Probabilistic ancestral sequences and Multiple alignments

Gaston H. Gonnet and Steven A. Benner
Informatik

Organic Chemistry
E.T.H.
Zurich, Switzerland

Abstract

An *evolutionary configuration* (EC) is a set of aligned sequences of characters (possibly representing amino acids, DNA, RNA or natural language). We define the probability of an EC, based on a given phylogenetic tree and give an algorithm to compute this probability efficiently. From these probabilities, we can compute the most likely sequence at any place in the phylogenetic tree, or its probability profile. The probability profile at the root of the tree is called the probabilistic ancestral sequence. By computing the probability of an EC, we can find by dynamic programming alignments over two subtrees. This gives an algorithm for computing multiple alignments. These multiple alignments are maximum likelihood, and are a compatible generalization of two sequence alignments.

1 Introduction

In our view the main contribution of this paper is to introduce the computation of probabilities of a given evolutionary configuration (EC), that is a phylogenetic tree with sequences at its leaves, based on a Markovian model of evolution. From this probabilities, we can compute the probability profile at any internal node or internal branch of the phylogenetic tree, and when these are computed at the root of the tree, we call them *probabilistic ancestral sequences* (PAS). The ability to compute probabilities of particular configurations, allows us to use a dynamic programming (DP) algorithm to align two subtrees. By starting from the leaves and working towards the root of the tree with this procedure, we construct a multiple alignment of all the sequences.

The problem of computing multiple alignments is a difficult one. All of the existing algorithms [7, 2, 5, 11, 14, 19, 16, 12, 1, 13] are either heuristic, or require exponential time. The algorithm here described is the first one to work in polynomial time and to be founded on an accepted model of evolution. These methods can be applied to DNA, RNA, natural language or any other similar problem. We will denote by Σ the alphabet over which these sequences are built. All of our algorithms work for arbitrary alphabets. In this paper, all the examples are on proteins. This particular case can be summarized as being over an alphabet of 20 symbols, the letters $\Sigma =$ ACDEFGHIKLMNPQRSTVWY.

This paper will not discuss how to construct the phylogenetic trees, this subject falls beyond our present scope. The algorithms we describe assume we start with the sequences and a phylogenetic tree.

1.1 Definitions

Mutation model. The evolution between sequences can be modelled by mutation matrices. A mutation matrix, denoted by M, describes the probabilities of amino acid mutations for a given period of evolution.

$$Pr\{\text{amino acid } i \longrightarrow \text{amino acid } j\} = M_{ji}$$

This corresponds to a model of evolution in which amino acids mutate randomly and independently from one another but according to some predefined probabilities depending on the amino acid itself. This is a Markovian model of evolution and while simple, it is one of the best models.

Intrinsic properties of amino acids, like hydrophobicity, size, charge, etc. can be modelled by appropriate mutation matrices. Dependencies which relate one amino acid characteristic to the characteristics of its neighbours are not possible to model through this mechanism.

Amino acids appear in nature with different frequencies. These frequencies are denoted by f_i and correspond to the steady state of the Markov process defined by the matrix M. I.e. the vector f is any of the columns of M^∞ or the eigenvector of M whose corresponding eigenvalue is 1 ($Mf = f$).

This model of evolution is symmetric, i.e. the probability of having an i which mutates to a j is the same as starting with a j which mutates into an i. This symmetry means

$$f_i M_{ji} = f_j M_{ij}$$

PAM distance. In the definition of mutation, we use the matrix M which implies certain amount of mutation. For all our work it is important to quantify mutation. We will use the terminology introduced by Dayhoff et al. [6] where amount of mutation is measured in PAM units.

A 1-PAM mutation matrix describes an amount of evolution which will change, on the average, 1% of the amino acids. In mathematical terms this is expressed as a matrix M such that

$$\sum_{i \in \Sigma} f_i(1 - M_{ii}) = 0.01$$

The diagonal elements of M are the probabilities that a given amino acid does not change, so $(1 - M_{ii})$ is the probability of mutating away from i.

If we have a probability or frequency vector p, the product Mp gives the probability vector or the expected frequency of p after an evolution equivalent to 1-PAM unit. Or, if we start with amino acid i (a probability vector which contains a 1 in position i and 0s in all others) M_{*i} (the i^{th} column of M) is the corresponding probability vector after one unit of random evolution. Similarly, after k units of evolution (what is called k-PAM evolution) a frequency vector p will be changed into the frequency vector $M^k p$. Notice that chronological time is not linearly dependent on PAM distance. Evolution rates may be very different for different species and different proteins.

