

Branch and Bound Algorithms to Determine Minimal Evolutionary Trees

M. D. HENDY

*Department of Mathematics and Statistics, Massey University,
Palmerston North, New Zealand*

AND

DAVID PENNY

Department of Botany and Zoology, Massey University, Palmerston North, New Zealand

Received 31 July 1981; revised 14 December 1981

ABSTRACT

Two practical branch and bound algorithms for determining minimal and near-minimal phylogenetic trees from protein sequence data are presented. A mathematical description and analysis of phylogenetic trees introduces these algorithms. A comment on efficiency and fine tuning completes the paper. An example is cited where computer time was reduced from an estimated 55 days for a total search, to just under 5 minutes.

1. INTRODUCTION

It has often been proposed that homologous protein sequences from sets of different taxa (biological species) contain information from which ancestral relationships could be deduced. These relationships are normally expressed in the form of a directed rooted tree (in the graph theoretic sense) called a phylogeny or phylogenetic tree. The hypothesized common ancestors for the set of taxa below it occur at the root of each subtree.

One simple model is to propose that each of the hypothetical ancestors represented in this tree had a homologous protein sequence from which the extant sequences were derived, so that we focus our attention on the evolution of the sequences as representing the evolution of the taxa. Between any pair of known sequences we can count the substitutions necessary to convert one sequence to the other. Each point of the phylogenetic tree will have a sequence associated with it: the terminal points representing the extant sequences, the internal points representing their hypothesised ancestors. Associated with each link of the tree we assign a link weight, being the

number of changes to convert the sequence of one end point to the other. The simplistic model is now to choose amongst all possible phylogenetic trees, that (those) which represents the smallest number of substitutions, i.e. that (those) trees whose sum of link weights is minimal. We call such a tree a minimal phylogenetic tree. This simplistic model was introduced by Cavalli-Sforza and Edwards [1] and has since been widely accepted. It is sometimes referred to as the maximum parsimony model.

There are two basic algorithmic requirements for solving this problem. The first is to minimally assign the ancestral sequences to any particular tree. Fitch [2] proposed an algorithm consisting of a double pass over the entire tree which guaranteed to give both the total weight of the tree and all minimal assignments of sequences to internal points. This algorithm was mathematically justified by Hartigan [8] and Moore et al. [9], and the algorithm subsequently improved by Fitch and Farris [4].

The second problem is how to find, among all possible phylogenies, that particular tree which has the smallest number of substitutions after the Fitch-Hartigan algorithm has been applied. Fitch [3] proposed a search of all possible trees, finding the minimal assignment of each, and choosing the smallest. It had been shown however (Cavalli-Sforza and Edwards [1]) that for $n \geq 3$ there are $(2n-5)!! = \prod_{i=1}^{n-2} (2i-1)$ nonrooted binary trees with n distinctly labeled terminal points, while Foulds and Robinson [6] computed a more general class of undirected phylogenetic trees, and counted numbers of trees many times greater than this figure. Fitch was able to find minimal phylogenetic trees for up to eight taxa, but beyond this number, the computing cost was too great.

The authors (Foulds et al. [5]) developed a technique of analysing a given phylogenetic tree, usually constructed heuristically, to test its minimality. The test could either confirm its minimality, indicate a refinement which would lead to a shorter tree, or be indecisive. Although this technique was successfully used to find minimal phylogenetic trees for sets of more than twenty taxa (Foulds et al. [6]), some subsequent applications to smaller sets of taxa proved indecisive.

In order to overcome this occasional indecision the authors developed two branch and bound search techniques which have been successfully applied to some of the previously unresolved problems. Using these, we have been able to determine all the minimal and near minimal phylogenetic trees, for small and medium size sets of data. A nonrigorous description of one method has been given (Penny [10]). The criterion of minimal weight (maximum parsimony) (Fitch [2]) is used throughout this paper, and the reliability (or sensitivity) of this criterion is discussed in an additional paper in preparation.

We have restricted our algorithms to phylogenetic trees which are non-rooted binary trees. The positioning of the root has no influence on the total weight of the tree, an undirected phylogenetic tree will become directed by

the insertion of a root on any link. The determination of the root will usually require additional information, and cannot be directly deduced from the sequences. A general phylogenetic tree can be converted to a binary tree by the replacement of any point of degree ≥ 4 with a number of points of degree 3 with identical sequences, connected by links of weight 0. Likewise a point of degree 2 can be converted to degree 3 by joining it to new pendant vertex of the same sequence again by link of weight 0 (Fitch [3]).

