# COACH: profile-profile alignment of protein families using hidden Markov models

Robert C. Edgar[1†] and Kimmen Sjölander[2]

[1]195 Roque Moraes Drive, Mill Valley, CA 94941, USA. bob@drive5.com
† Corresponding author

[2] Department of Bioengineering, University of California, Berkeley, CA 94720, USA. kimmen@uclink.berkeley.edu

## ABSTRACT

### Motivation

Alignments of two multiple sequence alignments, or statistical models of such alignments (profiles), have important applications in computational biology. The increased amount of information in a profile versus a single sequence can lead to more accurate alignments and more sensitive homolog detection in database searches. Several profile-profile alignment methods have been proposed and have been shown to improve sensitivity and alignment quality compared with sequence-sequence methods (such as BLAST) and profile-sequence methods (e.g. PSI-BLAST). Here we present a new approach to profile-profile alignment we call COACH, for Comparison Of Alignments by Constructing HMMs. COACH aligns two multiple sequence alignments by constructing a profile hidden Markov model (HMM) from one alignment and aligning the other to that HMM.

### Results

We compare the alignment accuracy of COACH with two recently published methods: Yona and Levitt's prof_sim and Sadreyev and Grishin's COMPASS. On two sets of reference alignments selected from the FSSP database, we find that COACH is able, on average, to produce alignments giving the best coverage or the fewest errors, depending on the chosen parameter settings.

### Availability

COACH is freely available, including source code, from: http://phylogenomics.berkeley.edu/COACH.

## 1. INTRODUCTION

### 1.1 Profiles and profile alignments

The construction of pairwise alignments of two protein sequences is a fundamental technique in computational biology. A pairwise alignment of an uncharacterized protein with a sequence of known function or structure can help identify homologous regions, from which inferences about function and structure can be made. A score or expectation value can be computed from the alignment, giving a statistical measure of the relatedness of the two sequences. This can be used to discriminate homologs from unrelated sequences and indicate the degree of functional or structural similarity that may be reliably inferred. Distantly related proteins may share a common fold and function, but their similarity can hard to detect from primary sequence alone (Brenner *et al.*, 1998). Multiple alignments of related proteins provide further information about the family, indicating patterns of conservation or variation at each position. A *profile* is a statistical model of a multiple alignment. Profiles typically contain the estimated probability of finding each amino acid type at each position, and may include position-specific gap penalties. We distinguish three classes of pairwise alignment algorithms. *Sequence-sequence* methods such as BLAST (Altschul *et al.*, 1990) and FASTA (Pearson, 1990) use the two primary sequences alone. *Profile-sequence* methods (Gribskov *et al.*, 1988; Tatusov *et al.*, 1994) such as PSI-BLAST (Altschul *et al.*, 1997) and SAM-T98 (Karplus *et al.*, 1998) align a query sequence to a profile. Recently, several *profile-profile* methods have been proposed (Pietrokovski, 1996; Lyngso *et al.*, 1999; Panchenko *et al.*, 2000; Rychlewski *et al.*, 2000; Yona and Levitt, 2002; von Öhsen *et al.*, 2003; Panchenko, 2003; Sadreyev and Grishin, 2003). These construct an alignment of two profiles, from which a similarity score and pairwise alignment of the two query sequences can be derived. Improvements in both alignment accuracy and homolog recognition have been reported for profile-profile methods over profile-sequence and sequence-sequence methods. Profile-profile methods have been used in genome annotation and protein classification (e.g. Pawlowski *et al.*, 1999; Henikoff *et al.*, 2000; Kunin *et al.*, 2001; Pawlowski *et al.*, 2001). Profile-profile alignment is also the iterated step in progressive multiple alignment algorithms such as CLUSTALW (Thompson *et al.*, 1994).

### 1.2 COACH: aligning a multiple sequence alignment to an HMM

In the following, we describe a new profile-profile algorithm we call COACH, for Comparison of Alignments by Constructing HMMs. COACH aligns two multiple sequence alignments by estimating a profile HMM from one alignment and aligning the other alignment to that HMM. COACH is the iterated step in SATCHMO, a progressive multiple sequence alignment method that has been shown (Edgar and Sjölander, 2003a) to produce alignments of comparable quality to CLUSTALW. We compare COACH to two recently published profile-profile methods: prof_sim (Yona and Levitt, 2002) and COMPASS (Sandreyev and Grishin, 2003). We assess alignment accuracy by comparing sequence alignments produced by these methods with structural alignments from the FSSP database (Holm and Sander, 1996).

### 1.3 Profile hidden Markov models

A profile HMM (Krogh *et al.*, 1994; Eddy, 1996) is a graphical model of a protein family that emits a sequence of letters representing amino acids. Each position (*node*) $k$ in a profile HMM has a match state ($M_k$), insert state ($I_k$) and delete state ($D_k$) (Figure 1). M states represent conserved positions, I states represent insertions relative to the consensus. M and I states emit letters; D states are silent, allowing for gaps. $M_k \rightarrow I_k$ and $M_k \rightarrow D_{k+1}$ transition probabilities correspond to scores for opening a gap in the profile and the emitted sequence respectively; $D_k \rightarrow D_{k+1}$ and $I_k \rightarrow I_k$ probabilities correspond to gap-extend scores. Gap penalties are thus position-specific, and the associated transition probabilities can be estimated from observed sequences.

### 1.4 Aligning an alignment to an HMM

Consider two multiple alignments, *A* and *T*. *A* has *N* sequences. Estimate an HMM from *T*. Run the HMM *N* times, generating *N* sequences. Align those sequences to each other by placing letters emitted by a given model state in the same column; call the resulting multiple alignment *B*. If *A* and *B* are identical (to within a trivial re-ordering of the sequences), we say that the HMM generated *A*. In that case, we achieve an alignment (call it *C*) of *A* to *T* via the model in which the columns of *A* and *T* are kept intact, as they must be if we trust that *A* and *T* are correct (Figure 2). Given this view, we can treat an HMM as an emitter of alignments. For example, we can compute the probability $P(A \mid \text{HMM})$ that the model generates *A*, and seek an optimal alignment of *A* to the HMM. Note that this differs from aligning the sequences individually to the model as all letters in a given column of *A* must be emitted from the same state, whereas the optimal alignment of an individual sequence might assign the letter in that column to a different state. If *A* is a good alignment, this may allow distantly related sequences to be more accurately aligned to the model. To make this more formal, we define the following terms. The *template alignment* (*T*) is the multiple sequence alignment from which the HMM is estimated. The *input alignment* (*A*) is the multiple alignment that is to be aligned to the HMM. An *output alignment* (*C*) is an alignment of the input alignment to the HMM or, equivalently, of the input alignment to the template alignment. A *gapless sequence* is a sequence in the input alignment that has a residue in every column. A gapless sequence need not be present, but is helpful conceptually. An output alignment is constructed by assigning each column in the input alignment to an emitter state in the HMM. Such an assignment uniquely determines the path that a gapless sequence must take through the HMM; we call this path a *route*. Given a route, a sequence with one or more gaps in the input alignment is uniquely constrained to take a related path, as illustrated in Figure 2. The probability of a route, given the input alignment and an HMM, is obtained by multiplying the probabilities of the paths implied by that route for each sequence in the input alignment. We construct a Viterbi output alignment (see section 2.1) by finding a most probable route that generates the input alignment. We fully account for transition probabilities associated with gapped positions in every sequence. Perhaps surprisingly, this turns out to be possible without increasing the effective time complexity of our algorithm compared with other profile-profile dynamic programming (DP) methods. We are thus able to exploit information about conserved patterns of gaps of different lengths, in contrast to methods that treat a gap as the 21st letter and are invariant under a permutation of a column (e.g. COMPASS), and those that apply position-independent affine gap penalties, ignoring internal gaps within a profile (e.g. prof_sim). Rather than estimating the HMM from the template alignment, it would be more natural to estimate an HMM from both the input alignment *A* and template alignment *T*, and align both *A* and *T* to that HMM. This would treat *A* and *T* symmetrically and avoid the question of which alignment is chosen to be the template. However, our approach appears to be more computationally tractable.

