_2^T \mathbf{d} - 2d_{wx} && (9) \\ &= 6 + 8 - 2 \times 1 && (10) \\ &= 12. && (11) \end{aligned}$$

Fast algorithms for OLS, WLS and GLS edge lengths

The fast algorithm for calculating $\mathbf{A}^T\mathbf{d}$ leads straight away to fast algorithms for calculating edge lengths.

Ordinary least squares

Edge lengths under OLS are given by the projection formula

$$\mathbf{b} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d}. \tag{12}$$

As mentioned earlier, this calculation takes $O(N^4)$ time using standard matrix multiplication, where N is the number of taxa. However, using the algorithm CALCULATEAd we can compute $\mathbf{A}^T\mathbf{x}$ in $O(N^2)$ time for any vector $\mathbf{x}$ of the appropriate size. Thus for each column δ_i of $\mathbf{A}$ we can calculate $\mathbf{A}^T\delta_i$ in $O(N^2)$ time, so $\mathbf{A}^T\mathbf{A}$ can be computed in $O(N^3)$ time. The inversion takes $O(N^3)$ time and the calculation of $\mathbf{A}^T\mathbf{d}$ takes $O(N^2)$ time so the entire calculation of edge lengths can be completed in $O(N^3)$ time.

We can go even faster! Below we describe an optimal $O(N^2)$ time algorithm for OLS edge lengths.

Weighted least squares

Edge lengths under WLS are given by the projection formula

$$\mathbf{b} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{d}. \tag{13}$$

Using standard matrix multiplication this calculation takes $O(N^4)$ time. Once again, a speed-up is possible.

Working from right to left, we can calculate $\mathbf{W}\mathbf{d}$ in $O(N^2)$ time, assuming that the input contains only the diagonal elements of $\mathbf{W}$. The vector $\mathbf{A}^T\mathbf{W}\mathbf{d}$ can then be calculated in $O(N^2)$ time using the CALCULATEAD algorithm.

The matrix $(\mathbf{WA})$ is equal to $\mathbf{A}$ with the rows scaled, so can be calculated in $O(N^3)$ time. We can then calculate $\mathbf{A}^T(\mathbf{WA})$ by applying the algorithm CALCULATEAD to each of the $O(N)$ columns of $(\mathbf{WA})$, so that $\mathbf{A}^T\mathbf{WA}$ is calculated in $O(N^3)$ time. This matrix is $N \times N$ so its inverse can be calculated in $O(N^3)$ time. Therefore the vector $\mathbf{b}$ of edge lengths can be retrieved in $O(N^3)$ time. This method works for binary and non-binary trees.

We have good reason to believe that the $O(N^2)$ time speed-up for OLS below does not extend to weighted least squares. The OLS formulae takes advantage of symmetries that disappear with the introduction of a scaling matrix $\mathbf{W}$.

Felsenstein (1997) has recently published an algorithm for calculating edge lengths under WLS. It also takes at least $O(N^3)$ time, but unlike the exact algorithm presented here, Felsenstein's algorithm is iterative and is not guaranteed to return the solution in this time.

Generalised least squares

Edge lengths under GLS are given by the projection formula

$$\mathbf{b} = (\mathbf{A}^T\mathbf{V}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{V}^{-1}\mathbf{d}. \tag{14}$$

Using standard matrix multiplication this calculation takes $O(N^5)$ time. We show how to complete it in $O(N^4)$ time.

We begin by using $O(N^2)$ applications of CALCULATEAD to construct $\mathbf{A}^T\mathbf{V}^{-1} = (\mathbf{V}^{-1}\mathbf{A})^T$. Then $\mathbf{A}^T\mathbf{V}^{-1}\mathbf{d}$ takes a further $O(N^3)$ operations, and $O(N^2)$ more applications of CALCULATEAD gives $(\mathbf{A}^T V^{-1}\mathbf{A})$.

The matrix $\mathbf{A}^T\mathbf{V}^{-1}\mathbf{A}$ is of size $N \times N$ so the inversion takes $O(N^3)$ time, giving a total time complexity for calculating edge lengths, and edge length variances and covariances, of $O(N^4)$. This is time optimal because there are $O(N^4)$ entries in $\mathbf{V}^{-1}$, none of which are redundant.

