BIOINFORMATICS

Vol. 19 no. 12 2003, pages 1505–1513
DOI: 10.1093/bioinformatics/btg193

A hidden Markov model for progressive multiple alignment

*Ari Löytynoja and Michel C. Milinkovitch**

Unit of Evolutionary Genetics, Free University of Brussels (ULB), cp 300, Institute of Molecular Biology and Medicine, rue Jeener & Brachet 12, B-6041 Gosselies, Belgium

Received on November 15, 2002; revised on February 11, 2003; accepted on February 21, 2003

ABSTRACT

Motivation: Progressive algorithms are widely used heuristics for the production of alignments among multiple nucleic-acid or protein sequences. Probabilistic approaches providing measures of global and/or local reliability of individual solutions would constitute valuable developments.

Results: We present here a new method for multiple sequence alignment that combines an HMM approach, a progressive alignment algorithm, and a probabilistic evolution model describing the character substitution process. Our method works by iterating pairwise alignments according to a guide tree and defining each ancestral sequence from the pairwise alignment of its child nodes, thus, progressively constructing a multiple alignment. Our method allows for the computation of each column minimum posterior probability and we show that this value correlates with the correctness of the result, hence, providing an efficient mean by which unreliably aligned columns can be filtered out from a multiple alignment.

Availability: The software is freely available at http://www.ulb.ac.be/sciences/ueg/

Contact: aloytyno@ulb.ac.be; mcmilink@ulb.ac.be

INTRODUCTION

Exact inference of optimal multiple sequence alignment is computationally impractical for more than a few short sequences. Heuristics are therefore used, and many of the available programs generating alignments among multiple nucleic-acid or protein sequences implement the progressive alignment method (Feng and Doolittle, 1987) that consists into iterating the classical dynamic programming algorithm (Needleman and Wunsch, 1970; Gotoh, 1982). Although the latter guarantees to find the minimum scoring alignment between two sequences, its iterative application does not guarantee the production of a globally-optimal multiple alignment, nor provides any measure of reliability across pairwise alignments. Methods to find sub-optimal pairwise alignments exist (Vingron, 1996), but are not widely implemented. Also, as the best-scoring alignment does not necessarily correspond to the true alignment, it may be more important to infer the reliability of the result than to optimize the absolute global score by all means.

The hidden Markov methods (HMM) can be viewed as stochastic regular grammars that generate sequences from a given alphabet. A HMM consists of states that emit characters: transition probabilities define the moves among the HMM states and the emission probabilities describe the character distribution in a given state. The path through the states is Markovian (i.e. it only depends on the previous state) but unknown from the outside (therefore 'hidden'). If let to run free, a HMM produces data according to the probabilistic model. Reversely, alternative histories that could have generated a given observed data set can be inferred and compared according to their respective probabilities under the HMM. Given a HMM, the Viterbi algorithm (Viterbi, 1967) allows to find the most probable state path given the data, and the combination of algorithms Forward and Backward (Rabiner, 1989) defines the joint posterior probability of all paths going through a given state.

HMMs were applied in speech recognition already in early 1970s (see Rabiner (1989) for historical, and general introduction to the topic), whereas they were introduced by Churchill (1989) in computational biology, and have since been applied in many other fields. Programs like HMMER (Eddy, 1995) and SAM (Karplus *et al.*, 1998) implement profile HMMs, and are widely used to perform multiple sequence alignments. They first generate a probabilistic description of a sequence family, and then either add more sequences to this profile, or match the profile against a query sequence. Although the profile HMMs describe well the functional constraints of the sequence family and help to find more distant homologs in databank searches (Karplus *et al.*, 1998), they completely ignore the evolutionary process that has generated the data.

*To whom correspondence should be addressed.

Indeed, in the vast majority of cases, the nucleic acid or protein sequences included in an alignment are evolutionary homologs, hence, related by a phylogenetic tree. We assumed that the combination of progressive alignment algorithms with probabilistic models of nucleotide or amino-acid substitution should improve the accuracy of multiple alignments and our understanding of the history and function of the sequences. Durbin *et al.* (1998) developed HMM methods for probabilistic pairwise-sequence alignments that include algorithms for the identification of the most probable state path, and computation of the forward and backward full probability of an affine-gap-score alignment of two sequences. Here we report on improvements of Durbin *et al.*'s approach and its generalization for computationally efficient production of probabilistic alignments among multiple sequences. We expand the probabilistic framework to the guide tree and describe sequence sites with vectors of character probabilities. Similarly to maximum likelihood (ML) evaluation of phylogenetic trees, we define the alignment match score as the probability of evolution from a parent node to its child nodes. Our model can be viewed as a probabilistic alternative to traditional progressive alignment methods. As in ClustalW (Thompson *et al.*, 1994), we perform pairwise alignments according to a neighbor-joining (NJ) (Saitou and Nei, 1987) guide tree.

An alternative approach to perform an alignment within a statistical framework was described by Thorne *et al.* (1991, 1992), and has been recently developed further (Hein *et al.*, 2000; Holmes and Bruno, 2001).