## 2. METHODS

### 2.1 Dynamic programming applied to a multiple alignment

Alignment of a sequence to an HMM is accomplished using the Viterbi algorithm (Bellman, 1957; Viterbi, 1967; Rabiner, 1989). The Viterbi algorithm finds a most probable path through the model that generates the input sequence, and relies on recursion relations that express the probability of a most probable partial path in terms of paths with one less edge. By developing analogous recursion relations for routes, we can extend the Viterbi algorithm to handle multiple alignments. We introduce the term *leg* to refer to an edge in a route. For a given consecutive pair of columns in the input alignment, with assignments of those columns to emitter states, the leg is the edge implied for a sequence that has no gaps in those columns. A difficulty arises related to gaps in columns emitted by insert (I) states, as illustrated in Figure 3. Suppose we know the probability of a most probable partial route that ends in state $I_k$ and emits the input alignment up to column $i$–1. Now we wish to compute the change in probability due to adding a leg that ends in the $I_k$ state again, emitting column $i$. It is clear that this adds an $I_k \rightarrow I_k$ edge for Seq1 in Figure 3, but the edge for Seq2 is undetermined by the information given because the predecessor state for Seq2's path is unknown (it could be $I_k$, $M_k$ or $D_k$). We overcome this difficulty by (1) computing tables that characterize the occurrence of gaps, and (2) introducing new DP matrices that track the leg by which a most probable route enters a given insert state. This enables us to "trace back" to the start of an insert of any length both in the route (by means of the new DP matrices) and in the input alignment (using the new gap tables) with an O(1) computation. We further show that the gap tables can be efficiently computed by visiting each position in the input alignment once only. Thus, we create two different types of profile: an HMM for the template alignment, and a new type of profile for the input alignment. This second type of profile includes observed residue frequencies at each position plus the frequencies of observed gaps with all starting positions and lengths. The recursion relations are relatively complicated; we derive and state them in full in the Appendix. As expected, they reduce to the familiar Viterbi algorithm in the case where the input alignment contains exactly one sequence.

## 3. VALIDATION

### 3.1 Reference alignments

In earlier work (Edgar and Sjölander, 2003b), we assessed the alignment accuracy of 23 different profile-profile scoring functions by comparing sequence alignments generated by those functions with 488 structural alignments from the FSSP database. We demonstrated improved accuracy of profile-profile over profile-sequence and sequence-sequence methods, but found no statistically significant difference between most of the scoring functions on this test set, which we call PP1. The PP1 dataset is composed of regions selected for a high degree of structural alignability in order to reduce possible ambiguities in the sequence alignment implied by a structural alignment. This was done by requiring the DALI structural alignment z-score to be ≥ 15, root-mean square distance (RMSD) to be ≤ 2.5Å and exact agreement between FSSP and the CE structural aligner (Shindyalov and Bourne, 1998) over a minimum of 50 consecutive positions. We speculate that these stringent criteria, which tend to limit the number of gapped positions, produced a test set for which the detection of weak sequence similarity is sufficient to produce high-quality alignments, making it relatively insensitive to possible performance differences between alignment methods and parameters. For the present work, we therefore designed a new test set based on criteria designed to include more structurally diverged proteins and hence more gaps. We call this new test set PP2. We selected pairs of sequences from the FSSP having ≤ 30% identity, DALI z-score ≥ 8 and ≤ 12, RMSD ≤ 3.5Å and alignment length ≥ 50. These criteria alone are sufficiently relaxed to allow matches between convergent folds and regions of similar secondary structure, so we additionally required that the two sequences were homologous according to the SCOP database (Murzin *et al.*, 1995). To reduce redundancy, these pairs were filtered so that no two sequences aligned to a common third sequence had greater than 30% identity. Finally, we selected 500 pairs at random from the remainder. We retained all positions considered alignable by FSSP (agreement with CE was not required, in contrast to PP1). These alignments can therefore be expected to contain regions upon which different structural aligners disagree, and within which consideration of probable homology rather than atom coordinates alone may produce some shifts (Cline, 2000). We consider this a reasonable price to pay in an attempt to improve the sensitivity of the reference data and see reason to suppose that our criteria might bias alignments in favor of one sequence-based method over another. Alignments were created by PSI-BLAST from a release of the NCBI non-redundant protein sequence database (Pruitt *et al.*, 2003) downloaded in January 2003. We used blastpgp version 2.2.5 with options –h 5 –e 0.1, keeping only alignments produced by the final iteration.

### 3.2 Quality scores

We use three quality scores for comparing a test alignment with a reference alignment. $Q_{\mathrm{Dev}}$ (the *developer's score*) is the number of correctly aligned pairs in the test alignment, $t_c$, divided by the length of the reference alignment. This score has been used, for example, by Thompson *et al.* (1994), who call it SP, Sauder *et al.* (2000), who refer to it as $f_D$, and Sadreyev and Grishin (2003) who call it $Q_{\mathrm{developer}}$. $Q_{\mathrm{Mod}}$ (the *modeler's score*) is $t_c$ divided by the length of the test alignment; this is Sauder *et al.*'s $f_M$, and Sadreyev and Grishin's $Q_{\mathrm{modeler}}$. Each of these scores is useful in some applications, but also has drawbacks. $Q_{\mathrm{Dev}}$ does not penalize over-alignment (i.e., aligning residue pairs that are not structurally alignable); $Q_{\mathrm{Mod}}$ does not penalize under-alignment. Neither gives credit for regions in the test alignment that are shifted by one or a few positions relative to the reference alignment; however, such regions may still be successfully used in homology modeling, and may even be more "correct" when probable homology is considered rather than atom coordinates alone. Cline *et al.* (2002) have proposed a score that is designed to address these issues; we call it $Q_{\mathrm{Cline}}$ (the *Cline score*). It penalizes both over- and under-alignment, and gives positive, although reduced, scores for positions with small shifts. $Q_{\mathrm{Cline}}$ has a parameter $\varepsilon$ that controls the range of shifts that get positive scores; following Cline *et al.* we set $\varepsilon = 0.2$. All three scores have a maximum value of one in the case of perfect agreement. $Q_{\mathrm{Dev}}$ and $Q_{\mathrm{Mod}}$ have a minimum of zero when no pairs are correctly aligned; $Q_{\mathrm{Cline}}$ can achieve negative values when there are many large shifts.

