A New Method That Simultaneously Aligns and Reconstructs Ancestral Sequences for Any Number of Homologous Sequences, When the Phylogeny Is Given[1]

Jotun Hein
NIEHS

Among the fundamental problems in molecular evolution and in the analysis of homologous sequences are alignment, phylogeny reconstruction, and the reconstruction of ancestral sequences. This paper presents a fast, combined solution to these problems. The new algorithm gives an approximation to the minimal history in terms of a distance function on sequences. The distance function on sequences is a minimal weighted path length constructed from substitutions and insertions-deletions of segments of any length. Substitutions are weighted with an arbitrary metric on the set of nucleotides or amino acids, and indels are weighted with a gap penalty function of the form $g_k = a + (b \times k)$, where k is the length of the indel and a and b are two positive numbers. A novel feature is the introduction of the concept of sequence graphs and a generalization of the traditional dynamic sequence comparison algorithm to the comparison of sequence graphs. Sequence graphs ease several computational problems. They are used to represent large sets of sequences that can then be compared simultaneously. Furthermore, they allow the handling of multiple, equally good, alignments, where previous methods were forced to make arbitrary choices. A program written in C implemented this method; it was tested first on 22 5S RNA sequences.

Introduction

The history of homologous DNA sequences can be described by a phylogeny with substitutions and insertions-deletions (indels) of nucleotides occurring on different branches on the phylogeny. If gene conversions or recombinations occur, a single phylogeny may not be sufficient to describe the relationships among the sequences. Other genetic events different from substitutions and indels are also imaginable.

When analyzing homologous DNA sequences, scientists often invoke the principle of parsimony: the best reconstructed history is the one that necessitates the smallest amount of evolution compatible with the extant sequences. The form of this problem that is addressed here is as follows: given a set of homologous sequences, what is the most parsimonious history, where the amount of evolution is measured by the weighted sum of substitutions and indels? The weight assigned to an indel will be restricted to the type $g_k = a + (b \times k)$, where k is the length of the indel and a and b are arbitrary positive constants. The weight of a substitution can be given by any metric on the set of nucleotides. Phylogenies will be restricted to trees involving only bifurcation. This is a legitimate restriction in that the root can be moved and branch lengths can be

1. Key words: parsimony, phylogeny, alignment, ancestral sequences, 5S RNA.

Address for correspondence and reprints: Dr. Jotun Hein, NIEHS, Research Triangle Park, North Carolina 27709.

Mol. Biol. Evol. 6(6):649–668. 1989.
© 1989 by The University of Chicago. All rights reserved.
0737-4038/89/0606-0006$02.00

zero. Thus, it is solely a question of representation and not of the actual form of evolution.

The general parsimony problem has some previously studied special cases. The first case is when either no indels are allowed or their positions are already known. This is the traditional parsimony tree reconstruction problem that is tackled by programs in Swofford's PAUP (1985) or in Felsenstein's (1989) PHYLIP (phylogeny inference package). The problem is NP-complete (Foulds and Graham 1982), which is to say that it is unlikely that a polynomial algorithm can guarantee to find the most parsimonious tree for a large number of sequences. For a review of the problems of phylogeny reconstruction, see Felsenstein (1982).

A second case is the two-sequence alignment problem, where there are only two sequences but indels are allowed. The criterion for alignment can be either similarity maximization or a distance minimization. Whether similarity or distance is chosen is not very important, and it can be shown (Smith et al. 1981) that for a large set of similarity parameters there exist distance parameters such that the distance and similarity alignments will be identical. The focus in the present article is solely on distance minimization.

The first methods for finding distance alignments of biological sequences were introduced by Sankoff (1972) and Sellers (1974). Both methods minimized the number of events, substitutions, or indels of single elements necessary to transform one sequence into the other. This approach was generalized to any number of sequences by Sankoff (1975). When more than three sequences are aligned, their phylogeny must be given to evaluate the overall number of changes that have taken place. There are three main problems with this method. First, the phylogeny is often not known beforehand. Second, the method is slow and not practical for more than three sequences. Last, molecular evolution of real sequences also includes indels of longer stretches of nucleotides and the method only considers the indel of single elements. Sankoff et al. (1973) had a faster approximate method in use, but it still had to be given the phylogeny, and it allowed only for indels of length one. Konings et al. (1987) have a method that improves on this by finding both the alignment and the phylogeny simultaneously. Waterman et al. (1976) introduced a method for aligning two sequences that allowed indels of several contiguous elements if a gap penalty function was provided that assigned weights to indels of different lengths. This was accomplished at some cost in the speed of computation. Gotoh (1981) improved on this in the case where the gap penalty function was of the form $a + (b \times k)$, where a and b are constants, and k is the length of the indel. Gotoh's method was generalized to three sequences by Fredman (1984) for the case $b = 0$. These methods can be extended to any number of sequences (author's unpublished data) whose phylogeny is known, but, again, it becomes unacceptably slow for more than three sequences. For reviews of the problems of sequence comparison, see Sankoff and Kruskal (1983) or the special 1984 issue of *Bulletin of Mathematical Biology* (vol. 46, no. 4) on sequence comparison algorithms.

There are several practical reasons to find an efficient approximation to the general parsimony problem. For example, a practical method will make the parsimony phylogeny reconstruction fully automated. Previously, an alignment would have to be constructed in advance of the phylogeny reconstruction; this was often done manually and with great labor. Often the alignments were very arbitrary and influenced the final phylogeny. Human interaction also precludes the use of computer simulations to assess the reliability of the method. Moreover, it is conceptually more consistent to apply parsimony to all evolutionary events, not just to substitutions. Finally, the method

reported gives proper alignments, even if history is not the prime concern. The sequences are put out in an order such that closely related sequences are adjacent, giving an overall simple alignment.

In the present paper an algorithm is presented that solves these problems, and a program that implements the algorithm has been written in C. The program can be applied to very large data sets and still yield acceptable computing times. Indeed, the present version of the program can align many (>100) sequences 2,000–3,000 bp long, so it is possible to compare a large number of even the largest RNA molecules. If events or relationships very far back in time are to be reconstructed, it is advantageous to compare as many sequences as possible, since the reliability of the reconstruction of an ancestral molecule is bound to improve if many descendants are used. Thus attempts to develop methods for handling very large sets of sequences are not without practical motivation.

It is clear from the known computational properties of the general parsimony problem that some approximations must be accepted in a practical algorithm. Two approximations were employed in the method described here. Since even the restricted problem of constructing an alignment, given the phylogeny, is computationally too difficult, a practical goal is to find a good—but perhaps not the best—alignment, given the phylogeny. Since the general parsimony problem is NP-complete, the best tree cannot be guaranteed to be found by a reasonable algorithm for large sets of sequences. Therefore a good—but not necessarily the best—tree must be accepted. The present article focuses on the alignment problem, given the phylogeny. The presentation of the method is as follows: First, two well-established parsimony problems and their solution are presented; these problems are (1) how to assign nucleotides to internal nodes when the tree with nucleotides at the tips is given and (2) how to parsimoniously align two sequences. Then it is shown how to construct a graph that represents sequences that most parsimoniously can be postulated as ancestor sequences in alignments of two sequences. Last, the traditional dynamic programming algorithm is generalized to allow alignment of graphs (actually aligning alignments). This new algorithm allows an efficient and fast approximation to the phylogenetic alignment problem.

In the accompanying article (Hein 1989) a tree algorithm will be described that, together with the phylogenetic alignment algorithm, will provide an algorithm that will both align and find the phylogeny of a set of sequences. The initial tree reconstruction in this method is approached through pairwise comparisons. An additional problem in this context is that pairwise comparisons are computationally expensive, since they correspond to alignments. Care should be taken not to make unnecessary comparisons when constructing the tree. This has not been a problem for previous methods that started from an alignment and/or were given the phylogeny. So the method is economical in the number of pairwise comparisons used to reconstruct the phylogeny.