Phylogenetic tree. A phylogenetic tree is a representation of evolution. In our case, since we are interested in amino acid sequences, these trees normally describe the evolution of the species where the sequences are found. All the phylogenetic trees are based on a model or theory of evolution. The model of evolution which we consider here is a simple model based on ancestor sequences mutating into descendant sequences where no parallel evolution or horizontal transfers will be considered.

We will assume that we do not have any sequence information corresponding to the internal nodes of the phylogenetic tree. This is the normal situation. All the sequence information is at the leaves. Our phylogenetic trees are always binary (each internal node has degree 3), that is to say, every internal node corresponds to a splitting point in the phylogeny.

In our case, the leaves of the phylogenetic tree are the sequences from present day species. The internal nodes of the tree represent the points of divergence where two different branches of evolution arose. The root of the tree represents the nearest (latest) common ancestor of all the species considered. For example Figure 1 is a phylogenetic tree where 1, 2, ..., 5 represent the present-day sequences, W represents the nearest common ancestor of 1 and 2, X is the nearest common ancestor of 4 and 5 and O, the root of the tree, represents the nearest common ancestor of all the sequences.

We are interested in trees with branches which measure amount of evolution in PAM units. We call this an evolutionary phylogenetic tree (EPT). There is no obvious constraint between the distances from the root to the leaves, as it is well known that similar proteins in different species may evolve at different rates.

In Figure 1, if lengths mean amount of evolution, we could say that 3 evolved more from O than 2 and 4. Since we assume that all these descended from a common ancestor, we can also say that 3 evolved more rapidly (mutated more rapidly) than 2 and 4, since in the same amount of time it evolved more.

When the ancestral sequences (O, Y, X and W) are not available for analysis, as it is usually the case, we have to infer their existence and location from the present-day sequences 1, 2, ..., 5. Phylogenetic trees derived only from the information on the leaves are normally called dendrograms.

The evolution from O to Y will be modelled by a mutation matrix M^{d_Y}, where d_Y is the PAM distance between O and Y; the evolution from O to 1 by

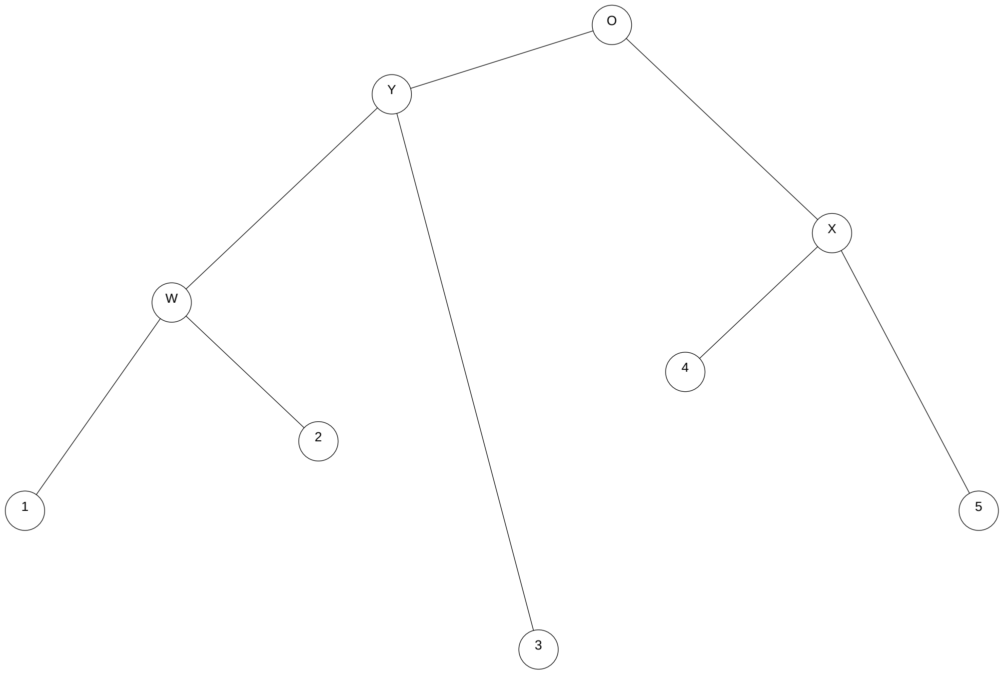

Figure 1: Example of an EPT with 5 leaf nodes

$M^{d_{O1}} = M^{d_Y+d_W+d_1}$, etc. The exponent d_{ij} denotes the distance between node i and node j. d_i (with a single index) denotes the distance between node i and its parent.