2. LABELED BINARY TREES

DEFINITION

A *binary tree* is a tree $T = \langle V, E \rangle : \forall v \in V, d(v) = (\text{degree of } v) = 1 \text{ or } 3$. Let $P = \{v \in V : d(v) = 1\}$ be the set of *pendant points* and $I = \{v \in V : d(v) = 3\}$ be the set of *internal points* so that $V = P \cup I$, the disjoint union of P and I . If $e \in E$ is incident at a *pendant point*, we call e a *pendant link*; otherwise we call e an *internal link*.

If each of the $n = |P|$ pendant points are distinctly labeled, we call T a *labeled binary tree* on n (pendant) points, abbreviated *lbt*(n). Let $T_n = \langle P_n \cup I_n, E_n \rangle$ be a *lbt*(n) with pendant points p_1, p_2, \dots, p_n , which we will identify by their labels. If p_i, p_j are two distinct pendant points adjacent to a common internal vertex we say $\{p_i, p_j\}$ is a *neighboring pair*. We note the following two results, which can be obtained by simple counting arguments.

LEMMA 1

For $n \geq 3$, we have $|P_n| = n, |I_n| = n - 2, |E_n| = 2n - 3$.

LEMMA 2

For $n \geq 4$, T_n has at least two disjoint neighboring pairs.

In the branch and bound algorithms we need to recursively generate all the *lbt*(n)'s for a range of values of n , and for efficiency, avoid repetition. In both algorithms we use the same principle, which we call the generating principle.

GENERATING PRINCIPLE

Let $m < n$ be positive integers. Let τ_n be the set of all *lbt*(n)'s with pendant points $P_n = \{p_1, \dots, p_n\}$. If $P_m \subset P_n$ is a subset of m pendant points, we define τ_m to be the set of all *lbt*(m)'s with pendant points p_m . Let $\phi : \tau_n \rightarrow \tau_m$ be a mapping [i.e., ϕ uniquely assigns a *lbt*(m) to each *lbt*(n)]. Given ϕ , we can define the inverse relation $\phi^{-1} : \tau_m \rightarrow \tau_n$ where for each $T_m \in \tau_m$,

$$\phi^{-1}(T_m) = \{T_n \in \tau_n : \phi(T_n) = T_m\}$$

is called the inverse image of T_m . The collection of inverse images $\{\phi^{-1}(T_m): T_m \in \tau_m\}$ forms a partition of τ_n , and so this gives us a recursive procedure of generating all the $\text{lbt}(n)$'s.

With ϕ the mapping which deletes P_n from T as in Theorem 2, we can derive the following well-known result.

THEOREM 1 (CAVALLI-SFORZA AND EDWARDS [1])

For $n \geq 3$ there are $(2n-5)!! = \prod_{i=1}^{n-2} (2i-1)$ distinct $\text{lbt}(n)$'s.

[Note: $(2n)!! = 2 \times 4 \times 6 \times \dots \times 2n$ and $(2n+1)!! = 1 \times 3 \times 5 \times \dots \times (2n+1)$ are known as double factorials.]

THEOREM 2

For $n \geq 4$ there are $N_{n,r}$ distinct $\text{lbt}(n)$'s with r neighboring pairs, where

$$N_{n,r} = \begin{cases} \frac{n!(n-4)!}{(n-2r)!r!(r-2)!2^{2r-2}} & \text{for } 2 < r < \left\lfloor \frac{n}{2} \right\rfloor, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Proof. Suppose in $T_n \in \tau_n$, p_n is adjacent to $u \in I_n$, and u is also adjacent to vertices v and w . Let $\phi: T_n \rightarrow T_{n-1}$ be the mapping which deletes the vertices p_n, u , and their incident links, replacing them by a new link (v, w) . Let $T_{n-1} = \phi(T_n)$. Suppose T_n has r neighboring pairs and T_{n-1} has r' neighboring pairs. We find $r' = r$ or $r-1$ depending on the position of p_n . If p_n is not a point of a neighboring pair of T_n , then the neighboring pairs of T_{n-1} and T_n are the same, so $r' = r$. If however $\{p_n, p_l\}$ is a neighboring pair, then the removal of p_n will mean that $r-1$ neighboring pairs of T_n will be in T_{n-1} . If one of v, w (say v) is pendant, then $\{p_l, v\}$ will be a new neighboring pair of T_{n-1} and $r' = r$; otherwise $r' = r-1$. Thus $r' = r$ unless (v, w) is one of the $n-1-2r' = n-2r+1$ pendant links which is not in a neighboring pair of T_{n-1} .