The Method to Align Sequences, Given the Phylogeny

The problem undertaken in this section is illustrated by the tree depicted in figure 1. The known sequences ($s_1, \ldots, s_5$) are at the tips of the tree, since the tips represent the present, while internal nodes represent sequences that are in the past and that must be reconstructed. The problem is to perform this assignment of sequences to internal nodes so that the total amount of evolution, the weight of the tree, is minimized. The simple case, where the sequences all consist of one nucleotide and no indels are allowed, is called the *small parsimony problem.* A solution to the small parsimony

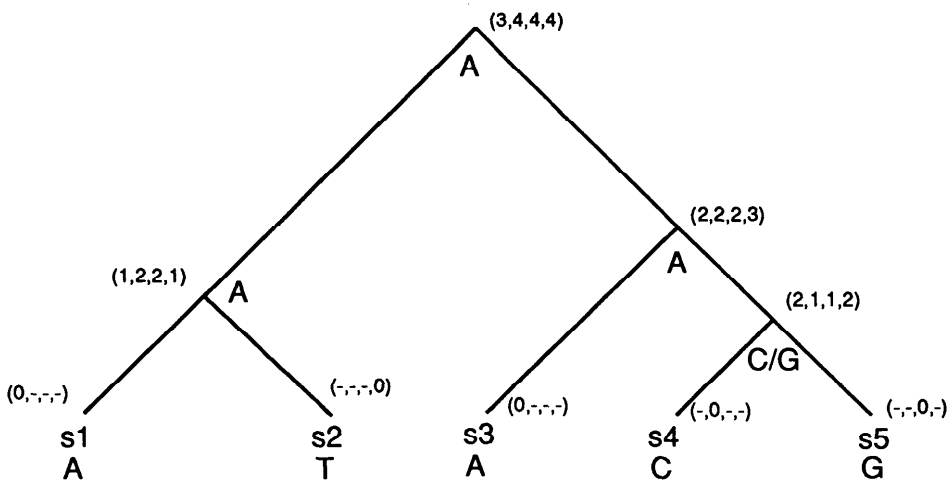


FIG. 1.—Rooted bifurcating tree with five tips, illustrating the FHS algorithm. At the tips five one-nucleotide sequences are located. The four-dimensional vector at each node is the weight of the minimal subtree hanging from that node if it is assigned A (position 1) to T (position 4) in the vector. A minimal assignment to all nodes can first take place when these vectors have been calculated for all nodes up to the root by using eq. (2). The simple metric only registering identities/nonidentities was used. In this simple case eq. (1) could have been used.

problem can be used for complete sequences if they are prealigned, by applying it to every column of the alignment. The big parsimony problem is analogous, except that the tree is not given and that it must be reconstructed also. The problem addressed in this section is a difficult extension of the small parsimony problem, since it involves complete sequences as well as indels.

The small parsimony problem in the case of the simple distance function between nucleotides that is 1 if the nucleotides are different and 0 if they are identical was first addressed by Fitch (1971) and, independently, by Hartigan (1973).

In the special case of the simple distance function on the set of nucleotides, the above procedure leads to the following simple rule. Let sn, sn_L, and sn_R be sets of nucleotides associated with a node, its left and right descendant, respectively. These sets obey the following recursion:

$$\begin{aligned} &\text{if } sn_L \cap sn_R \text{ is nonempty, then } sn \text{ is set to } sn_L \cap sn_R \text{ (case 1)}; \\ &\text{otherwise, } sn \text{ is set to } sn_L \cup sn_R \text{ (case 2)}. \end{aligned} \tag{1}$$

Each time case 2 occurs, an additional substitution must be added to the tree. This allows calculation of the weight of the complete tree. One parsimonious assignment (of many possible) can now be chosen in the following way by going down the tree: At the root assign any nucleotide, n, in sn. In case 1, n is also assigned to sn_L and sn_R. In case 2, n is assigned to whichever of sn_R and sn_L contains n and in the other set any nucleotide in sn can be assigned.

Sankoff (1975) presented the solution to the problem when the distance function between nucleotides was an arbitrary metric. Let dist be a metric on the set, N, of nucleotides and let $w(n)$, $w_L(n_L)$, and $w_R(n_R)$ be the most parsimonious weight of the subtrees hanging from a specified node, the subtree hanging from its left descendant,

and the subtree hanging from its right descendant, all under the restriction that the nucleotides assigned to these three nodes are n, n_L, and n_R, respectively. These quantities will satisfy the following recursion:

$$w(n) = \min_{n_L \varepsilon N}[w_L(n_L)+\text{dist}(n,n_L)] + \min_{n_R \varepsilon N}[w_R(n_R)+\text{dist}(n,n_R)] \quad n \varepsilon N, \quad (2)$$

subject to the initial condition $w(n) = 0$ if the node is a tip and n is the nucleotide present at this tip and to $w(n) = \infty$ if the node is a tip and n is not the nucleotide present there. The recursion is the basis of an algorithm that finds the minimal weight of the complete tree by a first upward pass through the tree from the tips and going toward an arbitrarily chosen root. A second downward pass will allow assignment of nucleotides to all internal nodes, by passing down the tree from the root toward the tips. At each node choose a nucleotide that can obey recursion (2), by entering the right side, when its ancestral nucleotide is at the left side in recursion (2). This will give one (of possibly many) most parsimonious assignments. The algorithm for solving the small parsimony problem will be called the FHS (Fitch-Hartigan-Sankoff) algorithm. The running time of the FHS algorithm is proportional to the number of tips in the tree.

The proof of recursion (2) relies only on the existence of a distance function on the set of nucleotides, so exactly the same recursion can be written for sequences instead of for nucleotides. Because of the size of the set of sequences in contrast to the set of nucleotides, the resulting recursion does not give a practical algorithm for calculating ancestral sequences.

One property of the solution of the recursion is the following: the state that is eventually assigned to an internal node can be different than that which would be assigned if the node had been the root in the small subtree hanging from it, and if the rest of the tree were unknown. In other words, when enlarging the tree one might have to give up what seemed to be the best solution. Assuming the contrary will be the kernel of an approximation that decomposes the many-sequence-alignment problem to a series of pairwise problems.

Let $d(\ ,\ )$ be a metric on sequences and let w, w_L, and w_R be the minimal weights of the trees hanging from the specified node and from its left and right descendant respectively. S, S_L, and S_R are the sets of sequences that are assigned to these nodes. The recursion now looks as follows:

$$w = \min[d(s_1,s_2)] + w_L + w_R \quad s_1 \varepsilon S_L,\ s_2 \varepsilon S_R . \quad (3)$$

The sequences associated with each node in the upward pass will be

$$S = [s: d(s_1,s)+d(s,s_2) = d(s_1,s_2) \text{ for some } s_1 \varepsilon S_L \text{ and } s_2 \varepsilon S_R] . \quad (4)$$

In other words, the sequences assigned to a node are those that do not impose extra evolution on the total history, relative to what would be necessary when the sequences at the left and right descendant are known. Sequences in S are said to be *between* S_L and S_R.

The set of sequences may still be huge, but fortunately it can be stored as sequences generated by a graph, itself inferred from a generalization of the traditional dynamic programming method. Sequence graphs are introduced to represent a large set of

sequences. The traditional dynamic programming algorithm is generalized to compare sequence graphs and to find sequence subgraphs that are close to each other according to some sequence metric. Simultaneously, the subgraphs are aligned. This permits a rational selection and representation of potential ancestral sequences among these sets of sequences, as more information is available in the form of closely related sequences. These graphs are used to represent potential ancestor sequences at each node in the phylogeny.