### 3.3 Other profile-profile methods

COMPASS is readily available in binary form. Dr. Golan Yona kindly provided a binary version of prof_sim. CLUSTALW includes a profile-profile algorithm but was unable to process many of the alignments in our test set, apparently because its sequence weighting scheme requires that all pairs of sequences in a profile have at least one position in common (T. J. Gibson, personal communication). We were unable to obtain implementations of other published profile-profile methods.

### 3.4 COACH HMM estimation

HMM parameters were estimated from PSI-BLAST alignments as follows. Sequence weights were applied using the method of Henikoff and Henikoff (1994). A match state was created for each column. Match state emission distributions were computed using the Dirichlet mixture prior given by Sjölander *et al*. (1996). Transition distributions were computed using the default Dirichlet density prior in the HMMER package (Eddy, 2001).

### 3.5 Alignment boundary conditions

COMPASS and prof_sim construct local alignments. At least one published profile-profile method (von Öhsen *et al.*, 2003) is based on global alignment. In COACH we allow a choice of boundary conditions: local to both sequences, global to the HMM but local to the sequence (*semi-global*), and global to both sequences. For database searching, local alignments are often chosen; however semi-global searches are useful for domain recognition, and a requirement of global similarity may be more appropriate if functional inferences are to be made for a multi-domain protein. COACH requires that one alignment be chosen as the template and the other as the target. In the semi-global case, we choose the shorter profile as the template (we found weak evidence, not presented here, that this gives more accurate alignments). With other boundary conditions we found no feature of the alignments that predicted the better choice, including the log-odds scores relative to different null models, and therefore choose arbitrarily.

### 3.6 COACH local alignment options

Local alignment requires special consideration. Adding a match state to a sub-path always reduces its probability; it is therefore not possible to define local alignment by seeking the most probable sub-path. A common solution is to add terminal insert states before and after the end of the main model and to allow transitions from the N-terminal insert state into any match state and from any match state into the C-terminal insert state. Letters before the locally aligned region are emitted by the N-terminal inserter; letters following the region are emitted by the C-terminal inserter. The self-loop transition probability $S$ of the terminal insert states controls the average length of a local alignment. If $S$ is chosen to be the average M→M probability, then, on average, local alignments will be extended only if there is a positive match state score for the additional letter (because the cost of adding an additional match  state is compensated by the reduced cost of making one less self-loop in the terminal insert state). However, this design may not be optimal for alignments of distant homologs because match state scores may be negative for residues that are plausible for a given position when sequence identity is low, causing a local alignment to be truncated. A simple heuristic to correct for this effect is to adjust $S$: reducing the terminal self-loop probability makes it more favorable to add match states to the local path, allowing weakly negative matches to extend the local alignment. A more rigorous approach would be to re-estimate match state probabilities for different degrees of divergence, e.g. as a function of sequence identity; however, this raises theoretical and practical issues beyond the scope of this report. COACH offers three alternatives for local alignment. (1) Set $S$=$S_{\mathrm{Null}}$, the self-loop probability in a simple null model consisting of a single insert state. $S_{\mathrm{Null}}$ is chosen, following HMMER, such that the null model emits sequences of length 350, the approximate average length of a protein. $S_{\mathrm{Null}}$ is larger than the typical $\mathrm{M}_k \rightarrow \mathrm{M}_{k+1}$ probability, so this setting requires positive local match scores and can therefore be considered very conservative, tending to produce short alignments of high confidence. (2) Set $S$=$S_{\mathrm{AvgMM}}$, the average $\mathrm{M}_k \rightarrow \mathrm{M}_{k+1}$ probability. With this setting, any non-negative match score will extend the alignment. (3) Set $S$=$S_{\mathrm{Div}}$, a value tuned to the estimated divergence of the two profiles by optimizing on training sets of different divergences. Moving from option (1) to (2) then (3) increases coverage (makes longer alignments) at the expense of higher error rates (over-alignment). Option (2) is roughly equivalent to introducing what we have previously called a *center parameter* (Edgar and Sjölander, 2003b) and Yona and Levitt (2002) have termed a *shift*, i.e. a constant value added to all match scores, having the effect of tuning the local alignment length. It should be emphasized that these issues with local alignment are found in most dynamic programming methods; they are not specific to COACH. Some profile HMM methods, e.g. SAM and HMMER, use approximate solutions with a pre-determined length bias. In BLAST, the choice of substitution matrix biases the length of the alignment; e.g., BLOSUM62 (Henikoff and Henikoff, 1992) has fewer positive scores than BLOSUM30 and therefore tends to create shorter alignments.

### 3.7 Results

We aligned profiles (or sequences) derived from our reference sets PP1 and PP2 using COACH, prof_sim, COMPASS, PSI-BLAST and BLAST. The average quality scores of the resulting alignments are summarized in Table 1. We find, in agreement with previous studies, that profile methods clearly out-perform BLAST, which uses primary sequence only. The improvements in score over BLAST are highly significant, with p-values computed by the Friedman rank test found to be typically $< 10^{-5}$. In most cases, profile-profile methods achieve higher scores than PSI-BLAST (a profile-sequence method), but a trade-off is now apparent. For example, the lowest error rate, i.e. best $Q_{\mathrm{Mod}}$ average score, is obtained with COACH local $S$=$S_{\mathrm{Null}}$, at the expense of a coverage ($Q_{\mathrm{Dev}}$) that is lower than PSI-BLAST. Similarly, the best $Q_{\mathrm{Dev}}$ is always obtained by COACH semi-global, but at the cost of error rates that are higher than other choices. We observe similar rankings between the methods and similar trade-offs between coverage and error on both sets of reference alignments (PP1 and PP2). As expected, PP2 is much more challenging, with all methods producing much lower quality scores than on PP1. The statistical significance of the differences in score between the different profile-profile methods vary from moderate to low, depending on the parameter settings and chosen quality score.

### 3.8 Execution speed

The recursion relations for COACH are relatively complicated. However, the algorithm can be implemented efficiently, and the CPU time needed by COACH on our reference data was competitive with other methods. For example, COACH required only 8 minutes to complete PP2 on a 2 GHz Pentium 4 PC, including HMM parameter estimation and construction of input profiles, compared with 40 minutes for COMPASS. (Direct comparison with prof_sim is not possible as that method relies on profiles previously constructed by PSI-BLAST).