We note that by Lemma 2, $r \geq 2$, and also that $2r \leq n$. So

$$N_{n,r} = 0 \quad \text{if } r < 2 \text{ or } r > \frac{n}{2}.$$

Inverting ϕ , we find if T_{n-1} has r' neighboring pairs, then $\phi^{-1}(T_{n-1})$ will contain $n-1-2r'$ trees with $r'+1$ neighboring pairs and $(2n-5)-(n-1-2r') = n-4+2r'$ trees with r' neighboring pairs. Thus

$$N_{n,r} = (n-2r+1)N_{n-1,r-1} + (n+2r-4)N_{n-1,r}. \quad (2)$$

If we assume, for induction (1) holds for $n-1$, then we find

$$\begin{aligned} N_{n,r} &= \frac{(n-2r+1)(n-1)!(n-5)!}{(n-2r+1)!(r-1)!(r-3)!2^{2r-4}} + \frac{(n+2r-4)(n-1)!(n-5)!}{(n-2r-1)!r!(r-2)!2^{2r-2}} \\ &= \frac{(n-1)!(n-5)!}{(n-2r)!r!(r-2)!2^{2r-2}} \{4r(r-2) + (n+2r-4)(n-2r)\}, \\ &= \frac{n!(n-4)!}{(n-2r)!r!(r-2)!2^{2r-2}}, \end{aligned}$$

which agrees with (1).

We could also derive Theorem 1 from this result, noting that

$$\begin{aligned} \sum_{r=2}^{[n/2]} N_{n,r} &= \sum_{r=2}^{[n/2]} \{(n-2r+1)N_{n-1,r-1} + (n+2r-4)N_{n-1,r}\} \\ &= \sum_{r=1}^{[n/2]-1} (n-2r-1)N_{n-1,r} + \sum_{r=2}^{[n/2]} (n+2r-4)N_{n-1,r} \\ &= (2n-5) \sum_{r=2}^{[(n-1)/2]} N_{n-1,r} \\ &= (2n-5)!! \end{aligned}$$

and using an inductive assumption.

3. PHYLOGENIES

The protein sequences are usually presented to us as a string of letters, each letter representing an amino acid (20 different values) or with each letter representing the lower level of nucleotides (4 different values) with each triplet of nucleotides coding for one amino acid. These letters are called character states. We generalize the situation with the following definitions.

DEFINITION

Let K be a finite set, whose elements we call *character states*; for some positive integer t we call the t -tuples of K^t *character sequences*. For $\mathbf{X} = (X_1, X_2, \dots, X_t) \in K^t$, the j th component X_j is the *character state at the j th site*. If $\mathbf{X}, \mathbf{Y} \in K^t$, we define $\delta(\mathbf{X}, \mathbf{Y})$, the *distance from \mathbf{X} to \mathbf{Y}* , to be the number of sites j , $1 \leq j \leq t$, where $X_j \neq Y_j$. Note that δ is a metric on K^t .

If $T = \langle V, E \rangle$ is a tree, and $\theta: V \rightarrow K^t$ is a mapping which assigns a character sequence $\theta(v) \in K^t$ to each point $v \in V$, then the labeled tree (T, θ)

will be called a *phylogeny*. For each link $e_{ij} = (v_i, v_j)$ of (T, θ) we define the weight of e_{ij} as

$$w(e_{ij}) = \delta(\theta(v_i), \theta(v_j)).$$

We take the *weight* of the phylogeny (T, θ) as the sum of all link weights, viz.

$$w(T, \theta) = \sum_{e \in E} w(e).$$

If $W \subseteq V$ is a subset of points with designated character sequences $\theta(W) = \{\theta(w) : w \in W\}$, and if (T, ϕ) is a phylogeny such that $\forall w \in W, \phi(w) = \theta(w)$, then we say (T, ϕ) is a phylogeny which *covers* $\theta(W)$.