The next section will show how to find sequences between two single sequences. Then the method will be generalized for finding sequences between two sets of sequences, generated by two graphs. Both problems are closely related to the alignment of two sequences.

Sequences between Two Sequences

The majority of sequence comparison methods use some variety of dynamic programming. In molecular biology this technique was first used by Needleman and Wunsch (1970) to compare two protein sequences, by maximizing a similarity measure. The following method was put forth by Gotoh (1981): Let the length of the two sequences, s_1 and s_2 be $l1$ and $l2$, and let $s_1(i)$ and $s_2(j)$ refer to the ith and jth nucleotide in s_1 and s_2, respectively. Let $s_{1,i}$ and $s_{2,j}$ refer to the subsequences of s_1 and s_2 consisting only of the first i and j nucleotides, respectively. The distance between two nucleotides will be dist, and the gap penalty function for a gap of length k is of the form $g_k = a + (b \times k)$. The distances between sequences are of three types: $d(s_{1,i},s_{2,j},0)$, which is the smallest genetic distance between $s_{1,i}$ and $s_{2,j}$, subject to the constraint that $s_1(i)$ and $s_2(j)$ are paired; $d(s_{1,i},s_{2,j},1)$, which is the smallest genetic distance between $s_{1,i}$ and $s_{2,j}$, subject to the constraint that $s_1(i)$ is paired with a gap sign; and $d(s_{2,j},s_{1,i},1)$, which is the smallest genetic distance between $s_{1,i}$ and $s_{2,j}$, subject to the constraint that $s_2(j)$ is paired with a gap sign. Alignments ending in these three possibilities are said to be of type 0, 1, or 2, respectively. Optimality requires that $\max\{\text{dist}(\,,\,)\}$ is smaller than $2 \times b$. This precludes the possibility that a gap sign in s_1 is followed by a gap sign in s_2. These genetic distances will now obey the following recursion:

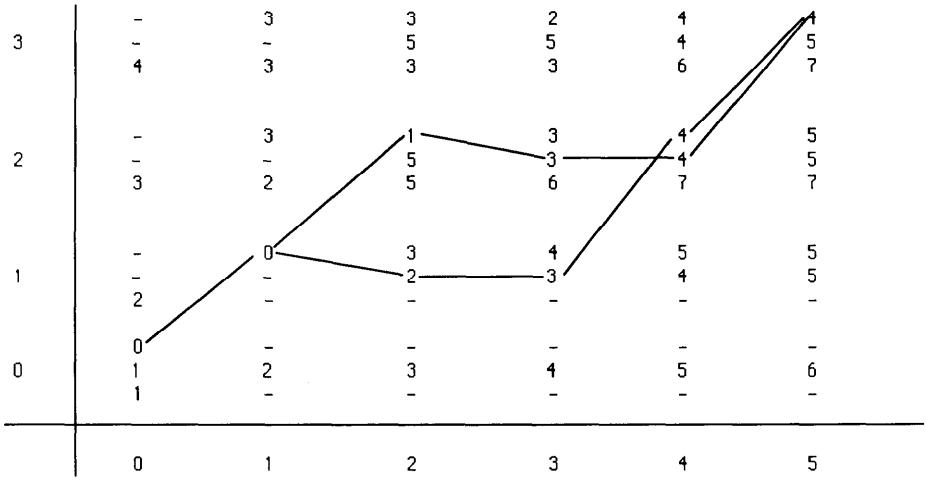
$$\begin{aligned} d(s_{1,i},s_{2,j},0) &= \min[d(s_{1,i-1},s_{2,j-1},0);\, d(s_{1,i-1},s_{2,j-1},1);\, d(s_{1,i-1},s_{2,j-1},2)] \\ &\quad + \text{dist}[s_1(i),s_2(j)]\,; \\ d(s_{1,i},s_{2,j},1) &= \min[d(s_{1,i-1},s_{2,j},1) + a;\, d(s_{1,i-1},s_{2,j},1)] + b\,; \\ d(s_{1,i},s_{2,j},2) &= \min[d(s_{1,i},s_{2,j-1},0) + a;\, d(s_{1,i},s_{2,j-1},2)] + b\,. \end{aligned} \tag{5}$$

These distances are subject to the boundary conditions $d(s_{1,0},s_{2,0},0) = 0$ and $d(s_{1,0},s_{2,0},1) = d(s_{1,0},s_{2,0},2) = a$. Again the alignment of the complete sequences can be found by backtracking. The algorithm can be implemented by storing $d(s_{1,i},s_{2,j},t)$ in a three-dimensional matrix at entry (i,j,t), where $t = 0, 1, 2$. The method is illustrated on two short sequences in figure 2. The performance of this algorithm is $0(1^2)$.

When recursion (5) is applied, not all entries in the distance matrices need to be calculated. One can cut corners, as observed by Fickett (1984) and Ukkonen (1985). The calculations can be confined to a belt around the two 45° diagonals in the matrix that go through (0,0) and ($l1$,$l2$), respectively. Let db be the thickness of the band above and below the two diagonals. Ukkonen showed, in the special case where a

a Sequences s1=TGAAT s2=TCT

b Distances between subsequences



c two alignments with minimal weight 4:

```
TGAAT        TGAAT
TC--T        T--CT
```

FIG. 2.—Gotoh's algorithm as applied to two short sequences. The weight of an indel of length k is here $1 + k$. a, The two sequences. b, The matrix with the three types of genetic distances. The three types are 0 at the top, then 1 and 2. A dash (-) indicates that the entry is not defined and therefore set to infinity. The entry (5,3,0) is underlined because it corresponds to the minimal alignment of the two sequences. If an entry is underlined and another entry can fulfill eq. (5) with it, then that entry will also be underlined and they will be connected by an edge. Thus, both (4,2,0) and (4,2,1) can be used in eq. (5) with (5,3,0), so they will also be underlined and connected to (5,3,0) with an edge. The graph defined in this fashion is called the backtracking graph. c, The two minimal alignments.

$= 0$ (thus three types of distances are identical), that if $d(s_1,s_2) < (|l1-l2|-db)/(2\times b)$, then the calculated area is large enough to guarantee that the alignment is minimal. This inequality is applied by first performing the distance calculations with a certain band width and then recalculating with a larger band width until the inequality is fulfilled. This inequality will give a band that in general can be made much thinner, while still permitting one to find an optimal solution. The band width is an input parameter.

The goal is now to construct a graph that will generate the set of sequences between s_1 and s_2. A graph, G, is a sequence graph if it is directed, noncyclic, and connected and has one sink (a node with only ingoing edges) and one source (a node

with only outgoing edges). (These last conditions are not necessary, but they are convenient for programming reasons.) Each edge can be associated with a nucleotide or with a set of nucleotides.

Traversing such a graph in all possible ways from source to sink will generate a set of sequences whose elements are sets of nucleotides. Selection of a nucleotide in each nucleotide set generates an even larger set of nucleotide sequences, S(G).

The sequence graph for an ordinary sequence of length l will be a linear graph with $l + 1$ nodes and one edge connecting them, and the ith edge will be associated with the ith nucleotide in the sequence.

The sequence graph generating the sequences between s_1 and s_2 will be closely related to the graph traversed when the backtracking graph [the BG(s_1,s_2) graph] is performed to obtain possible alignments. The nodes in the backtracking graph are parameterized by the coordinate position (i,j,t) in the three matrices.