## 4. DISCUSSION

We have presented a new algorithm, COACH, which computes an optimal alignment of a multiple sequence alignment to a profile hidden Markov model. COACH is the iterated step in SATCHMO, a multiple sequence alignment algorithm that has previously been shown to produce alignments of comparable accuracy to CLUSTALW. We introduced two novel features which enable the Viterbi algorithm for an alignment of length $L$ to be implemented with $O(L^2)$ time complexity, as follows. New dynamic programming matrices track the predecessor state through which a most probable path enters a given insert state. Gap tables store information about the frequencies of gaps of different lengths in the input alignment. These features enable a "trace back" to the start of an insert of any length both in the HMM and in the alignment with an O(1) computation, overcoming a difficulty in constructing the recursion relations due to gaps in inserted regions. The gap tables can be constructed efficiently by induction. The alignment accuracy of COACH was compared with two recent methods: Yona and Levitt's prof_sim, and Sadreyev and Grishin's COMPASS. Like COMPASS, but unlike prof_sim, COACH reduces to a method expected to work well in the case where one or both profiles are derived from a single sequence. Accuracy was assessed on two sets of reference alignments derived from the FSSP database. One set (PP1) was selected for a high degree of structural alignability, the other (PP2) for more diverged pairs of structures. Multiple alignments were generated from each selected FSSP sequence using PSI-BLAST, from which profiles were constructed. In agreement with previous studies, we find that profile methods are clearly superior to BLAST, and generally superior to PSI-BLAST. However, in the latter case, differences in performance are smaller, and vary according to parameter settings and the chosen measure of alignment quality. On both PP1 and PP2, on average, COACH gave the best coverage (with semi-global boundary conditions) and the fewest errors (with local boundary conditions and a suitably chosen null model). However, differences in quality score were not statistically significant in some cases. We conclude that COACH is competitive in accuracy and speed with other available profile-profile methods. We suggest that it is useful to have a choice of boundary conditions, and perhaps also a means to select increased coverage or reduced error rates when using local alignment.

## ACKNOWLEDGMENTS

The authors thank Melissa Cline, Sean Eddy and Kevin Karplus for helpful discussions.

# A. APPENDIX

## A.1 Viterbi recursion relations

For brevity, we develop the Viterbi recursion relations for typical model nodes and typical columns without considering boundary cases at the beginning and end of the model and of the input alignment; these are easily determined for a given model architecture and are used, for example, to choose local versus global alignments. We use the following notation.

| | |
|---|---|
| $A$ | The input alignment, i.e., the alignment to be aligned to the HMM. |
| $N$ | Number of sequences in $A$. |
| $L$ | Number of columns in $A$. |
| $A_i$ | The $i$th column of $A$, $i$=1...$L$. |
| $a_{iv}$ | Number of letters of type $v$ in $A_i$. |
| $A\{i\}$ | The first $i$ columns of $A$. |
| Q, R, q, r | HMM states. |
| t(QR) | Log-odds transition score for Q→R. |
| e(R, $v$) | Log-odds emission score of letter $v$ in R. |
| $\boldsymbol{\pi}(\mathrm{R}, i)$ | A most probable route that emits $A\{i\}$ and assigns state R to column $i$. |
| $\pi_\mathrm{s}(\mathrm{R}, i)$ | The path of sequence $s$ implied by $\boldsymbol{\pi}(\mathrm{R}, i)$. |
| $\sigma(\pi_\mathrm{s}(\mathrm{R}, i))$ | The log-odds score of $\pi_\mathrm{s}$. |

Here, a log-odds score is bit score, i.e. $\log_2(P)$ for a given probability $P$. We define:

$$\boldsymbol{\sigma}(\boldsymbol{\pi}(\mathrm{R}, i)) = \textstyle\sum_s \sigma(\pi_\mathrm{s}(\mathrm{R}, i)). \tag{1}$$

This is the log-odds score of a most probable route that generates the input alignment through column $i$ and assigns state R to that column. We define the *leg score* $\boldsymbol{\lambda}$ for column $i$ given two states Q and R connected by an edge Q→R as:

$$\boldsymbol{\lambda}(\mathrm{QR}, i+\delta_\mathrm{R}) = \boldsymbol{\sigma}(\boldsymbol{\pi}(\mathrm{Q}, i)) - \boldsymbol{\sigma}(\boldsymbol{\pi}(\mathrm{R}, i+\delta_\mathrm{R})), \tag{2}$$

where $\delta_\mathrm{R} = 1$ if R is an emitter state, $\delta_\mathrm{R} = 0$ otherwise. The leg score is the incremental cost of adding a QR leg to a most probable route. If $\boldsymbol{\lambda}(\mathrm{QR}, i)$ can be calculated for all Q, R and $i$, this gives the Viterbi recursion relations:

$$\boldsymbol{\sigma}(\boldsymbol{\pi}(\mathrm{R}, i+\delta_\mathrm{R})) = max_\mathrm{Q}\ \{\boldsymbol{\sigma}(\boldsymbol{\pi}(\mathrm{Q}, i)) + \boldsymbol{\lambda}(\mathrm{QR}, i+\delta_\mathrm{R})\} \tag{3}$$

The leg score can be expressed as an emission term $\boldsymbol{\varepsilon}$ and a transition term $\boldsymbol{\tau}$:

$$\boldsymbol{\lambda}(\mathrm{QR}, i) = \boldsymbol{\varepsilon}(\mathrm{QR}, i) + \boldsymbol{\tau}(\mathrm{QR}, i) \tag{4}$$

If R is a delete state, $\boldsymbol{\varepsilon}$ is zero, otherwise:

$$\boldsymbol{\varepsilon}(\mathrm{QR}, i) = \textstyle\sum_v a_{iv}\, \mathrm{e}(\mathrm{R}, v). \tag{5}$$

The transition score $\boldsymbol{\tau}$ can be expressed as:

$$\boldsymbol{\tau}(\mathrm{QR}, i) = \textstyle\sum_s \mathrm{t}(\mathrm{qr}_s), \tag{6}$$

where $\mathrm{qr}_s$ is the edge, if any, implied for sequence $s$ by leg QR. If no edge is implied, $\mathrm{t}(\mathrm{qr}_s)$ is understood to be zero. In many cases, $\mathrm{qr}_s$ can be deduced from the leg type by looking at column $i$ and the previous column $i$–1 to see if there is a gap or letter in those positions, as shown in Table 2.

We define the following occupancy vectors over the input alignment. Values are the number of sequences that have the given contents in columns $i$ and, if applicable, $i$–1.

| | | |
|---|---|---|
| $\mathrm{G}_i$ | Gap in column $i$. | (7) |
| $\mathrm{L}_i$ | Letter in column $i$. | (8) |
| $\mathrm{LL}_i$ | Letter in columns $i$–1 and $i$. | (9) |
| $\mathrm{GG}_i$ | Gaps in columns $i$–1 and $i$. | (10) |

$\mathrm{LG}_i$ Letter in column $i$–1, gap in column $i$. (11)
$\mathrm{GL}_i$ Gap in column $i$–1, letter in column $i$. (12)

These vectors enable us to calculate $\boldsymbol{\tau}(\mathrm{QR}, i)$ for all leg types except those for which q=? in Table 2. For example, by considering cases 1–4, it follows that:

$$\boldsymbol{\tau}(\mathrm{M}_{k-1}\mathrm{M}_k, i) = \mathrm{LL}_i\ \mathrm{t}(\mathrm{M}_{k-1}\mathrm{M}_k) + \mathrm{GL}_i\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{M}_k) + \mathrm{LG}_i\ \mathrm{t}(\mathrm{M}_{k-1}\mathrm{D}_k) + \mathrm{GG}_i\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{D}_k). \tag{13}$$

