Context-specific Methods for Sequence Homology Searching and Alignment

Andreas Biegert

München 2010

Dissertation zur Erlangung des Doktorgrades
der Fakultät für Chemie und Pharmazie
der Ludwig–Maximilians–Universität München

Context-specific Methods for Sequence Homology Searching and Alignment

Andreas Biegert
aus Bonn

2010

Erklärung:

Diese Dissertation wurde im Sinne von §13 Abs. 3 der Promotionsordnung vom 29. Januar 1998 von Herrn Professor Dr. Patrick Cramer betreut.

Ehrenwörtliche Versicherung:

Diese Dissertation wurde selbstständig, ohne unerlaubte Hilfe erarbeitet.

München, am 27. September 2010

Andreas Biegert

Dissertation eingereicht am: 28.09.2010

1. Gutachter: Prof. Dr. Patrick Cramer
2. Gutachter: Prof. Dr. Ulrike Gaul

Mündliche Prüfung am: 13.12.2010

Acknowledgements

I owe thanks to many people who supported and encouraged me throughout the last years during my thesis, and I can only try to acknowledge them here.

First and foremost, I would like to thank Dr. Johannes Söding for giving me the opportunity to work in his group and to contribute to many fascinating research projects. Moreover, I would like to thank Johannes for his constant support, friendly attitude, and helpful discussions. The past three years at the Gene Center have been an unforgettable experience - not least because of Johannes' sheer endless passion for science that has inspired me throughout my work.

I would like to thank Prof. Dr. Patrick Cramer for being my doctoral supervisor, and Prof. Dr. Ulrike Gaul for being my second PhD examiner. I am also very grateful to Prof. Dr. Dietmar Martin, Prof. Dr. Burkhard Rost, Prof. Dr. Klaus Förstemann, and Prof. Dr. Karl-Peter Hopfner for offering their time as members of my dissertation committee.

Furthermore, I would like to thank all present and former members of the Söding and Tresch group for their help, all the stimulating discussions, delicious cakes, and the enjoyable atmosphere. Thanks to Eckhart for always having an open ear for my LaTeX questions, thanks to Holger for his insightful comments on my figures, and thanks to Michael for reading parts of this thesis. Also thanks to Andi for keeping the compute-cluster, the lifeline of the computational biology, up and running.

Finally, I thank my parents for their generous and unwavering support throughout all these years. In particular, I would like to thank my girlfriend Claudia for her love and support in all situations of life, and forgiving me that I neglected her occasionally in the course of writing up this thesis.

Summary

Sequence alignment and database searching are essential tools in biology: This is because a protein's function can often be inferred from homologous proteins. Standard sequence comparison methods use substitution matrices to find the alignment with the best sum of similarity scores between aligned residues. These similarity scores do not take into account the local sequence context. In chapter 2 of this thesis, we present an approach that derives context-specific amino acid similarities from short windows centered on each query sequence residue. Our results demonstrate that the sequence context contains much more information about the expected mutations than just the residue itself: By employing our context-specific similarities (CS-BLAST) in combination with NCBI BLAST, we increase the sensitivity by up to two-fold on a difficult benchmark set, without loss of speed. Alignment quality is likewise improved significantly. Furthermore, we demonstrate considerable improvements when applying this paradigm to sequence profiles: Two iterations of CSI-BLAST, our context-specific version of PSI-BLAST, are more sensitive than five iterations of PSI-BLAST.

The concept of context-specificity for biological sequence comparison is very general. In chapter 3, we use sequence context to define column states, i.e. states for contexts with length $l = 1$, that describe profile columns optimally. In that way, we can encode a sequence profile as a sequence over an extended alphabet of column states while preserving most of the evolutionary information in the profile. We present a novel approach that uses such sequences of column states to derive a profile-to-profile score that can be efficiently computed at the same speed as standard sequence-to-sequence scores. By employing our fast profile-to-profile score in combination with the iterative HMM-HMM search tool HHBLITS, we significantly increase HHBLITS's sensitivity for detecting remote homologs, at no loss of speed.

Finally, we transfer the context-specific paradigm to the case of multiple alignment of noncoding DNA. This application is of particular interest because the low information content of noncoding sequences and the often weak overall conservation in these regions render alignments between related species difficult, while reliable alignments offer great promise to identify functional regions (such as *cis*-regulatory elements) through their inter-species conservation. In chapter 5, we show how to combine context-specific pseudocounts together with partial order graphs and profile HMMs to obtain a novel progressive alignment method CS-ALIGN. In a multiple alignment benchmark based on simulated promoter sequences, CS-ALIGN achieves considerable improvements with respect to binding site alignment, as well as overall alignment quality compared to the best current alignment programs.

Contents

List of Figures

List of Tables

Abbreviations

AMA ..... alignment metric accuracy

BLAST ..... basic local alignment search tool

BLOSUM ..... block substitution matrix

CRM ..... *cis*-regulatory module

CS-BLAST ... context-specific BLAST

CSI-BLAST ... context-specific iterated BLAST

DNA ..... deoxyribonucleic acid

EM ..... expectation maximization

HMM ..... hidden Markov model

MAC ..... maximum accuracy

MAMA ..... maximum alignment metric accuracy

MSA ..... multiple sequece alignment

NCBI ..... National Center for Biotechnology Information

PAM ..... point accepted mutations

PDB ..... protein data bank

POG ..... partial order graph

PPI ..... protein-protein interaction

PSI-BLAST ... position-specific iterated BLAST

PWM ..... positional weight matrix

ROC ..... rank order characteristic

SCOP ..... structural Classification of Proteins

TFBS ..... transcription factor binding site

UPGMA ..... unweighted pair group method with arithmetic mean

1. Introduction to homology searching and alignment

Protein sequences evolve from pre-existing ancestral sequences rather than being invented *ab initio*. By recognizing similarities between the sequence of an uncharacterized protein and sequences of proteins that are already characterized, it is therefore often possible to infer the structure or function of the new sequence from related proteins. Two sequences that are related through a common ancestor are said to be homologous. In computational sequence analysis the concept of homology has become a central theme because of the manifold applications of homology detection in areas such as protein function prediction, protein structure prediction, and protein evolution. The detection of homologous relationships between proteins has therefore become a routine procedure in investigating the function of new proteins. Likewise, biologists use inference from homology to explore fundamental molecular, cellular and developmental processes based on the study of simple and well studied model systems.

In sequence analysis, the alignment of two protein sequences is a way to identify regions of similarity that may be a consequence of evolutionary relationships between the sequences, which usually implies structural and often also functional similarity. The computation of alignments involves the insertion of gaps between residues such that residues thought to be homologous are placed in the same column. A common feature of virtually all alignment methods is the use of a scoring scheme which ideally assigns the highest score to the biologically most likely alignment. Since the accuracy of an alignment method often depends largely on the underlying scoring scheme, the development of more sensitive scoring schemes is of great importance in protein sequence analysis. The next sections introduce the issue of how to score alignments.

1.1. Scoring models and gap penalties

The objective of comparing sequences is to look for evidence that they have diverged from a common ancestor by mutation and selection. In the course of evolution, nature selects for favorable mutations, so that some sorts of changes may be seen more than others. The general approach for measuring how biologically meaningful an alignment is, is to compute a sum of terms for each aligned pair of residues in the alignment plus penalties for each gap. The intrinsic assumption of such an additive scoring theme is that mutations at

different sites in sequence have occurred independently. At least for protein sequences this assumption appears to be justified. Furthermore, it is reasonable to assume that conservative substitutions will contribute with a positive score whereas non-conservative substitutions will contribute with a negative score because non-conservative substitutions are expected to occur less frequently in real alignments. In bioinformatics such substitution scores $s(a,b)$ are commonly arranged in *substitution matrices* of dimension 20×20. In terms of statistics, one can describe each alignment term by the logarithm of the relative likelihood that the aligned residues are being related, as compared to being unrelated. In computational sequence analysis the former is often referred to as Match-model M and the latter as null- or Random-model R. In the Random model R each amino acid a occurs independently with frequency $f(a)$ and the Null model probability of an alignment of two residues a and b is simply the product of their background frequencies: $f(a)f(b)$. In the Match model M, two aligned residues a and b occur with the joint probability $p(a,b)$. Here, $p(a,b)$ is usually derived from the observed number of amino acid substitutions in a large set of reliable training alignments. An additive scoring scheme can then be described by a *log-odds-ratio*

$$s(a,b) = \log\left(\frac{P(a,b|M)}{P(a,b|R)}\right) = \log\left(\frac{p(a,b)}{f(a)f(b)}\right). \tag{1.1}$$

One of the first amino acid substitution matrices was the PAM (Point Accepted Mutation) matrix (Dayhoff et al., 1978). This substitution matrix estimates the substitution frequencies for 1% accepted mutations (PAM1) and extrapolates these to derive matrices as high as PAM250. However, it turned out that changes over evolutionarily long time scales are not very well approximated by this extrapolation method. This problem has been addressed by the BLOSUM matrix family (BLOck SUbstitution Matrix) developed by Henikoff and Henikoff (1992). Henikoff & Henikoff's approach is based on counting substitutions in ungapped regions in alignments of sequences (BLOCKS database) with a predefined maximal sequence similarity. For example, the widely used BLOSUM62 matrix, which is the matrix used by default in many alignment applications, is constructed from sequences with 62% or less sequence identity.

There are two approaches for penalizing gaps: first, a linear gap penalty for a gap of length g

$$\gamma(g) = -gd \tag{1.2}$$

or second, an affine score

$$\gamma(g) = -d - (g-1)e \tag{1.3}$$

with *gap-open* penalty d and *gap-extension* penalty e. Usually, e is set to something smaller than d to penalize long insertions or deletions less than in the simple linear model. This corresponds to the expectation that gaps should occur preferably in consecutive stretches as opposed to being dispersed.

1.2. Pairwise sequence alignment

Given a scoring scheme we now want to find an algorithm that efficiently computes an optimal alignment for a pair of sequences x and y with length n and m. For the case of global alignment, where sequences need to be aligned end to end, Needleman & Wunsch proposed a *dynamic programming* algorithm (Needleman and Wunsch, 1970) which was later revised by Gotoh to a more efficient version (Gotoh, 1982). The algorithm is guaranteed to find the optimal solution and proceeds by building an alignment using previous solutions for optimal alignments of smaller subsequences. To do so, it computes a $n \times m$ matrix S, where value $S(i,j)$ denotes the score of the best alignment between subsequences $x_{1...i}$ and $y_{1...j}$. Since in a pairwise alignment each column can either contain two aligned residues or a gap in the first sequence aligned to a residue in the second or vice versa, there are three ways in which an alignment can be extended up to positions (i,j) (see figure 1.1). This allows the recursive calculation of the best score up to positions (i,j):

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i,y_j), \\ S(i-1,j) - d, \\ S(i,j-1) - d. \end{cases} \tag{1.4}$$

Starting at the upper left corner of the matrix, the algorithm repeatedly applies the above formula to calculate each $S(i,j)$ value from the above-left, left, or above cell as depicted in the following figure.

$$\begin{array}{rcl} S(i-1,j-1) + s(x_i,y_j) & & S(i,j-1) - d \\ & \searrow \ \downarrow & \\ S(i-1,j) - d & \rightarrow & \mathbf{S(i,j)} \end{array}$$

After the matrix has been filled out in this way, the score of the best alignment is by definition the final value of $S(n,m)$. To find the alignment itself, one has to perform a traceback through the matrix and identify for each cell (i,j) which of the three choices contributed to its value. Since the traceback starts at $S(n,m)$, the alignment is actually built in reverse. In the *global alignment* case one assumes that there exists a biologically meaningful alignment between the *entire* sequences x and y. This assumption, however, is not always justified: for example when two proteins share a common domain but are unrelated in their remaining parts. *Local alignment* algorithms address this problem by looking for the best alignment between *subsequences* of x and y. Furthermore, local alignment is usually the most sensitive

I	G	A	x_i		A	I	G	A	x_i		G	A	x_i	-	-
L	G	V	y_j		G	V	y_j	-	-		S	L	G	V	y_j

Figure 1.1.: Three ways of how an alignment can be extended up to positions (i,j)

method for comparing highly diverged sequences, even if they may possibly share a common evolutionary origin over their entire length. The algorithm that computes the optimal local alignment for a pair of sequences, the so called Smith-Waterman algorithm (Smith and Waterman, 1981), is closely related to the Needleman-Wunsch algorithm for global alignments. With the difference that there is an additional possibility in the maximization equation which corresponds to the start of a new alignment. It allows $S(i,j)$ to become 0 if all other options have a value less than 0:

$$S(i,j) = \max \begin{cases} 0, \\ S(i-1,j) + s(x_i,y_j), \\ S(i-1,j) - d, \\ S(i,j-1) - d. \end{cases} \tag{1.5}$$

The second change is that a local alignment does not have to end in the bottom right corner. Therefore the traceback procedure is started from the cell with the highest value $S(i,j)$ instead of $S(n,m)$.

The Needleman-Wunsch and Smith-Waterman implementations, as described above, use the linear gap cost structure as in equation (1.2). In order to assume the affine gap cost as in equation (1.3), one has to keep track of three instead of only one value per pair of residue coefficients (i,j). However, the general concept of dynamic programming and runtime complexity of the algorithm stay the same. Further details regarding alignment with affine gap costs are discussed in (Durbin et al., 1998).

Pairwise alignments are used to detect homologs between different proteins sequences either as global or local alignments. From the above definitions it is easy to see that using the Needleman-Wunsch or Smith-Waterman algorithms this can be solved in time proportional to the product of the length of the two sequences being compared: $O(nm)$. However, this is too slow for searching current databases, and in practice algorithms are used that are much faster. This gain in speed comes at the cost of possibly missing significant alignments due to the heuristics employed.

1.3. Profile-sequence comparison

In many fields of bioinformatics such as 3D protein structure prediction, protein function prediction, and protein evolution, the identification of very strongly diverged homologs can be of great help. Frequently, however, sequence-sequence comparison methods are not sensitive enough to identify such remote homologies. Extending the limits of sensitivity is therefore of great practical importance.

The development of profile-sequence comparison methods such as PSI-BLAST (Altschul et al., 1997) led to a great improvement in sensitivity over the sequence-sequence comparison methods. The idea is to better exploit evolutionary information about the sequence family

of the query by building an alignment of homologous sequences. From the alignment one can then deduce a sequence profile which is simply a $20 \times n$ matrix recording the amino acid frequencies for each position in the alignment (including pseudocounts). A profile draws its power from the ability to distinguish between conserved residues that are important for defining members of the family, and residues that vary among the members of the family. More than that, it holds precise information of the boundaries of important motifs in the family. The alignment of a profile with a multiple sequence alignment embodied by a profile is almost completely analogous to the alignment of two simple sequences. The only real difference is that the substitution score of two positions i, j is now given by the profile itself, rather than with reference to a substitution matrix.

1.4. BLAST and PSI-BLAST

The first release of BLAST (Basic Local Alignment Search Tool) (Altschul et al., 1990) employed a *seed and extend* heuristic in which first small exact matches are found which are then extended to obtain longer inexact ones possibly containing gaps. The algorithm proceeds by first generating a list of all k-mers (usually $k = 3$ for proteins) that have similarity $\geq T$ to some k-mer in the query. Then, it scans the database for all occurrences of these k-mers in the database and extends each such seed until its score drops a certain distance below the best score computed so far for the seed. Finally, all hits with a score greater than a certain minimum match score are reported. However, with respect to sensitivity and speed, this approach involves a dilemma: short k-mers are required to be sensitive, but at the same time increase the likelihood of chance matches, which in turn triggers numerous unneeded extensions. To mitigate this problem, BLAST 2.0 and PSI-BLAST (Altschul et al., 1997) use a 2-hit strategy with a remarkably improved specificity. The 2-hit approach triggers extensions for consecutive matches of two similar short words on the same diagonal within a window of 40 residues. By lowering the k-mer word similarity threshold, this is more sensitive than the 1-hit approach. At the same time, it significantly reduces the number of extensions for chance matches. The extension method is based on the Smith-Waterman algorithm (Smith and Waterman, 1981) but is restricted to a proximate region by a maximum score drop-off value.

PSI-BLAST is the most popular tool for profile-based detection of distant protein relationships. It performs consecutive BLAST searches to iteratively build up a profile of homologous sequences. The search strategy of PSI-BLAST is illustrated in Figure 1.2. In the first iteration, the profile is derived from the mutation probabilities in the BLOSUM substitution matrix. At the beginning of the second and following iterations, the profile is refined with hits from the previous round whose E-values are better than a predefined inclusion threshold. The resulting profile is often much more sensitive to detect remote homologies than standard substitution matrix mutation probabilities.

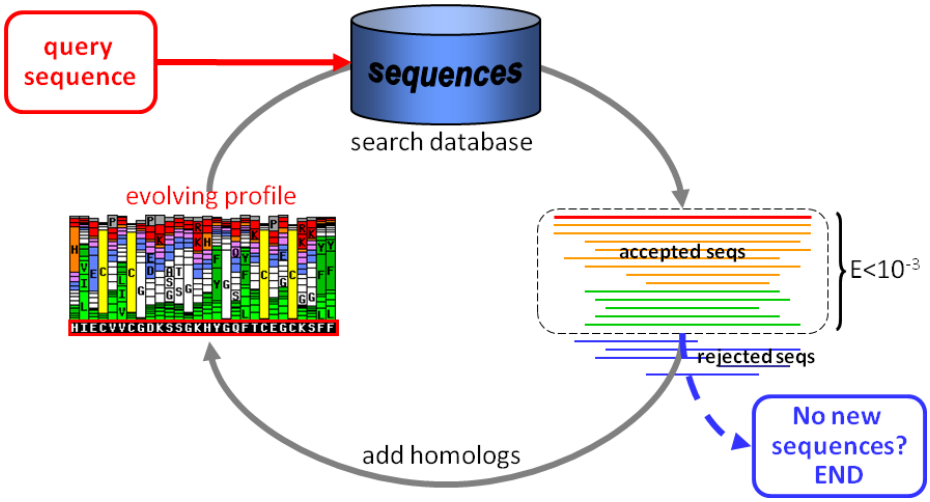


Figure 1.2.: Iterative homology search strategy performed by PSI-BLAST. In the first search iteration, the profile is derived from the mutation probabilities in the BLOSUM substitution matrix. At the beginning of the second and following iterations, the profile is refined with hits from the previous round. The resulting profile is often much more sensitive and more likely to detect remote homologies than standard substitution matrix mutation probabilities.

1.5. HMM-HMM alignment

HMM-HMM comparison as described by Söding (2005) has proven to be the most powerful method for remote homology detection in proteins. Profile HMMs are similar to the above described simple sequence profiles, but in addition to the amino acid frequencies in the columns of a multiple sequence alignment they contain the position-specific probabilities for inserts and deletions along the alignment. Thus, profile HMMs penalize chance hits much more than true positives which tend to contain insertions and deletions at the same positions as the sequence from which the HMM was build. In the following, we show how to generalize the log-odds score to the case of pairwise comparison of two HMMs and explain the method used to efficiently compute a pairwise alignment between two profile HMMs.

1.5.1. Log-sum-of-odds score

The log-odds score for sequence-HMM comparison measures how more probable it is that a sequence is emitted by an HMM rather than by a random Null model. More specifically, it can be written as

$$S_{\mathrm{LO}} = \log \frac{P(x_1,\ldots,x_L|\text{emission on path})}{P(x_1,\ldots,x_L|\text{Null})} \,. \tag{1.6}$$

We would like to to generalize this log-odds score to the case of HMM-HMM comparison, where an alignment between two profile HMMs corresponds to a certain path through the

two HMMs. The so called log-sum-of-odds score can be seen as such a generalization:

$$S_{\mathrm{LSO}} = \log \sum_{x_1,\ldots,x_L} \frac{P(x_1,\ldots,x_L|\text{co-emission on path})}{P(x_1,\ldots,x_L|\text{Null})} \,. \tag{1.7}$$

Here, the sum over $x_1,\ldots,x_L$ runs over all sequences with L residues. It is easy to see that the log-sum-of-odds score is in fact a generalization of the log-odds score because in the case of one of the HMMs representing only one sequence, only one term in the sum can contribute and equation (1.7) can be reduced to equation (1.6). In order to apply standard dynamic programming algorithms (e.g. the Viterbi algorithm) to find the HMM-HMM alignment with the maximum log-sum-of-odds score, we need some kind of similarity measure that allows us to compare the amino acid distributions of columns i and j in the two HMMs. Rewriting the log-sum-of-odds score in terms of HMM probabilities yields

$$S_{\mathrm{SLO}} = \sum_{k:X_kY_k=MM} S_{\mathrm{aa}}(q_{i(k)},p_{j(k)}) + \log \mathcal{P}_{tr} \tag{1.8}$$

where $\mathcal{P}_{tr}$ is the product of all transition probabilities for the path through q and p. $S_{\mathrm{aa}}(q_i,p_j)$ is the column score given by

$$S_{\mathrm{aa}}(q_i,p_j) = \log \sum_{a=1}^{20} \frac{q_i(a)p_j(a)}{f(a)}, \tag{1.9}$$

in which $q_i(a), p_j(a)$ denote the emission probability of amino acid a in columns i and j of the HMMs and $f(a)$ is the fixed amino acid background frequency. In this respect, the division by $f(a)$ can be seen as a weighting factor that increases the weighting of the rare amino acids compared to the more common ones. Furthermore, the column score is positive when the two amino acid distributions are similar, and negative otherwise. This property is important for local alignment.

1.5.2. Pairwise alignment of HMMs

Figure 1.3A shows a profile HMM with each column containing a match state M, a delete state D and an insert state I. In contrast to delete states, only match states and insert states emit amino acids. This implies that a match state can only be aligned with a match state (MM) or an insert state (MI) in the other HMM. A delete state, on the other hand, can only be aligned to a gap (DG). Thus, there are five possible pair states MM, MI, IM, DG, and GD. The interpretation of gaps in an alignment of HMMs is completely analogous to the case of sequence-sequence alignment. It denotes that for the column of the HMM that is aligned to the gap, there exists no column in the other alignment/HMM that evolved from the same residue in the ancestral alignment. In order to calculate the path with the maximal log-sum-of-odds score according to equation (1.7), one can apply a variant of the

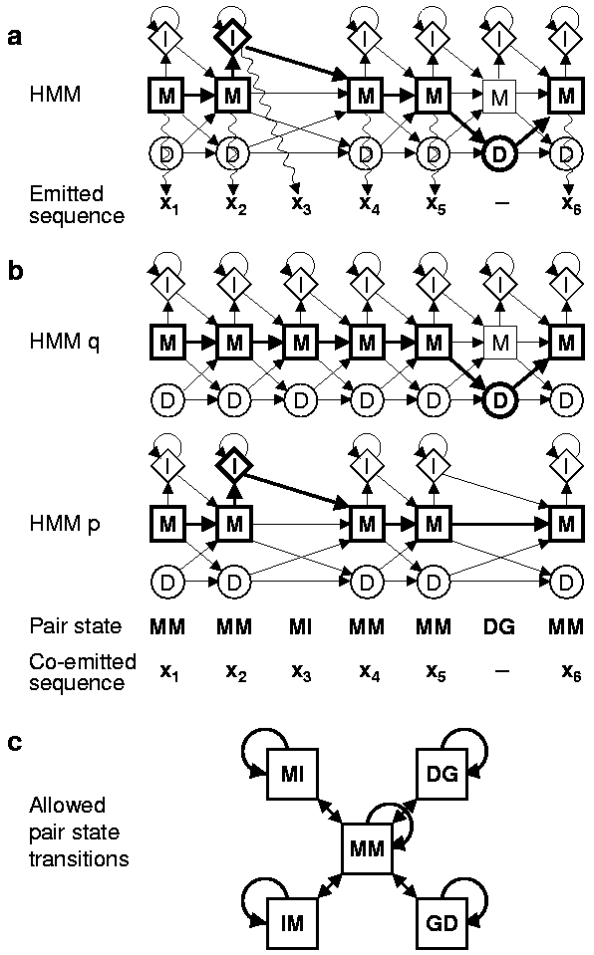


Figure 1.3: (**A**) The alignment of a sequence to a profile HMM can be represented by a path through the HMM (bold arrows). (**B**) Alignment of two HMMs by maximization of log-sum-of-odds score. The path through the two HMMs corresponds to a sequence that is co-emitted by both HMMs. (**C**) Allowed transitions between pairs states. (figure taken from Söding (2005))

Viterbi algorithm that maintains five dynamical programming matrices S_{XY}, one for each pair state $XY \in \{MM, MI, IM, DG, GD\}$. Similar to the Needleman-Wunsch algorithm, these matrices can be calculated recursively by

$$S_{MM}(i,j) = S_{\mathrm{aa}}(q_i,p_j) + \max \begin{cases} S_{MM}(i-1,j-1) + \log\left(q_{i-1}(M,M)\, p_{j-1}(M,M)\right) \\ S_{MI}(i-1,j-1) + \log\left(q_{i-1}(M,M)\, p_{j-1}(I,M)\right) \\ S_{IM}(i-1,j-1) + \log\left(q_{i-1}(I,M)\, p_{j-1}(M,M)\right) \\ S_{DG}(i-1,j-1) + \log\left(q_{i-1}(D,M)\, p_{j-1}(M,M)\right) \\ S_{GD}(i-1,j-1) + \log\left(q_{i-1}(M,M)\, p_{j-1}(D,M)\right) \end{cases} \tag{1.10}$$

$$S_{MI}(i,j) = \max \begin{cases} S_{MM}(i-1,j) + \log\left(q_{i-1}(M,M)\, p_j(M,I)\right) \\ S_{MI}(i-1,j) + \log\left(q_{i-1}(M,M)\, p_j(I,I)\right) \end{cases} \tag{1.11}$$

$$S_{DG}(i,j) = \max \begin{cases} S_{MM}(i-1,j) + \log\left(q_{i-1}(M,D)\right) \\ S_{DG}(i-1,j) + \log\left(q_{i-1}(D,D)\right) \end{cases} \tag{1.12}$$

where $q_i(X,X')$ and $p_j(Y,Y')$ denote the transition probabilities to go from state X or $Y \in \{M,I,D\}$ in column i or j to a state X' or $Y' \in \{M,I,D\}$. Equation (1.10) shows

the maximization required for global alignment. For local alignment a zero needs to be added as sixth case of the maximization. The optimal alignment is constructed as usual by backtracking from the cell with maximum score.

Part I.

Homology searching

2. Sequence context-specific profiles for homology searching

2.1. Introduction

Substitution matrices quantify the similarity between amino acids or nucleotides (Dayhoff et al., 1978; Gonnet et al., 1992; Henikoff and Henikoff, 1992). As a mainstay of biological sequence comparison, they are at the heart of standard alignment methods such as the Needleman-Wunsch and Smith-Waterman algorithms (Needleman and Wunsch, 1970; Smith and Waterman, 1981), which find the alignment with the maximum sum of similarity scores between aligned residues or bases. Sequence-search programs such as BLAST and FASTA (Altschul et al., 1990; Pearson, 1991) use substitution matrices to score short seeds and final alignments, multiple alignment programs such as CLUSTALW (Thompson et al., 1994) employ them in sum-of-pairs scoring to quantify the similarity between aligned sequence-profile columns, and in sequence profile-based methods such as PSI-BLAST (Altschul et al., 1997) or HHSEARCH (Söding, 2005) they are used for calculating pseudocounts (Henikoff and Henikoff, 1996; Tatusov et al., 1994).

For proteins, the importance of substitution matrices to identify homologs and calculate accurate alignments has stimulated various advances: Yu *et al.* have developed a rationale for compositional adjustment of amino acid substitution matrices by transforming the background frequencies implicit in a substitution matrix to frequencies appropriate for the comparison of protein sequences with nonstandard global amino acid composition (Yu et al., 2003). Others have derived specialized transmembrane substitution matrices from alignments of experimentally verified or predicted transmembrane segments to improve alignments of sequences with transmembrane regions (Jones et al., 1994; Mueller et al., 2001; Ng et al., 2000). The logic is that the structural environment of an amino acid residue partly influences into what amino acids it is likely to mutate.

Taking this idea a step further, so-called structure-dependent substitution matrices (see Figure 2.1) have been trained for a number of environments, defined by a combination of secondary structure state, solvent accessibility class, environmental polarity class and/or hydrogen bonding (Goonesekere and Lee, 2008; Overington et al., 1992; Rice and Eisenberg, 1997; Shi et al., 2001). EVDTREE (Gelly et al., 2005) also computes structure-dependent substitution scores, but the selected structural descriptors depend on residue types. All these structural environment-dependent matrices allow for the detection of more remotely

homologous proteins than standard substitution matrices. However, their application is limited by the need to know the structure of one of the proteins to be compared.

In contrast, sequence context-dependent methods do not rely on 3D structure information to define local environments. They describe the environment of a residue by the sequence surrounding it. Jung and Lee (2000) trained 400 × 400 substitution matrices for contexts consisting of pairs of residues up to four positions apart and obtained a 30% increase in sensitivity on a set of 107 proteins, although this result could not be confirmed in a large-scale study (Crooks et al., 2005). Gambin *et al.* derived 400 amino acid substitution matrices, one for each context consisting of the two residues neighboring the central residue (Gambin et al., 2002; Gambin and Otto, 2005). PHYBAL (Baussand et al., 2007) models the selective pressure inside and outside of hydrophobic blocks by two different substitution matrices and two different sets of gap penalties.