Our minimal phylogenetic tree model can now be interpreted as follows: We have a given point set W (the set of taxa to be examined) together with an associated set of homologous sequences $\theta(W)$. We wish to find a phylogeny (T, θ) which covers $\theta(W)$, which is of minimal weight among all trees T and all phylogenies (T, ϕ) which cover $\theta(W)$. Such a phylogeny will be called a *minimal phylogenetic spanning tree* (MPST) covering $\theta(W)$.

Although we must consider all possible trees for the MPSTs, the following theorem means that we can narrow our search to binary trees. In order to apply the preliminary lemmas we assume $n = |W| \geq 4$.

THEOREM 3

If (T', ψ') is a phylogeny covering $\theta(W)$ with $|W| = n$, then there exists a $\text{lbt}(n)$ $T = \langle V, E \rangle$ with $W \subseteq V$ the set of pendant points of T , and a character assigning function $\psi : V \rightarrow K'$, such that

- (1) (T, ψ) is a phylogeny covering $\theta(W)$;
- (2) $w(T, \psi) \leq w(T', \psi')$.

Proof. The basis of this proof is a procedure to replace all points of degree k , for $k = 2$ and $k \geq 4$, and to make W the pendant point set of T , and show that none of these procedures can increase the weight of the tree.

For all points v in T' we define $\psi(v) = \psi'(v)$. If $v \in W$ and v not pendant, we can introduce a new point v' with $\psi(v') = \psi(v)$. We replace v by v' and connect v by the new link (v, v') of weight zero. If $u \notin W$ and u is pendant, then we remove u and its incident link (u, w) , reducing the tree weight by $\delta(\psi(u), \psi(w)) \geq 0$. This pair of operations will produce a phylogeny (T'', ψ') satisfying (1) and (2), which however might not be binary.

If $d(v) = 2$, suppose v is adjacent to v_1, v_2 . If we delete v and its incident links, replacing them by the new link (v_1, v_2) , the tree weight will be reduced

by $\delta(\psi(v_1), \psi(v_1)) + \delta(\psi(v), \psi(v_2)) - \delta(\psi(v_1), \psi(v_2))$, which is nonnegative, as a δ is a metric.

If $d(v) = k \geq 4$, then suppose v adjacent to v_1, v_2, \dots, v_k . We introduce $k-3$ new points x_4, x_5, \dots, x_k with $\psi(x_i) = \psi(v)$, connect these with new links $(v, x_4), (x_4, x_5), \dots, (x_{k-1}, x_k)$ each of link weight zero, and replace the links $(v, v_3), (v, v_4), \dots, (v, v_k)$ by new links $(x_4, v_3), (x_5, v_4), \dots, (x_k, v_{k-1}), (x_k, v_k)$, each of the same weight as the link it replaces. Each point v, x_4, \dots, x_k , is now of degree 3, and the tree weight is unaffected.

After this operation has replaced all points of degree > 3 , the resulting tree T is a lbt(n) satisfying the required conditions.

4. ALGORITHM I (ADDING TAXA SINGLY)

Consider the process of constructing all lbt(n)'s recursively using the inverse relation ϕ^{-1} of Theorem 2. For each r , $3 \leq r < n$, we will have $(2r-5)!!$ lbt(r)'s whose pendant point set was $\{p_1, p_2, \dots, p_r\}$, and to each of these we can join p_{r+1} to each of the $2r-3$ branches to obtain $(2r-3)!!$ lbt($r+1$)'s. We choose to do this by a "depth first" scheme, starting with $r=3$; we then consider the first of the 3 lbt(4)'s, and then from these the first of the 5 lbt(5)'s, continuing until we construct the first of the $2n-5$ lbt(n)'s. We then successively construct each of the other $2n-6$ lbt(n)'s before we return to the $(n-1)$ th level and consider the second lbt($n-1$). Only when we have considered all the lbt(r)'s do we return to the $(r-1)$ th level and consider the next lbt($r-1$). When we have constructed all $(2n-5)!!$ lbt(n)'s, we are through.