Similarly,

$$\boldsymbol{\tau}(\mathrm{M}_k\mathrm{I}_k, i) = \mathrm{LL}_i\ \mathrm{t}(\mathrm{M}_k\mathrm{I}_k) + \mathrm{GL}_i\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{I}_k) \tag{14}$$

$$\boldsymbol{\tau}(\mathrm{M}_{k-1}\mathrm{D}_k, i) = \mathrm{L}_i\ \mathrm{t}(\mathrm{M}_{k-1}\mathrm{D}_k) + \mathrm{G}_i\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{D}_k) \tag{15}$$

$$\boldsymbol{\tau}(\mathrm{D}_{k-1}\mathrm{M}_k, i) = \mathrm{L}_i\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{M}_k) + \mathrm{G}_i\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{D}_k) \tag{16}$$

$$\boldsymbol{\tau}(\mathrm{D}_k\mathrm{I}_k, i) = \mathrm{L}_i\ \mathrm{t}(\mathrm{D}_k\mathrm{I}_k) \tag{17}$$

$$\boldsymbol{\tau}(\mathrm{D}_{k-1}\mathrm{D}_k, i) = N\ \mathrm{t}(\mathrm{D}_{k-1}\mathrm{D}_k). \tag{18}$$

We next consider the $\mathrm{I}_k\mathrm{I}_k$ leg shown in Figure 4. We assume that the $\mathrm{I}_k$ state was entered via $\mathrm{M}_k$, and denote by C the first column assigned to the $\mathrm{I}_k$ state. If there is a continuous series of gaps that extends exactly from column C to column $i$–1, as in Seq3, we can see that this implies an $\mathrm{M}_k \rightarrow \mathrm{I}_k$ edge. Given that we have a sequence exhibiting case 10, we know that there is a gap of length ≥ 1 that ends in column $i$–1. The edge type is determined by the length of that gap. Let c = $i$–C–1; then the possible scenarios are:

Gap length = c, edge is $\mathrm{M}_k \rightarrow \mathrm{I}_k$,
Gap length < c, edge is $\mathrm{I}_k \rightarrow \mathrm{I}_k$,
Gap length > c, edge is $\mathrm{D}_k \rightarrow \mathrm{I}_k$.

We now define the following *gap matrices*. Values in these matrices are the number of sequences that have the given type of gap.

$\mathrm{BE}_i[\mathrm{c}]$ Gaps of length = c ending in column $i$. (19)
$\mathrm{BL}_i[\mathrm{c}]$ Gaps of length < c ending in column $i$. (20)
$\mathrm{BG}_i[\mathrm{c}]$ Gaps of length > c ending in column $i$. (21)

Given that we know that the insert state was entered via the match state and the column number C assigned to that leg, by considering cases 9 and 10 we can express the transition score as:

$$\boldsymbol{\tau}(\mathrm{I}_k\mathrm{I}_k, i) = (\mathrm{LL}_i + \mathrm{BL}_{i-1}[\mathrm{c}])\ \mathrm{t}(\mathrm{I}_k\mathrm{I}_k) + \mathrm{BE}_{i-1}[\mathrm{c}]\ \mathrm{t}(\mathrm{M}_k\mathrm{I}_k) + \mathrm{BG}_{i-1}[\mathrm{c}]\ \mathrm{t}(\mathrm{D}_k\mathrm{I}_k). \tag{22}$$

Now suppose that the insert state was entered via the delete state, as shown in Figure 5. As this example illustrates, the scenarios are now:

Gap length < c, edge is $\mathrm{I}_k \rightarrow \mathrm{I}_k$,
Gap length ≥ c, edge is $\mathrm{D}_k \rightarrow \mathrm{I}_k$.

The transition score is therefore:

$$\boldsymbol{\tau}(\mathrm{I}_k\mathrm{I}_k, i) = (\mathrm{LL}_i\ + \mathrm{BL}_{i-1}[\mathrm{c}])\ \mathrm{t}(\mathrm{I}_k\mathrm{I}_k) + \mathrm{BG}_{i-1}[\mathrm{c}-1]\ \mathrm{t}(\mathrm{D}_k\mathrm{I}_k). \tag{23}$$

We introduce two new dynamic programming matrices that track the leg entering a given insert state. $\mathrm{N}_k[i]$ is the column number of the first column to be assigned to $\mathrm{I}_k$ in $\boldsymbol{\pi}(\mathrm{R}, i)$. $\mathrm{S}_k[i]$ is the last state prior to $\mathrm{I}_k$ in $\boldsymbol{\pi}(\mathrm{R}, i)$. The recursion relations for these matrices depend on the leg type added in an iteration, as follows.

| Leg | $N_k[i]$ | $S_k[i]$ |
|---|---|---|
| $M_kI_k$ | $i$ | $M_k$ |
| $D_kI_k$ | $i$ | $D_k$ |
| $I_kI_k$ | $N_k[i–1]$ | $S_k[i–1]$ |

We can then calculate c as:

$$c = i - N_{k-1}[i-1]. \tag{24}$$

This gives us an O(1) calculation of $\boldsymbol{\tau}(I_kI_k, i)$:

```
$\textit{if } S_k[i-1] \textit{ is } M_k \textit{ then}$
    $\boldsymbol{\tau}(I_kI_k, i) = (LL_i + BL_{i-1}[c])\ t(I_kI_k) + BE_{i-1}\ [c]\ t(M_kI_k) + BG_{i-1}[c]\ t(D_kI_k)$
$\textit{else}$
    $\boldsymbol{\tau}(I_kI_k, i) = (LL_i + BL_{i-1}[c])\ t(I_kI_k) + BG_{i-1}[c-1]\ t(D_kI_k)$
$\textit{endif.}$                                                                     (25)
```

The transition scores for the remaining legs types can be obtained through similar reasoning. We need to introduce one more gap matrix: $BT[i, j]$ is the number of sequences with a gap that includes columns $i$ and $j$. Note that for any three columns $a$, $b$, $c$ such that $a \leq b \leq c$, the number of gaps that include $a$ and $b$ but not $c$ is:

$$BT[a, b] - BT[a, c]. \tag{26}$$

Also, the number of gaps that include $b$ and $c$ but not $a$ is:

$$BT[b, c] - BT[a, c]. \tag{27}$$

This leads to the following calculations for $I_kM_k$ and $I_kD_k$ legs (here, $C = N_{k-1}[i-1]$):

```
$\textit{if } S_k[i-1] \textit{ is } M_k \textit{ then}$
    $\boldsymbol{\tau}(I_kM_k, i) = (LL_i + BL_{i-1}[c])\ t(I_{k-1}M_k) + BE_{i-1}[c]\ t(M_{k-1}M_k) + BG_{i-1}[c]\ t(D_{k-1}M_k) +$
        $(BT[C, i] - BT[C-1, i])\ t(M_{k-1}D_k) + (G_i - BT[C, i])\ t(I_{k-1}D_k) + BT[C, i]\ t(D_{k-1}D_k)$
$\textit{else}$
    $\boldsymbol{\tau}(I_kM_k, i) = (LL_i + BL_{i-1}[c])\ t(I_{k-1}M_k) + BG_{i-1}[c-1]\ t(D_{k-1}M_k) + (G_i - BT[C, i])\ t(I_{k-1}D_k) +$
        $BT[C, i]\ t(D_{k-1}D_k)$
$\textit{endif,}$                                                                     (28)

$\textit{if } S_k[i-1] \textit{ is } M_k \textit{ then}$
    $\boldsymbol{\tau}(I_{k-1}D_k, i) = BT[C, i]\ t(D_{k-1}D_k) + (R_i + BT[i, i] - BT[C, i])\ t(I_{k-1}D_k) +$
        $(BT[C, i] - BT[C-1, i])\ t(M_{k-1}D_k)$
$\textit{else}$
    $\boldsymbol{\tau}(I_kD_k, i) = BT[C, i]\ t(D_{k-1}D_k) + (R_i + BT[i, i] - BT[C, i])\ t(I_{k-1}D_k)$
$\textit{endif.}$                                                                     (29)
```

We now have O(1) expressions for the transition scores of all leg types which, via Equations (3-6), give the recursion relations for the Viterbi algorithm.

BE can be constructed using the following procedure. Create an $L \times L$ matrix and initialize all entries to zero. Traverse each sequence in the input alignment from left to right, maintaining a counter c. If a position $i$ contains a gap, add one to c; otherwise add one to $BE_i[c]$ and set c to zero. To construct BL, we start from its definition (20), which can be expressed as:

$$BL_i[c] = \sum_{x < c} BE_i[x]. \tag{30}$$

This implies the following recursion relation:

$$\begin{aligned} &BL_i[0] = 0, \\ &BL_i[c+1] = BL_i[c] + BE_i[c]. \end{aligned} \tag{31}$$

For BG, we can exploit the fact that the total number of gaps ending in column $i$ is $GL_{i-1}$:

$$\mathrm{GL}_{i-1} = \mathrm{BL}_i[\mathrm{c}] + \mathrm{BE}_i[\mathrm{c}] + \mathrm{BG}_i[\mathrm{c}],\ \forall \mathrm{c}, i. \tag{32}$$

Rearranging,

$$\mathrm{BG}_i[\mathrm{c}] = \mathrm{GL}_{i-1} - \mathrm{BL}_i[\mathrm{c}] - \mathrm{BE}_i[\mathrm{c}]. \tag{33}$$

Finally we can compute BT by applying Equation (26) to two consecutive columns:

$$\begin{aligned} &\mathrm{BT}[i, i] = \mathrm{G}_i, \\ &\mathrm{BT}[i, j+1] = \mathrm{BT}[i, j] - \mathrm{BG}_j[j - i]. \end{aligned} \tag{34}$$

### A.2 Complexity

The time and space complexity of our Viterbi algorithm is $O(L^2)$. Profile construction, both for the HMM and for the gap matrices and residue frequencies needed for the input alignment, can be accomplished in $O(LN)$ time and $O(L)$ space. Despite the formidable appearance of the recursion relations, the complexity is therefore comparable with traditional profile-profile alignment methods.

### A.3 Unaligned regions

Columns in the input alignment may be marked as containing positions that are not alignable due to structural information, low score for those positions, or other evidence (Altschul, 1998; Edgar and Sjölander, 2003a). Such columns should never be assigned to a match state—if they cannot be aligned to each other, they cannot be aligned to a profile position. We therefore require zero probability for such letters to be emitted by a match state, forcing unaligned columns to be assigned to insert states. It is advantageous to compress each unaligned region into a single column-like object that we call a *pillar*. We define a pillar to be either an aligned column or a maximal consecutive series of unaligned columns. Expressed in the form of pillars, an input alignment is transformed into a new data structure of equal or shorter length. The reduced length can significantly reduce memory use and speed due to the $O(L^2)$ complexity of our Viterbi algorithm. If no regions are marked as unaligned, this transformation has no effect. It is straightforward to develop recursion relations for pillars similar to those we have derived here for columns.

# TABLES

| *Reference set* | PP1 | | | PP2 | | |
|---|---|---|---|---|---|---|
| *Quality score* | $Q_{\mathrm{Dev}}$ | $Q_{\mathrm{Cline}}$ | $Q_{\mathrm{Mod}}$ | $Q_{\mathrm{Dev}}$ | $Q_{\mathrm{Cline}}$ | $Q_{\mathrm{Mod}}$ |
| *Algorithm* | | | | | | |
| COACH semi-global | 0.787 | 0.787 | 0.751 | 0.311 | 0.303 | 0.311 |
| prof_sim | 0.787 | 0.795 | 0.785 | 0.307 | 0.346 | 0.476 |
| COACH global | 0.780 | 0.779 | 0.779 | 0.288 | 0.278 | 0.301 |
| COACH local $S$=$S_{\mathrm{Div}}$ | 0.756 | 0.767 | 0.748 | 0.229 | 0.262 | 0.391 |
| COMPASS | 0.740 | 0.763 | 0.783 | 0.257 | 0.312 | 0.492 |
| COACH local $S$=$S_{\mathrm{avgMM}}$ | 0.667 | 0.712 | 0.796 | 0.171 | 0.234 | 0.580 |
| COACH local $S$=$S_{\mathrm{Null}}$ | 0.661 | 0.708 | 0.798 | 0.163 | 0.226 | 0.590 |
| PSI-BLAST | 0.733 | 0.758 | 0.768 | 0.231 | 0.265 | 0.383 |
| BLAST | 0.536 | 0.592 | 0.678 | 0.100 | 0.110 | 0.214 |

**Table 1. Alignment quality scores**

This table gives the mean developer, Cline and modeler scores for the tested algorithms on our two sets of reference alignments, PP1 and PP2. For comparison, PSI-BLAST (profile-sequence) and BLAST (sequence-sequence) scores are also shown. As expected, profile-profile methods generally show a small improvement over PSI-BLAST. In both reference sets, coverage is maximized (high developer score) by choosing COACH semi-global, errors are minimized (high modeler score) by COACH local $S$=$S_{\mathrm{Null}}$. We observe the expected trade-off between coverage and error produced by adjusting the null model self-loop probability $S$. Note, for example, that the low error rate of COACH local $S$=$S_{\mathrm{Null}}$ comes at the expense of lower coverage than PSI-BLAST.