Huang *et al.* took a decisive step forward, employing 281 substitution matrices for 281 states of a hidden Markov model trained on sequences of known structure. Each HMM state represents a single profile column. Context information is encoded essentially in the transition probabilities between the states. By mixing mutation probabilities from the substitution matrices weighted by posterior probabilities, HMMSUM achieved considerable improvements in alignment quality when compared to standard substitution matrices (Huang

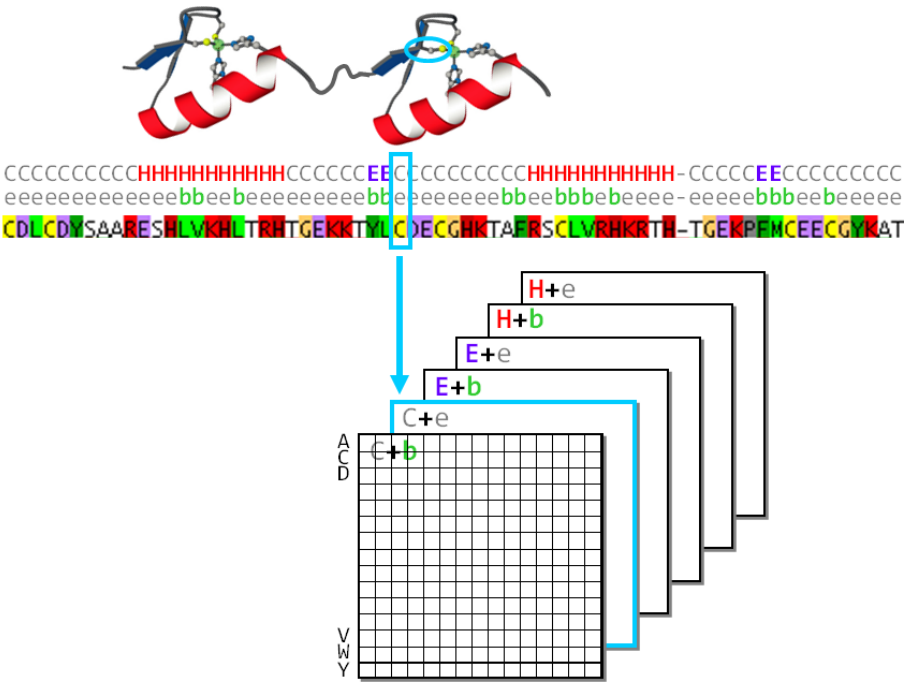


Figure 2.1.: Structure-dependent substitution matrices. For each combination of secondary structure state and solvent accessibility class, a separate substitution matrix is trained. The structural environment-dependent matrices allow for the detection of more remotely homologous proteins than standard substitution matrices.

and Bystroff, 2006). We expect such sequence contexts to predict mutation probabilities better than structural environments, since very different sequences with very specific amino acid preferences can adopt similar local structures (Han and Baker, 1996). When all these different sequences are pooled into the same structural environment, the specific amino acid preferences are lost.

In this work, we present a method that derives sequence context-specific amino acid similarities from 13-residue windows centered on each residue. We predict the expected mutation probabilities for each position by comparing its sequence window to a library with thousands of *context profiles*, generated by clustering a large, representative set of sequence-profile windows. The mutation probabilities are obtained by weighted mixing of the central columns of the most similar context profiles (see Figure 2.3B). Whereas iterative profile search tools such as PSI-BLAST align homologous, long sequence matches to the query with weights independent of the match quality, our method aligns mostly non-homologous, ungapped, short profiles, giving higher weights to better matching profiles. In contrast to HMMSUM, no substitution matrices are needed. Also, the context information is encoded explicitly in the context profiles with no need for transition probabilities. This leads to a simpler computation and to a much better runtime that scales linearly instead of quadratically with the number of states/contexts (see section 2.4). The context library can therefore be many times larger and hence finer-grained than in HMMSUM, enabling us to describe contexts such as "a large aliphatic residue with preference for leucine or isoleucine on the hydrophobic face of an amphipathic helix", zinc-finger, transmembrane helix, coiled coil, exposed β-sheet, GD-box, disordered region, and collagen, for example (see Figure 2.2).

A crucial insight for achieving speeds comparable to substitution matrix-based methods such as BLAST is this: Sequence-to-sequence comparison using a substitution matrix is exactly equivalent to profile-to-sequence comparison, if the sequence-profile is calculated from one of the sequences using full substitution matrix pseudocounts. Hence we can employ profile-based methods, which have similar speeds as their sequence-based counterparts, to implement sequence context-specific amino acid similarities.

CS-BLAST, our context-specific version of BLAST, works in the following way: We generate a sequence profile for the query sequence using context-specific pseudocounts and then jump-start NCBI's profile-to-sequence search method PSI-BLAST with this profile. We demonstrate that, on a difficult benchmark set, sequence searches with our new context-specific amino acid similarities are more than twice as sensitive as BLAST with the standard BLOSUM62 substitution matrix, produce higher quality alignments, and generate reliable E-values, all without loss of speed.

Finally, we apply the new paradigm to profile-to-sequence comparison by calculating context-specific pseudocounts for sequence profiles. The only difference to the previously described sequence-based scheme is that we now compare sequence-*profile* windows to our

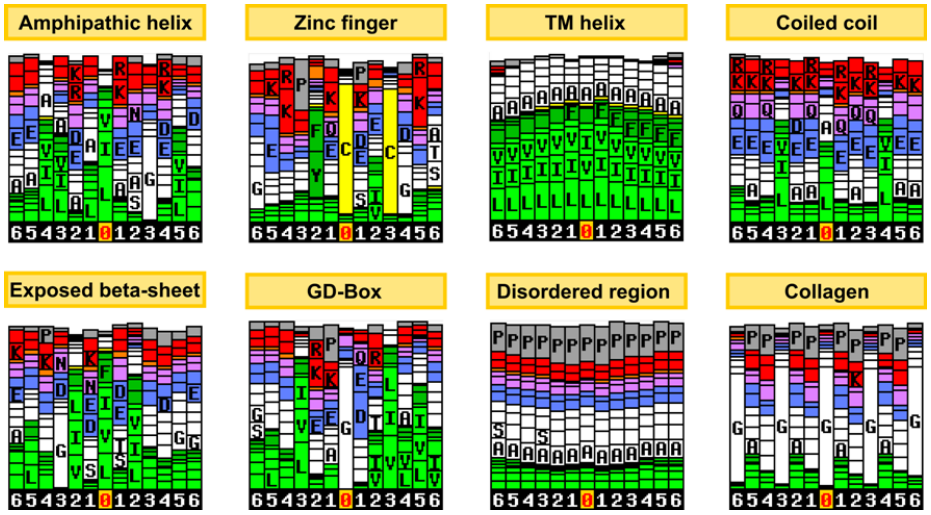


Figure 2.2.: Selected context profiles commonly found in proteins. Among many others, the context library is able to describe contexts such as "a large aliphatic residue with preference for leucine or isoleucine on the hydrophobic face of an amphipathic helix", zinc-finger, transmembrane helix, coiled coil, exposed β-sheet, GD-box, disordered region, and collagen. (left to right, top to bottom).

library of context profiles. In contrast to substitution matrix and Dirichlet pseudocounts (Durbin et al., 1998; Henikoff and Henikoff, 1996; Sjoelander et al., 1996; Tatusov et al., 1994), these pseudocounts do not depend only on the single profile column but on the entire sequence context of the profile column. We report considerable improvements of this context-specific scheme (CSI-BLAST) over PSI-BLAST.

2.2. Material and Methods

2.2.1. Context-specific mutation probabilities

We first show that amino acid substitution scores are directly related to pairwise amino acid mutation probabilities and sequence-profile pseudocounts. We can therefore derive sequence context-specific amino acid similarity scores from context-specific mutation probabilities. These mutation probabilities can be predicted with a probabilistic model using a large library of sequence profile windows representing very specific local sequence contexts.

Any matrix of substitution scores $S(x,y)$ describing the similarity between amino acids x and y can be written in the form (Altschul, 1991) $S(x,y) = \text{const} \times \log[P(x,y)/P(x)P(y)]$, where $P(x,y)$ is the probability that x and y occur aligned to each other in an alignment of homologous sequences, and $P(x)$ and $P(y)$ are the *background probabilities* of x and y to occur in representative sequences (whether aligned or unaligned). This can also be written as a log odds score, $S(x,y) = \log[P(y|x)/P(y)]$, where $P(y|x) = P(x,y)/P(x)$ is the conditional probability of y given x, i.e. the probability for amino acid x to mutate into y.

If y occurs more often in positions aligned with an x (described by $P(y|x)$) than what would be expected by chance (described by $P(y)$), then the score is positive, otherwise negative.

We next explore the connection of mutation probabilities $P(y|x)$ with sequence-profile pseudocounts. A *sequence profile* is a matrix $p(i,y)$ that succinctly represents a multiple alignment of homologous sequences. $p(i,y)$ is the frequency of amino acid y in column i of the multiple alignment. The profile describes what amino acids are likely to occur in related sequences at each position, or, in other words, the probability of a residue at position i to mutate into amino acid y. A single sequence (x_i) can be turned into a sequence profile by adding artificial mutations (i.e., *pseudocounts*) with the method of substitution matrix pseudocounts (Henikoff and Henikoff, 1996; Tatusov et al., 1994): $p(i,y) = P(y|x_i)$. Here, $P(y|x_i)$ are the conditional probabilities giving rise to substitution matrix $S(x,y)$. The profile-to-sequence score of column i of this single-sequence profile p with residue y_j of a second sequence (y_j) is

$$S\left(p(i,\cdot),y_j\right) := \log \frac{p(i,y_j)}{P(y_j)} = \log \frac{P(y_j|x_i)}{P(y_j)} = S(x_i,y_j). \tag{2.1}$$

Hence, substitution matrix scores can be seen as a special case of profile-to-sequence scores, where the profile is generated from one of the sequences using substitution matrix pseudocounts.

Figure 2.3A illustrates the equivalence of sequence-to-sequence and profile-to-sequences scoring with the alignment matrix of two zinc-finger sequences (x_i) and (y_j). The query profile resulting from the artificial mutations is illustrated as a histogram, in which the bar heights are proportional to the corresponding amino acid probabilities $p(i,y)$. The score of each matrix cell (i,j) can be interpreted in two ways: either as sequence-to-sequence score $S(x_i,y_j)$ between residues x_i and y_j, or as profile-to-sequence score $S(p(i,\cdot),y_j)$ between profile column $p(i,\cdot)$ and residue y_j.

In the above schemes, the expected mutation probabilities $P(y|x_i)$ at position i depend only on the single amino acid x_i. However, the sequence context X_i, defined below, contains much more information than just residue x_i itself about what amino acids to expect in related sequences. If we were able to calculate a context-specific mutation probability $P(y|X_i)$, we could define a score in a way analogous to equation (2.1), but using a *context-specific* profile $p_{\mathrm{cs}}(i,y) = P(y|X_i)$ instead of $P(y|x_i)$.

The context X_i is defined as the window of l residues surrounding x_i, i.e. $X_i = (x_{i-d},\ldots,x_{i+d})$ with $l = 2d + 1$. To predict the mutation probabilities for each position i, we compare its sequence window X_i to a precomputed library of K context profiles, $p_1,\ldots,p_K$, each of length l. The context-specific mutation probability $P(y|X_i)$, i.e. the probability of observing amino acid y in a homologous sequence given context X_i, will be calculated by a weighted mixing of the amino acids in the central columns of the most similar context profiles (Figure 2.3B). To derive the weight of each profile p_k, we first need the probabil-

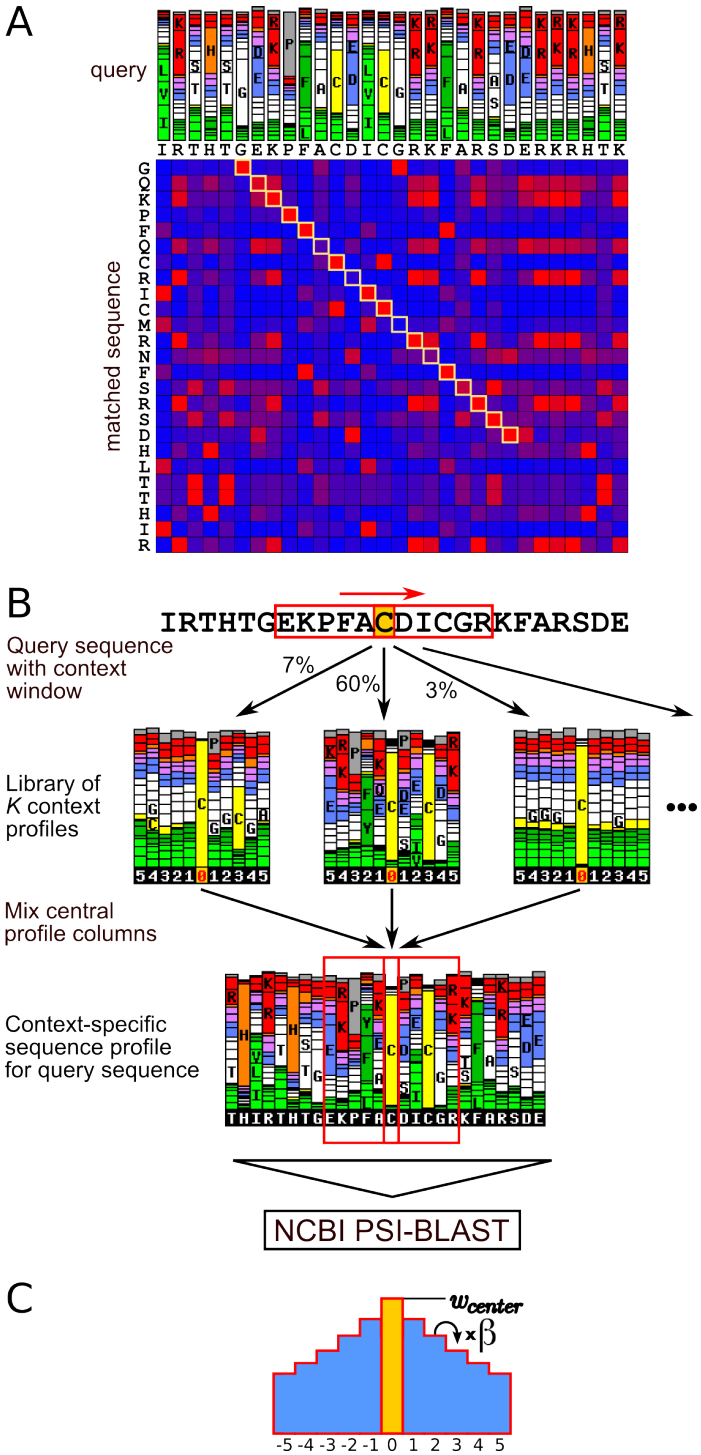


Figure 2.3: Method of context-specific sequence comparison. (**A**) Sequence search/alignment algorithms find the path that maximizes the sum of similarity scores (color-coded blue to red). Substitution matrix scores are equivalent to profile scores if the sequence profile (colored histogram) is generated from the query sequence by adding artificial mutations with the substitution matrix pseudocount scheme. Histogram bar heights represent the fraction of amino acids in profile columns. (**B**) Computation of context-specific pseudocounts: The expected mutations (i.e., pseudocounts) for a residue (highlighted in yellow) are calculated based on the sequence context around it (red box). Library profiles contribute to the context-specific sequence profile with weights determined by their similarity to the sequence context (see percentages). The resulting profile can be used to jump-start PSI-BLAST, which will then perform a sequence-to-sequence search with context-specific amino acid similarities. (**C**) Positional window weights decrease exponentially with the distance to the center position to model the decreasing information value of farther positions for the central profile column.

ity $P(X_i|p_k)$ that the sequence window X_i is *emitted* by profile p_k, which is equal to the product of probabilities of x_{i+j} ($j \in \{-d,\ldots,d\}$) being emitted by profile column $p_k(j,\cdot)$: $P(X_i|p_k) = \prod_{j=-d}^{d} p_k(j,x_{i+j})$. Since the inner positions in the window will be the most informative to predict the amino acid distribution for the central residue, we can refine the above formula by defining coefficients w_j which weight the contribution of each window position to the probability: $P(X_i|p_k) \propto \prod_{j=-d}^{d} p_k(j,x_{i+j})^{w_j}$. The values of w_j are parametrized by w_{center} and β (see Figure 2.3C). In the calculation of $P(X_i|p_k)$ for positions i within d residues of either end of the query sequence, overhanging context profile columns are simply ignored.

Next, we need to know the probability $P(p_k|X_i)$ that profile p_k was the one that emitted X_i. Using Bayes' theorem, we find

$$P(p_k|X_i) = \frac{P(X_i|p_k)P(p_k)}{P(X_i)} \propto P(p_k) \prod_{j=-d}^{d} p_k(j,x_{i+j})^{w_j}. \tag{2.2}$$

$P(p_k)$ is the Bayesian *prior probability* for profile p_k, determined in the process of computing the profile library. It quantifies the probability that a sequence window is emitted by profile p_k *prior* to knowing that sequence window. $P(X_i) = \sum_k P(X_i|p_k)P(p_k)$ is a normalization constant.

We can now calculate the context-specific mutation probabilities $P(y|X_i)$ by mixing the amino acid distributions $p_k(0,y)$ from the central columns of all K profiles with weights $P(p_k|X_i)$:

$$P(y|X_i) \propto \sum_{k=1}^{K} p_k(0,y)P(p_k|X_i). \tag{2.3}$$

Normalizing over all 20 amino acids yields the final expected mutation probability $P(y|X_i)$. To have more flexibility in adjusting the diversity of the context-specific profile $p_{\text{cs}}(i,\cdot)$, we mutate only a fraction $\tau \in [0,1]$ of (x_i) while leaving a fraction $1-\tau$ unchanged:

$$p_{\text{cs}}(i,y) = (1-\tau)\delta_{x_i,y} + \tau P(y|X_i). \tag{2.4}$$

Here, $\delta_{x_i,y} = 1$ if $x_i = y$ and 0 otherwise. In principle, τ needs to be optimized depending on the evolutionary distance over which homologous sequences are to be found, in a similar way as the substitution matrix with optimum diversity might be chosen. In practice we have found that, as in substitution matrices, a single diversity works well for the entire range of evolutionary distances.

Figure 2.3B illustrates the calculation of expected mutation probabilities $P(y|X_i)$ for a cysteine residue (highlighted in yellow) at position i belonging to a zinc-finger motif. Three profiles similar to the sequence window X_i (red box) are shown, whose central columns contribute to the context-specific sequence profile $p(i,y) = P(y|X_i)$ at position i with weights $P(p_k|X_i)$ of 7%, 60%, and 3%, respectively. With the resulting profile (bottom), a profile-to-

sequence search can be performed, e.g. using PSI-BLAST, which is equivalent to a sequence search with context-specific amino acid similarity scores (equation 2.1). In this example, the context-specific scheme recognizes the sequence context of the cysteine and correctly assigns a zinc-finger profile a high weight, resulting in a highly conserved cysteine.

The context-specificity paradigm is not restricted to sequences but applies equally well to sequence profiles or profile hidden Markov models (HMMs). It can therefore be used in profile-to-sequence (Altschul et al., 1997; Eddy, 1998; Gribskov et al., 1987) and profile-profile (Madera, 2008; Rychlewski et al., 2000; Sadreyev and Grishin, 2003; Söding, 2005; Thompson et al., 1994; Yona and Levitt, 2002) comparison, for example.

Our method CS-BLAST for context-specific protein sequence searching is a simple extension of BLAST: First, a context-specific sequence profile is generated for the query sequence using a library of context profiles. This step is very fast. Then PSI-BLAST is jump-started with this profile. PSI-BLAST is extended to the context-specific case in an analogous way (CSI-BLAST).

2.2.2. Generalization to sequence profiles

To apply the paradigm to sequence profiles and profile HMMs, we show how to generalize the calculation of pseudocounts from the single sequence case in equation (2.3) to the case of sequence alignments, from which the profile is derived. In analogy to the sequence context X_i, we define the context of the query alignment at position i as $Q_i = (c_q(i\text{–}d,\cdot),\ldots,c_q(i\text{+}d,\cdot))$, where $c_q(j,x)$ are the counts of amino acid x at position j of the query alignment. These counts are obtained from the sequence profile $q(j,x)$ by multiplying with the effective number of sequences $N_q(j)$ at position j in the query alignment: $c_q(j,x) = N_q(j)q(j,x)$ (see section 2.2.3 for details). We now merely need to show how to generalize $P(X_i|p_k)$ to $P(Q_i|p_k)$, since all other transformations leading to equation (2.3) remain essentially unchanged. To derive $P(Q_i|p_k)$, we model the amino acid counts $c_q(i)$ with multinomial distributions. Since $N_q(j)$ can be real-valued, however, we replace the factorials in the multinomial distribution by Gamma functions ($n! = \Gamma(n+1)$)

$$P(Q_i|p_k) = \prod_{j=-d}^{d} \left(\frac{\Gamma(N_q(i+j)+1)}{\prod_{x=1}^{20} \Gamma(c_q(i+j,x)+1)} \prod_{x=1}^{20} p_k(j,x)^{c_q(i+j,x)} \right)^{w_j}. \tag{2.5}$$

Note that, since the factor containing the Gamma functions does not depend on k, it will cancel out during the normalization of $P(p_k|Q_i)$ (cf. equation (2)). Similar to PSI-BLAST (Altschul et al., 1997), we choose the pseudocount admixture τ in equation (2.4) depending on the diversity of the query alignment, $\tau = a(b+1)/(b+N_q(i))$, where $a = 0.9$ and $b = 12.0$ have been determined on the training set as described in section 2.3.3.

2.2.3. Generation of context profile library

The quality of the predicted amino acid similarities depends to a large extent on the context profile library. The clustering procedure to derive this library is summarized in Figure 2.4. We start with all sequences from the NR (non redundant) database, clustered into groups with maximum inter-group sequence identity of 30% (NR30) (Mayer and Söding *et al.*, to be published). In contrast to other approaches, in which only sequences with solved structure in the protein data bank (PDB) were used (Gelly et al., 2005; Goonesekere and Lee, 2008; Huang and Bystroff, 2006; Overington et al., 1992; Rice and Eisenberg, 1997), this guarantees an appropriate representation of all classes of local sequence contexts, such as membrane helices, natively unfolded regions, or highly repetitive sequences. From the 1.5 million cluster alignments in this NR30 database, we discard those with an effective number of sequences below 2.5 and jump-start a PSI-BLAST search against the full NR database with each of the remaining alignments (E-value threshold 0.001). This ensures an alignment diversity that is sufficient to produce mutation probabilities in the same range as the BLOSUM62 matrix. After converting the alignments to profiles, we randomly sample 1 million training profile windows of length l from the full-length profile database. For a fixed number of context profiles ($K = 500, 1000, 2000, 4000$) we determine the profile amino acid probabilities as well as the profile prior probabilities $P(p_k)$ by maximizing the total

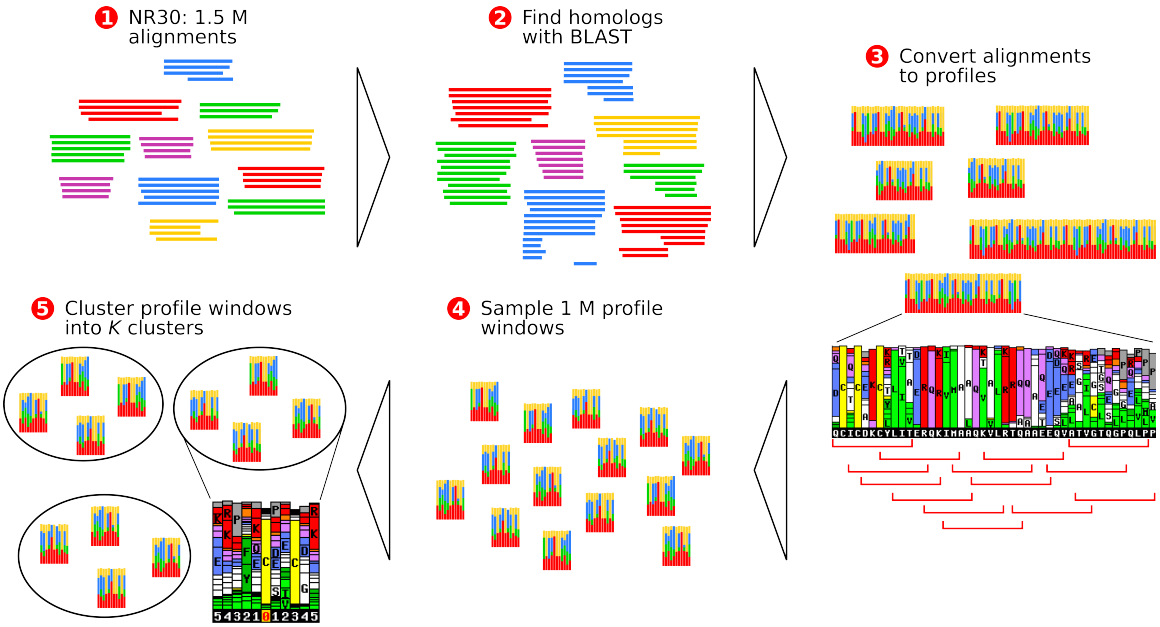


Figure 2.4.: Computation of the library of context profiles representing local sequence contexts. From a database (NR30) of 1.5M groups of aligned sequences covering the NR database, we select the 50 000 most diverse alignments and enrich these with homologs from a BLAST search. The alignments are converted to sequence profiles and 1 Mio. profile windows are randomly sampled and used to train K context profiles (K = 500,1000,2000,4000) with the expectation maximization algorithm.

likelihood that the training profile windows can be emitted by the context profiles. The maximization is done with the expectation maximization (EM) algorithm (Dempster et al., 1977).

Expectation maximization clustering

$N = 1$ million training profiles of length $l = 2d + 1$ were generated as described in Figure 2.4. Each training profile is represented by a count profile $c_n(j,x)$, which specifies the counts of amino acid $x \in \{1,\ldots,20\}$ at position $j \in \{-d,\ldots,d\}$. These counts are obtained by multiplying the sequence profile $t_n(j,x)$ by the effective number of sequences $N_n(j)$ at position j in the alignment from which training profile $t_n(j,x)$ was calculated: $c_n(j,x) = N_n(j)t_n(j,x)$ (see section 2.2.3 for details).

Here, we describe how these N profiles are clustered in order to obtain a set of K *context profiles* which recur frequently among the training profiles and which together can describe all training profiles. More precisely, we seek to determine context profiles $p = (p_1,\ldots,p_K)$ and their prior probabilities $\alpha = (\alpha_1,\ldots,\alpha_K)$ that maximize the likelihood $P(c|p,\alpha)$ that the training profile counts $c = (c_1,\ldots,c_N)$ were generated by the context profiles. We model the distribution of counts $c_n(j,x)$ in each column j by a multinomial distribution. Since $c_n(j,x)$ can be real-valued, however, we replace the factorials in the multinomial distribution by Gamma functions ($n! = \Gamma(n+1)$). The probability for context profile p_k to have emitted counts $c_n(j,x)$ ($j \in \{-d,\ldots,d\}$, $x \in \{1,\ldots,20\}$) is

$$P(c_n|p_k) = \prod_{j=-d}^{d} \left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)} \prod_{x=1}^{20} p_k(j,x)^{c_n(j,x)} \right)^{w_j}, \tag{2.6}$$

were the w_j are weights on the window positions as discussed in section 2 of the main text.