This process is computationally very expensive even for modest values such as $n=10$. [$15!! = 2,027,025$. By Stirling's formula $(2n-5)!! \sim \sqrt{2} \{(2/e)(n-2)\}^{n-2}$.] However, we do not really want all lbt(n)'s; we are really only interested in examining the phylogenies spanning a given $\theta(W)$, which are of minimal (or near-minimal) weight. Suppose we have a set of n taxa, $W = \{p_1, \dots, p_n\}$, and associated with each is a character sequence $\theta(p_i) \in K^l$, then using Fitch's assignment algorithm for determining sequences for internal vertices in a minimal way, we can determine the total minimal weight of each lbt(r) which spans $\theta(W_r)$, where $W_r = \{p_1, p_2, \dots, p_r\}$.

Suppose we had used a heuristic tree building algorithm to construct a near-minimal-weight tree (Foulds et al. [5]) of weight L . If during our search above we obtained a lbt(r) T spanning $\theta(W_r)$ with minimal weight which exceeded L , then it is easy to show that any lbt(s), $s > r$, constructed from T would also have minimal weight $> L$, and so no MPST could be found from T . Hence we terminate this branch of the algorithm. In general, whenever any lbt(r) is found whose weight exceeds L , we terminate that branch and proceed to the next lbt(r).

It is impossible to estimate in general how many of the $\sum_{r=3}^n (2r-5)!!$ trees need to be evaluated. In our experience with real data of fairly low complexity (Penny et al. [11]), only an insignificant portion of this number are considered. However, it is also possible to construct artificial data sets of low complexity in which all of the $(2n-5)!!$ $\text{lbt}(n)$'s must be evaluated. We have not encountered any naturally occurring data sets of this nature.

5. ALGORITHM II (ADDING TAXA IN PAIRS)

Again we consider a recursive process of potentially building up all $\text{lbt}(n)$'s on the set of n labels $P = \{p_1, p_2, \dots, p_n\}$, but in this case the recursion is in steps of two labels being added. As before, we consider the inverse relation of a decomposition function.

For $2 \leq s \leq n$, let τ_s be the set of all $\text{lbt}(s)$'s whose labeled pendant points are any subset of P containing s distinct elements. For $4 \leq s \leq n$ we define a function $\chi: \tau_s \rightarrow \tau_{s-2}$. If $T \in \tau_s$, then by Lemma 2 T has at least two neighboring pairs. Amongst the set of r neighboring pairs of T , we choose that pair $\{p_i, p_j\}$ of largest index, where we take the index of $\{p_i, p_j\}$ to be the smaller of i and j . Then p_i and p_j are adjacent to an internal point u_1 which is also adjacent to u_2 which in turn is adjacent to two further points u_3 and u_4 . We construct $\chi(T)$ by deleting the points p_i, p_j, u_1 , and u_2 and all links incident with them, replacing them with a new link (u_3, u_4) , so that $\chi(T) \in \tau_{s-2}$. The successive application of the function χ to a tree $T \in \tau_n$ will terminate in a tree of τ_2 if n is even, or τ_3 if n is odd. An example of this reduction is illustrated in Figure 1.

If $T \in \tau_s$ has r neighboring pairs, then the removal of the neighboring pair $\{p_i, p_j\}$ by χ will give $\chi(T)$ which retains the remaining $r-1$ neighboring pairs of T , but may also include an additional neighboring pair $\{p_k, p_l\}$ where one of p_k, p_l is now joined by the new link of $\chi(T)$, so $\chi(T)$ has either $r-1$ or r neighboring pairs. As there are always at least two neighboring pairs, the point p_1 will never be removed, and so our terminating tree of τ_2 or τ_3 will contain p_1 .

On inverting χ , we can add to our $\text{lbt}(s)$ T any pair of points $\{p_k, p_l\}$ as a neighboring pair, to any branch of T , provided that in any such tree T' , $\{p_k, p_l\}$ will then have the largest index of the neighboring pairs of T' . Suppose $k < l$, so the index of $\{p_k, p_l\}$ is k ; then if k is larger than the index of all neighboring pairs of T , then $\{p_k, p_l\}$ could be joined to any of the $2s-3$ links of T . If however there is one neighboring pair in T of index greater than k , then we only have a choice of the two links of this neighboring pair to join on $\{p_k, p_l\}$. If there were more than one neighboring pair in T of index greater than k , then we could not join on $\{p_k, p_l\}$ to T , nor to any tree subsequently constructed from T . Hence, if we have a sequence of neighboring pairs to be added to T , we must ensure that at each