We show that our probabilistic approach performs better than ClustalW on the alignment of simulated nucleotide data sets, and nearly equally well on the alignment of reference amino-acid sequences with residue identity >35%. The probabilistic sampling of locally sub-optimal solutions can further improve the correctness of the global alignment, although identification of the most correct alignment may be difficult because the global alignment correctness badly correlates with any of the alignment global probability scores. Most importantly, incorrectly aligned sites have, in average, lower posterior probabilities than correctly-aligned sites such that unreliably aligned positions can be identified and filtered out before additional (e.g. phylogenetic) analyses are undertaken.

SYSTEM AND METHODS

As in the classical progressive alignment method (Feng and Doolittle, 1987) we build the multiple alignment by performing pairwise alignments at the nodes of a guide tree in the order of decreasing similarity. The guide tree is produced with NJ-clustering (Saitou and Nei, 1987) after producing a distance matrix computed from the similarity scores of all pairwise alignments.

At each internal node of the guide tree a probabilistic alignment is performed between the two sequences corresponding to the two child nodes. Pointers from the parent to the child sites are stored. For each site of the ancestral sequence (from which the two child sequences have evolved), a vector of probabilities of the alternative character states ('A/C/G/T/-', or each of the 20 possible amino acids and '-', for nucleic acid or protein sequences, respectively) is computed. The newly built internal node is then aligned with another internal or tip sequence, and the procedure progresses through the guide tree. Once the root node is defined, the multiple alignment is build by recursively tracing back the emission of gap characters at the nodes below.

Substitution model

For a pairwise alignment we will consider two sequences, x and y, and their unknown parent, z, made of sites $x_{1\ldots n}$, $y_{1\ldots m}$ and $z_{1\ldots l}$. To allow for ambiguous characters, a sequence site is described as a vector of probabilities, $p_a(x_i)$, that the site x_i contains character a. At terminal (extant) nodes, the observed character is given a probability of 1 and alternative character-state probabilities are set to 0; if the observed character is ambiguous, the probability is shared among different characters. At internal (ancestral) sequences, different character states have different probabilities (summing to 1). The probability of each alternative state is computed as the sum of the probabilities that it yielded the child nodes, given the branch lengths.

Character state a at the parent node has the background probability q_a. The probability that this state has evolved to b at one of its two child nodes is s_{ab}, i.e. character states a and b are observed at the parent and its child site, respectively. If characters a and b are different, a substitution has occurred, otherwise the original character has not changed. A 'gap' is considered as an extra character with a non-zero probability of substituting any other character. Thus, internal nucleotide sequences consist of characters {A,C,G,T,-} whereas '-' always have a probability of zero in terminal sequences.

In other words, the probability p_{x_i,y_j}, that sites x_i and y_j are aligned, corresponds to $p_{z_k}(x_i, y_j)$, i.e. the probability that the site z_k at the parent node evolved to the sites x_i and y_j at the child nodes. As the ancestral character state z_k is unknown, the probability is summed over all possible characters a:

$$p_{x_i,y_j} = p_{z_k}(x_i, y_j) = \sum_a p_{z_k=a}(x_i, y_j). \qquad (1)$$

The probability of the event $z_k = a$ depends on $p_b(x_i)$ and $p_b(y_j)$, i.e. the probabilities of all possible characters b at the sites x_i and y_j, as well as on s_{ab}, the probability

of observing the character pair

$$p_{z_k=a}(x_i, y_j) = q_a \sum_b s_{ab}\, p_b(x_i) \sum_b s_{ab}\, p_b(y_j) \qquad (2)$$

where q_a is the character background probability, and s_{ab} is computed with the method of Jukes and Cantor (1969):

$$s_{ab}(v) = \frac{1}{n} + \frac{n-1}{n} e^{-\frac{n}{n-1}v} \text{ if } a = b, \text{ or} \qquad (3)$$

$$s_{ab}(v) = \frac{1}{n} - \frac{1}{n} e^{-\frac{n}{n-1}v} \text{ if } a \neq b$$

where n is the size of the alphabet ($n = 5$ for nucleotide sequences because 'gap' is considered a fifth character), v is the NJ-estimated branch length between the parent and the child. Although we adopted only the JC model, more complex models (incorporating parameters such as the transition versus transversion ratio, observed character state frequencies, and rate heterogeneity) could be implemented into our multiple alignment procedure.

An alignment gap is not different from a match except that the character state at one of the child sites is known to be '-' for sure, and the probability is summed over all possible characters a at the parent site z_k

$$p_{x_i,-} = p_{z_k}(x_i, -) = \sum_a p_{z_k=a}(x_i, -) \qquad (4)$$

where

$$p_{z_k=a}(x_i, -) = q_a \sum_b s_{ab}\, p_b(x_i)\, s_{a-} \qquad (5)$$

and similarly for p_{-,y_j}. As a '-' is an extra character, s_{a-}, the probability to observe a character and a gap, is computed as other s_{ab}.