We use the Expectation Maximization (EM) algorithm to find (p^*,α^*) which optimize the likelihood $P(c|p,\alpha)$. In order to make this problem tractable, we introduce hidden variables $z = (z_1,\ldots,z_N)$. Hidden variable $z_n \in \{1,\ldots,K\}$ indicates which context profile p_k has emitted training counts c_n. With this definition, we can solve the following optimization problem:

$$(p^*,\alpha^*) = \operatorname*{argmax}_{p,\alpha} P(c|p,\alpha) = \operatorname*{argmax}_{p,\alpha} \sum_z P(c,z|p,\alpha) \tag{2.7}$$

Suppose we have obtained values $\tilde{p}$ and $\tilde{\alpha}_k$ in a previous iteration of the EM algorithm. Then the new values are obtained by (M-step)

$$(p,\alpha) = \operatorname*{argmax}_{p,\alpha} Q(p,\alpha)\,. \tag{2.8}$$

where $Q(p,\alpha)$ is the expectation value of the log likelihood of the data over all possible

$z \in \{1,\ldots,K\}^N$:

$$\begin{aligned} Q(p,\alpha) &= \sum_z P(z|c,\tilde{p},\tilde{\alpha}) \log P(c,z|p,\alpha) && (2.9)\\ &= \mathbb{E}_z[\log P(c,z|p,\alpha)] && (2.10)\end{aligned}$$

Note that the probability of z is conditioned on the values $(\tilde{p},\tilde{\alpha})$ from the previous iteration and does not depend on (p,α). In order to maximize $Q(p,\alpha)$, we first apply Bayes' Theorem to calculate the probability distribution over z:

$$P(z|c,\tilde{p},\tilde{\alpha}) = \frac{P(c|z,\tilde{p})P(z|\tilde{\alpha})}{\sum_{z'} P(c|z',\tilde{p})P(z'|\tilde{\alpha})} \propto P(c|z,\tilde{p})P(z|\tilde{\alpha})\,. \tag{2.11}$$

The first term on the right is obtained by applying equation (2.6) to $P(c_n|\tilde{p}_{z_n})$ for all n,

$$P(c|z,\tilde{p}) = \prod_{n=1}^{N}\prod_{j=-d}^{d}\left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)}\prod_{x=1}^{20}\tilde{p}_{z_n}(j,x)^{c_n(j,x)}\right)^{w_j}, \tag{2.12}$$

while the second term on the right side is simply

$$P(z|\tilde{\alpha}) = \prod_{n=1}^{N}\tilde{\alpha}_{z_n}\,. \tag{2.13}$$

Therefore, we can calculate the probability that training counts c_n were emitted by context profile p_k (E-step):

$$\begin{aligned} &P(z_n = k|c,\tilde{p},\tilde{\alpha})\\ &= \frac{\tilde{\alpha}_k \prod_{j=-d}^{d}\left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)}\prod_{x=1}^{20}\tilde{p}_k(j,x)^{c_n(j,x)}\right)^{w_j}}{\sum_{k'=1}^{K}\tilde{\alpha}_{k'}\prod_{j=-d}^{d}\left(\frac{\Gamma(N_n(j)+1)}{\prod_{x=1}^{20}\Gamma(c_n(j,x)+1)}\prod_{x=1}^{20}\tilde{p}_{k'}(j,x)^{c_n(j,x)}\right)^{w_j}}\\ &= \frac{\tilde{\alpha}_k \prod_{j=-d}^{d}\left(\prod_{x=1}^{20}\tilde{p}_k(j,x)^{c_n(j,x)}\right)^{w_j}}{\sum_{k'=1}^{K}\tilde{\alpha}_{k'}\prod_{j=-d}^{d}\left(\prod_{x=1}^{20}\tilde{p}_{k'}(j,x)^{c_n(j,x)}\right)^{w_j}}\,. \end{aligned} \tag{2.14}$$

Let us now take a closer look at $Q(p,\alpha)$:

$$\begin{aligned} Q(p,\alpha) &= E_z[\log P(c,z|p,\alpha)]\\ &= E_z[\log(P(z|z,p,\alpha)P(z|\alpha))]\\ &= \sum_{n=1}^{N}\sum_{j=-d}^{d} w_j\Bigg(\log\Gamma(N_n(j)+1) - \sum_{x=1}^{20}\log\Gamma(c_n(j,x)+1)\\ &\quad + \sum_{x=1}^{20} c_n(j,x)\mathbb{E}_z[\log p_{z_n}(j,x)]\Bigg) + \sum_{n=1}^{N}\mathbb{E}_z[\log\alpha_{z_n}]\,. \end{aligned} \tag{2.15}$$

Here,

$$\mathbb{E}_z[\log p_{z_n}(j,x)] = \sum_{k=1}^{K} P(z_n = k|c,\tilde{p},\tilde{\alpha}) \log p_k(j,x) \tag{2.16}$$

and

$$\mathbb{E}_z[\log \alpha_{z_n}] = \sum_{k=1}^{K} P(z_n = k|c,\tilde{p},\tilde{\alpha}) \log \alpha_k \,. \tag{2.17}$$

To maximize $Q(p,\alpha)$ under the constraints

$$\sum_{k=1}^{K} \alpha_k = 1 \quad \text{and} \quad \sum_{x=1}^{20} p_k(j,x) = 1 \;\forall k,j \tag{2.18}$$

we define Lagrange multipliers $\lambda, \mu_{kj} \in \mathbb{R}$ for these constraints:

$$\frac{\partial Q}{\partial \alpha_k} = \sum_{n=1}^{N} \frac{P(z_k|c,\tilde{p},\tilde{\alpha})}{\alpha_k} = \lambda \cdot \underbrace{\frac{\partial}{\partial \alpha_k}\left(\sum_{k=1}^{K} \alpha_k - 1\right)}_{1}, \tag{2.19}$$

$$\begin{aligned} \frac{\partial Q}{\partial p_k(j,x)} &= \sum_{n=1}^{N} \frac{w_j c_n(j,x) P(z_n = k|c,\tilde{p},\tilde{\alpha})}{p_k(j,x)} \\ &= \mu_{kj} \cdot \underbrace{\frac{\partial}{\partial p_k(j,x)}\left(\sum_{x=1}^{20} p_k(j,x) - 1\right)}_{1} . \end{aligned} \tag{2.20}$$

Solving for α_k and $p_k(j,x)$ and using the normalization constraints in equation (2.18) to eliminate λ and the μ_{kj} yields

$$\alpha(k) = \frac{\sum_{n=1}^{N} P(z_n = k|c,\tilde{p},\tilde{\alpha})}{\sum_{k'=1}^{K}\sum_{n=1}^{N} P(z_n = k'|c,\tilde{p},\tilde{\alpha})} \tag{2.21}$$

$$p_k(j,x) = \frac{\sum_{n=1}^{N} P(z_n = k|c,\tilde{p},\tilde{\alpha}) c_n(j,x)}{\sum_{y=1}^{20}\sum_{n=1}^{N} P(z_n = k|c,\tilde{p},\tilde{\alpha}) c_n(j,y)} \tag{2.22}$$

These two equations give the recipe for updating model parameters p and α to new values in the M-step of the EM algorithm. In the E-step we use equation (2.14) to estimate hidden variables z.

Note that our clustering approach corresponds to a soft clustering of training profiles. Each training profile can be generated by any of the context profiles (see equation 2.7), and hence each context profile has contributions from all training profiles, as can be seen in equation (2.22). This kind of clustering is adapted to the intended use of our context profile library, in which we *mix* context profiles according to their posterior probabilities in order to generate pseudocounts, instead of deriving the pseudocounts from the most similar

context profile.

In practice, we have found that the EM algorithm converges reliably after 25 iterations for $K = 4000$, running about 100h on a single core of a 2.2 GHz 64-bit Quad-Core AMD Opteron processor. Several independent EM runs yield context profiles whose CS-BLAST sensitivity was within less than 2% of each other.

Effective number of sequences

The effective number of sequences at column i of a multiple alignment is calculated on the subalignment M_i formed by all sequences with a residue in column i and by all columns with at most 10% terminal gaps in these sequences. A terminal gap is a gap that lies either to the left or to the right of the entire sequence. For each column j of M_i we calculate amino acid frequencies $p(j,x)$, using the Henikoff sequence weighting scheme. Then the number of effective sequences is

$$N_n(i) = \exp\left(-\frac{1}{L_i}\sum_{j\in M_i}\sum_{x=1}^{20} p(j,x)\log\, p(j,x)\right) . \tag{2.23}$$

Here, L_i is the number of columns in M_i.

2.3. Results

2.3.1. Benchmark

The homology detection performance of our context-specific method CS-BLAST and standard NCBI BLAST is evaluated on a benchmark data set derived from SCOP version 1.73 (Murzin et al., 1995), filtered to a maximum pairwise sequence identity of 20% (SCOP20, 6616 domains). SCOP is a database of protein domains with known structure, hierarchically ordered by class, fold, superfamily, and family (see Figure 2.5). Following a standard procedure, we consider all domains from the same superfamily to be homologous (*true positives*) and all pairs from different SCOP folds to be non-homologous (*false positives*). Domain pairs from the same fold but different superfamilies are ignored.

We randomly assign members of every fifth fold in SCOP20 to the *optimization set* (1329 domains), the others to the *test set* (5287 domains). Using the optimization set, we determined the best values for the pseudocount admixture ($\tau = 0.9$) and the window weights ($w_{\text{center}} = 1.6$, $\beta = 0.85$). The values for the window length ($l = 13$) and the context library size ($K = 4000$) are a trade-off between sensitivity and time efficiency (see section 2.3.3).

We perform an all-against-all comparison of the test-set domains and count the true and false positive hits at various E-value thresholds (Figure 2.6A). To avoid a few large families from dominating the benchmark, we weight each true and false positive pair with 1 / (size of SCOP family of first domain). Compared with NCBI BLAST (version 2.2.19, BLOSUM62,

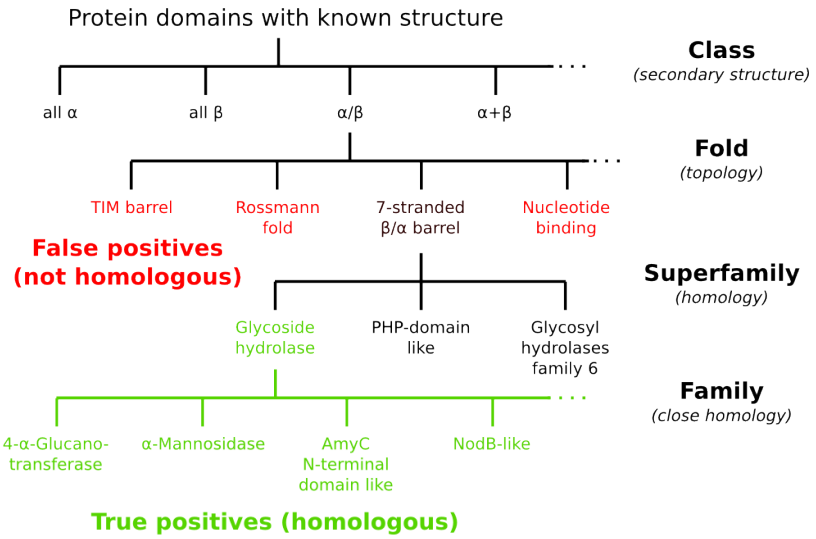


Figure 2.5.: Hierarchical organization of SCOP database consisting of protein domains with known structure. Each protein domain is characterized by class, fold, superfamily, and family. All domains from the same superfamily can be considered as homologous (*true positives*) and all pairs from different SCOP folds as non-homologous.

default parameters), CS-BLAST detects 139% more homologs at a cumulative error rate of 20%, 138% more at 10%, and, for the easiest cases at 1% error rate, still 96% more. To get an idea of the upside potential when parameters are trained on a larger set, we optimized w_{center}, β, and τ directly on the test set (red broken trace). These parameters ($w_{\text{center}} = 1.3, \beta = 0.9$, and $\tau = 0.95$) are used in the official version of CS-BLAST.

To assess the alignment quality, we compare predicted sequence alignments to gold-standard structural alignments generated by TM-ALIGN (Zhang and Skolnick, 2005). We start by randomly picking up to ten domain pairs from each family in SCOP 1.73, requiring a maximum sequence identity of 30%, and aligning each pair with TM-ALIGN. Those domains that are not well superposable (TM-ALIGN score < 0.6) are discarded. This results in 11 457 domain pairs from 5747 different domains. With each of the 5747 domains we perform a CS-BLAST and NCBI BLAST (version 2.2.19 with BLOSUM62) search against a database consisting of all domains belonging to the same family as the query domain and evaluate the quality of the predicted alignments for those pairs with a structural reference alignment. The alignment quality is assessed by two standard performance measures: *Alignment sensitivity* is the fraction of structurally aligned residue pairs that are correctly predicted, i.e. pairs correctly aligned/pairs struct. alignable. *Alignment precision* is defined as the fraction of aligned residue pairs in the predicted alignment that are correct, i.e. pairs correctly aligned/pairs aligned. Figure 2.6B plots alignment sensitivity and precision for various sequence identity bins. CS-BLAST is able to improve the BLAST results over the entire range of sequence identities, especially for the difficult alignments. Very similar results are obtained when reference alignments are generated with DALI (Holm and Sander,

1996) (data not shown).

Another critical aspect for database search tools is the reliability of the reported E-values. The E-value of a match is an estimate of the number of chance hits to be expected with a score better than that of the database match. We check the reliability of CS-BLAST E-values using the all-against-all searches of Figure 2.6A. We count the number of false positives at a given E-value threshold, which, together with the size of the benchmark database, allows us to derive the *actual* E-value. Figure 2.6C plots the actual against the reported E-value. NCBI BLAST's reported E-values are nearly identical to the observed ones. CS-BLAST E-values are too optimistic by a factor of about three to five, i.e. a reported E-value of 10^{-3} corresponds to an E-value of 5×10^{-3}. Considering that this deviation is quite small and that it changes little with E-value, it should be easy to accommodate in practice.

Finally, we evaluate the homology detection performance of CSI-BLAST, the context-

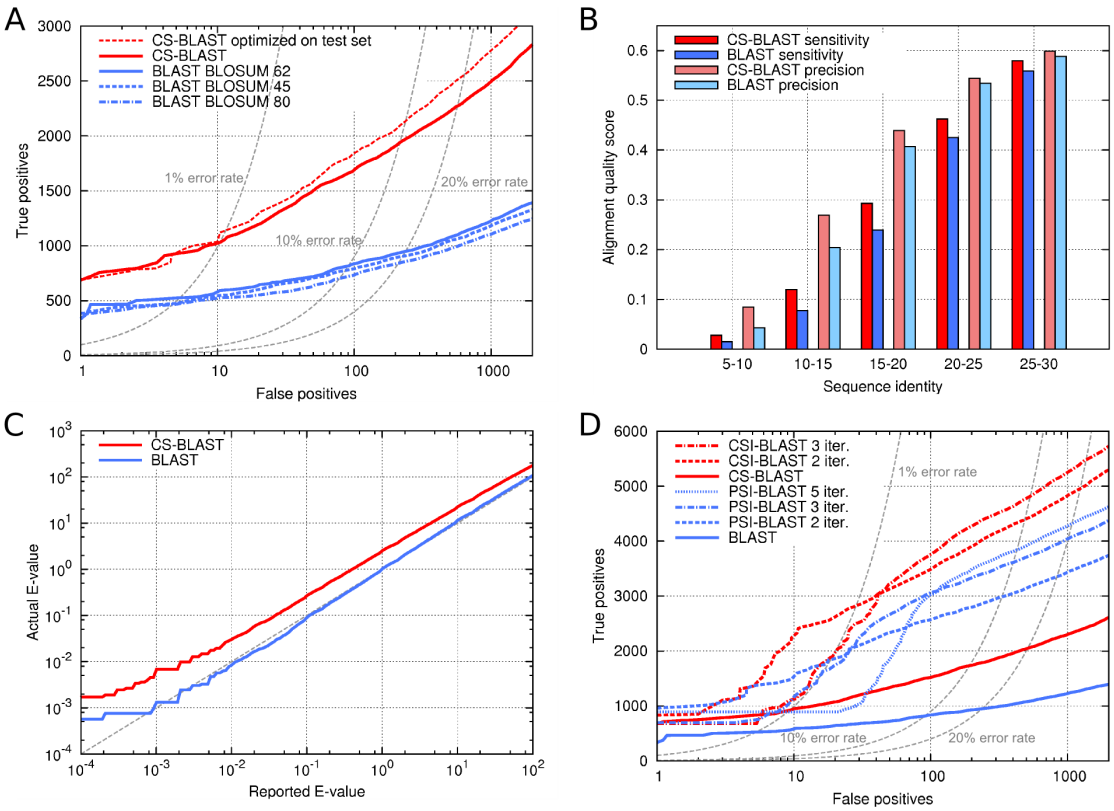


Figure 2.6.: (**A**) Homology detection benchmark on SCOP20 data set: true positives (pairs from the same SCOP superfamily) versus false positives (pairs from different folds). CS-BLAST detects 138% more true positives than BLAST at an error rate of 10 %. (**B**) CS-BLAST has better average alignment sensitivity and precision than BLAST over the entire range of sequence identities of the aligned pairs. (**C**) Actual versus reported E-values on the SCOP20 data set show that CS-BLAST E-values are too optimistic by a factor of 3 to 5. (**D**) Same benchmark as A (note different y-scales), but comparing CSI-BLAST with PSI-BLAST for one to five iterations. Two CSI-BLAST iterations are more sensitive than five PSI-BLAST iterations.

specific version of PSI-BLAST, on the benchmark of Figure 2.6A. Since, typically, PSI-BLAST searches are done with a large sequence database such as the non-redundant protein database (NR) at NCBI (Benson et al., 2008) in order to build diverse profiles, only the last search is performed against our benchmark database; all previous iterations use the full NR database (E-value inclusion thresholds set to 1×10^{-3} for PSI-BLASTand 2×10^{-4} for CSI-BLAST). Figure 2.6D plots true positives versus false positives detected by PSI-BLAST and CSI-BLAST after up to five search iterations. (The trace for CSI-BLAST with five iterations has been omitted since it does not significantly improve over three iterations anymore. The traces for one iteration are the same as in Figure 2.6A.) Remarkably, two iterations of CSI-BLAST are more sensitive than five rounds of standard PSI-BLAST ($\approx 15\%$ more homologs detected). This result surprised us. We had expected that context-specific profiles would only marginally improve sensitivity over standard sequence profiles, since profiles already contain family- and position-specific mutation rates. But the lead of CS-BLAST over BLAST is even extended in absolute terms after the second iteration, demonstrating that the context profiles contain local information from *analogous* sequences (i.e. with similar sequence context) that is partially independent of information from the *homologous* sequences in the profile.

2.3.2. Example: Activation domain of SOX-9

Figure 2.3B gave an example in which the context-specific method led to above-average conservation of Zinc-finger cysteines. In practice, it will be equally important to be able to guess which residues are conserved below average. As an example, Figure 2.7 presents profiles of a region from the activation domain of human SOX-9 transcription factor, generated with substitution matrix pseudocounts (left) and context-specific pseudocounts (right). Since this region is natively disordered, its sequence is only very weakly conserved. The substitution matrix method assigns the same amino acid distribution to the prolines as it would to a proline in a globular domain. The context-specific method, however, mixes the pseudocounts mainly from contexts which are also disordered, weakly conserved, and have a similar, biased amino acid distribution. Therefore, its profile exhibits below-average conservation of prolines, alanines, and glutamines while having higher overall probabilities for these residues.

2.3.3. Parameter optimization

Our context-specific search tools CS-BLAST and the EM clustering procedure for generating the library of context profiles (see section 2.2.3 about generation of context profiles) contain several adjustable parameters: The number of context profiles K, the profile window length l, the positional weight parameters w_{center} and β, and the pseudocount admixture τ. We optimize these parameters using a homology detection benchmark that is similar to the benchmark utilized to compare CS-BLAST with BLAST (see section 2.3.1 for details). The optimization runs, however, are performed on a small *optimization set* (1329 domains) that

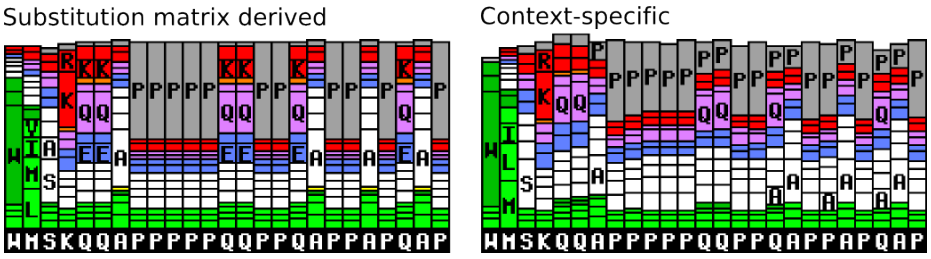


Figure 2.7.: Proline-rich region in human transcription factor SOX-9. The mutation profile computed with substitution matrix pseudocounts (left) overestimates the conservation in this region. The context-specific profile (right) shows weaker conservation of prolines, alanines, and glutamines, and increased presence of these residues in neighboring columns.

is distinct from the comprehensive *test set* (5287 domains) with no optimization set member sharing a common fold with any test set member. For a given parameter configuration ($K, l, w_{\text{center}}, \beta$, and τ) we first generate a library of K context profiles by EM clustering, then we perform an all-against-all comparison of the training-set domains and count the true positive (TP) and false positive (FP) hits at various E-value thresholds. From this we can infer a ROC5 score ($\in$ [0,1]) for each query. The ROC5 score is defined as the area under the TP-versus-FP ROC curve (receiver operating statistic) up to the fifth false positive hit, divided by the area under the optimal ROC curve. To assess the overall homology detection performance, one can plot the fraction of queries with ROC5 scores above a variable ROC threshold ($\in$ [0,1]). The area under such a curve, the mean ROC5 score, conveniently captures the homology detection performance in a single value and is therefore used as performance index in the parameter optimization. Optimization results by mean ROC5 score proved to be more robust and less noisy than results obtained by optimizing the overall sensitivity at a given error rate.

Ideally, we would like to compute mean ROC5 scores for a wide range of different parameter configurations. However, due to the considerable runtime of the EM clustering algorithm we have to restrict our optimization runs to a selected set of reasonable parameter settings. Parameter values for window size l and library size K result from a compromise between sensitivity and time efficiency in CS-BLAST. To limit the runtime of the profile generation in CS-BLAST for a typical protein to about $1s$, we choose $K = 4000$ and $l = 13$ (see Figure 2.8). With K and l fixed, we benchmark all 60 possible parameter combinations for $w_{\text{center}} \in \{1.3, 1.6, 2.0, 2.5\}$, $\beta \in \{0.8, 0.85, 0.9\}$, and $\tau \in \{0.8, 0.85, 0.9, 0.95, 1.0\}$. The optimization runs reveal that parameter setting $w_{\text{center}} = 1.6$, $\beta = 0.85$, and $\tau = 0.9$ gives the best mean ROC5 score of 0.2374. Around this optimal configuration we test the effect of different values for cluster size (K = 500,1000,2000,4000) and window length (l = 9,11,13,15). The results suggest that the homology detection performance improves not only with the number of context profiles K, as expected, but also with the length of the context window l. However, a subsequent evaluation of window lengths parameters $l = 13$ and $l = 15$ on the

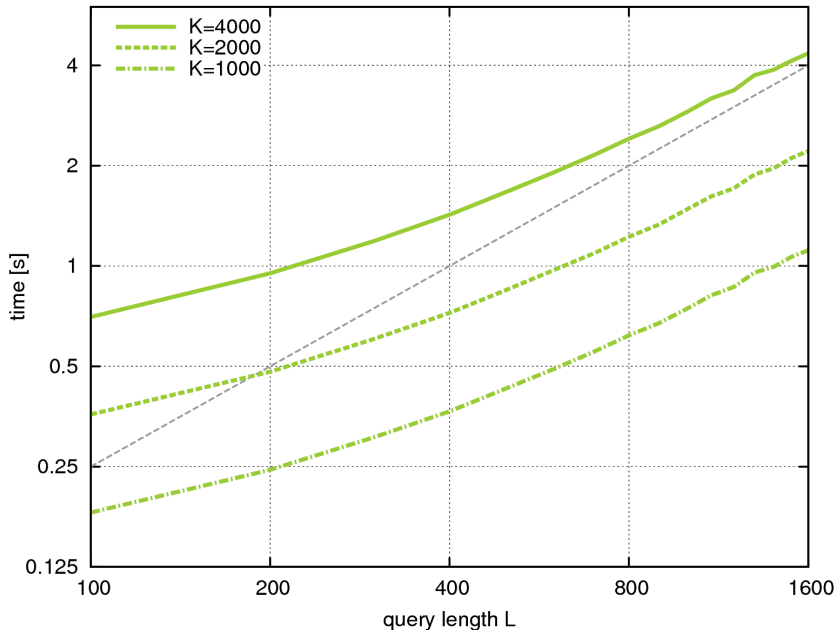


Figure 2.8.: Time required for generating the context-specific profile in CS-BLASTdepending on query length L and size of context library $K = 1000{,}2000, 4000$ (window length $l = 13$). The plot verifies that the runtime T scales roughly with $T \propto KlL$. To limit the runtime of the profile generation in CS-BLAST for a typical protein with 200 residues to $\sim 1s$, we choose $K = 4000$.

benchmark set revealed almost identical performance for both settings. We therefore chose $l = 13$ for the benefit of faster runtime. Figure 2.9A-E illustrates the effect of varying one parameter at a time while keeping the others fixed at the optimum configuration ($K = 4000$, $l = 13, w_{\text{center}} = 1.6, \beta = 0.85, \tau = 0.9$).

In addition to CS-BLAST parameters $(K, l, w_{\text{center}}, \beta, \tau)$, we also need to optimize pseudocount admixture parameters a and b for CSI-BLAST (see section 2.2.2). We employ the same benchmark procedure as described above, but perform three rounds of CSI-BLAST instead of CS-BLAST. The last search is performed against our training database; all previous iterations use the full NR database. Since we already know the optimal pseudocount admixture for the one-sequence case from the optimization of τ in CS-BLAST, we set a to 0.9. Benchmark runs for $b = 6, 8, 10, 12, 16, 20$ reveal that $b = 12$ gives the best results and is therefore the default setting in CSI-BLAST (Figure 2.9F).

2.4. Discussion

Sequence context is much more powerful than a single residue in predicting which amino acids that particular residue is likely to mutate into (Figure 2.6A,B). Since this context information is as easy to get as the sequence itself, it is surprising that sequence context is practically never exploited. The main reason seems to be the focus of past research on struc-

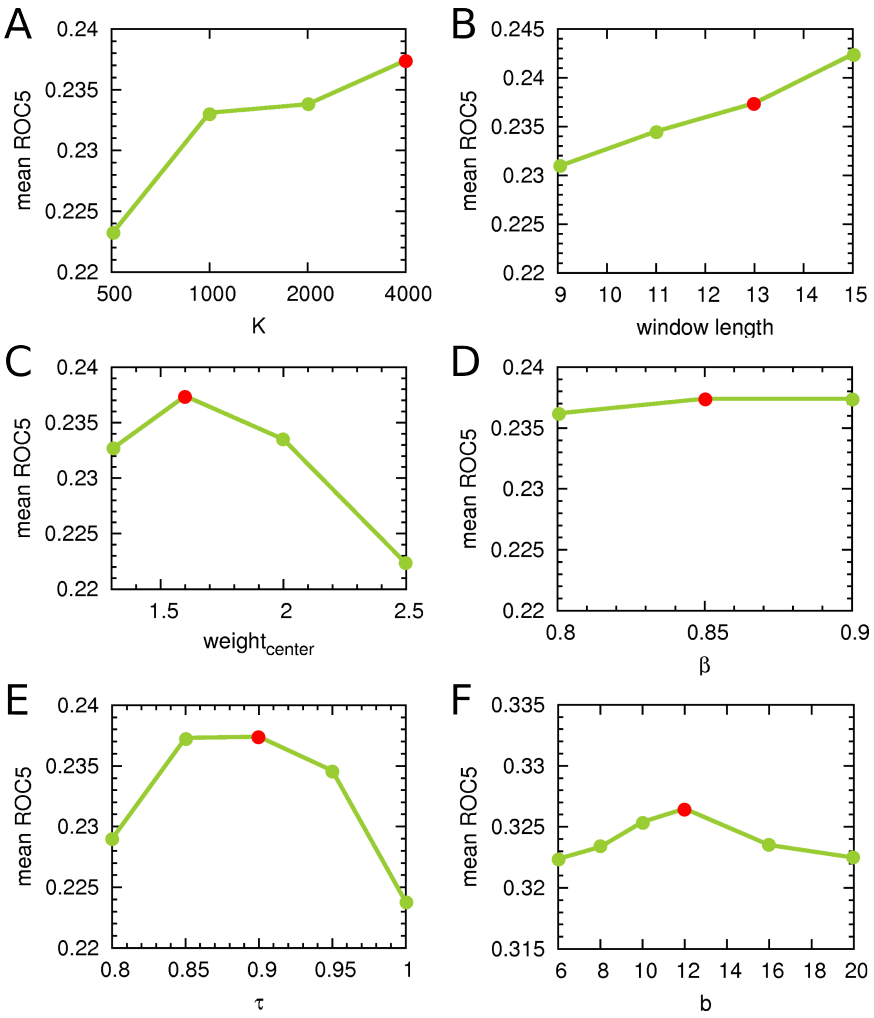


Figure 2.9.: Optimization of parameters $K, l, w_{\text{center}}, \beta, \tau$ and CSI-BLAST parameter b. The plots illustrate the effect of varying one parameter at a time while keeping the others fixed at the optimal configuration $K = 4000, l = 13, w_{\text{center}} = 1.6, \beta = 0.85$, and $\tau = 0.9$. Red circles indicate the optimal parameter value that is chosen as default in CS-BLAST/CSI-BLAST. (**A**) Number of context profiles K. (**B**) Window length l. (**C**) Weight of central profile column w_{center}. (**D**) Weight decay parameter β. (**E**) Pseudocount admixture τ. (**F**) Pseudocount extinction parameter b in CSI-BLAST ($a = 0.9$). Three iterations of CSI-BLAST were performed for this optimization. Note the different y-scale.

tural context, with its limitation to proteins of known structures (Gelly et al., 2005; Goonesekere and Lee, 2008; Huang and Bystroff, 2006; Overington et al., 1992; Rice and Eisenberg, 1997). Another reason may be the challenge to develop sequence context-specific methods that can compete with traditional context-free methods such as BLAST and PSI-BLAST in speed and usability (Baussand et al., 2007; Huang and Bystroff, 2006). We have shown how context-specific pseudocounts can be employed in combination with existing profile-based methods to extend residue-centered sequence comparison to the context-specific case, without loss of speed or usability.