One useful improvement to these methods would be a technique that somehow bypassed the need to calculate $\mathbf{V}^{-1}$. This would not, as explained above, accelerate the evaluation of each tree.

An unusually fast algorithm for OLS edge lengths

We have described an $O(N^3)$ algorithm for calculating OLS edge lengths. This is fast, but only as fast as several existing methods. In this section we develop 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 equation (3), the projection formula for OLS edge lengths, 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 utilise the properties of this matrix so that we can avoid 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 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 $\delta_i^T\mathbf{d}$, $\{\delta_{j_l}^T\mathbf{d} : e_{j_l} \text{ adjacent to } e_i\}$ and the numbers of leaves in the corresponding subtrees. This observation was made in slightly different terminology by Vach (1989) and later, independently, by Bryant (1997).

The actual formula is achieved by constructing, and solving, an appropriate set of linear equations. See Vach (1989) or Bryant (1997), pg. 131-137, for two ways of doing this. Rzhetsky and Nei (1993) employed a quite different, and more involved, technique to derive a formula for OLS edge lengths in binary trees.

We must consider two cases when presenting the OLS edge length formulas: internal edges and external edges. Diagram (i) in figure 2 represents a generic example of an internal edge e_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 leaf sets of these subtrees and put $N_l = |C_l|$ for $l = 1, \ldots, m$. 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 labelling in the case of an external edge e_i, a generic example of which is given in 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 leaf sets and put $N_l = |C_l|$ for all $l = 1, \ldots, m$. Let $\mathbf{v}$ be a vector of m ones, put $N_\alpha = N - 1$ and $N_\beta = 1$.

One formula, equation (15), suffices for both internal and external edges. What it might lack in aesthetic appeal it makes up for in usefulness.

Theorem 1 *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*

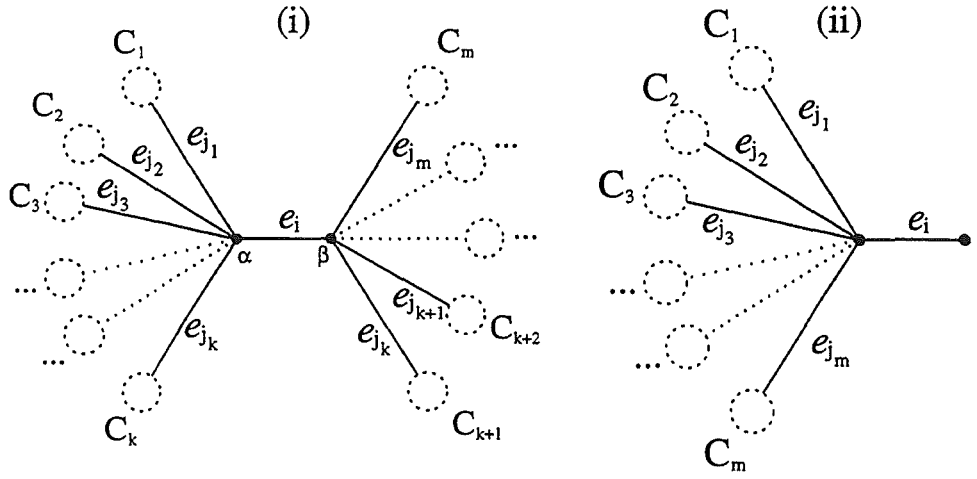

Figure 2 : Representation of the four generic cases for calculating edge lengths. (i) Internal edge in a non-binary tree. (ii) External edge in a non-binary tree.

by:

$$\mathbf{b}_i = \frac{\delta_i^T \mathbf{d} - \mathbf{w}^T \mathbf{N}^{-1} \mathbf{P}}{N_\alpha N_\beta - \mathbf{w}^T \mathbf{v}} \tag{15}$$

where N is the number of taxa,

$$\mathbf{P} = (\delta_{j1}^T \mathbf{d}, \delta_{j2}^T \mathbf{d}, \ldots, \delta_{jm}^T \mathbf{d})^T \tag{16}$$

and

$$\mathbf{w} = (N\mathbf{N}^{-1} - 2\mathbf{I} + \mathbf{U})^{-1}\mathbf{v}. \tag{17}$$

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 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 formulae for the elements of $\mathbf{w}$, with two cases to consider.