| *Case* | *Column* | | *Leg* | | *Edge* | |
|---|---|---|---|---|---|---|
| | $i$–1 | $i$ | Q | R | q | r |
| 1 | X | X | $M_{k-1}$ | $M_k$ | $M_{k-1}$ | $M_k$ |
| 2 | – | X | $M_{k-1}$ | $M_k$ | $D_{k-1}$ | $M_k$ |
| 3 | X | – | $M_{k-1}$ | $M_k$ | $M_{k-1}$ | $D_k$ |
| 4 | – | – | $M_{k-1}$ | $M_k$ | $D_{k-1}$ | $D_k$ |
| 5 | X | X | $M_k$ | $I_k$ | $M_k$ | $I_k$ |
| 6 | – | X | $M_k$ | $I_k$ | $D_{k-1}$ | $I_k$ |
| 7 | X | – | $M_k$ | $I_k$ | *none* | |
| 8 | – | – | $M_k$ | $I_k$ | *none* | |
| 9 | X | X | $I_k$ | $I_k$ | $I_k$ | $I_k$ |
| 10 | – | X | $I_k$ | $I_k$ | ? | $I_k$ |
| 11 | X | – | $I_k$ | $I_k$ | *none* | |
| 12 | – | – | $I_k$ | $I_k$ | *none* | |
| 13 | X | X | $I_{k-1}$ | $M_k$ | $I_{k-1}$ | $M_k$ |
| 14 | – | X | $I_{k-1}$ | $M_k$ | ? | $M_k$ |
| 15 | X | – | $I_{k-1}$ | $M_k$ | $I_{k-1}$ | $D_k$ |
| 16 | – | – | $I_{k-1}$ | $M_k$ | ? | $D_k$ |
| 17 | * | X | $M_{k-1}$ | $D_k$ | $M_{k-1}$ | $D_k$ |
| 18 | * | – | $M_{k-1}$ | $D_k$ | $D_{k-1}$ | $D_k$ |
| 19 | * | X | $I_{k-1}$ | $D_k$ | $I_{k-1}$ | $D_k$ |
| 20 | * | – | $I_{k-1}$ | $D_k$ | ? | $D_k$ |
| 21 | * | X | $D_{k-1}$ | $M_k$ | $D_{k-1}$ | $M_k$ |
| 22 | * | – | $D_{k-1}$ | $M_k$ | $D_{k-1}$ | $D_k$ |
| 23 | * | X | $D_k$ | $I_k$ | $D_k$ | $I_k$ |
| 24 | * | – | $D_k$ | $I_k$ | *none* | |
| 25 | * | * | $D_{k-1}$ | $D_k$ | $D_{k-1}$ | $D_k$ |

**Table 2. Edge type implied by leg type and gaps.**

For each leg type QR, the table shows the edge type qr implied for a sequence given that it contains a letter (X) or gap (–) in the columns assigned to Q and R. A star (*) indicates that the edge can be deduced without knowing whether there is a gap. A question mark (?) indicates that the q state for the sequence cannot be deduced without further information; these cases arise when a gap assigned to an insert state. In cases marked *none*, no new edge is implied by the leg. Cases are numbered for reference in the text.

# FIGURES

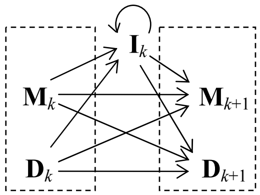

**Figure 1. Two consecutive nodes *k* and *k*+1 in a profile HMM**.
Letters represent states, arrows represent transitions. Match (M) and insert (I) states emit residues; delete (D) states are silent. Insert state $I_{k+1}$ and transitions out of $M_k$ and $D_k$ are not shown.

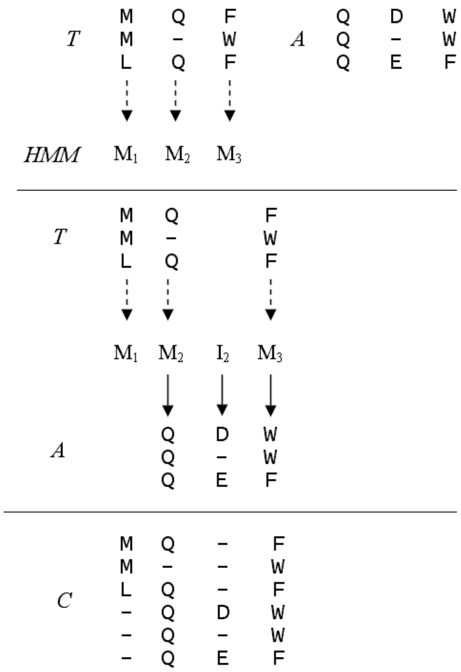

**Figure 2. Example alignment of two multiple alignments via an HMM.**
We start from two alignments, *A* and *T*. An HMM is estimated from *T* (top panel). In this example, we create a match state from each column (dotted arrows). *A* is then aligned to the HMM (middle panel) by assigning columns in *A* to emitter states in the model (solid arrows). The result is the output alignment *C* (bottom panel), in which columns of *A* and *T* are preserved intact. A column of gaps is inserted into *T* if one or more sequences in *A* visit an insert state; a column of gaps is inserted into *A* if all sequences visits a delete state. The first sequence in *A*, QDW, is a gapless sequence. It takes path $D_1 \rightarrow M_2 \rightarrow I_2 \rightarrow M_3$, which by definition is the route. The second sequence, Q–W, takes path $D_1 \rightarrow M_2 \rightarrow M_3$. The path taken by a sequence is uniquely determined given the route and the location of gaps in that sequence as it appears in *A*. An optimal alignment is determined by finding a route that maximizes the probability of *C*, which is computed by multiplying the probabilities of the paths for each sequence in *A*.

| | | | |
|---|---|---|---|
| Route | ? → | $I_k$ → | $I_k$ |
| Column | *i*–2 | *i*–1 | *i* |
| Seq1 | S | E | Q |
| Seq2 | S | – | Q |

**Figure 3. Gap in a column assigned to an insert state.**
In this example, two consecutive columns *i*–1,*i* in the input alignment are assigned to insert state $I_k$ in the model. Seq1 has letters in both columns; this means that Seq1 must take a self-loop in that insert state. Seq2 has a gap in the first of these columns, which implies that Seq2 makes one less visit to the insert state than Seq1. Columns *i*–1,*i* thus imply an edge $I_k \rightarrow I_k$ for Seq1. However, it is not possible to deduce the edge induced for Seq2 without looking further back in the path. If column *i*–2 is assigned to $I_k$, then a self-loop is also implied for Seq2. However, if column *i*–2 is assigned to $M_k$, then an $M_k \rightarrow I_k$ edge is implied. A similar situation arises whenever a gap appears in a column assigned to an insert state, and presents a difficulty in developing the recursion relations.

| | | | | | | |
|---|---|---|---|---|---|---|
| Route | $M_k$→ | $I_k$ → | $I_k$ → | | | |
| Last leg | | | | $I_k$ → | $I_k$ | |
| Column | C | | | *i*–1 | *i* | *Edge implied by last leg* |
| Seq1 | S | K | E | – | Q | $I_k \rightarrow I_k$ |
| Seq2 | S | – | K | – | Q | $I_k \rightarrow I_k$ |
| Seq3 | S | – | – | – | Q | $M_k \rightarrow I_k$ |
| Seq4 | – | – | – | – | Q | $D_k \rightarrow I_k$ |

**Figure 4. A route with sequences exhibiting case 10.**
For each sequence, the edge induced by the last leg is indicated.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Route | | $D_k$→ | $I_k$ → | $I_k$ → | | | |
| Last leg | | | | | $I_k$ → | $I_k$ | |
| Column | | | | C | *i*–1 | *i* | *Edge implied by last leg* |
| Seq1 | S | – | K | E | – | Q | $I_k \rightarrow I_k$ |
| Seq2 | S | – | – | K | – | Q | $I_k \rightarrow I_k$ |
| Seq3 | S | – | – | – | – | Q | $D_k \rightarrow I_k$ |
| Seq4 | – | – | – | – | – | Q | $D_k \rightarrow I_k$ |

**Figure 5. A route with sequences that exhibit case 10.**
This differs from Figure 4 in that the insert state is entered via the delete state rather than the match state.