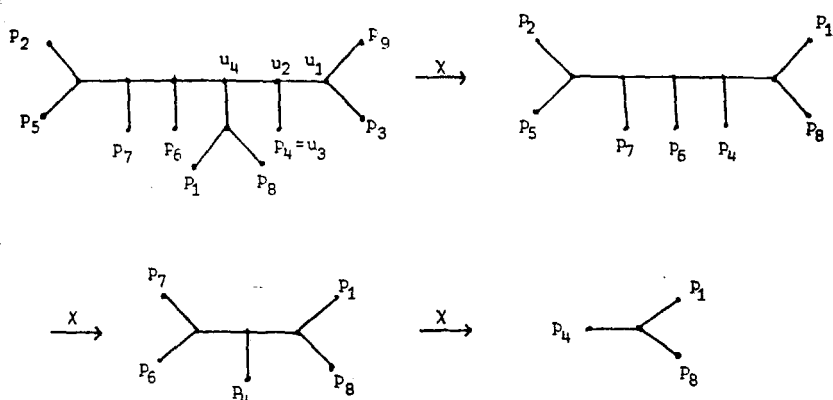


FIG. 1. The successive application of the decomposition function χ to a $\text{lbt}(9)$. In each tree the neighboring pair of largest index, viz. $\{p_3, p_9\}$, $\{p_2, p_5\}$, and $\{p_6, p_7\}$, have been deleted, terminating in a $\text{lbt}(3)$ with points $\{p_1, p_4, p_5\}$.

stage we never have more than one neighboring pair of index greater than the indices of the pairs yet to be connected. If $T \in \tau_m$ is a tree whose neighboring pair of greatest index is greater than the index of the points of P not in T , then we will call T a tree of type I; otherwise T will be of type II. If $T \in \tau_3$, then necessarily p_1 is one of the points of T . If p_2 is also a point of T , we will say T is of type II; otherwise it is of type I. If $T \in \tau_2$, then T will be of type I.

In order that we can ultimately construct a $\text{lbt}(n)$ by successive operation of χ^{-1} , we adopt the rule that if $T \in \tau_s$ is of type I, then the next neighboring pair is joined to T at either of the 2 links of the neighboring pair of largest index, but if T is of type II, then there is a choice of $2s - 3$ links to join the next neighboring pair. We show that this rule does not lead to redundancy by counting the number of trees so produced.

Firstly we note if $n = 2m$ is even, there are

$$\frac{\binom{2m}{2} \binom{2m-2}{2} \cdots \binom{4}{2} \binom{2}{2}}{m!} = \frac{2^{-m}(2m)!}{m!} = (2m-1)!! = (n-1)!!$$

ways of selecting m pairs of points from $\{p_1, p_1, \dots, p_n\}$, and similarly if $n = 2m + 1$ is odd, there are $(2m-1)!! = (n-2)!!$ ways of selecting m pairs from $\{p_2, p_3, \dots, p_n\}$.

For a given selection of pairs, let a_s be the number of trees of τ_s of type I, and b_s the number of type II, which can be constructed by our rule from that set of pairs. If $T \in \tau_s$, then T contains s points of P , so there will be $\frac{1}{2}(n-s)$ pairs remaining to choose for the next pair to join. If T is of type I, then the

chosen pair can only be joined to T at one of two links, whereas if T is of type II, we have $2s-3$ choices. Amongst the pairs to choose, the joining of that unjoined pair of smallest index will give a tree of type II; otherwise it will be of type I. Hence for $s \geq 2$,

$$b_{s+2} = (2s-3)b_s + 2a_s, \quad (3)$$

and

$$\begin{aligned} a_{s+2} &= \left\{ \frac{1}{2}(n-s)-1 \right\} [(2s-3)b_s + 2a_s] \\ &= \left\{ \frac{1}{2}(n-s)-1 \right\} b_{s+2}, \end{aligned}$$

so for $s \geq 4$,

$$a_s = \frac{1}{2}(n-s)b_s$$

and hence

$$b_s = (n+s-3)b_s. \quad (4)$$

If $s = n$, all the trees must be of type I, so b_n counts all the different trees constructible by our rule using the given selection of pairs. Hence

$$\begin{aligned} b_n &= (2n-5)b_{n-2} \\ &= (2n-5)(2n-7)b_{n-4}, \text{ etc.} \end{aligned}$$