The implementation of a more complex model of protein evolution than the JC model described above requires the modification of amino-acid substitution tables to incorporate gap characters. Let us assume that we have a 1 PAM substitution matrix S, and frequencies for the 20 amino-acids, q_a, and we want to convert them to 1 PAM 21×21 matrix S' and frequencies q'_a. If q'_- and s'_{i-} are the frequency of 'gap' and the probability of substituting a 'non-gap' character with a 'gap', the new frequencies for other characters and the non-diagonal substitution probabilities can be scaled to $q'_a = (1 - q'_-)q_a$ and $s'_{ij} = (1 - s'_{i-})s_{ij}$, respectively. To fulfill the conditions of an 1 PAM (i.e. one accepted mutation per 100 characters), the non-diagonal entries are scaled such that $\sum_i q'_i \sum_{i,j} s'_{ij} = 0.01\ (i \neq j)$, and, finally, the diagonal entries are defined to sum each row to one, $s'_{ii} = 1 - \sum_j s'_{ij}\ (i \neq j)$. Thus, given the 1 PAM-matrix S', the substitution probabilities $s_{ab}(v)$ are given by matrix S'^{100v}. As the position of root in the alignment guide tree can be arbitrary, the substitution probability table should be based on a symmetric matrix, such as WAG (Whelan and Goldman, 2001).

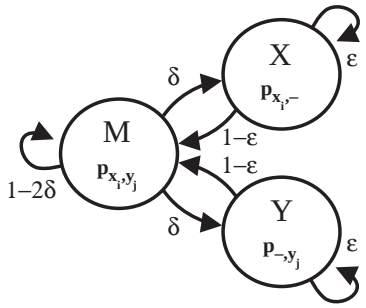

Fig. 1. A schematic presentation of the HMM that generates a pair of aligned sequences.

Hidden Markov model

The HMM that produces a pair of aligned sequences has three states (match, M; x-insert, X; and y-insert, Y) and two parameters for the among-states transition probabilities (Durbin *et al.*, 1998). The model is symmetric, and δ is the probability to move from M to an insert (X or Y), ε to stay at the insert, $1-\varepsilon$ to move back to M and $1-2\delta$ to stay at M (Fig. 1). At each state, the model emits characters according to emission probabilities: at M an aligned character pair is emitted with the probability p_{x_i,y_j}, whereas at X or Y a character against a gap sign or a gap sign against a character is emitted with probabilities $p_{x_i,-}$ or p_{-,y_j}, respectively.

Pairwise alignment

In the dynamic programming algorithm, the two sequences to be aligned define a matrix, and the algorithm proceeds recursively through all the cells by always choosing the best path from one of the previous cells. Once the recursion is completed, a trace-back algorithm is used to find the best path through the matrix.

In a probabilistic framework, the best alignment corresponds to the Viterbi path through the HMM. As the model has three states and the search space is two-dimensional, the alignment recursion requires three two-dimensional matrices: v^M for matches, and v^X and v^Y for gap states. Any move within the M or X or Y matrix corresponds to remaining in that state and produces an additional match, or the extension of an X-gap, or the extension of a Y-gap, respectively. On the other hand, a move between two matrices (M and X or M and Y) changes the state and either opens or closes a gap. The HMM state transition probabilities are static and pre-defined, but the character state emission probabilities at the guide tree nodes are computed dynamically according to the evolutionary substitution models described above.

The Viterbi algorithm for a sequence pair can be

described as follows:

Initialization:
$v^{\bullet}(0,0)=1$; $v^{\bullet}(i,-1)$, $v^{\bullet}(-1,j)$ are set to 0.

Recursion:
$i=0,\ldots,n$, $j=0,\ldots,m$, except (0,0);

$$v^{M}(i,j)=p_{x_i,y_j}\max\begin{cases}(1-2\delta)v^{M}(i-1,j-1),\\(1-\varepsilon)v^{X}(i-1,j-1),\\(1-\varepsilon)v^{Y}(i-1,j-1);\end{cases}$$

$$v^{X}(i,j)=p_{x_i,-}\max\begin{cases}\delta v^{M}(i-1,j),\\\varepsilon v^{X}(i-1,j);\end{cases}$$

$$v^{Y}(i,j)=p_{-,y_j}\max\begin{cases}\delta v^{M}(i,j-1),\\\varepsilon v^{Y}(i,j-1);\end{cases}$$

Termination:
$v^{E}=\max(v^{M}(n,m),v^{X}(n,m),v^{Y}(n,m))$.

During the recursion the relative probabilities of different paths entering the cell are stored. Thus, for $v^{M}(i,j)$ and $v^{X}(i,j)$ they are

$$p_{M\to M}(i,j)=\frac{(1-2\delta)v^{M}(i-1,j-1)}{p_{\bullet\to M}(i,j)}$$
$$p_{X\to M}(i,j)=\frac{(1-\varepsilon)v^{X}(i-1,j-1)}{p_{\bullet\to M}(i,j)}$$
$$p_{Y\to M}(i,j)=\frac{(1-\varepsilon)v^{Y}(i-1,j-1)}{p_{\bullet\to M}(i,j)}$$

and

$$p_{M\to X}(i,j)=\frac{\delta v^{M}(i-1,j)}{\delta v^{M}(i-1,j)+\varepsilon v^{X}(i-1,j)}$$
$$p_{X\to X}(i,j)=\frac{\varepsilon v^{X}(i-1,j)}{\delta v^{M}(i-1,j)+\varepsilon v^{X}(i-1,j)}$$

where

$$p_{\bullet\to M}(i,j)=(1-2\delta)v^{M}(i-1,j-1)+(1-\varepsilon)\times(v^{X}(i-1,j-1)+v^{Y}(i-1,j-1))$$

and similarly for $v^{Y}(i,j)$. A trace-back algorithm is then used either to select the best path or to sample different paths according to their posterior probabilities: choosing a 'match' step ($\to M$) creates pointers from the parent site to the two child sites, whereas taking a 'gap' step ($\to X$ or $\to Y$) creates a pointer to one child site and a gap as the second child site.