The first node(s) in BG(s_1,s_2) is the one that corresponds to the distance between the complete sequences. It will be $(l1,l2,0)$, $(l1,l2,1)$, or $(l1,l2,2)$, which gives the lowest value of $d(s_{1,l1},s_{2,l2},t)$. The nodes of BG(s_1,s_2) can be found recursively by using recursion (5) to determine which (i,j,t) among $(l1-1,l2-1,t)$, $(l1-1,l2,t)$, and $(l1,l2-1,t)$ actually led to the previous node in BG(s_1,s_2). If this process is continued until (0,0,0) is reached, all nodes in BG(s_1,s_2) will eventually be found. Each time a node is used to define a new node, the first node will have an edge pointing to the second node; the edge will be defined to be of the same type as the first node.

Edges of type 0 are associated with the set of nucleotides (possibly one) consisting of the union of the nucleotides of the associated pairing. Edges of type 1 or type 2 are associated with the nucleotide that is not gapped.

If this graph is constructed for the example in figure 2, it will generate two new sequences besides the two input sequences, TCAAT and TGACT, between s_1 and s_2. The first sequence will match the first possible alignment, and the second sequence will match the second alignment. But the graph BG(s_1,s_2) does not generate all sequences between s_1 and s_2; it fails, for example, to generate TGT. That occurs because the graph, BG(s_1,s_2), obtained from backtracking requires that any piece of DNA involved in an indel be present in BG(s_1,s_2). For such missing sequences to be included, extra edges must be added to the graph. They are called *indel edges* and will not be associated with nucleotide sets. Their function is to represent the possibility that an indel was an insertion relative to the ancestral sequence. They will point from a node where a series of horizontal (type 1) or vertical (type 2) edges start, to a node where such a series ends. This modified graph will be called MBG(s_1,s_2). The MBG(s_1,s_2) derived from the alignment of the two sequences in figure 2 is shown in figure 3a. S[MBG(s_1,s_2)] is exactly all sequences between s_1 and s_2 according to the given metric.

Sequences between Sets of Sequences Generated by Graphs

The above algorithm solves the problem of constructing the most parsimonious ancestors of two single sequences. In the process of approximating the FHS algorithm, this would allow one to go back from sequences at the tips to their immediate ancestors—but not further. Therefore an algorithm is needed to align these ancestor sequence graphs and then to construct new sequence graphs to represent the sequences between the two sequence graphs. The following generalization of an algorithm of Sankoff and Kruskal (1983), combined with the method of Gotoh (1981), allows this. Let G_1 and

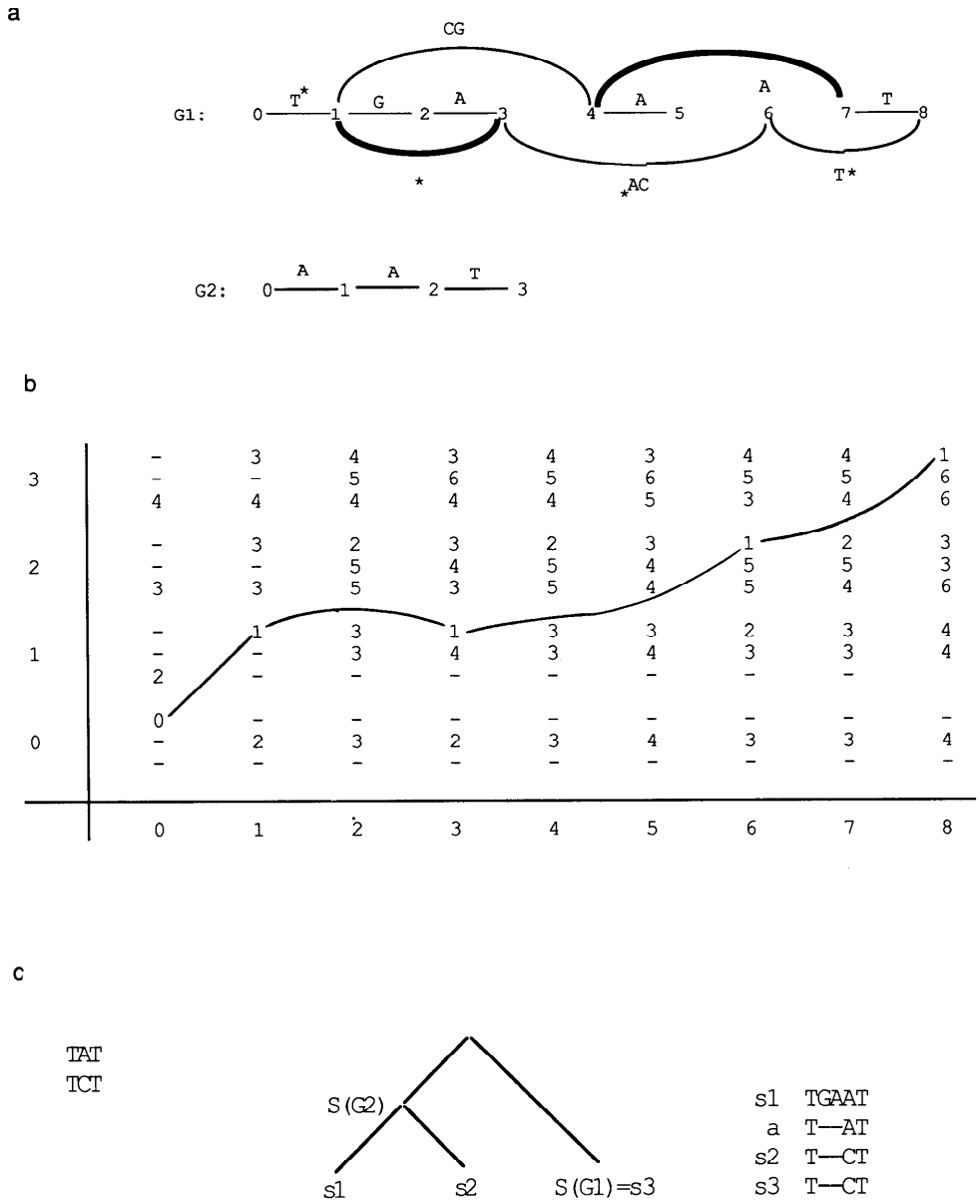


FIG. 3.—Sequence graph comparison algorithm. a, Two small sequence graphs. The length of a sequence graph is the number of nodes minus one. G1 is the modified backtracking graph from fig. 2, while G2 only represents one sequence. The edges and nucleotides in G1 that generated the closest sequence to S(G2) are indicated by asterisks. b, Dynamic programming algorithm. Analogous to ordinary sequence comparison, the distances between subgraphs are tabulated in a $\{0, \ldots, l1\} \times \{0, \ldots, l2\} \times \{0, \ldots, 2\}$ integer matrix. Minimal edges connecting minimal nodes are drawn. c, Overall use of the graph comparison algorithm. First, the two sequence graphs are compared, resulting in one alignment. S(G1) represents s1, s2, and four sequences between s1 and s2. S(G1) represents only one sequence, s3. The effect of introducing s3 is to select one alignment of s1 and s2 as more likely (the first alignment in fig. 2) and also to select the most likely ancestral sequence compatible with that alignment. The use of this algorithm makes it possible to reconstruct a complete history of the sequences. Since the comparison of G2 and s3 chose one alignment of s1 and s2 and also their immediate common ancestor, a, and since s3 was simultaneously aligned to a, all three sequences have been aligned and the one ancestral sequence in the tree relating them has been predicted.