Dayhoff matrices. Dayhoff et al. [6] presented a method for estimating the matrix M from the observation of 1572 accepted mutations between 34 superfamilies of closely related sequences. Their method was pioneering in the field. Nowadays we are able to estimate M by more accurate and better founded methods [8, 4].

A Dayhoff matrix (in honour of Margaret O. Dayhoff) is a matrix, computed from a 250-PAM mutation matrix, used for the standard dynamic programming method of sequence alignment. The Dayhoff matrix entries are related to M^{250} by

$$D_{ij} = 10 \log_{10} \frac{(M^{250})_{ij}}{f_i}$$

Sequence alignment. Aligning sequences is the process of associating some positions of each sequence with a position of the other sequence. This association preserves the order of the sequences. E.g.

```
VNRLQQNIVSL____________EVDHKVANYKPQVEPFGHGPIFMATALVPGLYLGVPWF
VNRLQQSIVSLRDAFNDGTKLLEELDHRVLNYKPQANPFGNGPIFMVTAIVPGLHLGAPWF
```

Unassociated positions are called *insertions* (or their counterpart *deletions*). Aligning protein sequences by dynamic programming (DP) using Dayhoff matrices is equivalent to finding the alignment which maximizes the probability that the two sequences evolved from an ancestral sequence as opposed to being random sequences. More precisely, we are comparing two events

a) that the two sequences are independent of each other, and hence an arbitrary position with amino acid i aligned to another arbitrary position with amino acid j has the probability equal to the product of the individual frequencies
$$Pr\{i \text{ and } j \text{ are independent}\} = f_i f_j$$

b) that the two sequences have evolved from some common ancestral sequence after t units of evolution.

$$\begin{aligned} Pr\{i \text{ and } j \text{ descended from } x\} &= \sum_x f_x Pr\{x \to i\} Pr\{x \to j\} \\ &= \sum_x f_x (M^t)_{ix} (M^t)_{jx} \\ &= \sum_x f_j (M^t)_{ix} (M^t)_{xj} \\ &= f_j (M^{2t})_{ij} = f_i (M^{2t})_{ji} \end{aligned}$$

We use $\sum_i$ as a shorthand for $\sum_{i \in \Sigma}$, that is a sum over all symbols of the alphabet. The entries of the Dayhoff matrix are the logarithm of the quotient of these two probabilities.

$$D_{ij} = 10 \log_{10} \left(\frac{Pr\{i \text{ and } j \text{ descended from } x\}}{Pr\{i \text{ and } j \text{ are independent}\}} \right)$$

Since DP maximizes the sum of the similarity measure, DP maximizes the sum of the logarithms or maximizes the product of these quotients of probabilities. As a conclusion, DP finds the alignment which maximizes the probability of having evolved from a common ancestor (a maximum likelihood alignment) against the null hypothesis of being independent. This makes aligning sequences using Dayhoff matrices a soundly based algorithm.

Multiple sequence alignment. A multiple sequence alignment, or multiple alignment, is a generalization of the case of 2 sequences to k sequences. Instead of relating the sequences by a single parent, we have an EPT which describes the parent-descendant relations. Dynamic programming is no longer applicable in practice, it requires exponential space and time in k. The following is an example of a multiple sequence alignment with $k = 5$ for the sequences described in the earlier examples.

```
1     FKQCCWNSLP____RGLSNVALVYQEFMAKCRGESENLQLVTALVINLPSMA
2   SMFRQCIWNSLS____HGLPETAPIYQPLKARCRGVSENLQLVTEIIINLPTLC
3   SLWCQCIKASLPLKVIRGTPEVAPLYDQLEQVCRSENQ____VSEIVAKFASLC
4   TMFKMCLWNALP____RGLPEVAPVYRPLKARCRGDSENLQLCAERLVNLPELC
5   AILRSCIWNLLP____RGLPEAAPIYEPLKARLRGESENYKLVTEIIMTLPSLC
```

Probabilistic ancestral sequence. A probabilistic ancestral sequence (PAS) is like a protein sequence where each position, instead of an amino acid, has a vector of dimension $|\Sigma|$ describing the probability of each amino acid being in that position. A PAS is normally associated to an EPT and to a particular place in this tree. It is normally most interesting to compute the PAS at divergence points or at the root of the EPT. E.g. find the PAS for the common ancestor of all mammals.