Once the alignment path is resolved, the site at the parent node is defined as the vector of probabilities corresponding to each possible character state (including 'gap') assignment. For a match that is

$$p_a(z_k)=\frac{p_{z_k=a}(x_i,y_j)}{\sum_b p_{z_k=b}(x_i,y_j)} \quad (6)$$

and similarly for the gaps. The ancestral sequence can then be aligned with another sequence.

Posterior probability

As the Viterbi algorithm always chooses the best solution, it finds a single path that maximizes the probability. On the other hand, by summing the probabilities of all the paths entering a given cell, one obtains the full probability of two characters being aligned independent of any specific path.

The Forward recursion:

Initialization:
$f^{\bullet}(0,0)=1$; $f^{\bullet}(i,-1)$, $f^{\bullet}(-1,j)$ are set to 0.

Recursion:
$i=0,\ldots,n$, $j=0,\ldots,m$, except $(0,0)$;

$$f^{M}(i,j)=p_{x_i,y_j}\Big[(1-2\delta)f^{M}(i-1,j-1)+(1-\varepsilon)(f^{X}(i-1,j-1)+f^{Y}(i-1,j-1))\Big];$$

$$f^{X}(i,j)=p_{x_i,-}\Big[\delta f^{M}(i-1,j)+\varepsilon f^{X}(i-1,j)\Big];$$

$$f^{Y}(i,j)=p_{-,y_j}\Big[\delta f^{M}(i,j-1)+\varepsilon f^{Y}(i,j-1)\Big];$$

Termination:
$f^{E}=f^{M}(n,m)+f^{X}(n,m)+f^{Y}(n,m)$.

The Backward recursion:

Initialization:
$b^{\bullet}(n,m)=1$; $b^{\bullet}(i,m+1)$, $b^{\bullet}(n+1,j)$ are set to 0.

Recursion:
$i=n,\ldots,1$, $j=m,\ldots,1$, except(n,m);

$$b^{M}(i,j)=(1-2\delta)p_{x_{i+1},y_{j+1}}b^{M}(i+1,j+1)+\delta\Big[p_{x_{i+1},-}b^{X}(i+1,j)+p_{-,y_{j+1}}b^{Y}(i,j+1)\Big]$$

$$b^{X}(i,j)=(1-\varepsilon)p_{x_{i+1},y_{j+1}}b^{M}(i+1,j+1)+\varepsilon p_{x_{i+1},-}b^{X}(i+1,j);$$

$$b^{Y}(i,j)=(1-\varepsilon)p_{x_{i+1},y_{j+1}}b^{M}(i+1,j+1)+\varepsilon p_{-,y_{j+1}}b^{Y}(i,j+1);$$

$f^{\bullet}(i, j)$ and $b^{\bullet}(i, j)$ sum the probabilities of all the possible alignments between sub-sequences $x_{1...i}$ and $y_{1...j}$, and $x_{i...n}$ and $y_{j...m}$, respectively. $f^{\bullet}(i, j)b^{\bullet}(i, j)$ is the full joint probability of all alignments passing by (i, j), i.e. aligning $x_i : y_j$. Dividing by f^E gives the proportion of the full probability corresponding to all alignments passing through the cell (i, j). Thus, if x_i and y_j are matching characters, the posterior probability that x_i and y_j are aligned is given by

$$P(x_i \diamond y_j | x, y) = \frac{f^M(i, j)b^M(i, j)}{f^E} \qquad (7)$$

and similarly for the insert-states *X* and *Y*. We assume that the posterior probability of the sites on the alignment path is a valid estimator of the local reliability of the alignment.

Multiple alignment

The pairwise alignment algorithm works progressively, from the tip-nodes towards an arbitrary (central) root of the tree, and defines each internal node by the alignment of its two child nodes. Once the root node is defined, the multiple alignment can be build by tracing back the characters at the nodes below. As each pairwise alignment stores, for each character, pointers to the two child characters, a recursive call from the root resolves characters at terminal sequences. If a gap is created in one of the internal nodes, a gap character state is introduced in all the sequences of that sub-tree, and the recursive call does not proceed further in that branch.

In a similar manner, the recursive call returns the posterior probability for an aligned column of sites at each internal node.

At a given column, most sequences may be well aligned but few problematic sequences either may contain gaps or have so much diverged that the full alignment is ambiguous. While the average posterior probability of such a site can be high, the minimum value at the column probably best reflects the ambiguity of the full alignment at that site. Hence, we use the minimum posterior probability value of a column as a measure of reliability of the full multiple alignment being correct.