G_2 be two sequence graphs. The distance between two such graphs is defined as the distance between the two closest nucleotide sequences in $S(G_1)$ and $S(G_2)$, i.e.,

$$d(G_1,G_2) = \min\{[d(s_1,s_2)]s_1 \in S(G_1), s_2 \in S(G_2)\} . \tag{6}$$

Let $G_{1,i}$ and $G_{2,j}$ be the subgraphs consisting only of nodes that can be reached from node i and node j (including these nodes), respectively, and only of edges involving these nodes. Let $d(G_{1,i},G_{2,j},t)$ be the distance between these two subgraphs, restricted to alignments of type t. The presence of indel edges leads to a nonstandard definition of a predecessor: Node i' is a predecessor of node i if there is a nondummy edge pointing from i to i' or if it is a predecessor of a node that has a dummy edge pointing to it from node i. Between each node i and a predecessor i' of it there will be exactly one edge with a set of associated nucleotides, and this will be denoted $n(i,j)$. Let $P(i,1)$ and $P(j,2)$ be the predecessors of i and j and G_1 and G_2, respectively. The distances between subgraphs will now obey a recursion similar to recursion (5).

$$\begin{aligned} d(G_{1,i},G_{2,j},0) &= \min[d(G_{1,i'},G_{2,j'},0); d(G_{1,i'},G_{2,j'},1); d(G_{1,i'},G_{2,j'},2)] \\ &\quad + \operatorname{dist}[n(i,i'),n(j,j')] ; \\ d(G_{1,i},G_{2,j},1) &= \min[d(G_{1,i'},G_{2,j'},0) + a; d(G_{1,i'},G_{2,j},1)] + b ; \\ d(G_{1,i},G_{2,j},2) &= \min[d(G_{1,i},G_{2,j'},0) + a; d(G_{1,i},G_{2,j'},2)] + b . \end{aligned} \tag{7}$$

The minimum is taken over i' in $P(1,i)$ and j' in $P(2,j)$. The function dist will now be a distance function on sets of nucleotides that is defined as the distance between the two closest members from each set. The initial conditions are $d(G_{1,0},G_{2,0},0) = 0$ and $d(G_{1,0},G_{2,0},1) = d(G_{1,0},G_{2,0},2) = a$. Again, successive applications of recursion (7) allow calculation of $d(G_1,G_2)$. Backtracking picks out the pairs of sequences in $S(G_1)$ and $S(G_2)$ closest to each other and simultaneously aligns them. If G_1 and G_2 represent ordinary sequences, then recursion (7) reduces to recursion (5).

For an illustration, see figure 3, where a sequence graph (G_1) and an ordinary sequence (G_2) are be compared and where the algorithm is used to decompose a triple sequence alignment problem to two pairwise problems.

The sequence graph that generates the sequences between $S(G_1)$ and $S(G_2)$ can be constructed in a manner strictly analogous to the method of constructing the sequence graph of two ordinary sequences. The resulting graph is called $MBG(G_1,G_2)$. $S[MBG(G_1,G_2)]$ coincides with the set defined by recursion (4). It is possible that $MBG(G_1,G_2)$ is too rich in the sense that it might contain a subgraph that will generate the same sequence set. An auxiliary algorithm was introduced to remove this redundancy and is described in the Appendix.

This algorithm functions as the upward pass in the FHS algorithm. The downward pass is a little more complex than in that algorithm and is now sketched. One history among these is to be chosen recursively according to the following principle: Assume that G_1G_2 and $MBG(G_1,G_2)$ are given and that one sequence, s, has been chosen in $S[MBG(G_1,G_2)]$. Now s_1 and s_2 in $S(G_1)$ and $S(G_2)$ must be chosen so that $d(s_1,s) + d(s,s_2) = d(G_1,G_2)$. When this is done it is possible to perform a downward pass that will choose one history with a weight consistent with the distances between all the sequence graphs.

Assume that s_1 is to be chosen. Label all edges in G_1 that have pointers pointing

to them from the path in MBG(G_1,G_2) that generated s. If these edges constitute a path from source to sink, then s_1 can be chosen in the set of sequences generated by this path by choosing in the nucleotide set just as in the one-nucleotide case in the original FHS algorithm. If the labeled edges do not connect source and sink in G_1, edges must be added to make it connect source and sink. The added edge corresponds to insertions in the history from s to s_1, and it will correspond to a dummy edge in MBG(G_1,G_2) that jumped over this segment. The segment will correspond to a shortest path connecting the same node as does the dummy edge which consists only of edges of type 1. Any nucleotides can be chosen from the sets of nucleotides associated with the edges of this path. A similar procedure is carried out for s_2.

The concept of sequence graph is now to be used to fashion an algorithm that reconstructs the history of many sequences by sequentially aligning sequence graphs, instead of by aligning all sequences simultaneously.

The algorithm described in the present article is then combined with the tree-constructing algorithm given in the accompanying article (Hein 1989) to give an algorithm that simultaneously aligns and reconstructs the phylogeny.

An Example

A program written in C implements the algorithm. When run on a Celerity C1260, many (>50) sequences ≤3,000 bp long could be compared. Because of growth in the size of the sequence graphs used to represent ancestral sequences, it is preferable that the sequences are reasonably related. If <50% of the positions of two aligned DNA sequences are different, then they are more than sufficiently related. When the sequences are proteins, the corresponding figure is 25%–30%. At the other extreme, if completely unrelated sequences are compared, the number of equally good alignments very quickly grows to unmanageable numbers. The method was applied to the following data set.

5S RNAs from Very Distantly Related Organisms

A set of 5S RNA sequences from very distantly related organisms were studied, not so much to offer new taxonomy but to illustrate the method. It is preferable to use more and longer sequences, but the size of the resulting alignment then makes it unsuitable for an illustration.

The 22 5S RNAs were taken from GENBANK 55. The gap penalty function was $10 + (3 \times k)$, and any substitution cost 4. It took 165 s on the Celerity computer. The total weight of the history is 1,628. The resulting alignment including ancestral sequences is given in figure 4. The corresponding phylogeny is shown in figure 5, and both the number of substitutions and the number of indels are shown in table 1. The substitutions and indels are without time direction, since the phylogeny is unrooted. A higher frequency of transitions than of transversions was observed. Of the six possible kinds of substitution without time direction, two are transitions and four are transversions. The number of transitions is overrepresented relative to the uniform distribution. This lends credibility to the method, since this has been observed for the evolution of other sequences, where the reconstruction of their history was unproblematic.

The alignment has 42 sequences, 20 of which are ancestral. The first group of sequences is from plants and is followed by groups from the following: chloroplasts, bacteria, fungi, and animals. The single sequences are the proposed ancestral sequences in the tree connecting these five groups. For instance, sequence 30 is the proposed