2 Probability of an evolutionary configuration

In this section we will describe how to compute the probability of an evolutionary configuration (EC), that is the probability of a particular alignment of sequences according to a given EPT.

First we should notice that our model of evolution treats each position independently of its neighbours. Hence the probability of an EC will be the product of the probabilities for each of the aligned positions. For each aligned position we have an EPT with one amino acid at each leaf. (We consider the special case of indels later). The tree in Figure 2 represents a slice of the EC at position 4 where sequence 1 has a `K`, sequence 2 has an `R`, etc. In this case, the amino acids placed at nodes W, X and Y are the unknown amino acids at the divergence points.

How to compute this probability efficiently. Let O be the point which identifies the root of the EPT. To compute the probability of a given position, we use the same idea as Dayhoff et al. [6] i.e. sum over all possible unknown

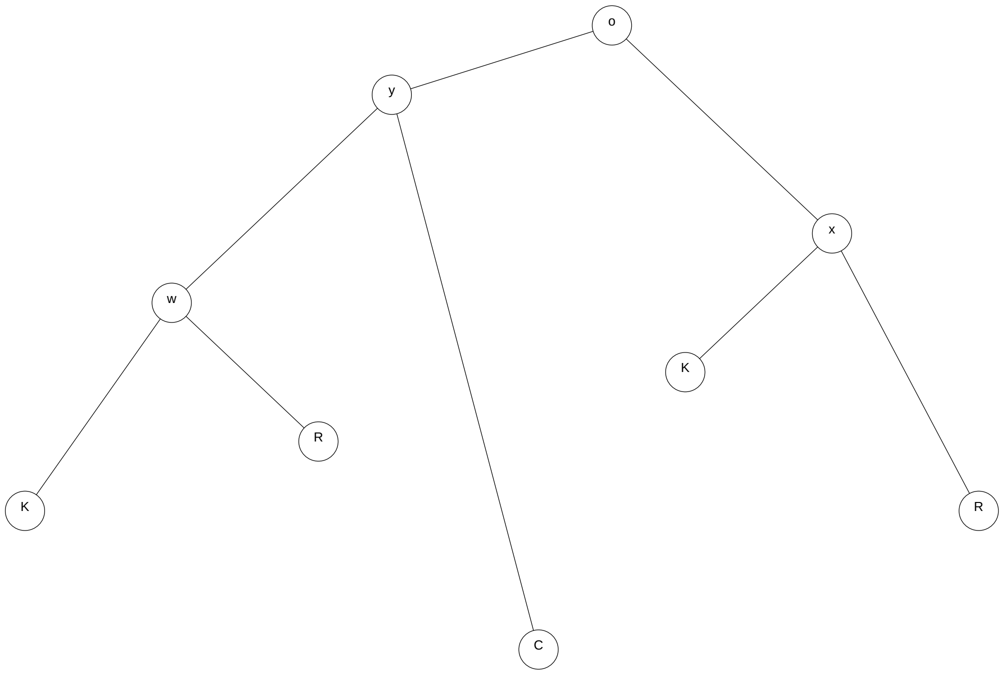

Figure 2: EPT slice at position 4 of the aligned sequences

amino acids o, w, x and y and evaluate all the mutation probabilities.

$$Pr\{\mathrm{EC}\} = \sum_{o} f_o \sum_{x} M_{xo}^{d_X} M_{Kx}^{d_4} M_{Rx}^{d_5} \sum_{y} M_{yo}^{d_Y} M_{Cy}^{d_3} \sum_{w} M_{wy}^{d_W} M_{Kw}^{d_1} M_{Rw}^{d_2}$$

The interpretation of each part of the product is simple. For each branch between $U \rightarrow V$ we will have a term of the form $M_{vu}^{d_V}$, for each unknown internal amino acid at node U we will have a sum for all its possibilities ($\sum_u$) and for the root we have the sum of each possible amino acid times its natural frequency of occurrence. Lower case letters denote the particular amino acid at a given position, e.g. x is the amino acid in a given position of sequence X. The upper case K, R and C indicate the known amino acids at the leaves.