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

$$\mathbf{w}_l = \frac{1}{(N/N_l - 2)}(\gamma + \mathbf{v}_l) \tag{18}$$

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

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

$$\mathbf{w}_l = \frac{\mathbf{v}_l - \mathbf{v}_\lambda}{N/N_l - 2} \tag{19}$$

whereas

$$\mathbf{w}_\lambda = \mathbf{v}_\lambda + \sum_{j \neq \lambda} \left(\frac{-N_j \mathbf{v}_j + N_j}{N - 2N_j} \right). \tag{20}$$

Note that for any external or internal edge there can be at most one adjacent subtree with $N/2$ leaves, so there can be at most one λ such that $N_\lambda = N/2$.

In both cases it takes only $O(m)$ to calculate all of the entries of $\mathbf{w}$. Looking at equation (15) we see that the length $\mathbf{b}_i$ can therefore be calculated in $O(m)$ time, provided that $\delta_i^T \mathbf{d}$ and the values $\delta_{j_l}^T \mathbf{d}$ in $\mathbf{P}$ are already known. We can calculate all of these values in $O(N^2)$ time during preprocessing using the algorithm CALCULATEAd, before we evaluate individual edge lengths. For each edge we have $m \leq N$ so the calculation of all of the edge lengths can be completed in $O(N^2)$ time. We summarise the entire process in algorithm CALCULATEEDGELENGTHS.

As an illustration we now calculate OLS edge lengths for the four taxa tree in figure 1 (ii), with a given distance vector $\mathbf{d} = (\mathbf{d}_{wx}, \mathbf{d}_{wy}, \mathbf{d}_{wz}, \mathbf{d}_{xy}, \mathbf{d}_{xz}, \mathbf{d}_{yz})^T = (1, 3, 2, 5, 2, 4)^T$. As before, we calculate the values $\delta_1^T \mathbf{d} = 6$, $\delta_2^T \mathbf{d} = 8$, $\delta_3^T \mathbf{d} = 12$, $\delta_4^T \mathbf{d} = 8$ and $\delta_5^T \mathbf{d} = 12$ using the algorithm CALCULATEAd.

First consider edge e_1. The adjacent edges are $e_{j_1} = e_2$ and $e_{j_2} = e_5$. Thus $N_1 = 1$, $N_2 = 2$, $N_\alpha = 3$, $N_\beta = 1$ and $\mathbf{v} = (1, 1)^T$. From line 12 we get $\mathbf{w}_1 = 0$ and from line 14, $\mathbf{w}_2 = 1$. In line 22 we obtain

$$\mathbf{b}_1 = (6 - (0 + \frac{1 \times 12}{2}))/2 = 0. \tag{21}$$

Similarly $\mathbf{b}_2 = 1$, $\mathbf{b}_3 = 3$ and $\mathbf{b}_4 = 1$.

When $i = 5$ we get $N_\alpha = N_\beta = 2$ and $\mathbf{v} = (2, 2, 2, 2)^T$. From lines 16 and 17:

$$\kappa = 1 + (\frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2}) = 3 \text{ and } \gamma = \frac{-1}{3}(\frac{2}{2} + \frac{2}{2} + \frac{2}{2} + \frac{2}{2}) = \frac{-4}{3} \tag{22}$$

Procedure CALCULATEEDGELENGTHS(T,$\mathbf{d}$)