660

```
-atc-c--gcggccatagcaacgagaaaacaccggatc--ccatcc-gaactccgaagttaagcacgttagcgcccggatagtactg-gggtgggggaccacccgggaatacct---ggtgctgcag---ga-------  38

-atc-c--acggccataggactctgaaagcactgcatc--ccgtcc-gatctgcaaagttaaccagagtaccgcccagttagtacca-cggtgggggaccacgcgggaatcctg---ggtgctgtgg---tt-------  12
-atc-c--acggccataggaccctgaaagcaccgcatc--ccgtcc-gatctgcgcagttaaccagggtaccgcccagttagtacca-cggtgggggaccacgcgggaatcctg---ggtgctgtgg---tt-------  31
-atc-c--acggccataggaccctgaaagcaccgcatc--ccgtcc-gatctgcgcagttaaccagggtgccgcctagttagtacca-cggtgggggaccacgcgggaatccta---ggtgctgtgg---tt-------  11
-atc-c--acggccataggaccctgaaaacaccgcatc--ccgtcc-gatctgcgcagttaagcagggtaccgcccagttagtaccg-gggtgggggaccacccgggaatcctg---ggtgctgtgg---tt-------  32
-atc-c--acggccataggacacagaaaacatcgcatc--ccgtcc-gatctgcgcaatcaagctgtgtaccgcccagtcagtaccg-gagtgggggaccatccgggaatcctgccaggtgctgtgg---tt-------  10

-atc-c--acggccatagcaccctgaaaacaccggatc--ccgtcc-gatctccgaagttaagcagggtagcgcccggttagtactg-gggtgggggaccacccgggaataccg---ggtgctgtag---tt-------  37

-gcc-a--acgaccataccacgctgaatacatcggttc--tcgtcc-gatcaccgaaattaagcagcgtcgcgggcggttagtactt-agatgggggaccgcttgggaacaccg---cgtgttgttg-gcct-------  22
-gcc-a--acggccataccaccctgaatacaccggatc--tcgtcc-gatctccgaagttaagcagggtcgcgcccggttagtactt-gggtgggggaccgcctgggaataccg---ggtgctgtag-gctt-------  36
-gct-t--acggccataccagcctgaatacgcccgatc--tcgtcc-gatctcggaagctaagcagggtcgggcctggttagtactt-ggatgggagaccgcctgggaatacca---ggtgctgtaa-gctt-------  20
-gcc-t--acggccataccaccctgaatacgcccgatc--tcgtcc-gatctcggaagctaagcagggtcgggcctggttagtactt-ggatgggagaccgcctgggaataccg---ggtgctgtag-gctt-------  35
-gcc-t--acggccacaccaccctgaaagtgcctgatc--tcgtct-gatctcagaagcgatacagggtcgggcctggttagtacct-ggatgggagaccgcctgggaatacca---ggtgtcgtag-gctt-------  18
-gcc-t--acggccataccaccctgaaagcgcccgatc--tcgtct-gatctcggaagctaagcagggtcgggcctggttagtactt-ggatgggagaccgcctgggaataccg---ggtgctgtag-gctt-------  33
-gtc-t--acggccataccaccctgaacgcgcccgatc--tcgtct-gatctcggaagctaagcagggtcgggcctggttagtactt-ggatgggagaccgcctgggaataccg---ggtgctgtag-gctt------t  19
-gcc-t--acggccataccaccctgaaaacgcccgatc--tcgtct-gatctcggaagctaagcagggtcgggcctggttagtactt-ggatgggagaccgcctgggaataccg---ggtgctgtag-gctt-------  34
-gcc-t--acggccatcccaccctggtaacgcccgatc--tcgtct-gatctcggaagctaagcagggtcgggcctggttagtactt-ggatgggagacctcctgggaataccg---ggtgctgtag-gctt-------  21
```

FIG. 4.—Alignment of the 42 [22 (extant sequences) + 20 (ancestral sequences)] sequences. The genetic events that have occurred in the history of these sequences can be obtained by combining the information in the alignment and the phylogeny in fig. 5. Take two sequences corresponding to the nodes connected by an edge in the phylogeny. Compare the sequences site by site, and a list of events will be obtained. The different types of events in the whole tree are shown in table 1.

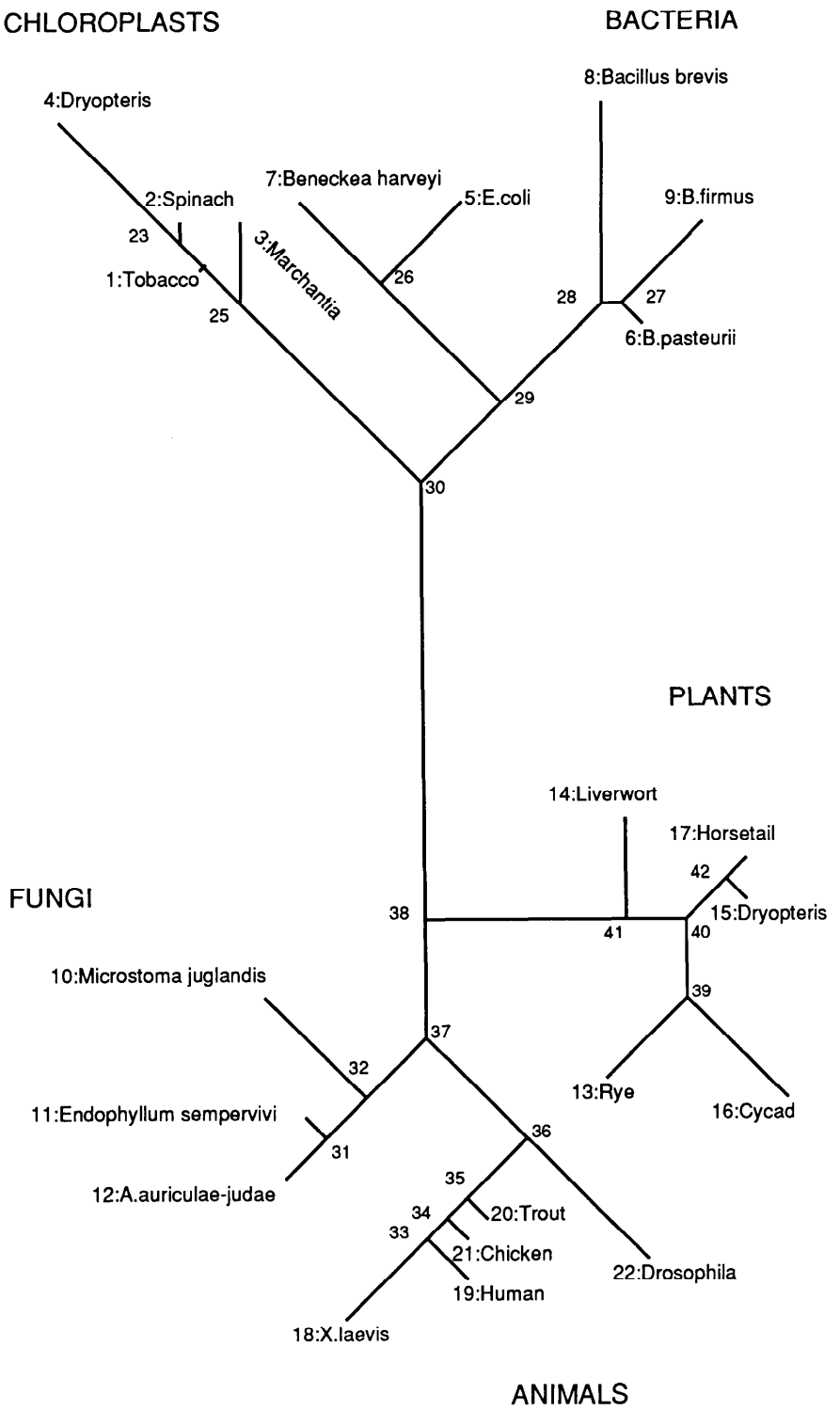


FIG. 5.—Phylogeny of the 22 5S RNA sequences. The total branch length of this phylogeny is 1,628. The length of each edge is proportional to the weight of the differences between the sequences at the end of the edge.

Table 1
Mutational Events (substitutions and indels) Occurring in the History of the Sequences Analyzed in Figures 4 and 5

A. Substitutions

	A	C	G
C	38		
G	84	48	
T	41	86	35

B. Indels

	Length		
	1	2	3
No.	13	7	1

NOTE.—Since the obtained tree is unrooted, these events have no time direction.

common ancestor for both bacteria and chloroplasts. This phylogeny is in reasonable accordance with other investigations of the same sequences (Hori and Osawa 1987).

If approximate rate constancy is assumed, it is likely that the root of this tree is on the long branch connecting bacteria and chloroplasts with the eukaryotes, a biologically reasonable situation. In all investigations of 5S RNAs by this method, fungi and animals were more closely related to each other than to plants. There are a few places where the phylogeny does not match what one would expect. Birds and reptiles (rather than mammals and reptiles, as is the case in this phylogeny) are normally assumed to be sister groups, but the lengths of the inner branches separating these sequences are very short. Within the chloroplasts, dryopteris should have been an outgroup and spinach and tobacco ought to have been brothers.