As examples, we have built context-specific versions of BLAST and PSI-BLAST which considerably improve their performance at very little runtime overhead. For a typical protein of length $L = 250$ and a library size of $K = 4000$, the computation of the context-specific profile requires about one second. Also, runtime scales favorably, $T \propto KlL$. (Note that HMMSUM's runtime scales as $T \propto K^2L$, which places a strict practical limit on the number K of states/contexts in HMMSUM.) Since the output of CS-BLAST and CSI-BLAST is generated by the BLAST and PSI-BLAST programs themselves, users do not have to get accustomed to different command line options or output formats, and updates to the BLAST package will directly benefit the context-specific versions. The only caveat is that E-values need to be corrected by a factor of three to five (Figure 2.6C). We expect CS-BLAST to be particularly useful to find homologs for *singleton sequences*, since for these the lack of homologs precludes the use of profile-to-sequence search methods such as PSI-BLAST.

A pleasant surprise is the extent of improvements of sequence profiles through context-specific pseudocounts (Figure 2.6D), even though profiles already contain evolutionary information on position- and family-specific mutation probabilities. Hence, the information from locally similar, *analogous* sequences that are contained in the context profiles is at least partly orthogonal to the evolutionary information in the *homologous* sequences that contribute to the sequence profiles. Consequently, we can expect improvements when applying the new paradigm to the pairwise comparison of sequence profiles (Rychlewski et al., 2000; Sadreyev and Grishin, 2003; Yona and Levitt, 2002) and profile HMMs (Madera, 2008; Söding, 2005), or to hierarchical multiple sequence alignment programs (Notredame, 2007; Thompson et al., 1994).

It is possible to extend Dirichlet mixture pseudocounts (Durbin et al., 1998; Sjoelander et al., 1996) to the context-specific case. This would yield an alternative formulation of context-specific sequence comparison that is worth exploring. In that scheme, the context library would have K meta-profiles, i.e., multicolumn pseudocount priors. Each meta-profile would consists of l Dirichlet distributions and would be able to *emit* a profile with l columns. An advantage over the presented scheme might be that the diversity of each column in the meta-profiles is encoded by one additional parameter per column (the sum of all pseudocounts in a column), which might lead to better modeling of the profile contexts.

The paradigm presented here should be easily transferable to nucleotide sequences. The

application to non-coding regions such as promoter regions and regions harboring putative non-coding RNAs (ncRNAs) is of particular interest: The low information content of nucleotide sequences and the often weak overall conservation in these regions render alignments between related species difficult, while reliable alignments offer great promise to identify functional regions (such as cis-regulatory elements or ncRNAs) through their inter-species conservation (see, e.g., Stark et al. (2007)).

In summary, the paradigm of sequence context specificity offers greatly improved sensitivity and alignment quality in protein sequence comparison and is likely to hold similar advantages for nucleotide sequences. We believe that these advantages are sufficient to warrant a paradigm shift in biological sequence comparison, alignment, and molecular evolution from amino acid- and nucleotide-centric to context-specific methods.

3. Column states alphabet for sequence profile encoding

3.1. Introduction

We can also use sequence contexts to define column states, i.e. states with contexts of length $l = 1$, that describe profile columns optimally. In this way, we can encode a sequence profile as a sequence over an extended alphabet of column states while preserving most of the evolutionary information in the profile. In this chapter we present a novel approach that uses sequences of column states to derive a profile-to-profile score that can be efficiently computed at the same speed as standard sequence-to-sequence scores. Finally, we show how to employ fast profile-to-profile scoring in combination with the iterative HMM-HMM search tool HHBLITS to improve homology detection performance with no runtime overhead. To understand why fast profile-to-profile scoring is of particular interest in HHBLITS, let us take a closer look at the search strategy which it uses to detect remote homologs.

3.1.1. Iterative HMM-HMM search with HHBLITS

Protein homology detection and sequence alignment are powerful techniques in computational biology because often a proteins function or three dimensional structure can be predicted by inference from known homologs. The workhorse of protein homology detection is PSI-BLAST, with over 5000 citations (Altschul et al., 1997). PSI-BLAST owes its sensitivity to its iterative search scheme, in which significant sequence hits are aligned to the query sequence to construct the profile for the next search iteration (see Figure 1.2).

To improve the sensitivity of picking up distant evolutionary relationships, our group has developed a new method, HHBLITS, that is based on iterative profile HMM-HMM comparison (Remmert and Söding *et al.*, to be published). HMM-HMM comparison has been shown to be much more sensitive than PSI-BLAST (Söding, 2005). However, the use of HMM-HMM comparison in an iterative search scheme would require a database of profiles (or, more precisely, profile HMMs) covering all known sequences. Also, profile HMM-HMM comparison is prohibitively slow (more than 2000× slower than PSI-BLAST). HHBLITS solves both problems as follows: firstly it uses a previously constructed database (NR30) of HMMs and sequence alignments which covers the entire sequence database, and secondly, it employs a fast implementation of the Smith-Waterman algorithm (Farrar, 2007; Smith and Waterman,

1981) for pre-filtering the NR30 database. The pre-filtered HMMs in the database are then compared against the evolving query HMM using HMM-HMM comparison. All significant hits are aligned to the query alignment and the new alignment is used as input for the next iteration of Smith-Waterman pre-filtering and HMM-HMM search, as illustrated in Figure 3.1.

The pre-filter step is the crucial step to achieve speeds comparable to PSI-BLAST without losing much sensitivity compared to standard HMM-HMM comparison methods. It is not only the time limiting step but also the one determining the sensitivity in the HHBLITS workflow. The pre-filter needs to be as fast and efficient as possible because the search is performed against the whole NR30 database with over a million alignments, but at the same time it has to be sensitive enough to separate the non-homologous HMMs in the database from the true homologs, such that most of the latter ones can be compared with the time-consuming HMM-HMM alignment. In the following section we will demonstrate that we can significantly increase the homology detection performance of the pre-filter step by encoding the NR30 database not in the form of standard amino acid consensus sequences but in an alphabet of 62 profile columns, that has been generated by clustering a large representative set of training profile columns.

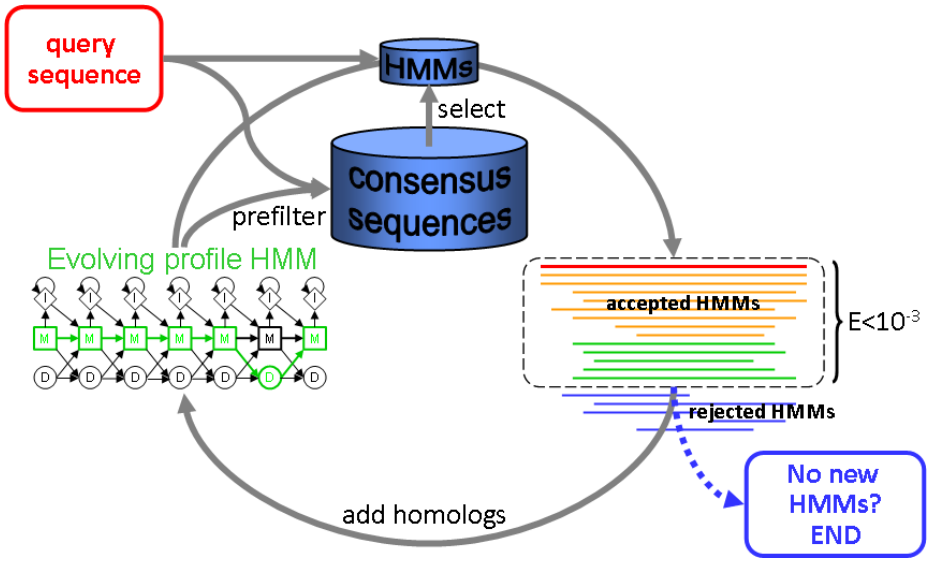


Figure 3.1.: Schematic workflow of HHBLITS. During the fast pre-filter step HHBLITS searches with the query sequence against all alignments in the NR30 database (large blue cylinder). All those HMMs in the NR30 database that were found with an E-value better than 100 are extracted into a temporary database (small blue cylinder). This much smaller temporary database can then be efficiently searched using very sensitive, but slow, HMM-HMM comparison. All significant hits are likely to be homologous and are aligned to the query profile HMM. This new profile HMM is then used as input for the next iteration of pre-filtering and HMM-HMM comparison. The search procedure is repeated until no new HMMs are found or the maximum number of iterations is reached.

3.2. Material and methods

3.2.1. Fast profile-to-profile match score

The central idea behind the pre-filter step in HHBLITS is as follows: to be able to make use of fast sequence comparison, we have to encode the database profiles as sequences. To balance speed and sensitivity to pick up even very distant evolutionary relationships, HHBLITS utilizes a fast implementation of the Smith-Waterman algorithm. Unlike BLAST and PSI-BLAST, the Smith-Waterman algorithm is guaranteed to find the optimal local alignment between the query sequence profile and a database sequence. As already explained in section 2.2.1, substitution matrix scores can be seen as a special case of profile-to-sequence scores, where the profile is generated from one of the query sequence by using substitution matrix pseudocounts. Therefore, the substitution score $S(x_i,y_j)$ used within the Smith-Waterman algorithm can be expressed as standard profile-to-sequence score of profile column i of the query sequence profile q with residue y_j of the database sequence (y_j):

$$S(q(i,\cdot),y_j) := \log \frac{q(i,y_j)}{P(y_j)} = \log \frac{P(y_j|x_i)}{P(y_j)} = S(x_i,y_j). \tag{3.1}$$

Note that, while this profile-to-sequence score can easily take into account additional evolutionary information on the query side through the query-side profile representation, the same is not true for the database side. On the database-side, we have to resort to encoding all evolutionary information into a single sequence. Therefore, the initial implementation of HHBLITS used consensus sequences instead of full NR30 alignments for the pre-filter step. Since in the translation of an alignment into a consensus sequence there is always a loss of information incurred, it is desirable to replace the profile-to-sequence score of the pre-filter by a profile-to-profile score that would make better use of evolutionary information on the database side.

However, simply replacing the profile-to-sequence score with a standard profile-to-profile score is not a viable option. The evaluation of the profile-profile co-emission score between two sequence profiles q and p

$$S(q(i,\cdot),p(j,\cdot)) := \log \sum_{a=1}^{20} \frac{q(i,a)p(j,a)}{P(a)} \tag{3.2}$$

requires iteration over all 20 amino acids and is therefore at least 20 times slower than the profile-to-sequence score introduced in equation (3.1).

In the following, we present a novel approach that derives a profile-to-profile score which can be efficiently computed at the same speed as the score in equation (3.1). Our approach is based on encoding each database-side alignment column as one letter from a pre-computed specialized alphabet, that consists of 62 column states. Each column state $k \in \{1,\ldots,62\}$ of this CS62 alphabet is characterized by a specific amino acid profile vector $s_k(\cdot)$ and individual

character $s_k \in \{0\text{-}9, A\text{-}Z, a\text{-}z\}$. We chose to use 62 column states as this is the number of all digits, uppercase and lowercase characters in the ASCII character-encoding scheme. Figure 3.2 illustrates each of the 62 columns states with their one letter codes and characteristic profile vectors. The CS62 alphabet itself was generated by clustering a large representative set of training profile columns (see section 3.2.2 for details). Through encoding each profile column q_j in the NR30 database with one of the 62 column states such that $y_j \in \{1,\ldots,62\}$, we can now formulate a new profile-to-column-state score that uses profile information on the database side but is fast to evaluate at the same time

$$S(q(i,\cdot),y_j) := \log \sum_{a=1}^{20} \frac{q(i,a)s_{y_j}(a)}{P(a)}. \tag{3.3}$$

Note that although equation (3.3) still contains a sum over all amino acids, we can easily pre-calculate all scores $S(q(i,\cdot),k)$ for $i \in \{1,\ldots,L\}, k \in \{1,\ldots,62\}$. Thus, we do not have to recalculate alignment match scores repeatedly during dynamic programming but simply perform a look-up of the pre-calculated scores $S(q(i,\cdot),k)$ for the query profile in a way completely analogous to profile-to-sequence comparison, except that the alphabet over 20 amino acids is replaced by an alphabet over 62 column states.

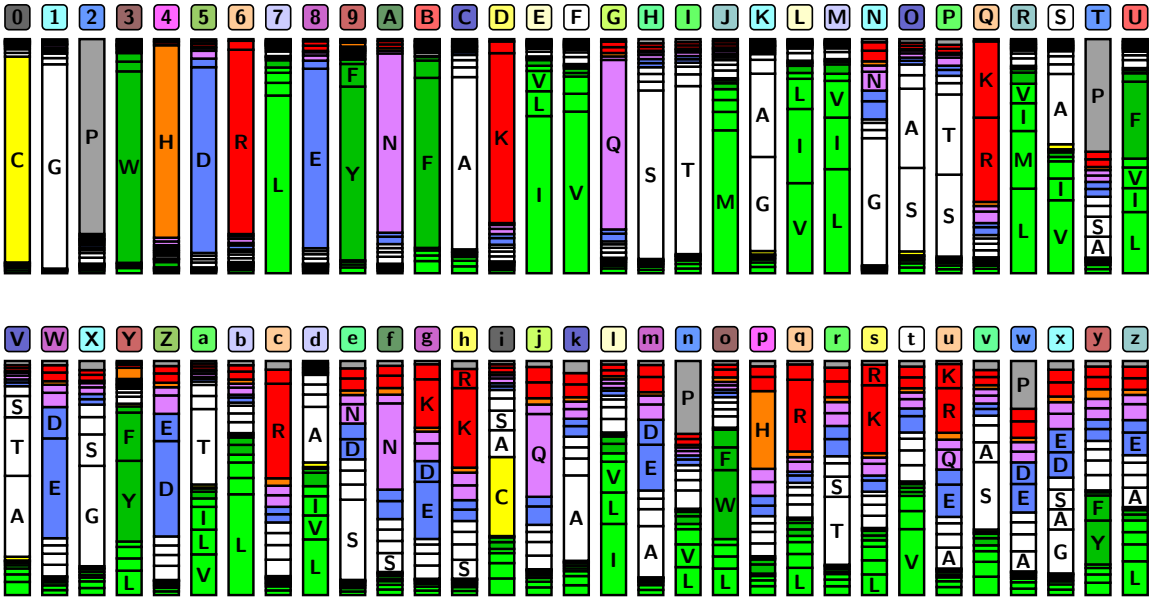


Figure 3.2.: Overview of all 62 column states in the CS62 alphabet. Each column state is characterized by a specific amino acid profile vector (histograms) and a one-letter code (colored boxes with rounded corners). Column states with similar amino acid profile vectors possess identical background colors in their one letter code. Note that the first 20 column states are more or less pure and encode for the 20 amino acids. Other non-pure column states contain groups of amino acids that readily substitute for each other, e.g. aliphatic amino acids or lysine and arginine. Column states at the very end (`x,y,z`) represent highly diverged alignment columns with slightly biased background distributions.

3.2.2. Generation of CS62 column state alphabet

The improvement of the profile-to-column-state score (equation 3.3) over the consensus based profile-to-sequence score as in equation (3.1) depends to a large extent on the column state alphabet. The clustering procedure to derive the 62 column states is almost equivalent to expectation maximization clustering (described in section 2.2.3) that we used to generate the context profile library for CS-BLAST. However, we make two important changes: First, instead of clustering the training profiles into a set of $K = 4000$ context profiles of length $l = 13$ we want to obtain $K = 62$ context profiles of length $l = 1$. Of course, the resulting 62 context profiles of length one are really nothing other than 62 profile vectors. As a training set we use $N = 10$ million training profile vectors randomly sampled from NR30 database alignments. Second, we utilize a window weight of $w = 1000$ instead of $w = 1.6$. This is motivated by the fact that *one* column state instead of a mixture of states will be used for encoding an alignment column. Thus, the CS62 alphabet should be optimized such that the observed counts in an alignment column are described as well as possible by a single column state, rather than through a mixture of column states. The high value of $w = 1000$ ensures that the expectation maximization algorithm effectively performs a hard clustering where each training profile column is generated by one column state only.

3.2.3. Translation of alignments into CS62 sequences

In order to use the profile-to-column-state score (equation 3.3) in the pre-filter step, HHBLITS needs as input the CS62 column state sequences of the NR30 database. To obtain the CS62 column state sequences we need to translate each NR30 alignment into the CS62 column states alphabet. This process is illustrated in Figure 3.3. First, the input alignment is converted into a sequence profile (colored histogram below alignment) to which a small amount of context-specific pseudocounts are added. In the following translation step each profile column is translated into the one CS62 alphabet letter (colored boxes with rounded corners) that best describes the observed counts. More precisely, profile column i is replaced with the alphabet letter s_k for which the probability to generate the observed amino acid counts $c(i,x)$ is maximum:

$$k = \underset{k}{\operatorname{argmax}}\, P(s_k) \left(\sum_{x=1}^{20} s_k(x)^{c(i,x)} \right)^w . \tag{3.4}$$

$P(s_k)$ is the Bayesian *prior probability* for column state s_k, determined in the process of computing the column states alphabet (section 3.2.2). It quantifies the probability that an alignment column is emitted by profile vector $s_k(\cdot)$ *prior* to knowing that alignment column. The counts $c(i,x)$ of amino acids x at position i of the input alignment are modeled with a multinomial distribution. In order to make the translation of alignments readily available to HHBLITS users, we have implemented the translation procedure in a small executable

called CS-TRANSLATE. The executable takes an alignment and the column state library as input, translates the alignment, and finally writes the CS62 column state sequence in FASTA format to an output file.

3.3. Results

The homology detection performance of HHBLITS with sequences over the CS62 alphabet and HHBLITS with standard amino acid consensus sequences is evaluated on a benchmark data set derived from SCOP version 1.73 (Murzin et al., 1995), filtered to a maximum pairwise sequence identity of 20% (SCOP20, 6616 domains). SCOP is a database of protein domains with known structure, hierarchically ordered by class, fold, superfamily, and family. Following a standard procedure, we consider all domains from the same superfamily to be homologous (*true positives*) and all pairs from different SCOP folds to be non-homologous (*false positives*). Domain pairs from the same fold but different superfamilies are ignored.

We randomly assign members of every fifth fold in SCOP20 to the *optimization set* (1329 domains), the others to the *test set* (5287 domains). Using the optimization set, we determined the best values for the admixture of the context-specific pseudocounts to be added to database alignment side as well as the P-value threshold of the HHBLITS pre-filter, which adjusts how many HMMs are included in the temporary database for HMM-HMM comparison.

We perform an all-against-all comparison of the test-set domains and count the true and false positive hits at various P-value thresholds (Figure 3.4A). To avoid a few large

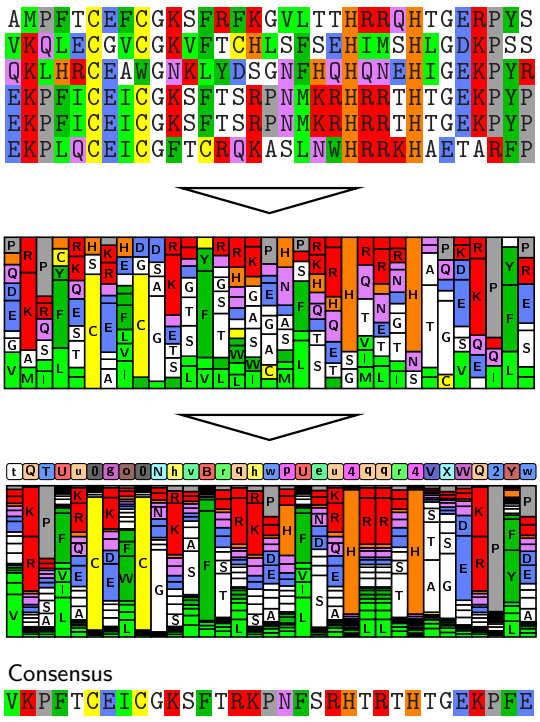


Figure 3.3: Translation of an alignment into a sequence over an extended alphabet of column states. First, the input alignment is converted into a sequence profile (colored histogram below alignment) to which a small amount of context-specific pseudocounts has been added. In the following translation step, each profile column is translated into the one CS62 alphabet letter that best describes the observed counts (colored boxes with rounded corners). Below each CS62 alphabet letter its characteristic profile column is shown in order to facilitate the comparison between original and translated profile columns. The sequence at the bottom is the consensus of the input alignment. Clearly, the consensus sequence fails to faithfully encode the observed profile counts in all but the most highly conserved columns consisting almost entirely of Cysteines and Histidines. The CS62 sequence on the other hand is able to represent all observed counts very well, regardless of whether they occur in a highly conserved or less conserved profile column.

families from dominating the benchmark, we weight each true and false positive pair with 1 / (size of SCOP superfamily of first domain). After one and two iterations, HHBLITS with CS62 column state sequences detects 9% and 8% more true positives than HHBLITS with standard amino acid consensus sequences at an error rate of 10%. To get an idea of the upside potential when we use a column state library of more than 62 states, we also generated a CS219 alphabet that consists of 219 column states and makes use of all ASCII characters as one-letter codes. Compared with the CS62 alphabet, the extended CS219 alphabet does not improve the homology detection performance after one iteration and the improvements after two iterations are only marginal (1% more homologs detected, data not shown).

In the homology detection benchmark illustrated in Figure 3.4A we pool the hits of *all* queries before counting the number of true and false positive hits at various P-values. Thus, not every query is guaranteed to contribute equally to the ROC curves shown in Figure 3.4A.

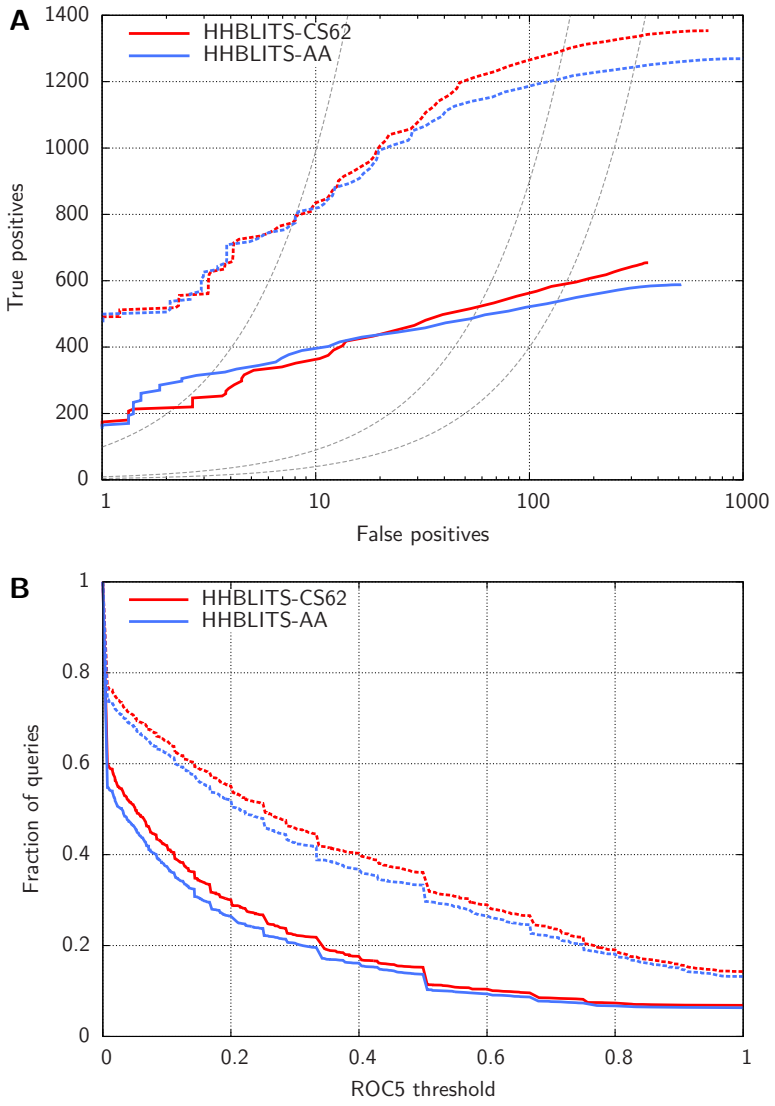


Figure 3.4: CS62 column states alphabet improves homology search performance of HHBLITS after one (solid lines) and two rounds (dashed lines). (**A**) Homology detection benchmark on SCOP20 data set: true positives (pairs from the same SCOP superfamily) versus false positives (pairs from different folds). After one and two iterations HHBLITS with CS62 column states alphabet detects 9% and 8% more true positives than HHBLITS with standard amino acid consensus sequences at an error rate of 10%. (**B**) Same benchmark set as in (A) but fraction of queries with ROC5 scores above a variable ROC5 threshold shown. Again, HHBLITS with column states alphabet is able to detect more homologs than HHBLITS with standard amino acid consensus sequences (10% and 7% improvements after one and two iterations respectively).

For example, the very first high scoring false positives might be caused by erroneous hits in the search results of only one query. To get an idea of the homology detection performance when each query search contributes equally, we have devised a second benchmark. For this benchmark we again perform an all-against-all comparison of the test-set domains but we count the true positive (TP) and false positive (FP) hits at various P-value thresholds for each query *separately*. From this we can infer a ROC5 score ($\in$ [0,1]) for each query. The ROC5 score is defined as the area under the TP-versus-FP ROC curve (receiver operating statistic) up to the fifth false positive hit, divided by the area under the optimal ROC curve. To assess the overall homology detection performance, we plot the fraction of queries with ROC5 scores above a variable ROC5 threshold ($\in$ [0,1], see Figure 3.4B.). The area under the curve, the mean ROC5 score, conveniently captures the homology detection performance in a single value and is also used as performance index in the parameter optimization. The ROC5 benchmark results illustrated in Figure 3.4B, confirm our earlier results. After one and two iterations, HHBLITS with CS62 column state sequences detects 10% and 7% more homologs than HHBLITS with standard amino acid consensus sequences.

3.4. Discussion

When representing a diverse sequence profile as a standard consensus sequence of amino acids, a loss of information is unavoidable. This is because diverse profile columns cannot adequately be modeled by one single amino acid. Here, we present an approach that encodes a profile as a sequence over an extended alphabet of column states, each of which is associated with a characteristic profile vector. This new approach preserves most of the evolutionary information in the profile because diverse profile columns can be represented by a column state with similar amino acid distribution. Since the column state sequence is made up of printable letters, it is easy to read both by humans and computer programs.

We have shown how sequences of column states can be employed in combination with the existing HMM-HMM search tool HHBLITS to improve its homology detection performance with no runtime overhead. Also, the translation of a sequence profile into a column state sequence is extremely fast (runtime less than 100 milliseconds for an average profile) which makes the translation of a profile databases with millions of profiles possible. The new approach to describe profile columns by CS62 columns states should be easily transferable to all search and alignment tools that use standard amino acids based consensus sequences. To utilize sequences over the CS62 column state alphabet in their programs, developers merely need to download the pre-computed CS62 column state alphabet and the CS-TRANSLATE executable to translate sequence profiles or alignments into column state sequences.

Part II.

Multiple alignment

4. Multiple alignment of regulatory DNA

4.1. Overview

With the rapid sequencing of many closely related genomes, comparative genomics has become a very promising area in computational biology (Cliften et al., 2003; Consortium, 2007a; Kellis et al., 2003). Comparative genomics methods use alignments of intergenic DNA sequences of related species to identify functional elements by their conservation signature. The basic assumption is that regulatory motifs underlie evolutionary constraints which cause them to evolve more slowly than the "background" surrounding DNA, that is free of any functional constraints. Therefore, it is standard practice to search for regulatory elements by scanning alignments of regulatory regions to identify short stretches that are well conserved across species (Woolfe et al., 2005). Besides the computational prediction of already characterized functional elements (Berman et al., 2004; Sinha et al., 2003; Wasserman et al., 2000), comparative sequence analysis can also help to discover novel sequence motifs (Li and Wong, 2005; Siddharthan et al., 2005) and to explore the principles of regulatory sequence evolution (Ludwig et al., 1998). Other important applications that make use of cross-species multiple alignments include the inference of phylogeny (Wong et al., 2008) and the estimation of substitution rates (Siepel and Haussler, 2004).

All these downstream analysis methods critically depend on the correctness of the underlying multiple alignment. For instance, in regions where sequences are misaligned, conservation-based motif detection methods may fail to identify conserved transcription factor binding sites (see Figure 4.1). The dependence of comparative genomics methods on the underlying multiple alignments also becomes evident in a study of Stark et al. (2007) who analyzed 12 *Drosophila* genomes for *de novo* discovery of functional elements. When investigating the effect of alignment accuracy on their predictions, they found only 59% agreement between different alignment strategies for regulatory motif instances.

Because misaligned sequences could easily produce false signals of evolutionary change, methods used to build up phylogenetic trees are at high risk of a loss of accuracy when sequences are misaligned. Wong et al. (2008) have shown that different alignment methods may seriously affect the results of comparative genomic analysis such as reconstruction of phylogenetic trees and inference of positive selection. In addition, a simulation study by Pollard et al. (2006) found that variation in alignment accuracy could result in significant errors in evolutionary studies, and that there is a need for phylogenetic tools that can control for alignment errors.

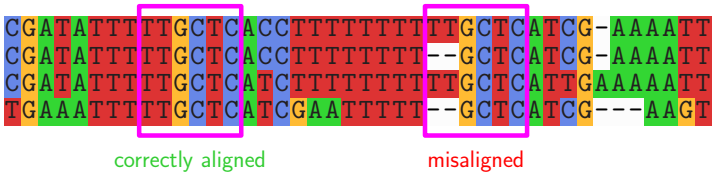


Figure 4.1.: Example of conserved transcription factor binding sites in orthologous yeast sequences that have been misaligned. The alignment algorithm has incorrectly inserted gaps in orthologous motif occurrences because it did not recognize the motif signature.