Testing the new algorithms

We tested the performance of our method in the alignment of (i) simulated nucleotide data sets and (ii) amino-acid data sets from a reference database. Nucleotide sequences (20 sequences, ~500 characters) were generated with the program Rose (Stoye *et al.*, 1998) (JC model, mean substitution rate = 0.013, and insert/delete probability = 0.03) as follows: a root random sequence (of length 500) was evolved on a random tree to yield sequences of 'low' or 'high' mean divergences, i.e. with an average number of substitutions per site of 0.5 or 1.0, respectively. Furthermore, the insertion/deletion length distribution was set to 'short' [frequencies of gaps of length 1–3 = 0.8, 0.1, 0.1] or 'long' [frequencies of gaps of length 1–7 = 0.3, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1]. Hence the four 'mean divergence—mean gap length' conditions tested here are 'low-short' 'low-long', 'high-short', and 'high-long'. 50 random data sets were generated under each of the four combinations of settings, and the average performance is reported.

As our algorithm, in its current form, models global alignments of sequences, we chose to test its performances in aligning the amino-acid sequences from the 'Reference 1' (Ref1) of the BAliBASE database (Thompson *et al.*, 1999). Ref1 contains alignments of less than six equi-distant sequences, i.e. the percent-identity between two sequences is within a specified range, such that all the sequences are of similar length, with no large insertion or extension. The alignments are divided into three subgroups according to their lengths, and were further divided into three classes according to their similarities: 'Test 1' consists of 27 alignments that are in average 97 amino-acid long, and from which 10, 10 and 7 alignments share identity of >35%, 40–20% and <25%, respectively; 'Test 2' consists of 27 alignments that are in average 270 amino-acid long, and from which 10, 9 and 8 alignments share identity of >35, 40–20 and <25%, respectively; 'Test 3' consists of 28 alignments that are in average 559 amino-acid long, and from which 8, 12 and 8 alignments share identity of >35, 40–20, and <25%, respectively.

The performance of our method was compared to that of ClustalW (version 1.81, default parameters). The same ClustalW-generated guide tree was used for both alignments but branch lengths were multiplied by 0.5 before the probabilistic alignment. The nucleotide sequences were aligned using equal character background frequencies ($q_a = 0.2$) and JC-model, whereas amino-acid alignments were performed using the 'WAG' substitution probability matrix with 'gap' background frequency and substitution probability of 0.1 and 0.001, respectively. The HMM parameters δ and ε were estimated from pairwise alignments of terminal sequence pairs such that $\delta = 1/2(l_m + 1)$, and $\varepsilon = (1 - 1/(l_g + 1))$, where l_m and l_g are the mean lengths of match- and gap-segments. To avoid null-estimates, one pseudo-count of 5 is added in both calculations. All pairwise alignments were computed with the affine-gap-score algorithm using (i) for nucleic acids, the 'swgap' substitution score table (from ClustalW 1.81) with gap-opening and gap-extension penalties of 15 and 7, respectively, and (ii) for amino-acid data, the PAM-120 table with gap-opening and gap-extension penalties of 10 and 0.1, respectively.

For each test data set, the probabilistic multiple alignment was iterated (i) 10 times by choosing the best trace-back path but breaking ties randomly, and (ii) 30 times by sampling the path from the posterior probability distribution. Each alignment was compared against the

Table 1. Nucleotide sequences were simulated under four conditions

	Nucleotide				Test 1			Test 2			Test 3		
Cond./%id.	A	B	C	D	>35	40–20	<25	>35	40–20	<25	>35	40–20	<25
ClustalW	0.933	0.893	0.633	0.529	0.818	0.608	0.344	0.840	0.662	0.198	0.811	0.669	0.247
ProAlign	0.941	0.909	0.696	0.604	0.812	0.466	0.099	0.801	0.638	0.098	0.782	0.622	0.158
Min. cor.	0.971	0.960	0.885	0.824	0.907	0.861	0.604	0.934	0.872	0.500	0.960	0.897	0.629
Min. inc.	0.740	0.656	0.641	0.620	0.685	0.584	0.459	0.601	0.589	0.403	0.623	0.607	0.517

A, 'low-short', B, 'low-long', C, 'high-short', and D, 'high-long', corresponding to the 'mean distance - gap length'. The tests 1, 2, and 3 correspond to amino-acid alignments of short, medium, and long sequences, respectively, and each test consists of classes of sequences that share identity of >35%, 40–20%, or <25%. For each condition or test and class of alignments, the proportion of correctly aligned columns is reported for ClustalW and for the probabilistic algorithm (ProAlign). The ProAlign-computed average minimum posterior probabilities among correctly (min. cor.) and incorrectly (min. inc.) aligned sites are also shown.

known (true) alignment, and the proportion of correctly aligned columns was computed with the program SOAP (Loytynoja and Milinkovitch, 2001).

Statistical analyses were performed with the program 'R' (Ihaka and Gentleman, 1996).

RESULTS

Performance of the algorithms

The algorithms were implemented as described in the 'System and methods' with one exception: the dynamic programming recursions (Viterbi, Forward and Backward) were not performed through the complete two-dimension matrices, but only within a fixed-size band. With this procedure, the space complexity of the algorithms is linear in sequence length and alphabet size, whereas the computational complexity increases linearly with sequence length and as the third power of alphabet size.