Discussion

Aligning n sequences takes $n - 1$ graph comparisons, when the tree relating the sequences is given. Nevertheless, the total performance of the new algorithm presented here is difficult to determine precisely. There are several reasons why exact upper bounds are hard to define. First, the graph-pruning algorithm consumes negligible time but probably has a poor worst-case complexity. Second, the sequence graphs could grow to limiting sizes, but for real data they stay constant in size. For reasonably homologous sequences, the average time used is probably proportional to the product of the average sequence length and their number, $1 \times n$, while the worst-case running time could be much worse.

The average performance of the algorithm when the tree is not given is also very good. The number of pairwise comparisons necessary to make the initial tree is probably $o(n^2)$ (a decreasing fraction of the distance matrix is calculated as the number of sequences increases), but, again, the number of cycles of nearest-neighbor interchanges necessary is hard to assess. The time consumed for a single sequence comparison grows slower than $s \times 1$, where s is the distance between the two sequences (or sequence

graphs). So the time consumption probably grows slower than $l^2 \times n^2$, which is an enormous increase in speed relative to the original many-sequence comparison algorithm. So it must be concluded that, as a practical algorithm, it performs well and can handle large data sets in a reasonable time (150 5S RNAs in <20 min).

Since the method claims to give a complete history of the sequences, it is natural to ask questions about the reliability of the method. Since the method has removed much that was arbitrary from previous methods, it should give more-reliable histories. The picture is probably the same as for other parsimony methods, where reliability decreases as the branch lengths in the phylogeny get longer.

There exists a series of methods that produce alignments without making explicit predictions about the history of the sequences, instead of just aligning similar regions of the sequences (e.g., see Bacon and Anderson 1986; Bain 1986; Johnson and Doolittle 1986; Sobel and Martinez 1986; Waterman 1986). The present method is fully competitive in terms of speed and size of data set it can handle, and the end result is much more ambitious, in that it gives a complete history of the sequences. However, some of these methods involve testing whether two (or more) sequences are homologous, i.e., have evolved from a common ancestor. This is usually taken as being equivalent to deciding whether the sequences are closer than should be expected if they were two (or more) "random" sequences. Although the present method does not include such a feature, it is still possible that it could be helpful in detecting very ancient relationships. The reason that certain homologous pairs of sequences fail to be detected by homology tests is that their common ancestor is too far back in time and that they have been subject to too many events. If the two sequences in question were members of a large family of known sequences, an obvious approach to this problem would be to reconstruct ancestral sequences as far back in time as possible and then apply a homology test to these sequences.

Feng and Doolittle (1987) and Konings et al. (1987) have recently published algorithms that simultaneously align and reconstruct phylogeny of a set of sequences. The method described in the present article should have the following advantages over these methods: First, it is based on a method that does not assume constancy in evolutionary rates. Second, thanks to the concept of sequence graph, it can consider many equally good alignments when reconstructing ancestral sequences and thus avoids an abitrary choice of alignment at this point. Third, it does not start by calculating all entries in the distance matrix, since this is not necessary, thereby giving it an advantage in speed. Last, the nature of the approximation to the general parsimony problem is described in recursions (3) and (4) and thus is very clear.

The following three problems were encountered in the course of devising this algorithm and writing the program:

1. Given a sequence graph, find an efficient algorithm that finds a minimal subgraph generating the same set of sequences.

2. Because the method always kept only the most parsimonious sequence sets when going upward in the tree, how much should this set have to be enlarged to guarantee finding the minimal alignment, given the phylogeny? Is it possible to combine with a suitable inequality methods giving close to optimal solutions (Waterman 1983; Byers and Waterman 1985) to make such an algorithm? Previously it was investigated how far the pairwise alignments were from being the corresponding marginal alignments in the three-sequences case, which is the same problem. How wrong is it to assume that the three-sequence problem can be reduced to two-sequence problems? Typically the three-sequence alignment could be deduced from the pairwise sequence

alignments or would be very close. It is not unreasonable to assume that only very close to minimal ancestors would have to be investigated in order to find the ancestors that would be minimal for the complete tree. This would allow a decomposition of Sankoff's original algorithm to two-sequence (graph) comparisons. It would likely mean an enormous speedup relative to the original algorithm and give a practical algorithm, without loss of minimality.

3. The two-sequence graph algorithm could be generalized to k-sequence graph comparisons (suggested to me by D. Sankoff), which undoubtedly would improve the result of the method but also would slow it down.

The following extension is being undertaken: Some molecular evolutionists (R. Garrett, personal communication) contend that, when reconstructing very ancestral events from ribosomal RNAs, it is essential to introduce a differential weighting to remove "noise" from highly variable sites. In present methods this must be done by letting the weighting be in the form of user-specified parameters, thereby introducing an ad-hoc flavor to the analysis. It should be very easy to introduce an automatic weighting into the presented algorithm as follows: The sequence (graph) algorithm can be generalized to include weighting of sites (positions). When the history of a set of closely related molecules has been reconstructed, it will be known which sites are rapidly evolving and which are not, and they could be weighted accordingly. This would both define important regions of the molecules and remove the noise of highly variable sites. Weighting in the phylogenetic parsimony problem has previously been undertaken by Farris (1969), Felsenstein (1981), and Williams and Fitch (1989), but without simultaneously incorporating alignment.

Acknowledgments

The author had useful discussions or received help with the manuscript from Russell Doolittle, Joseph Felsenstein, Walter Fitch, Charles Langley, Marcella McClure, David Sankoff, William Saurin, Jeff Thorne, and Michael Waterman. The author was supported by System Development Foundation grant GO15 and NIH grant GM36230.

APPENDIX

Pruning of Sequence Graphs

The sequence graphs constructed will have one difficulty; they are redundant in the sense that edges and perhaps nodes can be removed without diminishing the set of generated sequences. This is because different alignments can lead to the same set of ancestral sequences. To envision, notice that s_1 = CCC and s_2 = C can be aligned with a gap of length two either in the beginning or at the end but that both alignments will have the same set of potential ancestral sequences, namely, C and CCC. Since the algorithm will eventually choose only one sequence at each node, there is no reason to postpone the removal of this wasteful representation until the root of the complete tree has been reached. Not to remove this redundancy is computationally so wasteful as to render the algorithm useless for large number of sequences.

The following algorithm is introduced to improve on this: In the graph any node that must be traversed by a path going from source to sink is an articulation point (Wilson 1985). A subgraph in which the nodes with highest and lowest indices are articulation points but in which all points between them are not articulation points is called a *puff*. See figure 6a. It is not hard to see that any pruning algorithm can consider each puff separately. So assume that the given graph is a puff and that it starts at node b and ends at node e. At the beginning, all edges are unlabeled. The graph is now traversed by a depth-first search. Each time e is reached, a new path