Hence if $n = 2m$ is even, by (4),

$$b_n = (4m-5)(4m-7) \cdots (2m+1)b_4,$$

while by (3),

$$b_4 = 2a_2 + b_2.$$

But the first tree chosen in τ_2 must include p_1 and is of type II, so

$$a_2 = 0, \quad b_2 = b_4 = 1,$$

and

$$b_n = (4m-5)(4m-7) \cdots (2m+1) = \frac{(2n-5)!!}{(n-1)!!}. \quad (5)$$

Otherwise, if $n = 2m + 1$ is odd, then by (3)

$$b_n = (4m - 3)(4m - 5) \cdots (2m + 3)b_5.$$

while by (4),

$$b_5 = 2a_3 + 3b_3.$$

The initial tree $T \in \tau_3$ is obtained by joining p_1 to one of the m pairs given. If the selected pair contains p_2 , then T is of type II; otherwise if the selected pair is one of the $m - 1$ pairs not containing p_2 and T is of type I. Hence

$$a_3 = m - 1, \quad b_3 = 1, \quad b_5 = 2m + 1,$$

so

$$b_n = (4m - 3)(4m - 5) \cdots (2m + 1) = \frac{(2n - 5)!!}{(n - 2)!!}. \quad (6)$$

But with n even there were $(n - 1)!!$ pairings, and with n odd, $(n - 2)!!$ pairings; so in both cases we construct $(2n - 5)!!$ trees, which by Theorem 1 is the number of lbt(n)'s. Hence, as every tree is constructed at least once, this rule will not duplicate any trees, and there is no redundancy.

Suppose $T \in \tau_s$ and the weights of T and $\chi(T)$ are $w(T)$ and $w(\chi(T))$. If $\chi(T)$ was created by the removal of the neighboring pair $\{p_i, p_j\}$ whose common adjacent internal point was u_1 as previously, then

$$\begin{aligned} w(\chi(T)) &\leq w(T) - \{\delta(\theta(p_i), \theta(u_1)) + \delta(\theta(p_j), \theta(u_1))\} \\ &\leq w(T) - \delta(\theta(p_i), \theta(p_j)). \end{aligned}$$

Thus if $T' \in \tau_n$ was created from T by successively joining on the remaining $k = \frac{1}{2}(n - 5)$ pairs $\{p_{i_1}, p_{j_1}\}, \dots, \{p_{i_k}, p_{j_k}\}$, then

$$w(T') \geq w(T) + \delta(\theta(p_{i_1}, p_{j_1})) + \cdots + \delta(\theta(p_{i_k}, p_{j_k})). \quad (7)$$

Hence in this branch and bound algorithm we will always have a lower bound on the weights of all the lbt(n)'s by this inequality. If at any stage the right side of the inequality (7) exceeds the bound L , then we no longer pursue that path. This is a considerable advantage over the first method, where we effectively only had $w(T)$ as an estimate, but we have the disadvantage of having to consider each of the $(2k - 1)!! = (n - s - 1)!!$ pairings that could appear in the equality (7).

This branch and bound algorithm proceeds, like the previous algorithm, as a depth first search. If $n = 2m$ is even, the first step trees are the $n - 1$ trees $\in \tau_2$ with vertices $\{p_1, p_j\}$, $2 \leq j \leq n$. If $n = 2m + 1$ is odd, the first step trees are the $\frac{1}{2}(n - 1)(n - 2)$ trees $\in \tau_3$ with vertices $\{p_1, p_i, p_j\}$, $2 \leq i < j \leq n$.

We find with this algorithm again that many branches terminate early, and for some data nearly all pairings give rise to a value $\delta(\theta(p_i, p_j)) + \dots + \delta(\theta(p_i, p_{j_n})) > L$, so no trees at all need be considered for this set of pairings.

6. EFFICIENCY

The motivation for developing these new algorithms is to have algorithms that are of greater efficiency than previous methods so that we can extend the range of analysis to larger problems. It is extremely difficult to estimate the number of steps required, in general, to complete our branch and bound algorithms, as this varies considerably with the input data. Worst possible cases can be created which require all branches to be traversed, with no bounding occurring except at the final stage. These cases require $O(in^n)$ steps, where i is the length of the character sequences and n is the number of sequences. This function is obviously growing too rapidly to be practical. However the biological sequence data are far from worst possible, indicating that these sequences contain information. Hence we presume that they contain significant phylogenetic information. Of course, in a branch and bound algorithm it is important to have a good upper bound. For this we usually use heuristic methods developed by Foulds et al. [5], which generate a near-minimal tree in $O(in^2)$ steps.