Unfortunately, alignments of noncoding DNA are often very challenging, since noncoding sequences are much less structured than coding sequences and functional elements can drift away while being functionally substituted by analogous motifs in nearby locations (Moses et al., 2006). Even in the case of sequence pairs for which homology can be established over a larger stretch of a few hundred base pairs, the actual alignment of these orthologous noncoding sequences can often be ambiguous. These technical challenges have triggered the development of probabilistic alignment methods that can quantify the alignment uncertainty (Bradley et al., 2009; Lunter et al., 2008; Paten et al., 2008; Satija et al., 2008).

4.1.1. Alignment of regulatory motifs

Most alignment tools are not customized to regulatory sequences, and thus cannot take advantage of their specific structural and evolutionary properties. These alignment algorithms may incorrectly insert gaps in orthologous motif occurrences because they do not take into account sequence signature of binding sites. For instance, Figure 4.1 shows conserved transcription factor binding sites (boxes) in orthologous yeast sequences that are likely to be misaligned. While there are indeed two successful programs, EMMA and MORPH, that explicitly model transcription factor binding sites (TFBS) affinity matrices in the alignment algorithm (He et al., 2009; Sinha and He, 2007), their application is limited to the case of pairwise alignment. MORPH detects and aligns instances of known motifs by a probabilistic alignment algorithm. Although this approach has the additional requirement that binding site motifs have to be known *a priori*, the authors report that binding-site predictions are robust to alignment ambiguities. EMMA, the successor of MORPH, possesses the additional ability to model the gain and loss of TFBSs in its alignment process. The development of this feature has been stimulated by accumulating evidence that functional noncoding sequence in general and TFBSs in particular, are not always conserved in an alignable sequence even in relatively close species (Consortium, 2007b; Moses et al., 2006). Another recent work, the program SAPF (Satija et al., 2008), aims to combine probabilistic model-based alignment with *phylogenetic footprinting*, which refers to the identification of evolutionarily constrained sequences based on their lower substitution rates. However, TFBSs are not explicitly represented in the SAPF model, and the program is not designed to predict targets of specific

transcription factors.

4.1.2. General approaches to multiple sequence alignment

Even the classical problem of finding the optimal multiple sequence alignment under the assumption that the sequences have changed only by mutations that insert, delete and substitute residues, without the added complexity of accounting for functional binding sites, is a challenge by itself (Durbin et al., 1998). The computational challenges present in this problem can generally be related to the number and length of the sequences being aligned. The former increases the dimensionality of the problem, the latter increases the basic search space that needs to be considered. The combination of both means that naive implementations of even the most standard objective functions, such as the sum-of-pairs function, are simply unfeasible (Elias, 2006; Wang and Jiang, 1994). In practice, heuristic methods are therefore used for all but the smallest data sets.

The most commonly used heuristic methods are based on the progressive-alignment strategy (Feng and Doolittle, 1987; Hogeweg and Hesper, 1984; Taylor, 1988) with CLUSTALW (Thompson et al., 1994) being the most widely used implementation. The idea is to take an initial, approximate, phylogenetic tree between the sequences and to gradually build up the alignment, following the order in the tree. Therefore, the initial tree is also often called the guide tree. Although successful in a wide variety of cases, this method suffers from its greediness. Errors made in the first alignment stages are propagated through to the final result and cannot be corrected later as the rest of the sequences are added in. This is illustrated in Figure 4.2A in which four hypothetical sequences are progressively aligned. In the resulting alignment the word `CAT` is misaligned due to an error made in the very first pairwise alignment between sequence 1 and 2. The problem of erroneous sub-alignments being fixed before more distant sequences are added in occurs often when there are many nearly equivalent possibilities for a gap placement, a phenomenon described as edge wander (Holmes and Durbin, 1998). One possible approach to remedy this problem is iterative refinement (Gotoh, 1996), which improves on progressive alignment by iteratively realigning subsets of sequences, but this still often fails to correct complex alignment errors involving multiple sequences.

4.1.3. Consistency

Another approach to mitigate the problems of progressive alignment are so called consistency-based methods, which incorporate information from more diverged sequences in the alignment of sub-trees. Thus, these methods have a flavor of global optimization while still inherently working in the pairwise fashion known from progressive alignment. The textbook consistency-based approach, however, leads to algorithms with impractical runtimes that scale with $O(N^3L^3)$ for the alignment of N sequences of average length L. The first

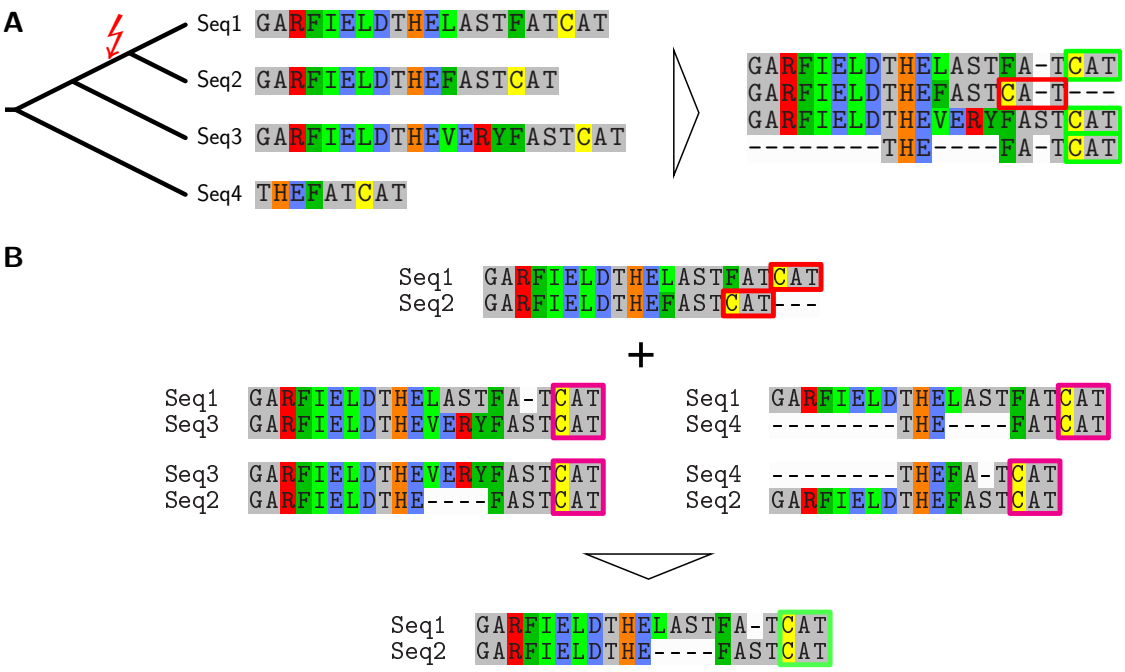


Figure 4.2.: Progressive sequence alignment. (**A**) If errors are made in standard progressive alignment, these errors are propagated through to the final result and cannot be corrected at later stages. The guide tree indicates the order in which four hypothetical sequences are aligned. In the resulting alignment on the right the word `CAT` (red box) is misaligned due to an error made in the very first pairwise alignment between sequence 1 and 2 (red lightning). (**B**) Consistency-based approach to progressive alignment. The basic idea is to incorporate information from more diverged sequences in the alignment of sequences 1 and 2 in order to prevent alignment errors (red boxes). The transitive pairwise alignments of sequences 1 with sequence 2 over sequences 3/4 suggest that that the words `CAT` in sequences 1 and 2 should be aligned to each other (magenta boxes). This information is then used in the first progressive alignment step between sequence 1 and 2 resulting in a correct alignment between sequences 1 and 2 (green box).

widely used multiple alignment algorithm that used consistency information was the program T-COFFEE (Notredame et al., 2000). It first computes a collection of global and local alignment for every pair of input sequences. A *consistency transformation* is then applied that incorporates scores created transitively by triangular projection of scores between the different constituent pairwise alignments (Figure 4.2B). Later, *Do et al.* have formulated the idea of using consistency information in a probabilistic framework (Do et al., 2005). Their program PROBCONS employs a probabilistic consistency transformation, based upon posterior match probabilities computed using the Forward and Backward algorithms (Durbin et al., 1998). While PROBCONS is limited to the alignment of protein sequences, its cousin PECANwas the first program to make posterior decoding and consistency-based alignment practical for very large multiple alignments of DNA (Paten et al., 2008).

4.2. Multiple alignment tools

In the following sections, we will explain the unique characteristics and principles behind the alignment algorithms of three current multiple alignment programs: PECAN FSA, and PRANK. All three methods have also been evaluated in the multiple alignment benchmark described in section 5.3.

4.2.1. PECAN

The alignment method of PECAN has a close relation to the consistency methodology first introduced in PROBCONS and T-COFFEE (Do et al., 2005; Notredame et al., 2000). By combining the methods of probabilistic consistency alignment with multi-sequence alignment with constraints (Myers et al., 1996), PECAN has made probabilistic consistency alignment practical for the alignment of large genomic sequences. Therefore, PECAN is now one of the default alignment programs that is being used by a number of whole-genome comparative genomic projects.

In brief, PECAN introduces two major technical advances. First, it utilizes a banded form of the standard forward–backward algorithm (Durbin et al., 1998) for pair-HMMs, to allow posterior probabilities of pairwise alignment to be generated with heuristics on large sequences. Second, the authors conceived a novel technique to handle the computation of multiple pairwise posterior probability calculations during the generation of a single consistency alignment across very large genomic sequences, which they call *sequence progressive alignment*. At its core, PECAN utilizes a pair-HMM to calculate match posterior probabilities $P(x_i \diamond y_j \,|\, x,y)$, which quantify how probable it is that residue i in a sequence x is aligned to residue j in a sequence y, followed by re-estimating these posterior probabilities by applying a probabilistic consistency transformation introduced by Do *et al.*. The consistency transformation incorporates similarities of x and y to other sequences z into the comparison of x to y:

$$P(x_i \diamond y_j \,|\, x,y) = \text{const} \times \sum_z \sum_k P(x_i \diamond z_k \,|\, x,z) P(z_k \diamond y_j \,|\, z,y). \tag{4.1}$$

The idea behind the above consistency transformation is to make use of the transitive nature of homology. It refers to the fact that if residue i is aligned to k and k to j then residue i must be aligned to j.

4.2.2. FSA

FSA is a multiple sequence alignment program for aligning proteins, RNA or long genomic DNA sequences (Bradley et al., 2009). FSA's approach to alignment seeks to maximize the expected accuracy of the calculated alignment, which allows it to more reliably identify non-homologous sequences than most other multiple alignment programs, although this increased accuracy comes at the cost of decreased speed. The alignment algorithm of FSA is

based on a pair-HMM which describes an insertion, deletion, and mutation process on a tree. FSA also uses a sequence annealing algorithm to combine the posterior match probabilities estimated from the pair-HMM into a multiple alignment. The sequence annealing algorithm, an improved version of the original algorithm in the multiple alignment program AMAP (Schwartz, 2007), begins with the null alignment, in which all sequences are unaligned, and merges single columns by aligning residues according to the corresponding increase in expected alignment accuracy (see Figure 4.3). Whereas standard progressive alignment methods such as CLUSTALW are forced to take rather large steps in alignment space by aligning entire sequences, the sequence annealing approach in FSA takes the smallest-possible steps of aligning single residues. This approach makes FSA much less prone to incorporating alignment errors at early alignment stages, even though it does not explicitly utilize consistency information.

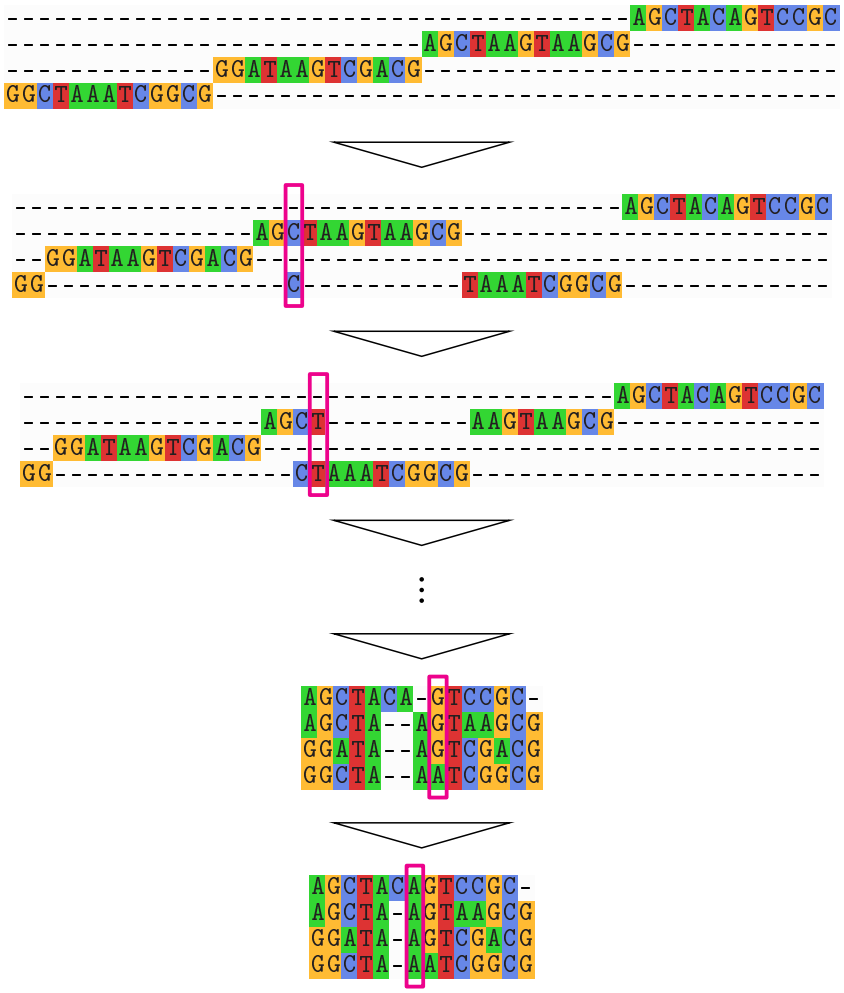


Figure 4.3: Multiple alignment by sequence annealing as utilized in the program FSA. The sequence annealing algorithm begins with the null alignment, in which all sequences are unaligned, and merges single columns by aligning residues according to the corresponding increase in expected alignment accuracy. At each step, the most recently merged column is highlighted (magenta box).

4.2.3. PRANK

PRANK is a probabilistic multiple alignment program for DNA and protein sequences (Löytynoja and Goldman, 2005, 2008). Unlike traditional multiple alignment methods that usually disregard the phylogenetic implications of predicted gap patterns, PRANK's alignment algorithm recognizes insertions and deletions as distinct evolutionary events and tries to find an alignment that corresponds to plausible insertion- and deletion-events. Thereby it avoids the over-estimation of the number of deletion and substitution events that often occur when using traditional progressive alignment methods (see Figure 4.4A). More precisely, the problem that PRANK tries to address is that in progressive sequence alignment insertions and deletions are treated differently. A gap for a deletion, with its associated penalty, is created only once, but a gap for an insertion has to be opened up multiple times (Figure 4.4). Simple iteration of pairwise alignment associates a full penalty with each of these gap-opening events, which leads to excessive penalization of single insertion events. This asymmetric handling of insertions and deletions has also been confirmed in an evaluation of alignment tools by Kim and Sinha (2010), who found that the performance of most tools degrades more rapidly when there are more insertions than deletions in the data set. In the example illustrated in Figure 4.4A, the phylogeny-aware alignment algorithm in PRANK uses evolutionary information from the third sequence to confirm the evolutionary event as insertion, which allows the site to be skipped in subsequent pairwise alignment steps without penalization (green arrows). Besides its phylogeny-aware progressive alignment algorithm, PRANK borrows ideas from maximum likelihood methods used in phylogenetics and to model the evolutionary distances between sequences.

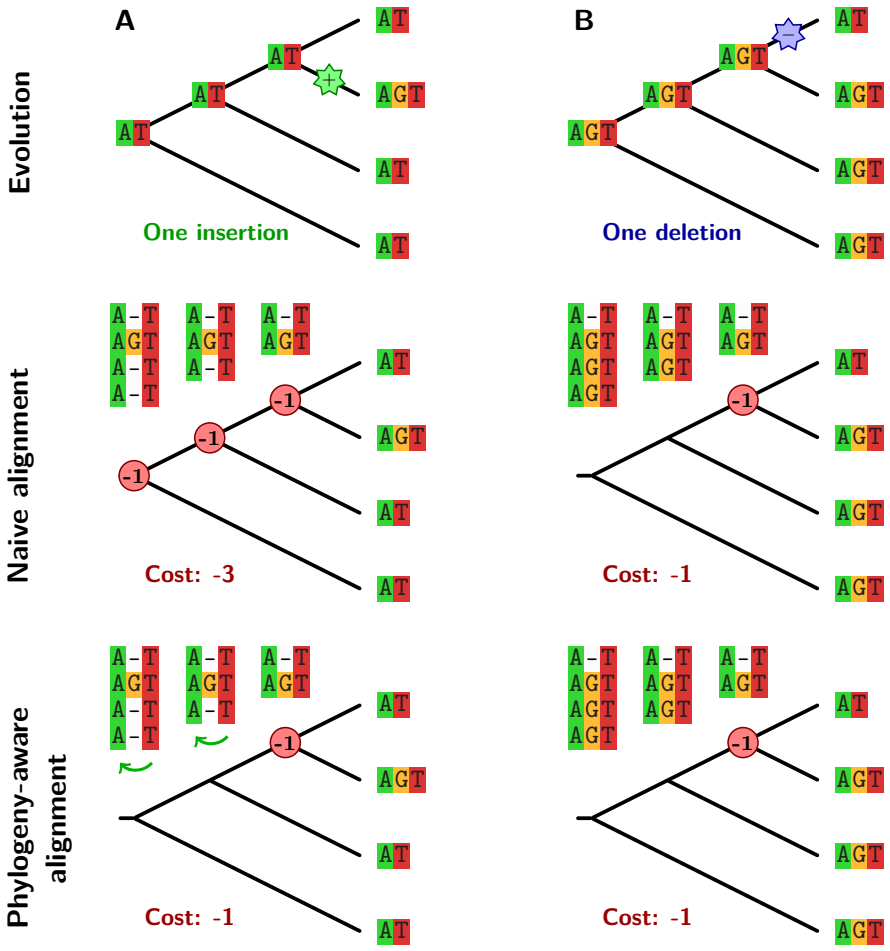


Figure 4.4.: Insertions (green star) and deletions (blue star) are treated differently in progressive sequence alignment. (**A**) Progressive alignment algorithms build a multiple alignment from sequential pairwise alignments, and an insertion requires a new gap to be opened in each of them. Naive iteration of pairwise alignment penalizes this single evolutionary event multiple times (red circles), giving an inappropriately high cost for the correct alignment. The phylogeny-aware alignment algorithm in PRANK uses evolutionary information from the third sequence to confirm the evolutionary event as insertion, which allows that site to be skipped over in subsequent pairwise alignment steps without penalization (green arrows). (**B**) A deletion is penalized only once in both progressive alignment schemes. (red circle).

5. Context-specific multiple alignment with partial-order HMMs

5.1. Introduction

Progressive multiple alignment methods depend on reducing a multiple sequence alignment (MSA) to a linear profile at each alignment step along the guide tree. However, this can lead to loss of information and gap scoring artifacts (Lee et al., 2002). Grasso and Lee (2004) proposed an elegant approach to this problem, which utilizes a partial order graph (POG) representation of an alignment at each progressive alignment step. In this chapter, we present a new approach to alignment representation that generalizes classical partial order graphs to a full probabilistic graph by combining profile HMMs and partial order graphs. Such a "partial order HMM" (PO-HMM) does not only prevent loss of information and gap scoring artifacts but has the added advantage that it can store several alternative alignments, including their probabilities. The ability to represent whole sets of alignments as a PO-HMM is particularly helpful because it prevents freezing of alignment errors at early stages of hierarchical alignment. In contrast to PO-HMMs, the POG s introduced by Grasso and Lee (2004) are not probabilistic and not able to be combined with HMMs.

Let us take a closer look at progressive alignment to understand why the graph based approach by Lee *et al.* is beneficial. Traditional progressive alignment methods, e.g. CLUSTALW, build a multiple sequence alignment through a series of pairwise alignments in the order dictated by the evolutionary guide tree. This requires aligning pairs of multiple alignments, to build up larger alignments. In practice, however, pairwise dynamic programming is not applied directly to align two multiple alignments. Instead, commonly each

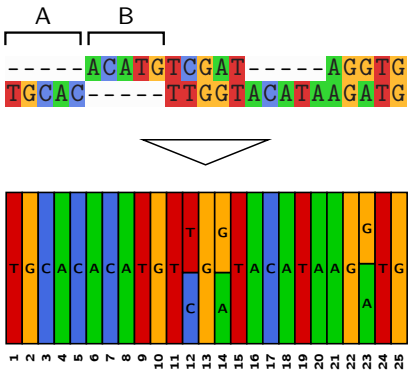


Figure 5.1: Profile representation of a multiple sequence alignment consisting of two hypothetical sequences (Lee et al., 2002). While alignment blocks A and B could be swapped without changing the meaning of the alignment, this would change the order of the first ten columns of the sequence profile, which indeed could make a drastic difference when other sequences are aligned to the profile.

alignment is reduced to a sequence profile which can be aligned to each other in pairwise dynamic programming alignment. Unfortunately, the reduction of an multiple alignment to a sequence profile inevitably involves loss of information. That is because while the multiple alignment contains all the information needed to produce the profile, the profile does not contain all the information to recover the original multiple alignment. For example, one might choose to exclude all alignment columns that correspond to insertions from the sequence profile. However, even if we assume that all columns of the alignment are represented in the profile there is a loss of information incurred because the profile may keep residue and gap frequencies for each column, but it has no information on which sequence a given residue comes from. This makes scoring of gaps problematic, for reasons that are illustrated in Figure 5.1. While the profile contains *all* columns from *all* sequences in the alignment, it is likely that no sequence in the alignment actually has a residue in all these columns. As a result, any sequence not containing *all* these columns, will be charged artificial gap penalties even if its gaps are exactly the same as in one or more sequences in the alignment. Even worse, the only sequence that will not be penalized when aligned to the profile is the consensus sequence of the multiple alignment. Another problem becomes apparent if we consider the two unaligned blocks A,B in Figure 5.1 with subsequences `TGCAC` and `ACATG` at the beginning of the alignment. As far as the meaning of the multiple alignment is concerned, the order of these blocks, AB or BA, does not matter. This is because the blocks do not share any sequences with each other. However, changing the order from AB to BA does indeed drastically change the structure of the sequence profile. If the blocks were in BA order, aligning the sequence `TGCACTTGGTACAT`$\cdots$ would no longer be charged a five residue gap penalty because now `TGCACTTGGTACAT`$\cdots$ matches without gaps. There is tremendous degeneracy in the representation of a multiple alignment because through rearrangement of blocks such as A an B in Figure 5.1 many equivalent alignments exist but each of these will give rise to a *different* sequence profile. The real problem here is that the one dimensional sequence profile is unable to properly represent the fact that there is *no* ordering relation between blocks A and B because no sequence in block A is aligned to any sequence in block B and vice-versa.

In this chapter, we present a new approach to MSA representation that generalizes classical partial order graphs to a full probabilistic graph by combining profile HMMs and partial order graphs. Such a "partial order HMM" (PO-HMM), is ideal for representing not only one but possibly a set of potentially correct, probable multiple alignments in a way that is ideal from a statistical as well as an algorithmic perspective. Furthermore, it is alignable by pairwise PO-HMM-to-PO-HMM (PO-PO) alignment, similarly to sequence-sequence or profile-profile alignment. The PO-HMM also adequately models the degeneracy inherent to multiple sequence alignments.

5.2. Materials and Methods

5.2.1. Representing multiple alignments by PO-HMMs

To explain the structure of a PO-HMM let us consider the multiple alignment of nine sequences depicted in Figure 5.2A. We can represent the alignment as a partially ordered graph of nodes such that each graph node corresponds to an alignment column (Figure 5.2B). In addition, we introduce special `BEGIN` and `END` nodes in order to adequately model leading and trailing gaps in the multiple alignment. Each node, with exception of `BEGIN` and `END` nodes, stores information about sequence IDs and sequence positions of all aligned residues in the column corresponding to that node. The residue counts at each alignment column are needed for the calculation of nucleotide emission probabilities of PO-HMM match states (see section 5.2.4). In the partial order graph, directed edges are drawn between two nodes if, and only if, they contain at least one pair of residues that occur consecutively in a sequence. For instance, there is an edge connecting nodes 6 and 7 because the residues in alignment columns 6 and 7 are consecutive in all sequences. The fact that *all* sequences contribute to this edge gives the edge a probability of 1.0 (black edge color). The edge between nodes 7 and 14, on the other hand, is a result of the consecutive pair of nucleotides `AT` in sequence 6. However, the edge probability of this edge is $\frac{1}{9}$ (assuming equal sequence weights). Thus, gaps in the alignment appear in the partial order graph as edges that jump one or more nodes. Edge probabilities indicate the fraction of sequences in the alignment that contribute to the edge. Therefore, a node may have any number of incoming and outgoing edges because edges simply indicate the preceding and subsequent letters of every sequence that has a letter in the alignment column of this particular node. Thus, it is possible to trace the path of each individual sequence through the partial order graph. Therefore, it becomes evident that there is a one-to-one relationship between an alignment and a POG. Each multiple alignment maps to a PO-HMM and vice-versa. Even the arbitrary placement of gaps is elegantly described by the partial order graph. For instance, columns 22-25 and 18-21 in Figure 5.2A could be swapped without changing the meaning of the alignment. The POG representation accounts for this degeneracy by not imposing a strict ordering relation to branch 22-25 with respect to branch 18-21 since there exists no path of directed edges connecting the two branches.

Although up to now we have treated each node in the POG as a single entity, a PO-HMM is actually a POG that contains a pair of match and insert states at each of its nodes and that has transition probabilities (and hence penalties) associated with each of its transitions. The PO-HMM transition structure around nodes 7 and 34 is shown in Figure 5.2. Readers who are familiar with profile HMMs will immediately recognize that the PO-HMM structure resembles the form of standard profile HMMs, as they were first introduced by Haussler *et al.* and Krogh *et al.* (Krogh et al., 1994). However, a PO-HMM has allowed transitions between match states M_i and I_i and match state M_j not only for $i = j - 1$ as profile

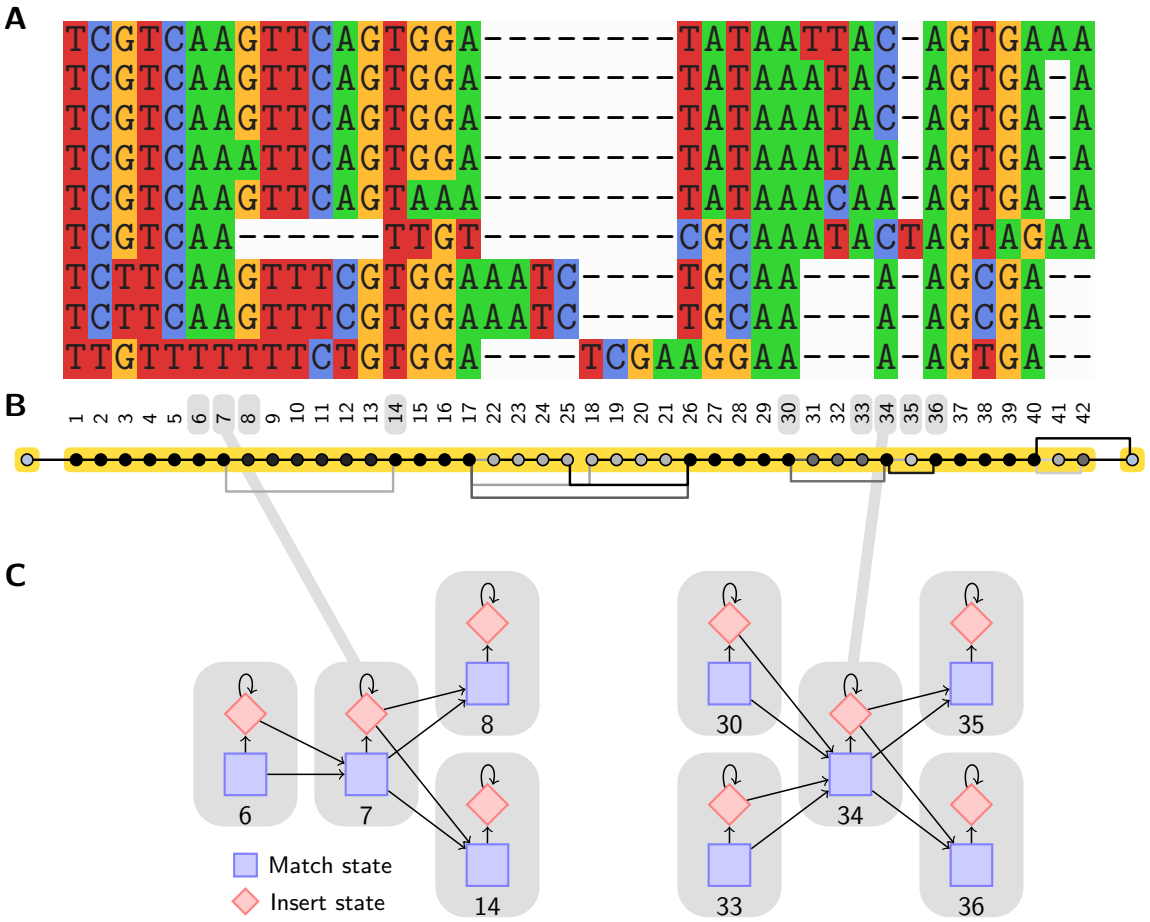


Figure 5.2.: Multiple sequence alignment in PO-HMM representation. (**A**) Multiple sequence alignment of nine sequences. Note that columns 17-25 and columns 18-21 could be swapped without changing the meaning of the multiple alignment. (**B**) Partial order graph (POG) representing the multiple alignment in (A). Each node in the POG corresponds to one alignment column. Nodes are connected by directed edges from left to right where indicated. Edges are shaded to indicate the fraction of sequences in the alignment that contribute to the edge. Note that nodes 22-25 have no ordering relation to nodes 18-21 since there exists no path of directed edges between the two branches. (**C**) PO-HMM transition structure around nodes 7 and 34. Each node in the POG in (B) consists of a a match state (blue square) and an insert state (red diamond).