1. Count the number of leaves on each side of each edge.
2. Calculate $\delta_i^T\mathbf{d}$ for each edge using CALCULATEAD.
3. FOR each edge e_i DO
4. IF e_i is internal THEN
5. Let α and β be the endpoints of e_i. Let $e_{j_1},\ldots,e_{j_k}$ be the edges adjacent to e_i at α and let $e_{j_{k+1}},\ldots,e_{j_m}$ be the edges adjacent to e_i at β. For each l let N_l be the number of leaves on the other side of e_{j_l} from e_i (See figure 2 (i)). Let N_α and N_β be the number of leaves on either side of e_i.
6. ELSE
7. Let $e_{j_1},\ldots,e_{j_m}$ be the edges adjacent to e_i. For each l let N_l be the number of leaves on the other side of e_{j_l} from e_i. Put $k=m$, $N_\alpha = n-1$ and $N_\beta = 1$.
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. $\mathbf{w}_l \leftarrow \frac{\mathbf{v}_l-\mathbf{v}_\lambda}{N/N_l-2}$
13. END(FOR)
14. $\mathbf{w}_\lambda \leftarrow \mathbf{v}_\lambda + \sum_{j\neq\lambda}\left(\frac{-N_j\mathbf{v}_j+N_j}{N-2N_j}\right)$.
15. ELSE
16. $\kappa \leftarrow 1+\sum_{j=1}^m \frac{N_j}{N-2N_j}$
17. $\gamma \leftarrow \frac{-1}{\kappa}\sum_{j=1}^m \frac{\mathbf{v}_j}{(N/N_j-2)}$
18. FOR $l=1,\ldots,m$ DO
19. $\mathbf{w}_l \leftarrow \frac{1}{(N/N_l-2)}(\gamma+\mathbf{v}_l)$
20. END(FOR)
21. END(IF-ELSE)
22. $$\mathbf{b}_i \leftarrow \left(\delta_i^T\mathbf{d} - \sum_{l=1}^m \frac{w_l\delta_{j_l}^T\mathbf{d}}{n_l}\right) / \left(N_\alpha N_\beta - \sum_{j=1}^k \mathbf{w}_j N_\beta - \sum_{j=k+1}^m \mathbf{w}_j N_\alpha\right)$$
23. END(FOR)

END.

Algorithm 2 : CALCULATEEDGELENGTHS

and from line 19, $\mathbf{w} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$. Finally, in line 22,

$$\mathbf{b}_5 = \left(12 - \left(\frac{2}{1} + \frac{2}{1} + \frac{4}{1} + \frac{8}{3}\right)\right)\left(2 \times 2 - \frac{4}{3} - \frac{4}{3}\right) \quad (23)$$

$$= 1/2. \quad (24)$$

Calculating leaf to leaf distances in minimal time

We have now 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 optimal time algorithms for minimum evolution (e.g. ME(OLS), ME(FM), ME(GLS)). However other popular tree selection criteria, namely sums of squares, require the calculation of the leaf to leaf distance in the tree with these edge lengths. Given any two leaves, 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: we are aiming for an $O(N^2)$ time evaluation method.

We introduce a simple new method, CALCULATEDISTANCE which calculates leaf to leaf distances in $O(N^2)$ time from a tree T and its edge lengths.

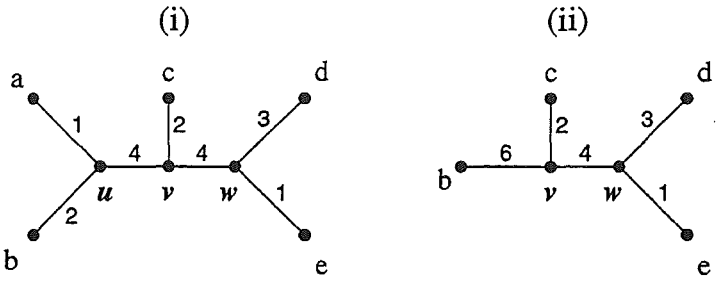

Figure 3 : An example illustrating the application of Algorithm CALCULATEDISTANCE

```
Procedure CALCULATEDISTANCE(T)
1.  FOR each leaf 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)
7.    Remove leaf 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)
END.
```

Algorithm 3 : CALCULATEDISTANCE

For example, consider tree (i) in figure 3. We have labelled the internal vertices for convenience. Starting with leaf a we calculate, in order, $\mathbf{p}_{au} = 1$ then $\mathbf{p}_{ab} = \mathbf{p}_{au} + 2 = 3$ and $\mathbf{p}_{av} = \mathbf{p}_{au} + 4 = 5$, and then $\mathbf{p}_{ac} = 7$, $\mathbf{p}_{aw} = 9$, $\mathbf{p}_{ad} = 12$ and $\mathbf{p}_{ae} = 10$. We then remove a and contract vertices to obtain tree (ii) which is then used to calculate the remaining values of $\mathbf{p}$.

By calculating distances to internal vertices we can calculate the distance from a single leaf to all other vertices and leaves in just $O(N)$ time, giving $O(N^2)$ time for the entire operation. The contraction in lines 7 and 8 speeds up the process by taking advantage of the fact that for each x and y we need only calculate one of $\mathbf{p}_{xy}$ and $\mathbf{p}_{yx}$.