There are several ways of improving efficiency within the algorithm however. Fitch's algorithm for assigning minimal labelings to a given tree requires two passes through the tree. However, at the end of the first pass the minimal weight is determined, which can be compared with the bound and terminated at that stage. Further, we really only need to consider the additional cost of adding the one or two species at each recursive stage, and this can be done without recoding the whole tree if the intermediate first pass labelings from the previous step in the recursion have been recorded.

In the first branch and bound algorithm we found that rearranging the order of presentation of the sequences and permuting the sites in the sequence (row and column permutations of the table of sequences) does have a considerable effect on the time taken by the algorithm. The minimal trees found are of course independent of the order of presentation, but if we can terminate as many branches of the algorithm as early as possible, we decrease the number of steps required.

Using some test data on $n = 10$ taxa with $i = 7$ sites, we compared the times required to find the 12 minimal trees when we reordered the species

and reordered the sites. When we ordered the species so that the most unlike sequences were presented first, we required less than a third of the time taken with the order reversed. As an indicator, the algorithm examined 1027 lbt(8)'s in the first case, compared with 4615 lbt(8)'s in the second. When we ordered the sites with the most variables sites first, we had a 15–30% reduction in time over the same sequences reversed.

In most cases the first algorithm (adding taxa singly) proved to take significantly less time than the second (adding taxa in pairs), but they were both of the same order of magnitude. However, when the minimal trees where "stringy" (i.e., the number of links in the path between pairs of pendant points was large), we found the second algorithm performed better than the first. This was due to the very small number of combinations of pairs that did not exceed the bound before any tree construction commenced.

We have used the first branch and bound algorithm to construct all the minimal and near-minimal trees for a set of eleven mammalian taxa using five different protein sequences, as well as a concatenated sequence made by adjoining all the sequences together. We had been unable to complete this study using earlier methods of tree construction. An example from this study illustrates the saving by the branch and bound algorithm in comparison with a total search. It was estimated, from the time taken for a total search on eight taxa, that it would take 55 days for a total search for eleven fibrinopeptide A sequences. However, algorithm 1 took less than 5 minutes, even though this was a fairly complex data set. A detailed analysis of the trees from these five data sets will appear elsewhere.

REFERENCES

- 1 L. L. Cavalli-Sforza and A. W. F. Edwards, Phylogenetic analysis: models and estimation procedures, *Evolution* 21:550–570 (1967).
- 2 W. M. Fitch, Toward defining the course of evolution: minimum change for a specific tree topology, *Syst. Zool.* 20:406–416 (1971).
- 3 W. M. Fitch, On the problem of discovering the most parsimonious tree, *Amer. Nat.* 111:223 (1977).
- 4 W. M. Fitch and J. S. Farris, Evolutionary trees with minimum nucleotide replacements from amino and sequences, *J. Mol. Evol.* 3:263–278 (1974).
- 5 L. R. Foulds, M. D. Hendy, and David Penny, A graph theoretic approach to the development of minimal phylogenetic trees, *J. Mol. Evol.* 13:127–149 (1979).
- 6 L. R. Foulds, David Penny, and M. D. Hendy, A general approach to proving the minimality of phylogenetic trees illustrated by an example with a set of 23 vertebrates, *J. Mol. Evol.* 13:150–166 (1979).
- 7 L. R. Foulds and R. W. Robinson, Determining the asymptotic number of phylogenetic trees, in *Combinatorial Mathematics VII*, Lecture Notes in Mathematics, 829, Springer, Berlin, 1980, pp. 110–126.
- 8 J. A. Hartigan, Minimum mutation fits to a given tree, *Biometrics* 29:53–65 (1973).

- 9 G. W. Moore, J. Barnabas, and M. Goodman, A method for constructing maximum parsimony ancestral amino acid sequences on a given network, *J. Theoret. Biol.* 38:459-485 (1973).
- 10 David Penny, Branch and bound method for minimal trees and maximal cliques, *Syst. Zool.*, to appear.
- 11 David Penny, M. D. Hendy, and L. R. Foulds, Techniques for the verification of minimal phylogenetic trees illustrated with ten mammalian haemoglobin sequences, *Biochem. J.* 187:65-74 (1980).