HMMs have but generally for $i < j$. A second difference to standard HMMs is the absence of delete states to make pairwise alignment of PO-HMMs more easily made. The M_i carry emission probabilities derived from the residues aligned in node i of the PO-HMM. Emission distributions of insert states are set to the background distribution, just as in pairwise sequence alignment.

5.2.2. Representing several alternative alignments by a PO-HMM

So far we have learned how the PO-HMM structure can store a multiple alignment in a very condensed fashion. However, a PO-HMM is able to represent not only one but also

a set of probable multiple alignments in a way that is ideal from a statistical as well as an algorithmic perspective. Let us have a look at a short example. Figures 5.3A and B depict two alternative pairwise alignments of two sequences x and y. The two alignments have slightly differently placed gaps: while the first alignment aligns residues $y_3y_4y_5$ to gaps, the second alignment places the gaps at residues $y_5y_6y_7$. We would like to represent both alignments in *one* PO-HMM in order to preserve the information about alternative alignment paths until the later progressive alignment stages. The POG structure given in 5.3B illustrates how we can encode both alignments as PO-HMM in a condensed fashion. While alignment columns that differ between the two alternative paths are stored separately, the PO-HMM represents two identical alignment columns only once, in the form of a single

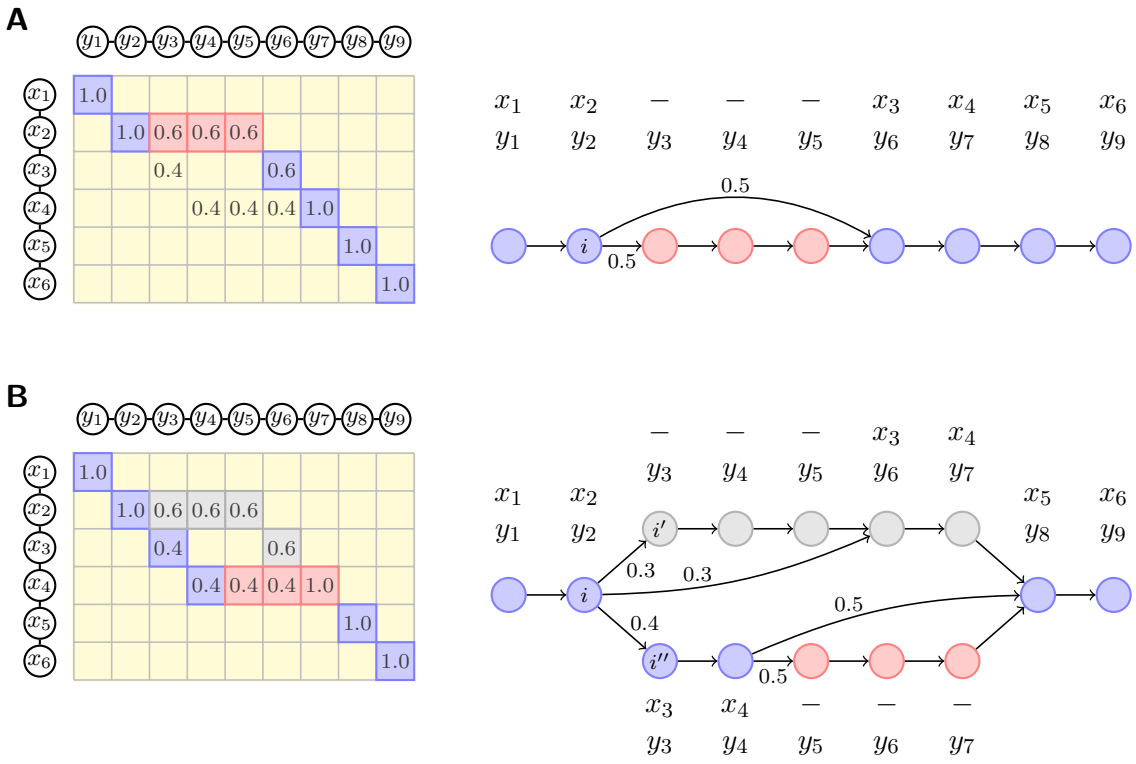


Figure 5.3.: Partial order graph representation of two alternative alignments between sequences x and y. The numbers within matrix cells give the posterior probability that two residues are aligned with each other (blue cells) or that one residue is aligned with a gap (red cells). (**A**) The best alignment with probability 0.6 uses the "upper" path through the matrix and aligns residues $y_3y_4y_5$ to gaps. The POG representation of this pairwise alignment is given on the right. Weights of outgoing edges at node i are both 0.5 because of equal sequence weights. (**B**) The alternative alignment with probability 0.4 uses the "lower" path through the matrix and aligns residues $y_5y_6y_7$ to gaps. The POG on the right represents both alternative alignments in a condensed fashion. Note that the weights of the outgoing edges at node i model the probabilities for branching off into the two alternative alignment segments ($0.3 + 0.3 = 0.6$ versus 0.4).

node. Furthermore, the probability of branching off into alternatively aligned segments can be conveniently expressed through edge probabilities. For instance, the outgoing edges at node i in Figure 5.3B mirror the probabilities of the two alternative paths ($0.3 + 0.3 = 0.6$ versus 0.4). The advantage of the condensed encoding is that the number of nodes in the PO-HMM merely grows with the number of alternative aligned multiplets of residues, not the sum of all alignment columns. The ability to encode whole sets of alignments into a PO-HMM is especially helpful when aligning sequences and PO-HMMs progressively from the leaves to the root node because one incurs much less risk of choosing a wrong alignment early on that will then be frozen.

5.2.3. Pairwise PO-HMM alignment

To calculate pairwise PO-HMM alignments needed at each step of the progressive PO-HMM alignment procedure, we use global PO-HMM-to-PO-HMM (PO-PO) comparison that computes posterior probabilities. The concept of posterior probabilities for alignments was first introduced by Miyazawa *et al.* (Miyazawa, 1995): given two sequences x and y, the posterior probability $P(x_i \diamond y_j \,|\, x,y)$ quantifies how probable it is that residue i in sequence x is aligned to residue j in sequence y. This approach was later extended to the case of *local* HMM-HMM alignment (Biegert and Söding, 2008). Here, we generalize the concept of probabilistic alignment based on posterior probabilities to the case of *global* PO-PO comparison. The details about the derivation of posterior probabilities for *global* PO-PO comparison are given in the following section.

Posterior probabilities for PO-PO alignment

To quantify the local reliability of an alignment, we would like to calculate posterior probabilities for global PO-PO comparison. Assume we align two PO-HMMs q and p of length L_q, L_p. We adopt the thermodynamic interpretation of PO-PO alignment that was developed for sequence-sequence alignment by Kschischo and Lässig (2000); Miyazawa (1995); Mückstein et al. (2002). The probability for an alignment $\mathcal{A}$ between q and p is given by

$$P(\mathcal{A}) = \frac{e^{\beta S(\mathcal{A})}}{Z}\,, \tag{5.1}$$

where $S(\mathcal{A})$ is the score for alignment $\mathcal{A}$ and $\beta = 1/kT$ (T = temperature, k = Boltzmann constant) can be assumed to be 1 in the following. Z is the *partition function*:

$$Z = \sum_{\mathcal{A}} e^{\beta S(\mathcal{A})}. \tag{5.2}$$

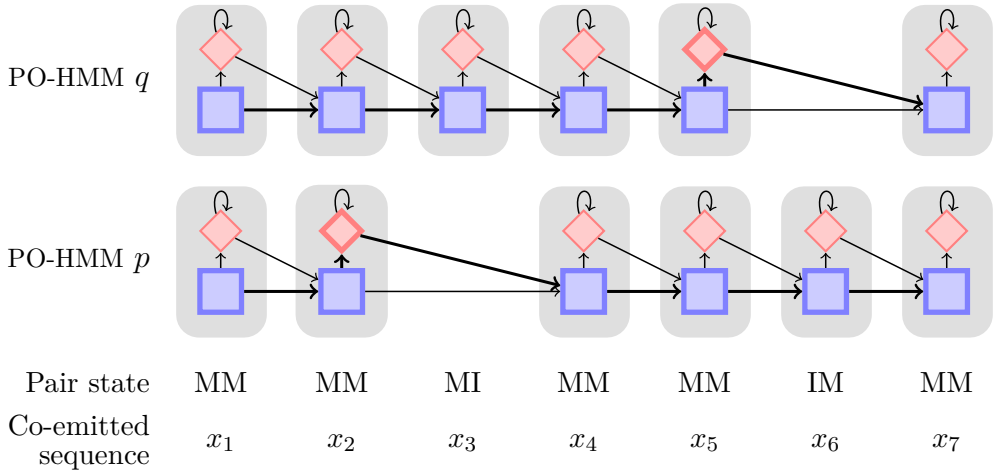


Figure 5.4.: Alignment of two PO-HMMs by maximization of log-sum-of-odds score. The path through the two PO-HMMs corresponds to a sequence that is co-emitted by both PO-HMMs. The alignment pair state MI signifies that the node of the first PO-HMM does not have a homologous partner.

The sum runs over all alignments such that Z is effectively a normalization constant in equation (5.1). The score $S(\mathcal{A})$ is defined as in Söding (2005), equation (2):

$$S(\mathcal{A}) = \log \sum_{x_1,\ldots,x_L} \frac{P(x_1,\ldots,x_L|\text{co-emitted on } \mathcal{A})}{P(x_1,\ldots,x_L|\text{Null})} \,. \tag{5.3}$$

The sum runs over all sequences $x_1,\ldots,x_L$ co-emitted along the alignment $\mathcal{A}$ of q and p (e.g. $L = 7$ in Figure 5.4). The numerator is the probability that $x_1,\ldots,x_L$ is co-emitted by both PO-HMMs along the alignment path and the denominator is the probability of the standardly used null model $P(x_1,\ldots,x_L|\text{Null}) = \prod_{l=1}^{L} f(x_l)$, where $f(a)$ are the fixed nucleotide background frequencies.

In order to apply dynamic programming in the Forward and Backward algorithm, we need to be more explicit about what equation (5.3) means in terms of PO-HMM probabilities. Let the two PO-HMMs q and p have probabilities $q_i(a)$ and $p_j(a)$ to emit nucleotide a in match state i or j and transition probabilities $q_i(X,X')$ and $p_j(Y,Y')$ to go from state X or $Y \in \{M,I\}$ in column i or j to a state X' or $Y' \in \{M,I\}$. Furthermore, let $q(i,i')$ and $p(j,j')$ be *edge probabilities* to go from node i or j to a node i' and j', irrespective of source and target state. Thus, the probability of going from X_i to $X'_{i'}$ is just $q_i(X_i,X'_{i'}) = q_i(X,X') \times q(i,i')$ for $(X,X') \in \{(M,M),(I,M)\}$, as illustrated in Figure 5.5. Insert states emit nucleotides according to the fixed nucleotide background frequencies $f(a)$. Suppose we are given an alignment of q and p, or rather the path $\mathcal{A}$ through the two PO-HMMs (Figure 5.4). We define K as the number of columns of the alignment of q with p. Let $X_k, Y_k \in \{M,I\}$ be the states in q and p in the kth column of the pairwise alignment of q and p and let $i(k)$ and $j(k)$ be the respective nodes from q and p. For the residues $x_l (l = 1,\ldots,L)$ emitted

along the path, we define $q^{\mathcal{A}}_{k(l)}(a)$ and $p^{\mathcal{A}}_{k(l)}(a)$ as the emission probabilities from q and p. More explicitly, $q^{\mathcal{A}}_k(a) = q_{i(k)}(a)$ for $X_k = M$ and $q^{\mathcal{A}}_k(a) = f(a)$ for $X_k = I$. Finally, we define $P_{tr}(\mathcal{A})$ as the product of all transition and edge probabilities for the path through p and q. In an analogous manner to Söding (2005), equation (3), we can show that the log-sum-of-odds score in equation (5.3) simplifies to

$$S(\mathcal{A}) = \sum_{k:X_kY_k=MM} S_{\text{col}}(q_{i(k)},p_{j(k)}) \; + \; \log P_{tr}(\mathcal{A})\,. \tag{5.4}$$

The sum runs over all alignment columns k with aligned Match-Match states $X_kY_k = MM$. We have used the column score

$$S_{\text{col}}(q_i,p_j) = \log \sum_{a=1}^{4} \frac{q_i(a),p_j(a)}{f(a)} \tag{5.5}$$

that measures the similarity of match state i of q and match state j of p. It is a weighted version of the co-emission probability, with weights being equal to $1/f(a)$. The background frequencies $f(a)$ have entered through the random sequence model probability for the co-emitted sequence, $P(x_1,\ldots,x_L|\text{Null}) = \prod_l f(x_l)$. Their effect is to up-weight rare nucleotides in the column score. This makes sense since rare nucleotides are less likely to be co-emitted by chance.

We now define the Forward and Backward partition functions in analogy to the Forward and Backward probabilities for the case of global HMM-sequence alignment (Durbin et al., 1998):

$$F_{XY}(i,j) = \sum_{\mathcal{A}\in\mathcal{F}_{i,j}} e^{\beta S(\mathcal{A})} \tag{5.6}$$

and similarly

$$B_{XY}(i,j) = \sum_{\mathcal{A}\in\mathcal{B}_{i,j}} e^{\beta S(\mathcal{A})}. \tag{5.7}$$

Here, $\mathcal{F}_{i,j}$ is the set of all global alignments between $q_{1\ldots i}$ and $p_{1\ldots j}$ ending in the aligned pair state (X^q_i,Y^p_j) for a given pair state $XY \in \{MM,\, MI,\, IM\}$. Similarly, $\mathcal{B}_{i,j}$ is the set

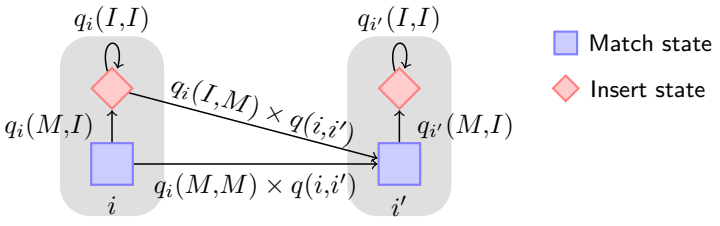


Figure 5.5.: Description of PO-HMM transition probabilities. The probability of going from state X_i to state $X'_{i'}$ for $(X,X') \in \{(M,M),(I,M)\}$ is the product of the state transition probability $q_i(X,X')$ and the edge probability $q(i,i')$.

of all global alignments between $q_{i+1\ldots L_q}$ and $p_{j+1\ldots L_p}$ starting *after* the aligned pair state (X_i^q,Y_j^p). From these definitions and equation (5.1), the posterior probability for pair state (X_i^q,Y_j^p) to be part of an alignment between q and p is

$$P(X_i^q \diamond Y_j^p \,|\, q,p) = \frac{F_{XY}(i,j) \times B_{XY}(i,j)}{Z} \,. \tag{5.8}$$

The *match* posterior probability $P(M_i^q \diamond M_j^p \,|\, q,p)$ quantifies how probable it is that match state i in PO-HMM q is aligned to match state j in PO-HMM p, while the *gap* posterior probability $P(M_i^q \diamond I_j^p \mid q,p)$ assigns a probability to the event that match state i in q is aligned to a gap. Furthermore, Z can be expressed in terms of the Forward partition function at the cell corresponding to the `END` nodes of PO-HMMs q and p:

$$Z = F_{MM}(L_q + 1,L_p + 1). \tag{5.9}$$

To compute the Forward partition functions F_{XY}, we need three dynamic programming matrices F_{XY}, one for each pair state $XY \in \{MM,\ MI,\ IM\}$. We begin the algorithm by initializing the top row and left column of the $F_{MM}(i,j)$ matrix to 0. Only cell $F_{MM}(0,0)$ corresponding to the `START` nodes in both PO-HMMs is set to 1. Then we proceed to fill the matrix recursively from top left to bottom right using the recursion relations given below. Note that although a PO-HMM is not a linear chain of nodes, such as a sequence or profile for example, it is still possible to fill the dynamic programming matrices recursively from top to left if we sort the nodes in *topological order*. A topological sort is simply a linear ordering of the PO-HMM nodes in which each node comes before all nodes to which it has outgoing edges (see Figure 5.6A).

$$\begin{aligned} F_{MM}(i,j) = S_{\mathrm{col}}(q_i,p_j) \sum_{i',j'} \Big[q(i',i)p(j',j) \,\big[& F_{MM}(i',j')q_{i'}(M,M)p_{j'}(M,M) \\ & + F_{MI}(i',j')q_{i'}(M,M)p_{j'}(I,M) \\ & + F_{IM}(i',j')q_{i'}(I,M)p_{j'}(M,M) \big] \Big] \end{aligned} \tag{5.10}$$

$$\begin{aligned} F_{MI}(i,j) = \sum_{i'} \Big[q(i',i) \,\big[& F_{MM}(i',j)q_{i'}(M,M)p_j(M,I) \\ & + F_{MI}(i',j)q_{i'}(M,M)p_j(I,I) \big] \Big] \end{aligned} \tag{5.11}$$

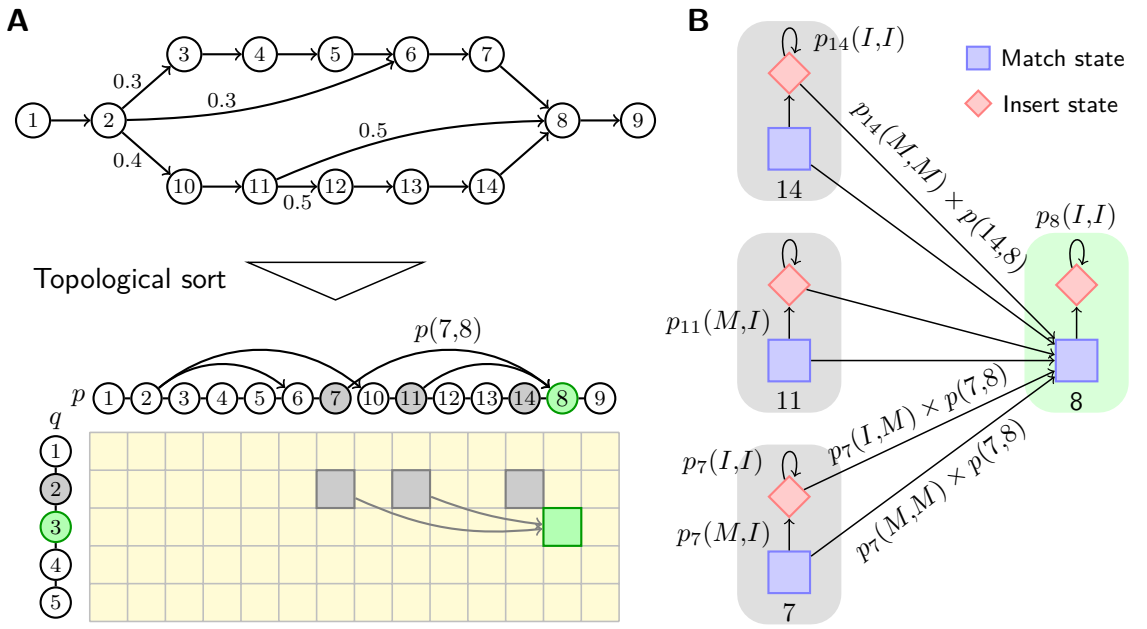


Figure 5.6.: Pairwise alignment of two PO-HMMs q and p using dynamic programming. (**A**) Dynamic programming matrix of Forward algorithm with PO-HMM q on the left and p at the top. The matrix entries are filled from top left to bottom right in topological order of the nodes in q and p. For instance, the value of the green cell is recursively calculated based on the values of the gray cells, which have already been filled in. (**B**) Possible state transitions at node 8 and its predecessors in PO-HMM p.

$$F_{IM}(i,j) = \sum_{j'} \Big[p(j',j) \left[F_{MM}(i,j') q_i(M,I) p_{j'}(M,M) + F_{IM}(i,j') q_i(I,I) p_j(M,M) \right] \Big] \tag{5.12}$$

The sum $\sum_{i',j'}$ runs over all pairs of nodes i' or j' that have outgoing edges leading into nodes i or j. Figure 5.6A illustrates the recursive calculation of the Forward matrix $F_{MM}(i,j)$. Node 8 (green) in PO-HMM p has three predecessor nodes (gray). Thus, the value of cell $F_{MM}(3,8)$ (green cells) is recursively calculated from three matrix cells (gray cells). The mathematical notation for transition probabilities in equations (5.10), (5.11) and (5.12) follows the labels in Figure 5.6B, which shows all possible transitions between states at node 8 and its predecessors. In an analogous manner to the Forward algorithm, our Backward algorithm recursively computes each $B_{XY}(i,j)$ starting from the bottom right of the matrix.

$$\begin{aligned} B_{MM}(i,j) = &\sum_{i',j'} B_{MM}(i',j') S_{\text{col}}(q_{i'}, p_{j'}) q_i(M,M) p_j(M,M) q(i,i') p(j,j') \\ &+ \sum_{i'} B_{MI}(i',j) q_i(M,M) p_j(M,I) q(i,i') \\ &+ \sum_{j'} B_{IM}(i,j') q_i(M,I) p_j(M,M) p(j,j') \end{aligned} \tag{5.13}$$

$$B_{MI}(i,j) = \sum_{i'j'} B_{MM}(i',j') S_{\text{col}}(q_{i'}, p_{j'}) q_i(M,M) p_j(I,M) q(i,i') p(j,j') + \sum_{i'} B_{MI}(i',j) q_i(M,M) p_j(I,I) q(i,i') \tag{5.14}$$

$$B_{IM}(i,j) = \sum_{i'j'} B_{MM}(i',j') S_{\text{col}}(q_{i'}, p_{j'}) q_i(I,M) p_j(M,M) q(i,i') p(j,j') + \sum_{j'} B_{IM}(i,j') q_i(I,I) p_j(M,M) p(j,j') \tag{5.15}$$

Here the sum $\sum_{i',j'}$ runs over all pairs of nodes i' or j' that have incoming edges from nodes i or j. Once Forward and Backward functions are calculated, the posterior probabilities can be found according to equations (5.8) and (5.9).

Maximum accuracy alignment

To derive an alignment from a posterior probability matrix, Holmes and Durbin (1998) proposed a maximum accuracy (MAC) algorithm, which maximizes the expected number of correctly aligned pairs of residues. Later, Schwartz *et al.* showed that approaches based on the expected accuracy alignment tend to maximize sensitivity at the expense of specificity (Schwartz et al., 2007). Thus, programs that are considered to be very sensitive may actually produce greedy alignments that do not distinguish between related (homologous) and unrelated sequence-regions. To solve this problem Schwartz *et al.* introduced a new accuracy measure for global multiple sequence alignment named *alignment metric accuracy* (AMA), which is based on a metric for multiple alignments. AMA is defined as the fraction of residues that are aligned correctly to another residue or to a gap.

Here, we adapt the AMA score to the case of pairwise alignment of PO-HMMs. A maximum AMA (MAMA) alignment of PO-HMMs q and p maximizes the expected number of correctly aligned resides (correctly aligned to insert states I or match states M):

$$\begin{aligned} f^{G_f}(\mathcal{A}) = & \sum_{k:X_kY_k=MM} P(M^q_{i(k)} \diamond M^p_{j(k)} \,|\, q,p) \,[n_q(i(k)) + n_p(j(k))] \\ & + G_f \sum_{k:X_kY_k=MI} P(M^q_{i(k)} \diamond I^p_{j(k)} \,|\, q,p) \, n_q(i(k)) \\ & + G_f \sum_{k:X_kY_k=IM} P(I^q_{i(k)} \diamond M^p_{j(k)} \,|\, q,p) \, n_p(j(k)) \to \max \ . \end{aligned} \tag{5.16}$$

Here, $P(X_i^q \diamond Y_j^p \,|\, q,p)$ denotes the posterior probability of state X_i in PO-HMM q to be aligned to state Y_j in PO-HMM p (see section 5.2.3 for details about posterior probabilities). $n_q(i(k))$ gives the number of residues at node $i(k)$ of q. Because posterior probabilities $P(X_i^q \diamond Y_j^p \,|\, q,p)$ are always positive, MAMA alignments maximizing $f^{G_f}(\mathcal{A})$ are always global. The objective function in equation (5.16) can be used to obtain a wide range of alignments, from the most

specific to the most sensitive. When the gap-factor parameter G_f is set to 0, $f^0(\mathcal{A})$ evaluates to the expected accuracy alignment score by Holmes and Durbin (1998), $f^{0.5}(\mathcal{A})$ evaluates to the expected AMA score. In general, setting G_f to higher values results in alignments with higher specificity, while reducing the value of G_f results in higher sensitivity.

The method to derive our global MAMA alignment is a modified version of the method proposed by (Holmes and Durbin, 1998), which uses the posterior probabilities as substitution scores:

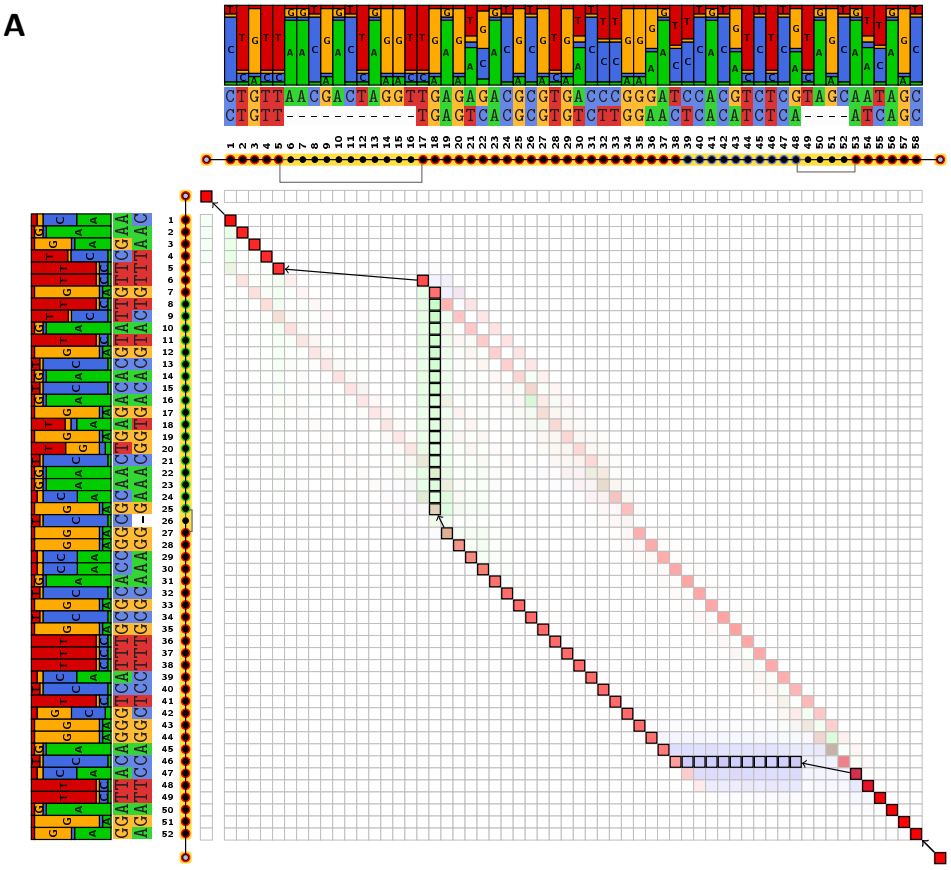
$$S(i,j) = \max \begin{cases} \max_{i',j'} S(i',j') + P(M_i^q \diamond M_j^p \,|\, q,p)\,[n_q(i) + n_p(j)] \\ \max_{i'} S(i',j) + G_f P(M_i^q \diamond I_j^p \,|\, q,p)\, n_q(i) \\ \max_{j'} S(i,j') + G_f P(I_i^q \diamond M_j^p \,|\, q,p)\, n_p(j) \end{cases} \tag{5.17}$$

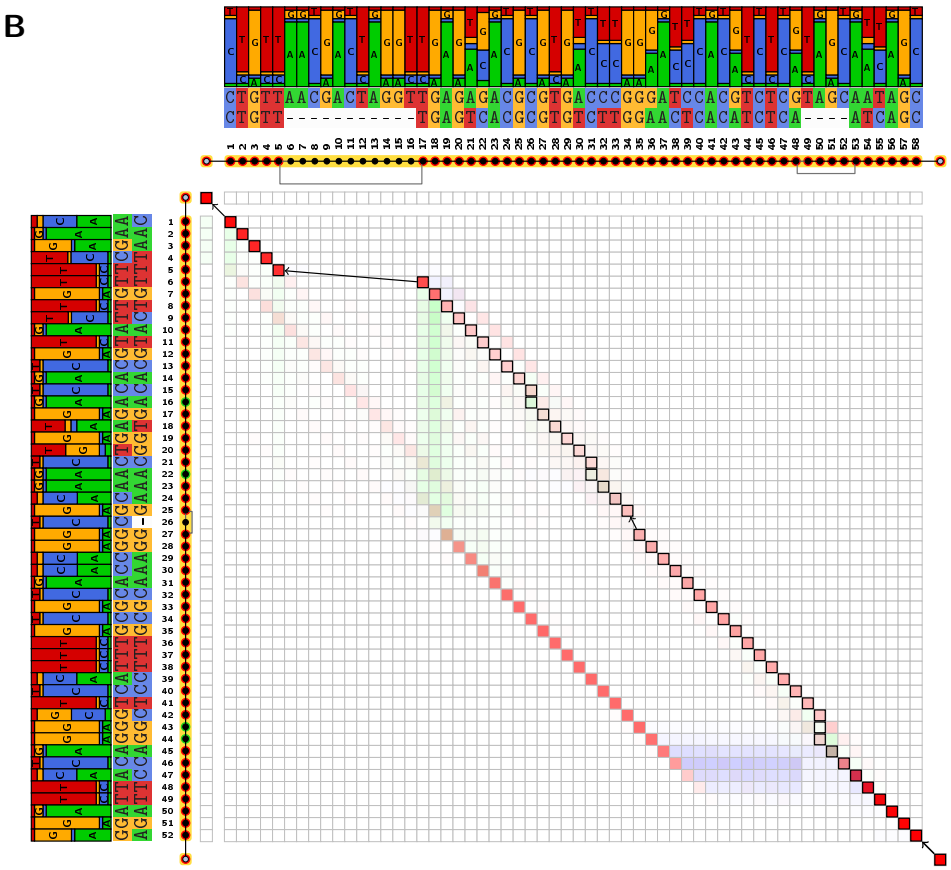
Similar to the recursion relations of the Forward algorithm given in equation (5.10), the maximization $\max_{i',j'}$ runs over all pairs of nodes i' or j' that have outgoing edges leading into nodes i or j. After matrix S has been filled, a standard traceback procedure will produce the best alignment. Figure 5.7 depicts two MAMA alignments (black squares) that have been computed from the three probability matrices with Match-Match (red), Match-Insert (green) and Insert-Match (blue) posterior probabilities. For the alignment shown in figure 5.7A, the gap factor parameter G_f was set to 0.5, which causes only positions with high confidence to be aligned. Consequently, the best alignment follows the "lower" path and aligns long regions of both PO-HMMs to gaps. The alignment in Figure 5.7B is based on the same posterior matrices but with gap-factor parameter G_f set to 0.01 This gives rise to a very greedy alignment, which follows the "upper" path through the matrix avoiding the placement of many gaps.