As written, this formula is very expensive to compute, it requires $|\Sigma|^{k-1}$ products, where k is the number of sequences. We can reorganize this summation by the introduction of new vectors T^X and S^X for each internal or external node X. The value of the vector depends on the position of the root of the tree and it is computed as follows

- if X is an external leaf, i.e. a known amino acid, x

$$T_i^X = \begin{cases} 1 & i = x \\ 0 & i \neq x \end{cases}$$

$$S^X = M^{d_X} \cdot T^X = M_{*x}^{d_X}$$

  (i.e. S^X is just the x^{th} column of M^{d_X}).

- if X is an internal node, then it is a ternary node. Let Y and Z be the adjacent nodes whose subtrees do not include the root of the tree (away from the root). Then

$$T_i^X = \left(\sum_y M_{yi}^{d_Y} T_y^Y \right) \left(\sum_z M_{zi}^{d_Z} T_z^Z \right) = S_i^Y \cdot S_i^Z$$

$$S_i^X = \sum_x M_{xi}^{d_X} T_x^X$$

$$S^X = \left(M^{d_X}\right)' T^X$$

These recursive definitions can be used if we do the computation from the leaves towards the root. Each S corresponding to an internal node requires $|\Sigma|^2$ multiplications, and each T requires $|\Sigma|$ multiplications. For k sequences, each of length m, this requires $m(k-1)|\Sigma|(|\Sigma|+1)$ multiplications ($420m(k-1)$ for proteins), which is not inexpensive, but perfectly feasible.

The computation of the probability of the EC is done with the two immediate descendants of the root, call them X and Y.

$$Pr\{\mathrm{EC}\} = \sum_i f_i \left(\sum_x M_{xi}^{d_X} T_x^X \right) \left(\sum_y M_{yi}^{d_Y} T_y^Y \right) = \sum_i f_i T_i^O = f \cdot T^O$$

The PAS at the root of the tree can be computed as follows: for each amino acid at the root compute the probability of such a configuration

$$Pr\{\text{aa } i \text{ at the root}\} = \mathrm{PAS}_i = \left(\sum_x M_{xi}^{d_X} T_x^X \right) \left(\sum_y M_{yi}^{d_Y} T_y^Y \right) = T_i^O$$

where X and Y are the descendants of the root. These are probabilities of the entire EC, to find the relative probabilities for each amino acid the PAS is normalized to sum 1, i.e.

$$\text{PAS} = T^O / \sum_i T_i^O$$

Independence of the position of the root. It can be shown that the probability of an EC is independent of where we place the root of the tree. This is a direct consequence of the symmetry of the evolution model mentioned in 1.1. To prove this, we first show that the probabilities are not affected if we move the root along the edge liking its two descendants (X and Y). The length of the branch from X to Y will remain constant, we will just change d_X and d_Y so that $d_X + d_Y = d_{XY} =$ constant. The probability of the EC is

$$\sum_i f_i \left(\sum_x M_{xi}^{d_X} T_x^X \right) \left(\sum_y M_{yi}^{d_Y} T_y^Y \right)$$

Moving the summations around and using the symmetry relation of f and M and noticing that the summation in i gives a matrix product, we obtain

$$\sum_y \sum_x T_y^Y T_x^X \sum_i f_i M_{yi}^{d_Y} M_{xi}^{d_X} = \sum_y \sum_x T_y^Y T_x^X f_y (M^{d_X + d_Y})_{yx}$$

and the formula is independent of d_X and d_Y, only depends on their sum. By continuity, this can be extended to any position within the EPT. Consequently for the purposes of evaluating the probability of a particular EC, we can place the root wherever it is most convenient.

3 Probabilistic Dynamic Programming and Multiple Alignments

The standard DP algorithm for aligning two sequences [15, 18, 10], requires that we estimate a cost of matching any two individual positions. This is done by the method described in the introduction or by completely empirical *cost/similarity* functions which assign some value to the alignment of each pair of amino acids. The DP algorithm for aligning sequences can be viewed as an algorithm which takes as parameter such a cost/similarity function,

$$C \; : \; \{\Sigma \cup \phi\} \times \{\Sigma \cup \phi\} \rightarrow \Re$$

and produces an alignment which maximizes the sum of the C function applied to all aligned pairs[1].