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,

$$SS(OLS) = (\mathbf{Ab} - \mathbf{d})^T(\mathbf{Ab} - \mathbf{d}). \tag{25}$$

The optimal edge lengths $\mathbf{b}$ are provided already by the $O(N^2)$ time algorithm CALCULATEEDGELENGTHS, while the calculation of $\mathbf{Ab}$ can be reduced from $O(N^3)$ time to $O(N^2)$ time using the the algorithm CALCULATEDISTANCE. Thus the time taken for the whole calculation is reduced from $O(N^3)$ time to $O(N^2)$ time.

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

$$SS(WLS) = (\mathbf{Ab} - \mathbf{d})^T\mathbf{W}(\mathbf{Ab} - \mathbf{d}) \tag{26}$$

takes a total of $O(N^3)$ time.

By a similar method the sum of squares calculation

$$SS(GLS) = (\mathbf{Ab} - \mathbf{d})^T\mathbf{V}^{-1}(\mathbf{Ab} - \mathbf{d}) \tag{27}$$

can be completed in $O(N^4)$ time, provided that, as above, the inverse matrix $\mathbf{V}^{-1}$ is calculated beforehand.

Summary of Results

A summary of the speed increases due to our results is presented in Table 1:

Table 1: Summary of evaluation speed increases

Evaluation criterion				Fastest previous	New speed
	Edge lengths				
OLS	$\mathbf{b}$	$=$	$(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d}$	$O(N^3)$	$O(N^2)$
WLS	$\mathbf{b}$	$=$	$(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{d}$	$O(N^4)$	$O(N^3)$
GLS	$\mathbf{b}$	$=$	$(\mathbf{A}^T\mathbf{V}^{-1}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{V}^{-1}\mathbf{d}$	$O(N^5)$	$O(N^4)$
	Sum of squares				
OLS	$SS(OLS)$	$=$	$(\mathbf{A}\mathbf{b}-\mathbf{d})^T(\mathbf{A}\mathbf{b}-\mathbf{d})$	$O(N^3)$	$O(N^2)$
WLS	$SS(WLS)$	$=$	$(\mathbf{A}\mathbf{b}-\mathbf{d})^T\mathbf{W}(\mathbf{A}\mathbf{b}-\mathbf{d})$	$O(N^4)$	$O(N^3)$
GLS	$SS(GLS)$	$=$	$(\mathbf{A}\mathbf{b}-\mathbf{d})^T\mathbf{V}^{-1}(\mathbf{A}\mathbf{b}-\mathbf{d})$	$O(N^5)$	$O(N^4)$
	Variance-covariance matrix for edge lengths				
GLS	$Var(\mathbf{b})$	$=$	$(\mathbf{A}^T\mathbf{V}^{-1}\mathbf{A})^{-1}$	$O(N^5)$	$O(N^4)$

Discussion

The speed up of least squares tree criterion evaluation described in this paper to avoid ambiguity should allow faster and more extensive tree search strategies for distance based methods. Rzhetsky and Nei (1992a) describe a localised, but often effective, method of evaluating all trees within a small partition distance of a good starting tree. Use of our algorithm CALCULATEEDGELENGTHS makes this method run $O(N)$ times faster. A new, even faster, algorithm has recently been developed for localised search with ME(OLS) (Bryant 1997, pg. 149-154).

A wide range of alternative search strategies, offering better protection against being trapped in local optima, are now available in PAUP 4.0. These too may be accelerated using the algorithms developed here.

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), Swofford et al. (1996), Waddell et al. (1997)). 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 (e.g. Kidd and Sgaramella-Zonta 1971). Another approach is to define the ME score of a tree as

$$\sum_i |b_i| \text{ instead of } \sum_i b_i \tag{28}$$

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 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, Tarasov and Khachiyan 1979 and Goldfarb and Liu 1991). It is reasonable to expect 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.

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.

1 Acknowledgements

We are grateful to David Penny, Masami Hasegawa, Hidetoshi Shimodaira, and Mike Steel for comments on the manuscript. This work was supported by a University of Canterbury Doctoral Scholarship (D.B.) and the Marsden Fund (P.W. and D.B.).