Generation of suboptimal alignments

Given that there are frequently alternative alignments with nearly the same probability as the best alignment, it is naturally of interest to compute all alternative alignment paths instead of choosing only the optimal alignment. Such alternative alignments are known as *suboptimal* alignments. In the setting of progressive multiple alignment, suboptimal alignments are particularly interesting. This is because a suboptimal alignment computed at an early stage in the guide tree may later be confirmed by more diverged sequences.

Figure 5.7. (following page): Probabilistic pairwise alignment of two PO-HMMs. The best pairwise alignment (black squares) has the maximum sum of posterior probabilities along its path. Cells are shaded from white to full saturation according to posterior probabilities for Match-Match, Match-Insert and Insert-Match alignment at this cell. Histograms above the partial order graphs illustrate the nucleotide emission probabilities of match states. (**A**) Optimal pairwise alignment with gap-factor parameter G_f set to 0.5 (eq. 5.16), which causes only positions with high posterior probability to be aligned. Consequently, the best alignment follows the "lower" path and aligns long regions of both PO-HMMs to gaps. (**B**) The same posterior probability matrix as in (A) but with gap-factor parameter G_f set to 0.01. This parameter setting results in a very aggressive alignment which follows the "upper" path through the matrix, avoiding the placement of many gaps.


A


B

However, to our knowledge, there is to date no progressive alignment method that preserves information about suboptimal alignment paths as sequences are progressively aligned in the order dictated by the guide tree. This may lead to alignment errors at early stages in the guide tree if the best pairwise alignment is arbitrary and many alternative alignments exist with almost equal probability. In the following, we will present a method that employs probabilistic alignment to generate all probable suboptimal alignments and represent them as a PO-HMM. This should prevent early alignment errors and allow more diverged sequences that are subsequently aligned to the PO-HMM to select which of the alignment alternatives was actually correct. First let us examine the characteristics of suboptimal alignments.

One class of suboptimal alignments with probabilities close to the probability of the optimal alignment will be those that only differ in a few positions from the optimal alignment. Because minor variations at different places in the alignment can be combined independently, the number of these "local" variants grows exponentially as the difference in probability from the optimal probability increases. Therefore, naive enumeration of all such variants is impractical. However, the flexibility in varying the alignment can differ substantially with position along the alignment. Posterior probabilities quantify for each cell in the dynamic programming matrix how "close" it is to being in the alignment. For instance, a cell in the dynamic programming matrix with a match posterior probability of 1.0 is certainly part of the alignment, whereas a match posterior probability of 0.1 suggests that the alignment path at that position is rather flexible. Another class of suboptimal alignment is the one that differs substantially from the optimal alignment (e.g. Figure 5.7). This is often the case when there are repeats in one or both of the sequences.

A number of different methods have been developed for searching suboptimal alignments. The most commonly used algorithm was first introduced by Waterman and Eggert (1987). Their algorithm is able to find the next best alignment that has no aligned pair of residues in common with any previously determined alignment. Once a top alignment has been obtained, the standard dynamic programming matrix is recalculated, ensuring that cells corresponding to residue pairs in already obtained alignments are set to zero (crossed out), which effectively prevents them from contributing to the next alignment. This procedure can be repeated until no more significant suboptimal alignments can be found. This amounts to approximating the posterior probability matrix by a sum of paths (alignments) with constant weights (probabilities). Although the Waterman & Eggert algorithm is widely used, it is not well suited to our case of pairwise PO-HMM alignment because it cannot deal with suboptimal alignments that are merely variants of the optimal alignment. Alignments inferred by the Waterman & Eggert algorithm are always distinct in all residue pairs.

Our method for generating suboptimal alignments - similar to the Waterman & Eggert algorithm - successively extracts suboptimal alignments but allows for shared pairs of aligned residues between the alignment paths. Thus, it is possible that the first and second suboptimal alignment may differ in the placement of one gap only. Furthermore, through the

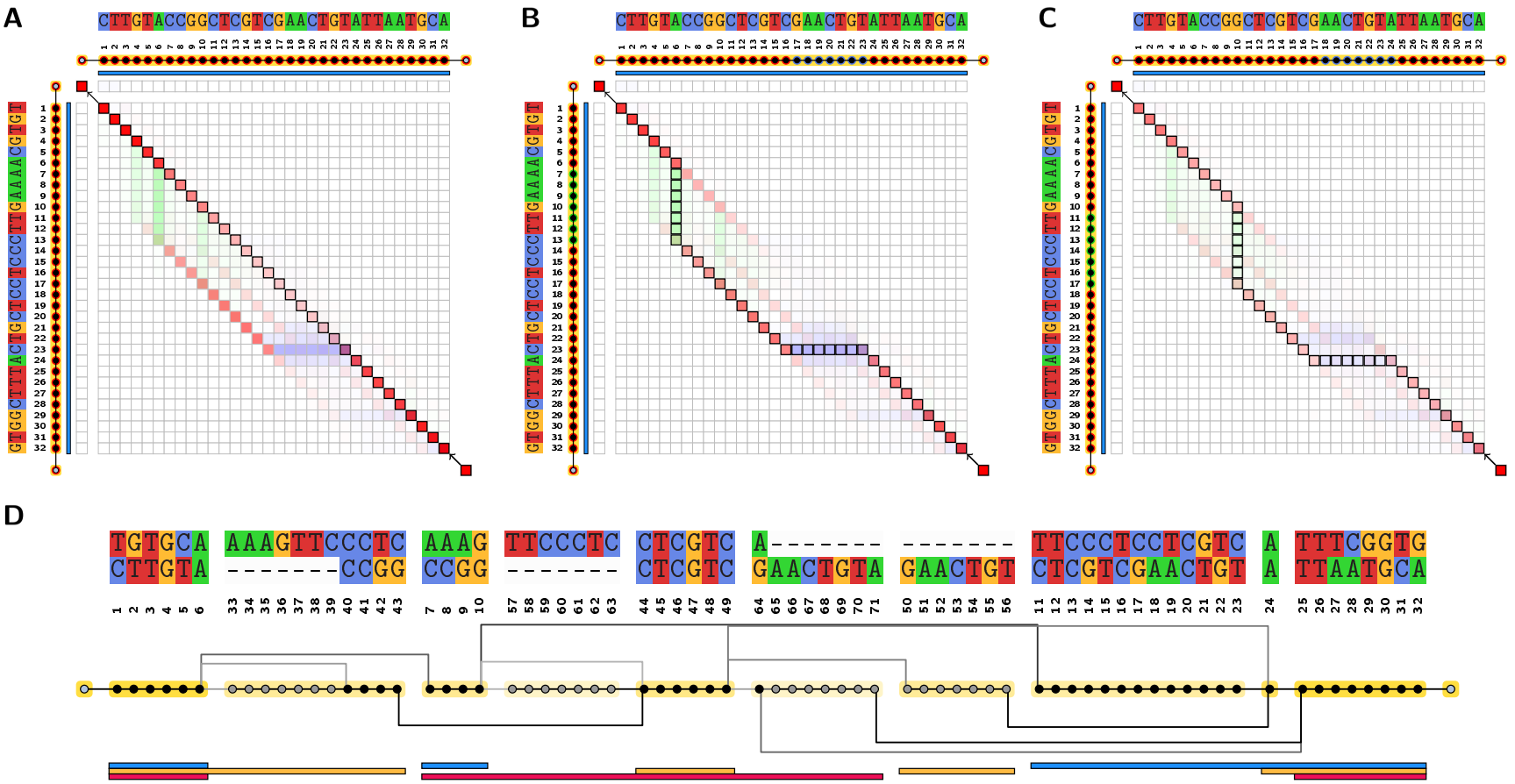


Figure 5.8.: Generation of suboptimal alignments in pairwise PO-HMM comparison. Matrix cells are shaded from white to full saturation according to posterior probabilities for Match-Match, Match-Insert and Insert-Match alignment at this cell. (**A**) The path of the optimal alignment follows the main diagonal (black squares). (**B**) To determine the next best suboptimal alignment, we subtract the minimal posterior probability of the optimal alignment from all cells along the optimal path. Thereby, Match-Match posterior probabilities on the main diagonal are reduced. Dynamic programming on the altered matrix determines the next best suboptimal alignment path (black squares). (**C**) The second suboptimal alignment is determined in the same way as in (B). (**E**) PO-HMM representing all four alignment paths shown in (A)-(C). Colored bars below the partial order graph indicate to which alignment a PO-HMM node belongs, i.e. optimal alignment (blue), first (yellow), or second suboptimal alignment (pink).

Algorithm 1: GENERATESUBOPTIMALALIGNMENTS($q, p, N_{\max}, P_{\min}$)
Generate suboptimal, pairwise alignments of PO-HMMs q and p. The algorithm successively extracts up to $N_{\max}$ suboptimal alignments, each with a minimum posterior probability of at least $P_{\min}$ along its path.

Input: q,p PO-HMMs to be aligned
$N_{\max}$ maximal number of suboptimal alignments to be extracted
$P_{\min}$ minimal posterior probability along each alignment
Output: merged PO-HMM r representing all alignments between q and p
$r \leftarrow \varnothing$
calculate $P(X_i^q \diamond Y_j^p), \forall i \in \{1,\ldots,L_q\}, j \in \{1,\ldots,L_p\}, XY \in \{MM,MI,IM\}$ with Forward-Backward algorithm
$P_{i,j} \leftarrow P(M_i^q \diamond M_j^p)$
for $n \leftarrow 1$ **to** $N_{\max}$ **do**
  determine next best alignment $\mathcal{A}_n$ on $P_{i,j}$ using eq. (5.17)
  $P_n \leftarrow$ minimum $P_{i,j}$ of all MM pair states along path of $\mathcal{A}_n$
  if $P_n < P_{\min}$ **then break**
  add aligned nodes in $\mathcal{A}_n$ to PO-HMM r
  subtract P_n from all cells $P_{i,j}$ with MM pair states in $\mathcal{A}_n$

fully probabilistic treatment of alignments, our approach is able to represent the inferred alignment paths within a single PO-HMM in a way that is ideal from a statistical as well as an algorithmic perspective. The pseudocode of the procedure for generating suboptimal alignments is given in Algorithm 1.

Figure 5.8 illustrates the calculation of one optimal and three suboptimal alignments between two PO-HMMs. After the posterior probability matrices for Match-Match (red), Match-Insert (green) and Insert-Match (blue) have been calculated with the Forward-Backward algorithm, the optimal alignment (Figure 5.8A) is retrieved by dynamic programming as in equation (5.17). As an approximation for the true probability of the optimal alignment, we use the minimum posterior probability P_1 of all MM cells in the alignment path. To account for the fact that the best alignment has already been retrieved, we then subtract this minimum probability P_1 from all cells in the match posterior matrix, for which the optimal alignment had a MM pair state. Dynamic programming on the altered probability matrices will then generate the next best alignment as illustrated in Figure 5.8A. This procedure is repeated until no more significant suboptimal alignments can be found.

5.2.4. PO-HMM construction

Given the path of a pairwise alignment between two PO-HMMs q and p, we would like to represent this alignment in the form of a new PO-HMM r. If nodes i in q and j in p were aligned with each other, they should be "merged" together and represented as *one* node in r. First, let us consider how to construct the node and edge set of r given the pairwise alignment illustrated in Figure 5.9A. The construction algorithm for PO-HMM r proceeds

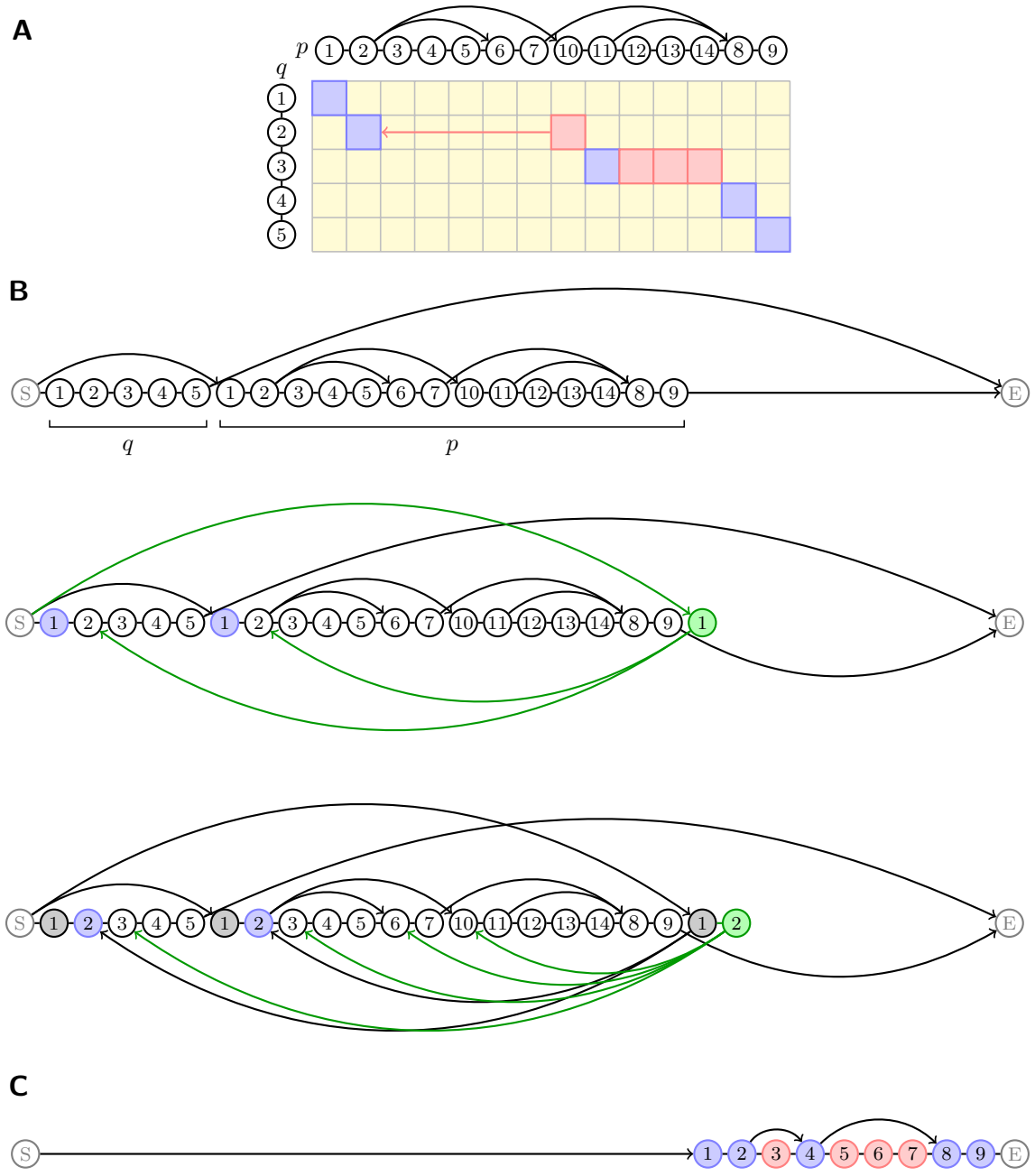


Figure 5.9.: Construction of a new PO-HMM based on the pairwise alignment of PO-HMMs q and p. (**A**) Alignment path through dynamic programming matrix with aligned Match-Match states indicated in blue and Insert-Match states in red. Note that PO-HMM p contains two alternative segments (nodes 3-7 and 10-14). (**B**) Procedure of PO-HMM construction. At the beginning, PO-HMMs q and p are stored as *one*, combined PO-HMM with q and p representing alternative branches. For each Match-Match alignment step, a new node (green) is added to the combined PO-HMM by merging the two aligned nodes from q and p. Also, all incoming and outgoing edges of the aligned pair of nodes are copied to the new node (green edges). (**C**) To obtain the final PO-HMM, all old nodes of PO-HMMs q and p are discarded and only the newly added nodes representing Match-Match (blue) and Insert-Match pair states (red) remain. Note that nodes 3-7 in p have not been merged into the final PO-HMM because these nodes were not part of the pairwise alignment.

in two parts. At the beginning, all nodes and edges in PO-HMMs q and p are temporarily added to the empty PO-HMM r (see Figure 5.9B top). In the main construction stage, we create a new node (green) in r for each pair state in the alignment path and update the edge set of r accordingly. More precisely, for a Match-Match pair state with aligned nodes i and j, we create a new node k by merging nodes i and j followed by copying all incoming and outgoing edges of nodes i and j to node k (green edges). In case of a Match-Insert or Insert-Match pair state, we copy the matched node into a new node k along with its incoming and outgoing edges. To obtain the final PO-HMM r, all temporary nodes and edges of PO-HMMs q and p are discarded. Only the newly created nodes and edges between them remain (see Figure 5.9C).

In the example of Figure 5.9, only a single alignment between PO-HMMs q and p was encoded in r. The above construction scheme can be easily extended to the case of multiple, alternative pairwise alignments. Here, we would like to obtain a condensed PO-HMM representation, in which only those aligned states in the dynamic programming matrix that differ between the alternative alignment paths are stored separately, while identical aligned states are represented only once. We can do so by keeping track of all aligned states that have already been stored in PO-HMM r and creating new nodes k only if the corresponding aligned states are not already part of r. In addition, we also keep track of which nodes belong to which suboptimal alignments (see Figure 5.8E) as this information is needed for the calculation of edge probabilities.

Calculation of edge-weights

After we have constructed the node and edge set of the new PO-HMM r, we need to calculate edge probabilities for all edges in r. For the calculation of an edge probability $r(i,i')$ between nodes i and i', we need to take into account two factors: first, the fraction of sequences that contribute to the edge, i.e. what proportion of sequences with an aligned residue at node i also have an aligned residue at node i', and second, the probability of branching off into alternatively aligned segments.

Let us assume that we have already calculated sequence weights w_k for each sequence k that is part of r, using the sequence weighting scheme of Henikoff and Henikoff (1994). For each node i, we first sort all out-edges $e_{ii'}$ of i by ascending topological index of the target node i' (the position of node v in the topologically sorted PO-HMM). Furthermore, we define $\mathcal{S}_j$ as the set of all sequences that have an aligned residue at node j and initialize the set of all "active" sequences $\Omega = \mathcal{S}_i$. We iterate over the sorted out-edges $e_{ii'}$ of i and set

$$r(i,i') = \frac{\sum_{k \in \mathcal{S}_{i'} \cap \Omega} w_k}{\sum_{l \in \mathcal{S}_i} w_l} \times f(i,i'). \tag{5.18}$$

After each assignment of $r(i,i')$, we remove all sequences with a residue at node i' from the active set: $\Omega = \Omega - \{k \in \mathcal{S}_{i'}\}$. The factor $f(i,i')$ accounts for the probability of going from

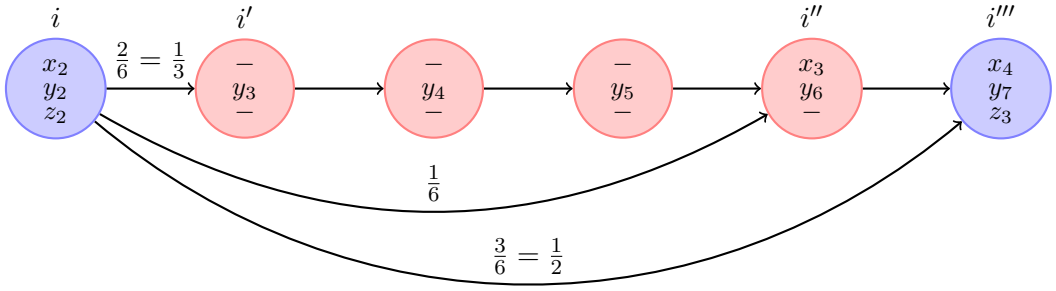


Figure 5.10.: Assignment of edge probabilities for a hypothetical PO-HMM consisting of three sequences x,y,z with sequence weights $w_x = 1, w_y = 2, w_z = 3$.

node i to node i' based on the probabilities of alternative alignments. We can calculate $f(i,i')$ from the probability P_n of each alternative alignment n represented in the PO-HMM and the set of all alignment paths $\mathcal{A}_j$ going through a node j:

$$f(i,i') = \frac{\sum_{n \in \mathcal{A}_i \cap \mathcal{A}_{i'}} P_n}{\sum_{m \in \mathcal{A}_i} P_n}. \tag{5.19}$$

Figure 5.10 illustrates the assignment of edge probabilities for a hypothetical PO-HMM consisting of three sequences x,y,z with sequence weights $w_x = 1, w_y = 2, w_z = 3$. In Figure 5.3B, the upper two outgoing edges at node i belong to alternative alignment 1, the lower outgoing edge to alternative alignment 2. In this example, $f(i,i') = 0.6$ and $f(i,i'') = 0.4$.

Match state emission probabilities

To make use of the column score (see equation 5.5) in pairwise PO-HMM alignment, we need to assign nucleotide emission probabilities to each PO-HMM match state. Following a standard procedure, we use maximum-likelihood estimation to calculate $q_i(a)$, the probability that nucleotide a is emitted by match state i in PO-HMM q:

$$q_i(a) = \frac{n_i(a)}{\sum_b n_i(b)}. \tag{5.20}$$

Here, $n_i(a)$ is the sum of sequence weights w_k over all sequences k that have an aligned nucleotide a at node i. As always, maximum likelihood estimators are vulnerable to overfitting if there are insufficient data. Indeed, if the nucleotide a was never aligned in state i, then the emission probability $q_i(a)$ would be zero. To avoid such problems, we add pseudocounts to PO-HMM emission probabilities with a substitution matrix method similar to PSI-BLAST using the Tamura & Nei substitution matrix (Tamura and Nei, 1993).

5.2.5. Multiple alignment of genomic sequences with PO-HMMs

Having presented algorithms for pairwise PO-HMM comparison and computation of suboptimal alignments, we now turn to the overall algorithm for generating multiple alignments, which has been implemented in the program CS-ALIGN. The basic strategy used by CS-ALIGN is similar to that used by MUSCLE (Edgar, 2004) and MAFFT (Katoh et al., 2002). CS-ALIGN progressively builds up a multiple alignment along a guide tree, but unlike MUSCLE and MAFFT it does not apply horizontal refinement afterwards. Another difference is that CS-ALIGN has been designed to take advantage of the structural and evolutionary properties of regulatory sequences. It takes as input not only one but *multiple* sets of orthologous sequences as this procedure allows CS-ALIGN to make use of the sequence signature of conserved regulatory motifs in the alignment process.

In the following we give an overview of the basic steps used in the CS-ALIGN algorithm, followed by a more detailed discussion of specific parts of the method. CS-ALIGN proceeds in three main stages. The first stage builds up a draft progressive alignment for each set of orthologous sequences. This stage emphasizes speed over accuracy. Therefore, CS-ALIGN uses fast k-mer distances for the calculation of the initial guide tree. The goal of the second stage is to learn the sequence signature of regulatory motifs that are present in the draft alignments. More precisely, CS-ALIGN learns a library with hundreds of *context profiles*, generated by clustering all sequence profile windows in the draft alignments. The third stage builds improved progressive alignments but tries to prevent some of the errors made in the draft progressive stage, using more sensitive Kimura distances instead of k-mer distances to build the guide tree, and using context-context alignment match scores and context-specific pseudocounts calculated from the library of context profiles.

Distance measures and guide tree construction

Similar to MUSCLE, CS-ALIGN uses two kinds of distance measures to construct the matrix of pairwise distances needed for the construction of the evolutionary guide tree: k-mer distances and Kimura distances. The former are obtained by k-mer counting in two unaligned sequences while the latter requires a global alignment between two sequences. A k-mer is a contiguous subsequence of length k and related sequences tend to have more k-mers in common than to be expected by chance. The primary motivation for k-mer counting is improved speed compared to distance measures that require an alignment. CS-ALIGN uses the following similarity measure as a proxy for the fractional identity between to sequences x and y:

$$F = \frac{1}{\min(L_x,L_y) - k + 1} \sum_{\tau} \min\left(N_x(\tau), N_y(\tau)\right). \tag{5.21}$$

Here, τ is a k-mer, L_x, L_y are the sequence lengths, and $N_x(\tau), N_y(\tau)$ are the number of times τ occurs in x and y respectively. The k-mer-based distance estimate for sequences x

and y is set to

$$d_{\text{kmer}}(x,y) = 1 - F. \tag{5.22}$$

Alternatively, given an alignment of sequences x and y, we can determine their fractional identity D directly from the alignment. For closely related sequences $1 - D$ is a good approximation to the mutation distance, i.e. the number of mutations that occurred on the evolutionary path between the two sequences. To correct for the increasing probability of multiple mutations at a single site as sequences diverge, we use the following distance estimate of Kimura (1991):

$$d_{\text{Kimura}}(x,y) = -\log_e(1 - D - D^2/5). \tag{5.23}$$

Given a matrix of pairwise distances, a binary tree is constructed by hierarchical clustering with UPGMA. There are three variants of UPGMA that differ in their assignment of distances to new clusters. We have implemented the "standard version" in which the distance between any two clusters $\mathcal{A}$ and $\mathcal{B}$ is taken to be the average of all distances between pairs of sequences x in $\mathcal{A}$ and y in $\mathcal{B}$ and equation (5.22) or (5.23) is used for the distances:

$$d(\mathcal{A},\mathcal{B}) = \frac{1}{|\mathcal{A}||\mathcal{B}|} \sum_{x\in\mathcal{A}} \sum_{y\in\mathcal{B}} d(x,y). \tag{5.24}$$

Progressive PO-HMM alignment

The progressive PO-HMM alignment algorithm takes as input both a set of sequences and their estimated evolutionary tree. The application of progressive PO-HMM alignment to a set of four hypothetical sequences is shown in Figure 5.11. The construction of the multiple sequence alignment follows the order indicated by the guide tree, where sequences that are most similar to each other will be aligned first. The progressive PO-HMM alignment differs from existing progressive alignment methods by the fact that alignments at the branch points are not reduced to linear sequence profiles. Instead, they are stored as PO-HMMs, which are aligned to each other using the pairwise PO-PO alignment algorithm explained in section 5.2.3. Another important difference is that at each branch point, CS-ALIGN computes not only the optimal alignment but *all* suboptimal alignments with a probability greater than P_{min}. The suboptimal alignment paths are stored as PO-HMM with probabilities of the suboptimal alignments being modeled by edge probabilities of nodes leading into alternatively aligned segments in the PO-HMM. For sequences with many equally probable alignments, a suboptimal alignment may turn out to be actually correct as more diverged sequences are aligned to the growing PO-HMM. For instance, Figure 5.11D illustrates how the alignment of sequence 3 actually confirms that the suboptimal alignment between sequences 1 and 2 shown in Figure 5.11B is actually correct. Thus, we would have made an alignment error if we had stored only the optimal alignment between sequences 1 and 2.