Our probabilistic algorithm performs in average better than ClustalW in the alignment of nucleotide sequences (Table 1). When the sequences are reasonably similar ($\bar{d} = 0.5$), more than 90% of the columns are correctly aligned, and the difference between the two methods is small. However, in the case of highly diverged data sets ($\bar{d} = 1.0$), the probabilistic algorithm infers approximately 10% (relative) more of the columns correctly. In the alignment of amino-acid data, the relative performances of the methods depend on the average identity of residues among sequences. The probabilistic alignments of sequences from the class >35% are nearly as accurate as those performed with ClustalW, the performance in the class 40–20% is still reasonably good, whereas ClustalW alignments of sequences from the class <25% are clearly superior (Table 1). For the nucleotide data sets, the average estimates for δ and ε are 0.003 and 0.75–0.78 or 0.009 and 0.65–0.68 under 'low' or 'high' conditions, respectively. For the protein data sets, the estimates of δ and ε depend on the average sequence similarity rather than on the mean sequence length. Both in classes >35% and 40–20%, the estimates for δ and ε are 0.01–0.03 and 0.75–0.80, respectively, whereas in the class <25% (where gaps are more frequent), the corresponding estimates are higher: 0.05–0.07 and 0.93–0.96, respectively.

One of the most important results of our analyses is that the average minimum posterior probability of correct columns is significantly higher than that of incorrect columns (Table 1), such that the minimum posterior probability of an alignment site and its correctness strongly correlate. Indeed, as illustrated in Figure 2, the posterior probability value of alignment columns is an efficient criterion for the identification of mis-aligned positions. The minimum, rather than the mean, posterior probability of an alignment site is a better indicator of its correctness (Table 2). However, the minimum posterior probability values of correctly- and incorrectly-aligned sites slightly overlap, and excluding ('filtering out') columns from the alignment according to a fixed minimum posterior probability threshold can cause two types of errors: some correctly aligned columns are lost and/or false columns are retained. As shown in Figure 3, optimal reduction of both type-I and type-II errors require different values of minimum probability threshold for closely and distantly related sequences. Model parameter values probably have an impact on both types of errors, and optimization of δ and ε estimates warrants further studies.

As we used known (i.e. 'true') alignments for testing the efficiency of our method, we could demonstrate that random breaking of 'ties' (i.e. choosing among equally good solutions) and sampling locally sub-optimal solutions can yield improved alignments (data not shown). However, we failed to find a criterion allowing to identify the most correct result when the true alignment is unknown. Indeed, after producing 10 alignments by random tie-breaking, and 30 alignments by sampling, we compared their correctness (percent correctly aligned columns) to their probability scores (the sum of Viterbi paths), as well as to their average posterior probability across sites (column minimum

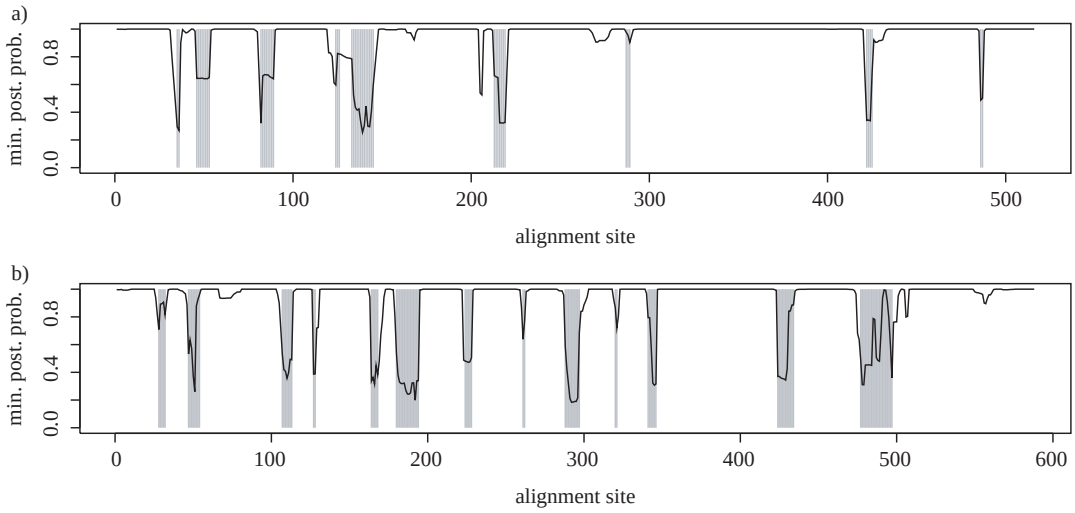

Fig. 2. The minimum posterior probability of an alignment column correlates well with its correctness. Incorrectly-aligned sites are indicated in gray, and minimum posterior probability of alignment sites is plotted for (a) a simulated nucleotide data set under the 'low-long' condition; and (b) an amino-acid data set ('3pmg' from the BAliBASE Reference1/Test3). The coefficients of correlation between correctness and minimum posterior probability are 0.705 and 0.735 for the alignments a) and b), respectively.

Table 2. The minimum, rather than the mean, posterior probability of an alignment site best indicates its correctness.