Correctness and complexity proofs for the algorithm CALCULATEAd

Theorem 2 *The algorithm* CALCULATEAD *correctly calculates* $\delta_i^T \mathbf{d}$ *for all* $i = 1, 2, \ldots, K$.

Proof

The algorithm calculates the values for external edges correctly. Suppose we have the situation as in figure 1 where the values $\delta_{j_1}^T \mathbf{d}, \delta_{j_2}^T \mathbf{d}, \ldots, \delta_{j_m}^T \mathbf{d}$ have been calculated correctly. For all $k = i, j_1, j_2, \ldots, j_m$ the value $\delta_k^T \mathbf{d}$ equals the sum of all the distances going from one side to the other side of the corresponding split. Hence for all $k = i, j_1, j_2, \ldots, j_m$

$$\delta_k^T \mathbf{d} = \sum_{a \in C_k, b \in L - C_k} \mathbf{d}_{ab} \tag{29}$$

Furthermore

$$\begin{aligned}
\sum_{k=1}^{m} \delta_{j_k}^T \mathbf{d} &= \sum_{k=1}^{m} \left(\sum_{a \in C_{j_k}, b \in L - C_{j_k}} \mathbf{d}_{ab} \right) && (30)\\
& && (31)\\
&= \sum_{k=1}^{m} \left(\sum_{a \in C_{j_k}, b \in C_i} \mathbf{d}_{ab} + \sum_{a \in C_{j_k}, b \in L - (C_i \cup C_{j_k})} \mathbf{d}_{ab} \right) && (32)\\
&= \sum_{a \in C_i, b \in L - C_i} \mathbf{d}_{ab} + \sum_{k=1}^{m} \sum_{\substack{l=1 \\ l \neq k}}^{m} \sum_{a \in C_{j_k}, b \in C_{j_l}} \mathbf{d}_{ab} && (33)\\
&= \delta_i^T \mathbf{d} + 2 \sum_{k<l} \left(\sum_{a \in C_{j_k}, b \in C_{j_l}} \mathbf{d}_{ab} \right) && (34)
\end{aligned}$$

as required.

Theorem 3 *Given a tree* T *and distance vector* $\mathbf{d}$ *we can calculate the vector* $\mathbf{A}^T \mathbf{d}$ *in* $O(N^2)$ *time, where* $\mathbf{A}$ *is the topological matrix of* T.

We have shown that the algorithm CALCULATEAd correctly calculates $\mathbf{A}^T\mathbf{d}$. It remains to show that the algorithm completes the calculation in $O(N^2)$ time.

If δ_i corresponds to a trivial split then the vector has exactly $N-1$ ones. Hence $\delta_i^T\mathbf{d}$ can be calculated in $O(N)$ time and calculating $\delta_i^T\mathbf{d}$ for all the trivial splits takes $O(N^2)$ time.

The loop in lines 2 to 5 of algorithm CALCULATEAd iterates once for every edge in T, that is, $O(N)$ times. Within each iteration the sum in line 4

$$\sum_k \delta_{j_k}^T \mathbf{d} \tag{35}$$

takes at most $O(N)$ time. The second part of line 4:

$$2\sum_{k<l}\left(\sum_{a\in C_{j_k}, b\in C_{j_l}} \mathbf{d}_{ab}\right) \tag{36}$$

could potentially take $O(N^2)$ time per iteration. However the total amount of time taken by this calculation over all the iterations is $O(N^2)$, the reason being that each individual distance $\mathbf{d}_{ab}$ is added at most once. This is easily verified by rooting the tree along the last edge evaluated—the individual distance $\mathbf{d}_{ab}$ is only part of the calculation for the edge directly above the least common ancestor of a and b. Hence the entire algorithm takes $O(N^2)$ time. □

References

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

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.

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

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

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.

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.

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. Polyno-

mial solvability of convex quadratic programming. Doklady Akademiia 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.
NEI, M. 1987. Molecular evolutionary genetics. Columbia University Press, New York.
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.
SATTATH, S. and A. TVERSKY. 1977. Additive similarity trees. Psychometrika **42**:319-345.
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.
SWOFFORD, D. L. 1997. Phylogenetic Analysis Under Parsimony, Version 4.0 (PAUP* 4.0). Sinaur Associates, 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. 1997. 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.