[1] This algorithm has been well described in the literature. Finding the cost of the best alignment, and producing the aligned sequences can be done with $m_A m_B$ evaluations of C and $O(\min(m_A, m_B))$ storage for sequences A and B with lengths m_A and m_B.

We will use DP over partially built multiple alignments and the C function will be derived from the probability of an EC. We call this aligning algorithm *probabilistic dynamic programming.*

More precisely, our DP algorithm works over two partial multiple alignments. A partial multiple alignment is a multiple alignment of all the sequences of a subtree of the EPT. Hence a partial multiple alignment is identified by an internal node and a direction in the EPT. E.g. node Y has as descendants the sequence 3 and the node W; node W has sequences 1 and 2 as descendants. The partial multiple alignment for node Y is:

```
1     FKQCCWNSLP____RGLSNVALVYQEFMAKCRGESENLQLVTALVINLPSMA
2    SMFRQCIWNSLS____HGLPETAPIYQPLKARCRGVSENLQLVTEIIINLPTLC
3    SLWCQCIKASLPLKVIRGTPEVAPLYDQLEQVCRSENQ____VSEIVAKFASLC
```

For each subtree X and its partial multiple alignment we also compute the T^X and S^X vectors. The C function for this algorithm will be derived from the probability of alignment of two slices of a multiple alignment under the same ancestor. In essence, this is exactly the same idea as in Dayhoff [6], extended so that we can compute the probabilities of alignments not just with two descendant sequences, but with two subtrees of descendants.

The C function is the logarithm of the quotient of probabilities, or

$$C(Y,Z) = 10\log_{10}\left(\frac{Pr\{Y \text{ and } Z \text{ are descendants of } x\}}{Pr\{Y\}Pr\{Z\}}\right)$$

(The factor 10 is retained for purely historical reasons, it was used first by Dayhoff et al. Any constant factor can be used without any material effect.) Using the T and S vectors this becomes

$$C(Y,Z) = 10\log_{10}\left(\frac{\sum_x f_x\left(\sum_y M_{yx}^{d_Y}T_y^Y\right)\left(\sum_z M_{zx}^{d_Z}T_z^Z\right)}{(f\cdot T^Y)(f\cdot T^Z)}\right)$$

$$= 10\log_{10}\frac{\sum_x f_x S_x^Y S_x^Z}{(f\cdot T^Y)(f\cdot T^Z)}$$

At this point, we are using DP to align the two subtrees rooted at Y and Z. Hence we still do not have a correspondence between each position of Y and each position of Z. That is why we have to use the sum $\sum_x f_x S_x^Y S_x^Z$ in C. Once that we have found the alignment of Y and Z, then we can compute T^X and S^X for the node we have just joined, the parent of Y and Z.

The cost of each computation of C is $O(|\Sigma|)$ multiplications. The cost of one probabilistic DP alignment is $O(m^2|\Sigma|)$ and the cost of the entire process (including the computation of S and T) is $O(mk|\Sigma|(|\Sigma|+m))$.

The alignment at the root of the EPT is the multiple alignment of all the sequences. Notice that for two sequences, this algorithm coincides exactly with DP using Dayhoff matrices, so this presents a continuous generalization of the 2-sequence algorithm.

Normalization. For any node Y, multiplying S^Y and T^Y by an arbitrary constant k_1 has no effect on the dynamic programming algorithm. (Both S^Y and T^Y have to be multiplied by the same constant so that the equation $S^Y = \left(M^{d_Y}\right)' T^Y$ is preserved). This can be readily verified from the definition of the cost function for DP, $C(Y, X)$ above. Similarly, the PAS are not altered by such multiplication, as the last step normalizes the probabilities to add to 1. It is then advantageous to normalize the T and S vectors to prevent underflows which are very likely to occur for large multiple alignments. This independence from a constant factor also comes handy for indels.