Cond. / %id.	Min. post. prob.	Mean post. prob.
Low-short	0.471 (0.152)	0.290 (0.106)
Low-long	0.506 (0.160)	0.338 (0.111)
High-short	0.418 (0.098)	0.222 (0.067)
High-long	0.411 (0.110)	0.197 (0.075)
>35%	0.598 (0.094)	0.500 (0.094)
40–20%	0.446 (0.132)	0.394 (0.118)
<25%	0.129 (0.170)	0.109 (0.146)

The table shows, for each of the four conditions of nucleotide evolution and the three classes of residue identity (Ref1/Test 3 of BAliBASE), the mean coefficient of correlation (and its standard deviation) between alignment site correctness and their minimum (min. post. prob.) or mean (mean post. prob.) posterior probability.

or mean). The resulting correlations were largely insufficient to be used as criteria for the selection of the most correct solution: R^2 is rarely >0.20, and every criterion includes cases where the slope is negative. Generally, the score from sampled alignments correlates more strongly with correctness than tie-breaking alignments do, but this result is due to the much larger variance of score exhibited by the former than by the latter.

Computer program

The novel algorithms described here are implemented in the software 'ProAlign'. A graphical interface allows the user to (i) perform alignments of nucleotide or amino-acid sequences, (ii) view the quality of solutions, (iii) filter the unreliable alignment regions and (iv) export the alignments to other softwares. ProAlign also offers the possibility to iterate the multiple alignment procedure by either choosing (with tie-breaking) among multiple best paths or sampling paths according to their posterior probabilities. A command-line interface is available for automating tasks.

ProAlign aligns one of the test nucleotide data set in 35 s on a standard Intel 500 MHz computer, whereas ClustalW performs an alignment with the same data set in less than 8 s. ProAlign is written in Java, and runs on Linux, Mac OSX and Windows computers. It is licensed under the GNU General Public License, and is freely downloadable at http://www.ulb.ac.be/sciences/ueg/.

DISCUSSION

We describe here a novel method for probabilistic multiple alignment by combining a pair HMM, a progressive algorithm, and an evolutionary model describing the nucleotide or amino-acid substitution process. We model sequence sites with vectors of character probabilities (including gaps), and define ancestral sequences from the pairwise alignment of their child sequences. The alignment of two internal nodes (i.e. 'multiple' alignment) is not distinguished from the alignment of two external nodes (classical pairwise alignment). As the complete alignment procedure is probabilistic, this new method allows realistic sampling of alternative solutions, as well as probabilistic evaluation of alignments.

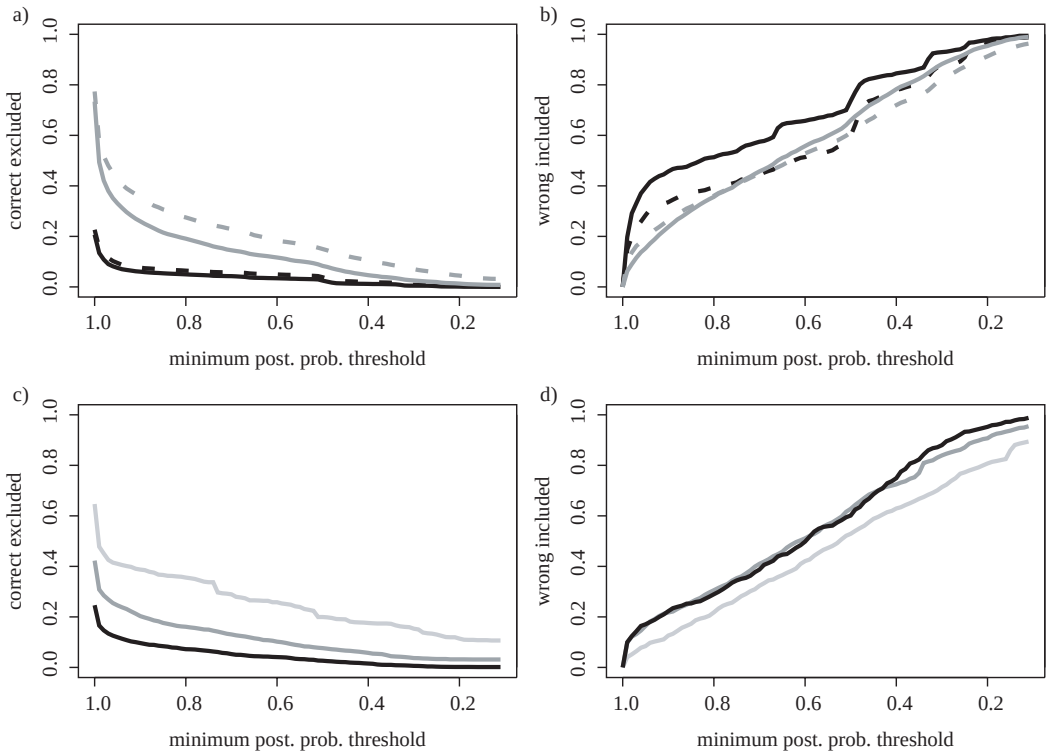

Fig. 3. The minimum posterior probability of correctly-aligned columns is in average higher than that of incorrectly-aligned columns, such that the minimum posterior probabilities of columns can be used as a criterion for filtering out bad data. The threshold of probability for column exclusion is plotted against (a) & (c), proportion of rejected correctly-aligned columns (type I error); (b) & (d), proportion of accepted incorrectly-aligned columns (type II error). (a) and (b) graphs: simulated nucleotide sequences; black and gray lines correspond to closely- and distantly-related sequences, solid and dashed to short and long gaps, respectively. (c) and (d) graphs: amino-acid sequences from the BAliBASE Reference1/Test 3; black, dark gray and light gray correspond to sequences that share >35%, 40–20%, and <25% residue identity, respectively.