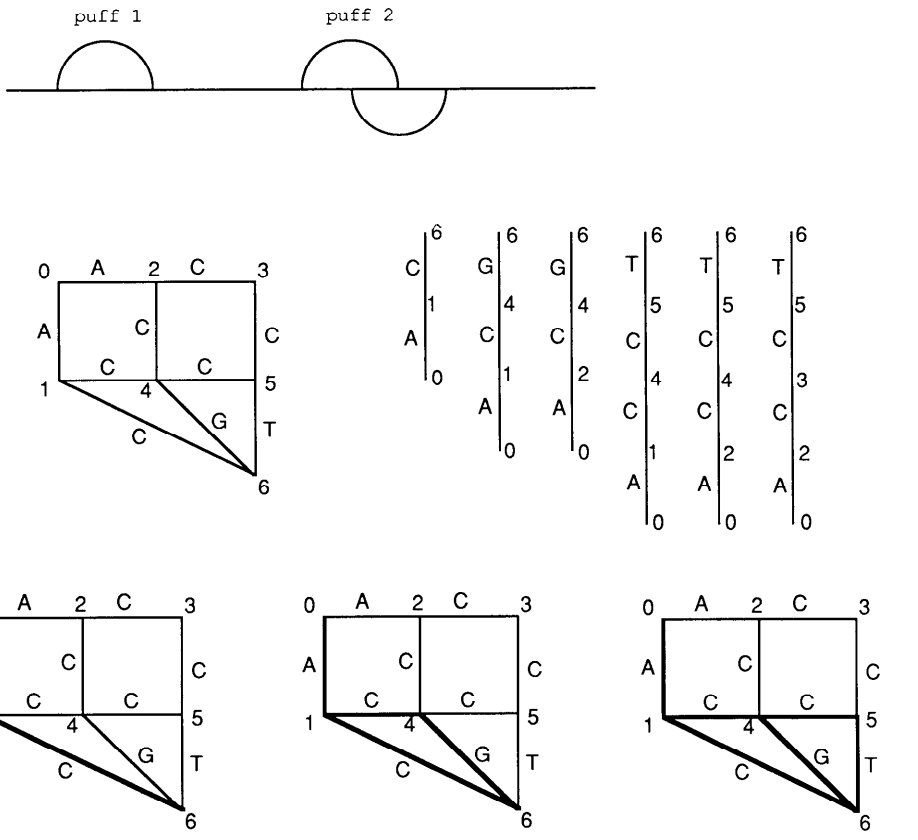


FIG. 6.—Pruning algorithm. *Top,* Sequence graph with two puffs on it that can be treated separately when checking the graph for redundancy. *Middle,* Another sequence graph, one with the six possible paths from node 6 to 0. *Bottom,* Consecutive labeling of the sequence graph as the paths are investigated.

through the graph has been generated, and all possible paths through the graph are gencrated by doing this. During this search, edges of the graph will be labeled in the following manner: Each time a new path is generated it is checked to see whether it can be embedded in the subgraph with only labeled edges; if not, all its edges are labeled as well. *Embedded* means that a path exists in the subgraph of the same length and that the nucleotide sets associated with corresponding edges are such that each nucleotide set of the new path is equal to or contained in the nucleotide set of the path in the subgraph.

For an illustration of this algorithm, see figures 6b–6c. In figure 6b is shown a sequence graph and the six possible paths in the order in which they are generated by a depth-first search. The path of length two has its edges labeled, since it is the first path; so does the second path, since it is the first path of length 3; but the third path will not, since the corresponding sequence is generated by the subgraph with only labeled edges. The fourth path will be labeled, since it is the first of length four, but the last two paths will not be labeled. Thus, in total, two nodes and four edges can be removed without diminishing the set of generated sequences.

LITERATURE CITED

BACON, D. J., and W. F. ANDERSON. 1986. Multiple sequence alignment. J. Mol. Biol. **91**:153–161.

BAIN, W. 1986. MULTAN: a program to align multiple DNA sequences. Nucleic Acids Res. **14**:159–179.

BYERS, T. H., and M. S. WATERMAN. 1985. A dynamic programming algorithm to find all solutions in a neighborhood of the optimum. Math. Biosci. **77**:179–188.

FARRIS, J. S. 1969. A successive approximations approach to character weighting. Syst. Zool. **18**:374–385.

FELSENSTEIN, J. 1981. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. Biol. J. Linnaean Soc. **16**:183–196.

———. 1982. Inferring phylogenies from quantitative data. Q. Rev. Biol. **57**:379–404.

———. 1989. PHYLIP manual version 3.2. University of California Herbarium, Berkeley.

FENG, D. F., and R. F. DOOLITTLE. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. **25**:351–361.

FICKETT, J. 1984. Fast optimal alignment. Nucleic Acids Res. **12**:175–180.

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

FOULDS, L. R., and R. L. GRAHAM. 1982. The Steiner problem in phylogeny is NP-complete. Adv. Appl. Math. **3**:43–49.

FREDMAN, M. L. 1984. Algorithms for computing evolutionary similarity measures with length independent gap penalties. Bull. Math. Biol. **46**:545–563.

GOTOH, O. 1981. An improved algorithm for matching biological sequences. J. Mol. Biol. **162**: 705–708.

HARTIGAN, J. A. 1973. Minimum mutation fits to a given tree. Biometrics **29**:53–65.

HEIN, J. J. 1989. A tree reconstruction method that is economical in the number of pairwise comparisons used. Mol. Biol. Evol. **6**:669–684.

———. A large version of the small parsimony problem. J. Theor. Biol. (submitted).

HORI, H., and S. OSAWA. 1987. Origin and evolution of organisms as deduced from 5S ribosomal RNA sequences. Mol. Biol. Evol. **4**:445–472.

JOHNSON, M. S., and R. F. DOOLITTLE. 1986. A method for the simultaneous alignment of three or more amino acid sequences. J. Mol. Evol. **23**:267–278.

KONINGS, D. A. M., P. HOGEWEG, and B. HESPER. 1987. Evolution of the primary and secondary structures of the E1a mRNAs of the adenovirus. Mol. Biol. Evol. **4**:300–314.

NEEDLEMANN, S. B., and C. D. WUNSCH. 1970. A general method applicable to the search for similarities in the amino acid sequences of two proteins. J. Mol. Biol. **48**:444–453.

SANKOFF, D. 1972. Matching sequences under deletion/insertion constraints. Proc. Natl. Acad. Sci. USA **69**:4–6.

———. 1975. Minimal mutation trees of sequences. SIAM J. Appl. Math. **78**:35–42.

SANKOFF, D., and J. KRUSKAL. 1983. Time warps, string edits and macromolecules: the theory and practice of sequence comparison. Addison-Wesley,

SANKOFF, D., C. MOREL, and R. J. CEDERGREN. 1973. Evolution of 5S RNA and the non-randomness of base replacements. Nature New Biol. **245**:232–234.

SELLERS, P. 1974. An algorithm for the distance between two finite sequences. J. Combination Theory **16**:253–258.

SMITH, T. F., M. S. WATERMAN, and W. M. FITCH. 1981. Comparative biosequence metrics. J. Mol. Evol. **18**:38–46.

SOBEL, E., and H. MARTINEZ. 1986. A multiple sequence alignment program. Nucleic Acids Res. **14**:363–375.

SWOFFORD, D. L. 1985. Phylogenetic analysis using parsimony: users manual. Illinois Natural History Survey, Champaign.

UKKONEN, E. 1985. Algorithms for approximate string matching. Inf. Control **64**:100–118.

WATERMAN, M. S. 1983. Sequence alignments in the neighborhood of the optimum with general applications to dynamic programming. Proc. Natl. Acad. Sci. USA **80**:1381–1384.

WATERMAN, M. S., T. F. SMITH, and W. A. BEYER. 1976. Some biological sequence metrics. Adv. Math. **20**:367–387.

WATERMAN, M. S. 1986. Multiple sequence alignment by consensus. Nucleic Acids Res. **14:** 9095–9102.

WILLIAMS, P. L., and W. M. FITCH. 1989. Finding the minimal change in a given tree. Pp. 453–470 *in* H. FERMHOLM, K. BREMER, and H. JORN, eds. The hierarchy of life. Elsevier Science, Amsterdam.

WILSON, R. B. 1985. An introduction to graph theory. 3d ed. Longman, Essex, England.

WALTER M. FITCH, reviewing editor

Received April 13, 1989; revision received July 14, 1989.

Accepted July 17, 1989.