Indels. Insertions/deletions are handled in the same way they would be handled by the standard DP algorithms [15, 10]. The cost of a deletion is known to depend on the PAM distance of the aligned sequences [3] and there exist approximations to compute them based on the distance alone. In case DP chooses an indel, one of the subtrees is viewed as having had a deletion and this deletion has to be propagated to the entire aligned subtree. If X is a node with descendants Y and Z, and DP chooses Y to be an insertion, i.e. Y is not aligned to anything, then T^X and S^X are computed as

$$T^X = S^Y Pr\{\text{deletion of } Z\}$$

and

$$S^X = \left(M^{d_X}\right)' T^X$$

as usual. This corresponds to computing the EC when all the Z subtree is missing. Since multiplying T^X and S^X by a constant has no effect (as discussed above), we can simply ignore $Pr\{\text{deletion of } Z\}$ and use $T^X = S^Y$.

4 Conclusions

This algorithm computes maximum likelihood multiple alignments given the sequences and their EPT. A maximum likelihood alignment under a Markovian model of evolution is a very desirable type of alignment, as it has well understood properties. This algorithm was implemented in Darwin between 1990 and 1991. It went through several (some major) revisions until about 1993 and it has been in use by the community since then. This is one of the rare cases where the algorithm had a successful service life *before* being published. It can be used through the Web (`http://cbrg.inf.ethz.ch/`) or by e-mail to (`cbrg.inf.ethz.ch`) from an automatic Darwin server. The server handles about 800 requests per month, of which about 34% involve multiple alignments. So far the feedback from users has been very positive.

A more formal evaluation of this algorithm is required. This has been done with a technique where we generate random EPTs and randomly mutate the sequences. From the resulting mutated sequences at the leaves of the tree we try to reconstruct the multiple alignment. It is easy to verify the accuracy, as we know the correct alignment by construction. The comparisons were done for this and several other algorithms available to us. These results and the description of the methodology exceed the scope of the paper and will be available in [9].

One problem affects the quality of these multiple alignments. The results are dependent on a correct EPT, which is not always available. Minor errors in the EPT are of little consequence, unless they straddle incorrectly a point where an indel occurred. This causes the same indel to be in separate subtrees of the EPT. The above algorithm will handle this by repeating the indel. As a result the two indels may not match. Heuristics to try to fix this problem include weighting the indels in a special way in the subtrees, but have not been entirely successful. It should be noted that if the EPT is correct then the algorithm described here will work correctly.

References

[1] L. Allison and C.S. Wallace. The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimization of multiple alignments. *J. Molecular Evolution*, 39:418–430, 1994.

[2] Lachlan H. Bell, John R. Coggins, and James E. Milner-White. Mix'n'match: an improved multiple sequence alignment procedure for distantly related proteins using secondary structure predictions, designed to be independent of the choice of gap penalty and scoring matrix. *Protein Engineering*, 6(7):683–690, 1993.

[3] Steven A. Benner, Mark A. Cohen, and Gaston H. Gonnet. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Molecular Biology*, 229:1065–1082, 1993.

[4] Steven A. Benner, Mark A. Cohen, and Gaston H. Gonnet. Amino acid substitution during functionally constrained divergent evolution of protein sequences. *Protein Engineering*, 7(11), 1994.

[5] Humberto Carillo and David. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48(5):1073–1082, 1988.

[6] Margaret O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model for evolutionary change in proteins. In Margaret O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352. National Biochemical Research Foundation, Washington DC, 1978.

[7] Adam Godzik and Jeffrey Skolnick. Flexible algorithm for direct multiple alignment of protein structures and sequences. *CABIOS*, 10(6):587–596, 1994.

[8] Gaston H. Gonnet, Mark A. Cohen, and Steven A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[9] Gaston H. Gonnet and Chantal Korostensky. Evaluation measures of multiple sequence alignments. In preparation, 1996.

[10] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[11] Sandeep K. Gupta, John Kececioglu, and Alejandro A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J. Computational Biology*, 1996. To appear.

[12] Xiaoqiu Huang. On global sequence alignment. *CABIOS*, 10(3):227–235, 1994.

[13] Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: A gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, October 1993.

[14] David J. Lipman, Stephen F. Altschul, and John D. Kececioglu. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA*, 86:4412–4415, June 1989.

[15] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[16] Gregory D. Schuler, Stephen F. Altschul, and David J. Lipman. A workbench for multiple alignment construction and analysis. *PROTEINS: Structure, Function, and Genetics*, 9:180–190, 1991.

[17] Peter H. Sellers. On the theory and computation of evolutionary distances. *SIAM J Appl. Math.*, 26(4):787–793, Jun 1974.

[18] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[19] J.D. Thompson, D.G. Higgins, and T.J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.