We tested our method with simulated nucleotide data sets, as well as with amino-acid data sets from the BAliBASE reference database, and compared its performance to that of the widely used program ClustalW. Our new method performs better than ClustalW for the alignment of nucleotide data, while the two approaches are nearly equally efficient for the alignment of protein sequences that share >35% residue identity. On the other hand, our model is inefficient, in its current form, for the alignment of protein sequences that share less than 25% residue identity. Still, we feel that the probabilistic method described here provides a valuable improvement over existing progressive multiple alignment methods. In addition to constructing alignments that are comparable to those of ClustalW, our algorithm provides, for each aligned position, its posterior probability of being correct. We demonstrate that the minimum posterior probability of an alignment site correlates well with its correctness, and can therefore be used as a criterion for the detection (and removal) of unreliably aligned regions.

Our results from the nucleotide alignments can probably be generalized, although we only generated data sets under a homogeneous process, and used the same model (i.e. JC) both in our algorithm and for the generation of data sets. One of the ClustalW built-in heuristics that can help in the alignment of nucleotide sequences is the increase of both gap-opening and gap-extension penalties if there are no gaps in the column but gaps already occur in adjacent sites (Thompson *et al.*, 1994). Our method does not require such heuristics because a gap, as other characters, is given a non-null probability in the ancestral sequence. On the other hand, when aligning protein sequences, the gap-opening penalty of ClustalW also depends on the amino-acid context, i.e. the penalty is further decreased if the region is hydrophilic. That makes biologically sense, as protein's hydrophobic core regions are usually best conserved, and the length variation mainly occurs in the exposed (hydrophilic) loops. Our method does not model protein's secondary structure, which likely explains its inferior performances in the alignment of

highly diverged amino-acid sequences. However, HMMs can be adapted for incorporating secondary structure parameters: states would correspond to different protein secondary structural elements and, in each state, the character background and substitution probabilities would be independently estimated (Goldman *et al.*, 1996). Such a model would significantly increase the algorithm's computational complexity.

ACKNOWLEDGEMENTS

We thank three anonymous reviewers for their constructive comments on an earlier version of the manuscript.

REFERENCES

Churchill,G.A. (1989) Stochastic models for heterogeneous DNA sequences. *Bull. Math. Biol.*, **51**, 79–94.

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge.

Eddy,S.R. (1995) Multiple alignment using hidden Markov models. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **3**, 114–120.

Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Goldman,N., Thorne,J.L. and Jones,D.T. (1996) Using evolutionary trees in protein secondary structure prediction and other comparative sequence analyses. *J. Mol. Biol.*, **263**, 196–208.

Gotoh,O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Hein,J., Wiuf,C., Knudsen,B., Moller,M.B. and Wibling,G. (2000) Statistical alignment: computational properties, homology testing and goodness-of-fit. *J. Mol. Biol.*, **302**, 265–279.

Holmes,I. and Bruno,W.J. (2001) Evolutionary hmms: a Bayesian approach to multiple alignment. *Bioinformatics*, **17**, 803–820.

Ihaka,R. and Gentleman,R. (1996) R: a language for data analysis and graphics. *J. Comput. Graph. Stat.*, **5**, 299–314.

Jukes,T.H. and Cantor,C. (1969) *Mammalian Protein Metabolism*. Academic Press, New York, pp. 121–132.

Karplus,K., Barrett,C. and Hughey,R. (1998) Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, **14**, 846–856.

Loytynoja,A. and Milinkovitch,M.C. (2001) SOAP, cleaning multiple alignments from unstable blocks. *Bioinformatics*, **17**, 573–574.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–553.

Rabiner,L.R. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, **77**, 257–286.

Saitou,N. and Nei,M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.

Stoye,J., Evers,D. and Meyer,F. (1998) Rose: generating sequence families. *Bioinformatics*, **14**, 157–163.

Thompson,J.D., Higgins,D.G. and Gibson,T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Thompson,J.D, Plewniak,F. and Poch,O. (1999) BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, **15**, 87–88.

Thorne,J.L., Kishino,H. and Felsenstein,J. (1991) An evolutionary model for maximum likelihood alignment of DNA sequences. *J. Mol. Evol.*, **33**, 114–124.

Thorne,J.L., Kishino,H. and Felsenstein,J. (1992) Inching toward reality: an improved likelihood model of sequence evolution. *J. Mol. Evol.*, **34**, 3–16.

Vingron,M. (1996) Near-optimal sequence alignment. *Curr. Opin. Struct. Biol.*, **6**, 346–352.

Viterbi,A.J. (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Informat. Theor.*, **IT-13**, 260–269.

Whelan,S. and Goldman,N. (2001) A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Mol. Biol. Evol.*, **18**, 691–699.