## REFERENCES

Altschul,S.F., Gish,W., Miller,W., Myers,E.E., Lipman,D.J. (1990), Basic local alignment search tool, *J. Mol. Biol.* **215**, 403-410.

Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W., Lipman,D.J. (1997), Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucl. Acids Res.* **25**, 3389-3402.

Bellman,R (1957), *Dynamic Programming*, Princeton University Press, Boston.

Brenner,S.E., Chothia,C. and Hubbard,T.J.P. (1998), Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships, *Proc. Natl. Acad. Sci. USA* **95**, 6073-6078.

Cline,M. (2000), Protein sequence alignment reliability: prediction and measurement, PhD thesis, University of California Santa Cruz.

Cline,M., Hughey,R., and Karplus,K. (2002), Predicting reliable regions in protein sequence alignments, *Bioinformatics* **18**(2), 306-14.

Eddy,S.R. (1996), Hidden Markov models*, Curr. Opin. Struct. Biol*., **6**(3), 361-5.

Eddy,S.R. (2001), HMMER: Profile hidden Markov models for biological sequence analysis, http://hmmer.wustl.edu/.

Edgar,R.C. and Sjölander,K. (2003a), SATCHMO: simultaneous alignment and tree construction using hidden Markov models, *Bioinformatics* **19**(11), 1404-1411.

Edgar,R.C. and Sjölander,K. (2003b), A comparison of scoring functions for protein sequence profile alignment, *Bioinformatics* (to appear), preprint at: http://www.drive5.com/papers/profprofacc.pdf.

Gribskov,M., Homyak,M., Edenfield,J., Eisenberg,D. (1988), Profile scanning for three-dimensional structural patterns in protein sequences, *Comput. Appl. Biosci.* **4**(1), 61-66.

Henikoff,J.G., Greene,E.A., Pietrokovski,S. and Henikoff,S. (2000), Increased coverage of protein families with the blocks database servers, *Nucl. Acids Res.* **28**, 228-230.

Henikoff,S. and Henikoff,J.G. (1992), Amino acid substitution matrices from protein blocks, *Proc. Natl. Acad. Sci. USA* **89**, 10915–19.

Henikoff.S. and Henikoff,J.G. (1994), Position-based sequence weights, *J. Mol. Biol.* **243**(4), 574-8.

Holm,L. and Sander,C. (1996), Mapping the protein universe, *Science* **273**, 595-602.

Karplus,K., Barrett,C. and Hughey,R. (1998), Hidden Markov models for detecting remote protein homologies, *Bioinformatics* **14**(10), 846-56.

Krogh,A., Brown,M., Mian,I.S., Sjolander,K., and Haussler.D. (1994), Hidden Markov models in computational biology. Applications to protein modeling, *J. Mol. Biol.* **235**(5), 1501-31.

Kunin,V., Chan,B., Sitbon,E., Lithwick,G. and Pietrokovski,S. (2001), Consistency analysis of similarity between multiple alignments: prediction of protein function and fold structure from analysis of local sequence motifs, *J. Mol. Biol*. **307**(3), 939-949.

Lyngso,R.B., Pedersen,C.N., Nielsen,H. (1999), Metrics and similarity measures for hidden Markov models, *Proc. Int. Conf. Intell. Syst. Mol. Biol. 1999*, 178-86.

Panchenko,A.R., Finding weak similarities between proteins by sequence profile comparison (2003), *Nucl. Acids Res.* **31**(2), 683-689.

Panchenko,A.R., Marchler-Bauer,A. and Bryant,S.H. (2000), Combination of threading potentials and sequence profiles improves fold recognition, *J. Mol. Biol.* **296**, 1319-1331.

Pawlowski,K., Rychlewski,L., Zhang,B. and Godzik,A. (2001), Fold predictions for bacterial genomes, *J. Struct. Biol.* **134**, 219-231.

Pawlowski,K., Zhang,B., Rychlewski,L. and Godzik,A. (1999), The Helicobacter pylori genome: from sequence analysis to structural and functional predictions, *Proteins* **36**, 20-30.

Pearson,W.R. (1990), Rapid and sensitive sequence comparison with FASTP and FASTA, *Meth. Enzym.* **183**, 63-98.

Pietrokovski,S. (1996), Searching databases of conserved sequence regions by aligning protein multiple-alignments, *Nucl. Acids Res.* **24**(19), 3836-3845.

Pruitt,K.D., Tatusova,T., Maglott,D.R. (2003), NCBI Reference Sequence project: update and current status, *Nucl. Acids Res.* **31**(1), 34-37.

Rychlewski,L., Jaroszewski,L., Li,W., Godzik,A. (2000), Comparison of sequence profiles, strategies for structural predictions using sequence information, *Protein Sci*. **9**, 232-241.

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

Sadreyev,R. and Grishin,N. (2003), COMPASS: A tool for comparison of multiple protein alignments with assessment of statistical significance, *J. Mol. Biol.* **326**, 317-36.

Sauder,J.M., Arthur,J.W., Dunbrack,R.L. (2000), Large-scale comparison of protein sequence alignments with structure alignments, *Proteins* **40**, 6-22.

Shindyalov,I.N., Bourne,P.E. (1998), Protein structure alignment by incremental combinatorial extension (CE) of the optimal path, *Protein Eng.* **11**(9), 739-747.

Sjolander,K., Karplus,K., Brown,M., Hughey,R., Krogh,A., Mian,I.S. and Haussler,D. (1996), Dirichlet mixtures: a method for improving detection of weak but significant protein sequence homology, *Comput. Appl. Biosci.* **12**(4), 327-45.

Tatusov,R.L., Altschul,S.F. and Koonin,E.V. (1994), Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks, *Proc. Natl. Acad. Sci. USA* **91**, 12091-95.

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, *Nucl. Acids Res*., **22**, 4673-80.

Thompson,J.D., Plewniak,F. and Poch,O. (1999), A comprehensive comparison of multiple sequence alignment programs, *Nucl. Acids Res.*, **27**(13), 2682-90.

Viterbi,A.J. (1967), Error bounds for convolutional codes and an asymptotically optimal decoding algorithm, *IEEE Trans. Info. Theory*, IT-13, 260-269.

von Öhsen,N., Sommer,I. and Zimmer,R. (2003), Profile-profile alignment, a powerful tool for protein structure prediction, *Proc. Pacific Symp. Biocomp. 2003*, 252-63.

Yona,G. and Levitt,M. (2002), Within the twilight zone: a sensitive profile-profile comparison tool based on information theory, *J. Mol. Biol.* **315**, 1257-75.