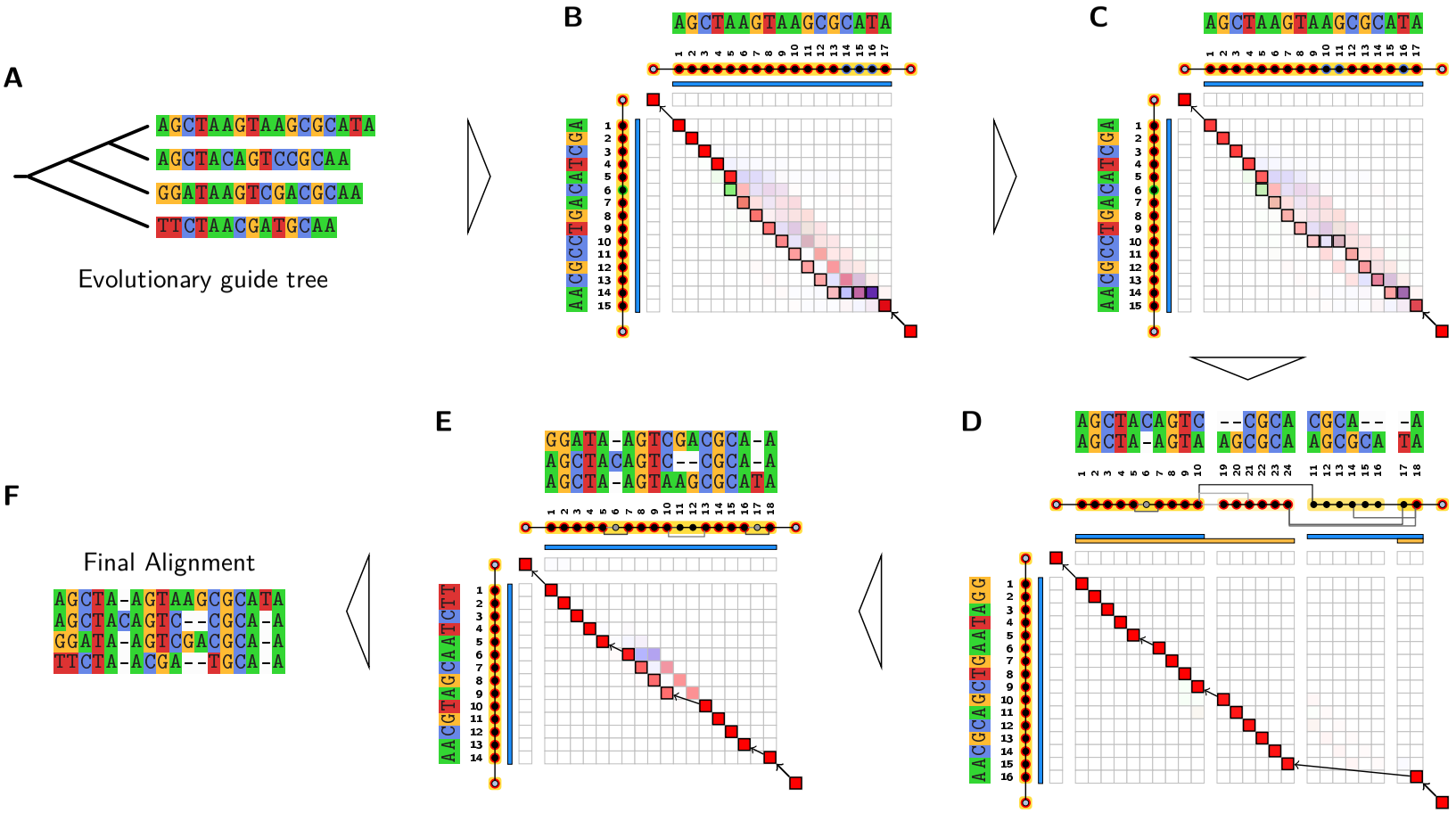


Figure 5.11.: Ambiguity during initial PO-HMM alignment steps can be resolved at a later stage. (**A**) Evolutionary guide tree indicating the progressive alignment order of four hypothetical sequences. (**B**) Optimal pairwise alignment of sequence 1 (top) and 2 (left). The region with a fluffy posterior probability distribution indicates multiple alignment paths with almost equal score. (**C**) Next best suboptimal alignment between sequences 1 and 2 with different gap placement. (**D**) The alignment between sequence 3 (left) and the PO-HMM of sequences 1 and 2 (top) reveals that the suboptimal alignment shown in (C) is actually correct. (**E**) Progressive alignment step between sequence 4 (left) and the PO-HMM of sequences 1, 2 and 3 (top). The final multiple alignment is shown in (**F**).

Context-specific pseudocounts for PO-HMMs

Before the pairwise alignment of two PO-HMMs q and p in the draft progressive stage, CS-ALIGN adds nucleotide pseudocounts to both PO-HMMs with a substitution matrix method similar to PSI-BLAST, employing the Tamura & Nei matrix (Tamura and Nei, 1993) in place of the BLOSUM matrix. In this scheme, the resulting emission probabilities $q_i(a)$ and $p_j(a)$ depend only on the nucleotides aligned in columns i and j of PO-HMMs q and p respectively. However, the context of the partial order alignment at position i may contain much more information than just the aligned residues at column i themselves about which nucleotides to expect in related sequences. This applies particularly to positions that are part of a regulatory motif, whose expected mutations generally deviate markedly from the background mutation rates in the Tamura & Nei matrix.

We have already demonstrated that we are able to significantly improve alignment quality in protein sequence comparison by employing context-specific similarities in CS-BLAST. The context-specific paradigm is likely to hold similar advantages for multiple alignment of nucleotide sequences with PO-HMMs. In the following, we show how to generalize the calculation of context-specific pseudocounts from the single sequence case in equation (2.3) to the case of PO-HMMs. In analogy to the sequence context X_i, we define the context at PO-HMM node i as the set of nodes $\mathcal{Q}_i$ such that the length $l(i,j)$ of the shortest path between any node $j \in \mathcal{Q}_i$ and i is at most $d = (l-1)/2$. Due to branches and jumping edges in the partial order graph there may be multiple nodes at distance d to any side of node i. $c_q(j,x)$ are the counts of aligned nucleotides at node j of the PO-HMM. These counts are obtained from the observed nucleotide frequencies $p(j,x)$ by multiplying with the effective number of sequences N_q in PO-HMM q: $c_q(j,x) = N_q p(j,x)$, where

$$N_q = \frac{1}{L_q} \sum_{i}^{L_q} \left(1 + N - N \sum_{x=1}^{4} p(j,x)^2 \right) \tag{5.25}$$

and N is the *total* number of sequences in q. We now merely need to show how to calculate the probability $P(\mathcal{Q}_i|p_k)$ for generating context $\mathcal{Q}_i$ given context profile p_k. All other

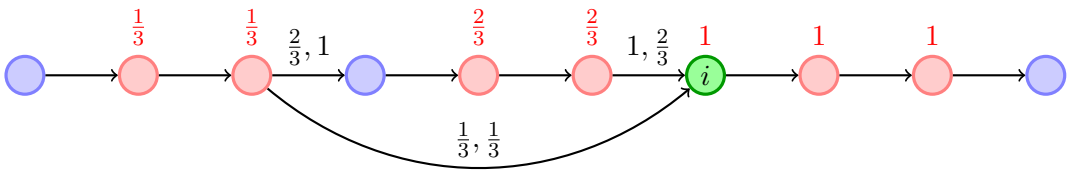


Figure 5.12.: Context window of length $l = 5$ in a PO-HMM with a jumping edge. The central node i is shown in green, nodes within the context window in red. Red labels above context nodes indicate context weights π_j. Edges are labeled with their forward and reverse probability (black numbers) where different from 1.0,1.0.

transformations leading to equation (2.3) remain essentially unchanged. As in the generalization of context-specific pseudocounts to sequence profiles, we model $P(\mathcal{Q}_i|p_k)$ using a multinomial distribution and replace the factorials by Gamma functions ($n! = \Gamma(n+1)$)

$$P(\mathcal{Q}_i|p_k) = \prod_{j \in \mathcal{Q}_i} \left(\frac{\Gamma(N_q+1)}{\prod_{x=1}^{4} \Gamma(c_q(l(i,j),x)+1)} \prod_{x=1}^{4} p_k(j,x)^{c_q(j,x)} \right)^{w_j \pi_j}. \tag{5.26}$$

Note that, since the factor containing the Gamma functions does not depend on k, it will cancel out during the normalization of $P(p_k|\mathcal{Q}_i)$. Furthermore, we have introduced a factor π_j to account for the probabilities of branching off into alternative context paths. For instance, Figure 5.12 illustrates the context nodes (red) within a context window of length $l = 5$ around node i (green). Each edge is assigned a forward as well as a backward probability and the weight factors π_j are indicated in red above each context node j. Note that for each node, the forward probabilities of all out-edges sum to 1 as do the backward probabilities of all in-edges. The algorithm that calculates $P(\mathcal{Q}_i|p_k)$ recursively explores the context neighborhood around node i to the right and to the left. To explore the context to the right of node i, it starts with $\pi_i = 1$ and follows the direction of graph edges. At each step from a current node j to a new node j', we set $\pi_{j'} = \pi_j \times q_{\text{fwd}}(j,j')$, where $q_{\text{fwd}}(j,j')$ is the forward probability of the edge connecting j and j'. Similarly, the context neighborhood to the left of node i is explored by following graph edges in backward direction setting $\pi_{j'} = \pi_j \times q_{\text{bwd}}(j,j')$ at each step.

Let us assume that we have already generated a library of K context profiles in the second stage of CS-ALIGN. In the third, improved progressive stage, we can now calculate context-specific mutation probabilities $P(y|\mathcal{Q}_i)$ by mixing the amino acid distributions $p_k(0,y)$ from the central columns of all K profiles with weights $P(p_k|\mathcal{Q}_i)$:

$$P(p_k|\mathcal{Q}_i) = \frac{P(\mathcal{Q}_i|p_k)P(p_k)}{P(\mathcal{Q}_i)} \propto P(p_k)P(\mathcal{Q}_i|p_k). \tag{5.27}$$

$$P(y|\mathcal{Q}_i) \propto \sum_{k=1}^{K} p_k(0,y)P(p_k|\mathcal{Q}_i). \tag{5.28}$$

Normalizing over all four nucleotides yields the final expected mutation probability $P(y|\mathcal{Q}_i)$. To have more flexibility in adjusting the diversity of the context-specific emission probabilities $q_i^{\text{cs}}(\cdot)$ in PO-HMM q, we mutate only a fraction $\tau \in [0,1]$ of the observed nucleotides $p(i,x)$ while leaving a fraction $1-\tau$ unchanged:

$$q_i^{\text{cs}}(y) = (1-\tau)p(i,y) + \tau P(y|\mathcal{Q}_i). \tag{5.29}$$

The pseudocount admixture τ needs to be optimized depending on the evolutionary distance of the two PO-HMMs to be aligned, in a similar way as the substitution matrix with optimum diversity might be chosen. We choose the pseudocount admixture depending on the diversity

of the alignment underlying the PO-HMM, $\tau = a(b+1)/(b+N_q(i))$, where $a = 0.6$ and $b = 5.0$ have been determined on a small optimization set (see section 5.3.1).

Context-context alignment match scores

In addition to context-specific pseudocounts for PO-HMM emission probabilities, we have implemented a second approach for taking context information into account in PO-HMM alignment. The idea is to calculate the similarity of two PO-HMM nodes not only based on the column score given in equation (5.5), but also by comparing posterior context probabilities $P(p_k|\mathcal{Q}_i)$ and $P(p_k|\mathcal{P}_j)$ of PO-HMMs q and p directly.

To understand the reasoning behind this context-context scoring, let us consider a short example. The canonical promoter illustrated in Figure 5.13A contains a TATA-Box and an Initiator motif, both containing a conserved adenine. Assuming both motifs are represented in our library of context profiles (Figure 5.13B), context-specific pseudocounts would correctly predict high adenine emission probabilities for both conserved positions, which in turn would give a high column score if the two motif positions were aligned. Of course, the high column score is not justified since both positions belong to different motifs and are not homologous. The actual problem is that the column score is unable to discern between columns with equal nucleotide emissions within *different* sequence contexts since it only considers the nucleotide emission probabilities. We therefore propose a *context-context* match score by which we compare posterior context probabilities $P(p_k|\mathcal{Q}_i)$ and $P(p_k|\mathcal{P}_j)$ in the alignment of PO-HMMs q and p

$$M_{\text{ctx}}(q_i,p_j) = \log\left(\sum_{k=1}^{K} \frac{P(p_k|\mathcal{Q}_i)P(p_k|\mathcal{P}_j)}{P(p_k)} \times \text{cons}(p_k)\right) \tag{5.30}$$

where $\text{cons}(p_k) \in [0,1]$ is the median FRcons conservation score (Fischer et al., 2008) over all columns j in context profile p_k

$$\text{cons}(p_k,j) = \frac{\log \sum_{x=1}^{4} p_k(j,x)^2/P(x)}{\log \sum_{x=1}^{4} p_k(j,x)/P(x)}. \tag{5.31}$$

If we omitted the factor $\text{cons}(k)/P(p_k)$ in equation (5.30), we would obtain the logarithm of the co-emission probability over all contexts k. In this respect, $\text{cons}(p_k)/P(p_k)$ can be interpreted as weighting factors to the co-emission probability. They increase the weight of rare and conserved contexts. This makes sense since co-emission of rare contexts is harder to produce by chance and regulatory motifs are often highly conserved. The context-context match score for q_i and p_j utilized during dynamic programming is obtained by multiplying M_{ctx} with a weight w_{ctx},

$$S_{\text{ctx}}(q_i,p_j) = w_{\text{ctx}} M_{\text{ctx}}(q_i,p_j). \tag{5.32}$$

This score is added to the nucleotide column score in equation (5.5). The weight coeffi-

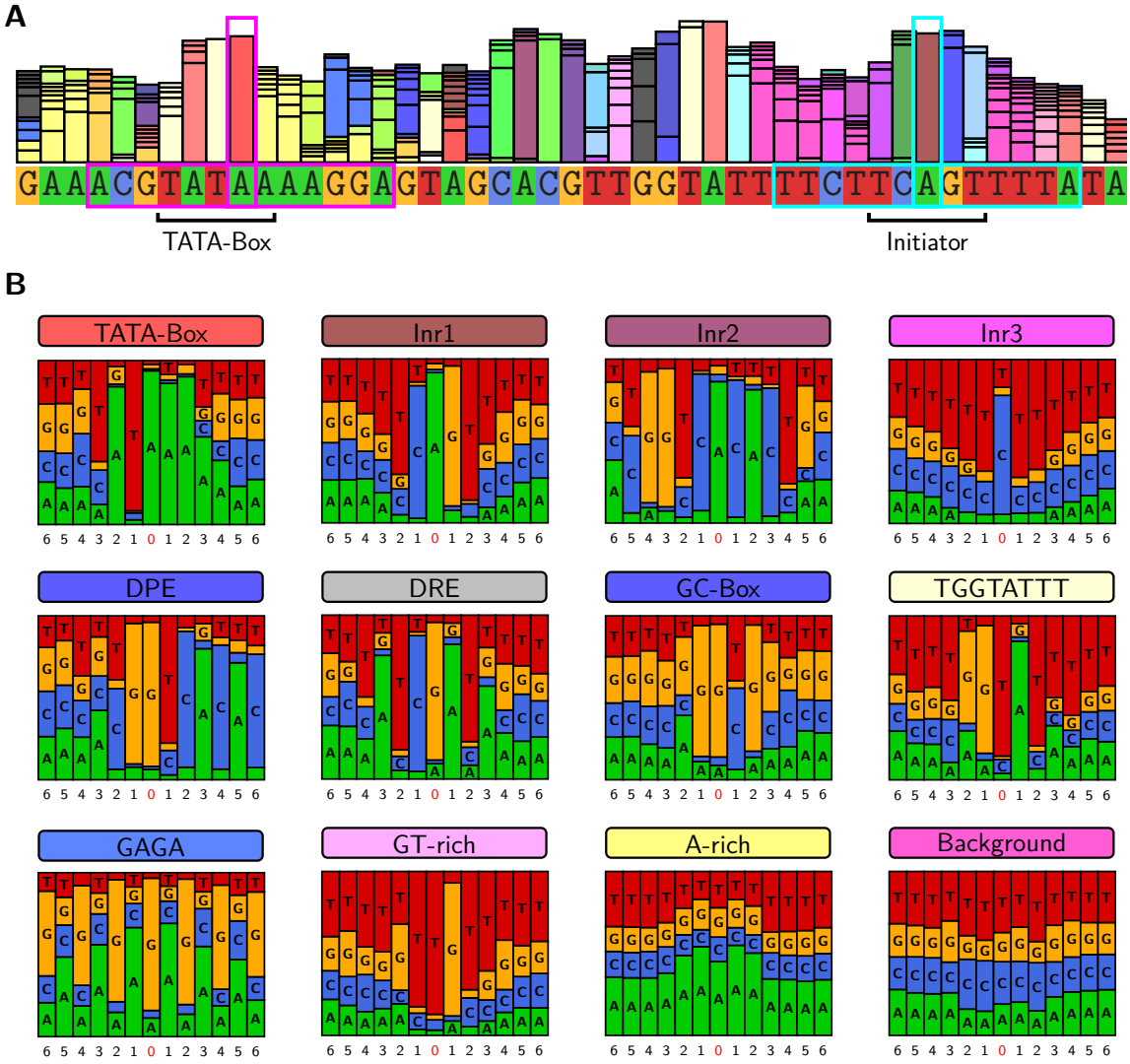


Figure 5.13.: CS-ALIGN uses context-aware alignment match scores that depend on context windows around each DNA residue. (**A**) For each sequence position we calculate a profile column (colored histogram) by comparing its sequence window to a library with hundreds of *context profiles* shown in (**B**), generated by clustering a large, representative set of sequence-profile windows. Each profile column contains the likelihood of each context profile at that specific position. The histogram colors correspond to the background colors of context names in the profile library. The similarity between two profile columns is utilized as context-context score in pairwise PO-HMM alignment.

cient w_{ctx} accounts for the fact that the context probabilities are not independent of their neighbors. Note that for the efficient evaluation of context-context profile match scores, it is necessary to pre-calculate the posterior context probabilities before the pairwise alignment of two PO-HMMs q and p. In addition, we take advantage of the sparse distribution of context probabilities by storing only those $P(p_k|\mathcal{Q}_i)$ with probability greater than 0.01. Let us again have a look at Figure 5.13A to understand how our context-context score

evaluates the alignment of the two conserved adenine positions in the TATA-Box and an initiator motifs. The posterior probability vectors $P(\cdot|\mathcal{Q}_i)$, depicted as histogram above the sequence, show a markedly different posterior probability distribution at the two adenine positions (magenta and blue box) reflecting their different sequence context. This in turn results in a low context-context match score which effectively penalizes the alignment of the two positions.

5.3. Results

To assess the alignment performance of the CS-ALIGN algorithm in comparison to other published multiple programs, we have devised a multiple alignment benchmark based on simulated promoter sequences, since alignments of regulatory regions are particularly interesting with regard to comparative genomics. For protein-coding sequences, often additional information is available in the form of the three dimensional structure of the protein sequences that can be used to generate gold-standard alignments to assess the quality of predicted alignments. This approach is not possible for noncoding DNA because here no gold-standard alignments are available to evaluate alignment quality. Therefore, we follow a widely adopted simulation-based benchmark approach (Lunter et al., 2008; Pollard et al., 2004, 2006; Rosenberg, 2005). The idea is to simulate sequence divergence *in silico* to obtain sequences that are related by a known reference alignment. It has to be kept in mind, however, that the validity of simulation-based alignment benchmarks is highly dependent on "realistic" models and parameters that reflect the underlying evolutionary processes.

5.3.1. Benchmark sets

For the generation of gold-standard reference alignments, we use the simulation program CIS-EVOLVER (Pollard et al., 2006), that incorporates a key characteristic of noncoding regulatory DNA: it can evolve a mixture of background genomic DNA sequences and transcription factor binding sites. CIS-EVOLVER takes as input a mutation guide tree and an ancestral sequence along with the sequence positions and PWMs of all transcription factor binding sites (TFBS) within that sequence. The ancestral sequence is evolved from the root down the branches of the tree using at each sequence site either a background or a binding-site mutation model. Non-binding site regions mutate according to the Hasegawa-Kashina-Yano (Hasegawa et al., 1985) substitution model whereas binding sites evolve according to the Halpern-Bruno model of position specific substitution rates (Halpern and Bruno, 1998), which requires conserved positions in a binding site to evolve more slowly and more specifically according to the PWM of the binding site. The result is an alignment of the leaf sequences generated by the simulation. For instance, Figure 5.14A shows the PWMs of transcription factors INO2, GLN3, DAL80, PHD1 and HAP1 in histogram representation. A hypothetical ancestral sequence with a DAL80 binding site is evolved along a 3-taxa muta-

tion guide undergoing three substitutions and one insertion (Figure 5.14B). The multiple alignment of the leaf sequences generated by the simulation is shown in Figure 5.14C.

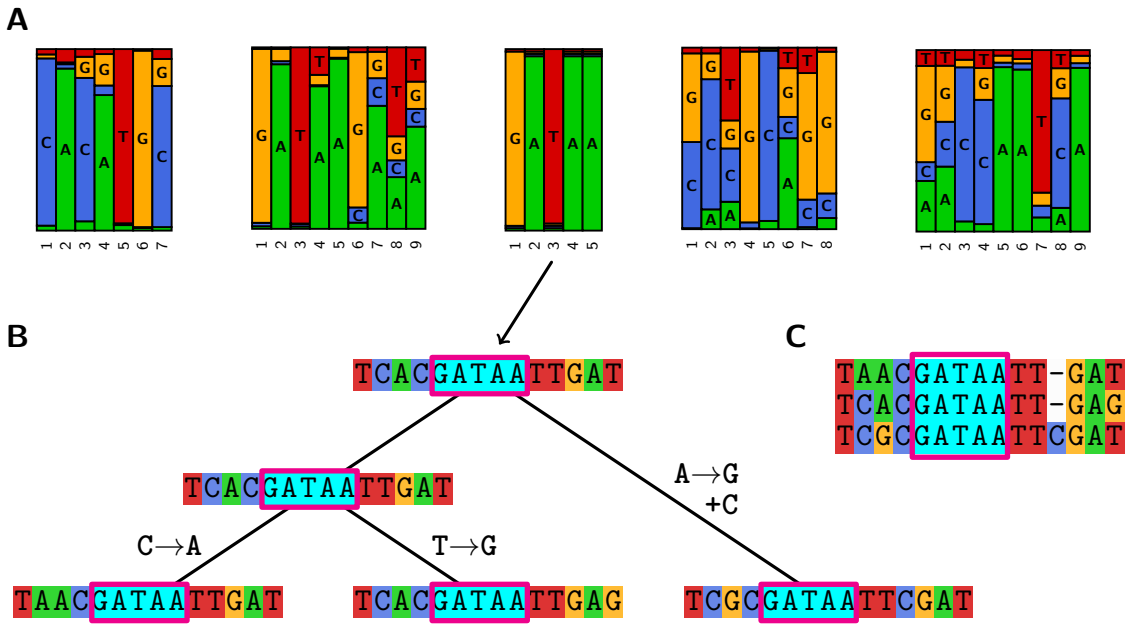


Figure 5.14.: Multiple alignment benchmark based on simulated sequences that are a mixture of background genomic DNA and transcription factor binding sites. (**A**) Histogram representation of binding affinity matrices for transcription factors INO2, GLN3, DAL80, PHD1 and HAP1 (from left to right). (**B**) A hypothetical ancestral sequence with a DAL80 binding site (magenta boxes) is evolved along a mutation guide tree. The labels indicate substitutions (→) and insertions (+) along tree edges. (**C**) Multiple alignment of the leaf sequences generated by the simulation.

In order to obtain alignments that closely mirror the sequence properties of noncoding DNA, we use as ancestral sequences yeast intergenic regions from a genomic location analysis of transcriptional regulators by Harbison et al. (2004). The Harbison dataset also provides us with sequence affinity matrices of 102 transcription factors and their genomic locations, both of which we use as input for the evolution of binding sites in CIS-EVOLVER. Out of a total of 6723 intergenic regions with a binding site proportion of 7%, we select all those sequences with at least one mapped binding site. Of the selected 3253 intergenic regions, we assign those from chromosome xvi to the *optimization set* (258 sequences), the others to the *test set* (5287 sequences). By using the optimization set, we determined the best values for the pseudocount admixture ($\tau = 0.6$), the context window length ($l = 7$), and the window weights ($w_{\text{center}} = 1.6$, $\beta = 0.80$). The context library size ($K = 4000$) is a tradeoff between sensitivity and time efficiency. If not stated otherwise, all reported benchmark results have been evaluated on the test set.

For each ancestral sequence we simulate synthetic DNA sequence data according to 8- and 16-taxon trees with symmetric and leftist branching topologies, using realistic evolutionary parameters for the evolution of genomic noncoding DNA. For each combination

of taxon number and topology, we simulate three datasets representing close, intermediate and distant evolutionary relationships, with a maximum pairwise distance of 0.8, 1.2 and 1.6 substitutions per site, respectively. These divergence estimates are in good agreement with values estimated based on mutations at silent positions in codons, and have also been used in previous multiple alignment simulation studies (Jayaraman and Siddharthan, 2010; Pollard et al., 2004). Figure 5.15 depicts the scale and branching order of the 6 mutation guide trees with leftist tree topology used in our sequence simulations.

5.3.2. Alignment quality measures

To assess the alignment quality, we compare predicted sequence alignments to gold-standard alignments generated with CIS-EVOLVER. The alignment quality is assessed by four sum-of-pairs performance measures: *Alignment sensitivity* is the fraction of aligned residue pairs in the gold-standard alignment that are correctly predicted, i.e., number of pairs correctly predicted/number of pairs in the reference alignment. *Alignment precision* is defined as the fraction of aligned residue pairs in the predicted alignment that are correct, i.e., number of pairs correctly aligned/number of pairs in the predicted alignment. *Motif sensitivity* is the fraction of aligned binding site residue pairs in the gold-standard alignment that are correctly predicted, i.e., number of pairs within binding sites that are correctly predicted/number of pairs within binding sites in the reference alignment. The motif sen-

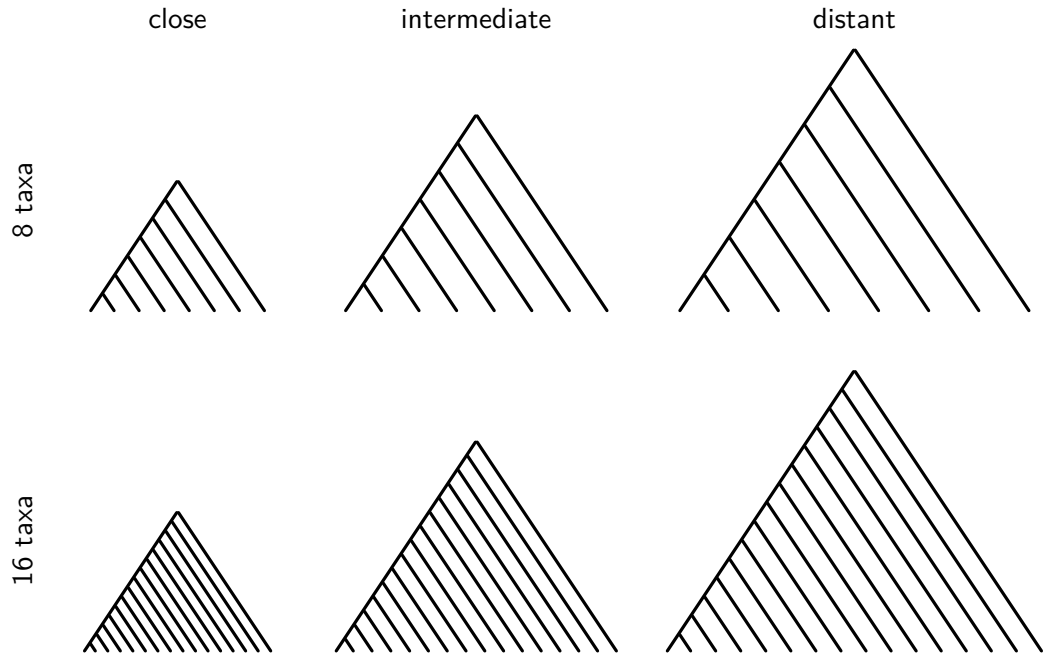


Figure 5.15.: Panel of 8- and 16-taxon leftist trees used for sequence simulation. All trees are drawn to the same scale representing close, intermediate, and distant evolutionary relationships (from left to right).

sitivity, therefore, quantifies to what extent residues that are part of a binding site are correctly aligned to each other. Usually, different alignment tools strike a different balance with respect to alignment sensitivity and alignment accuracy, i.e. some tools tend to align quite aggressively, producing alignments with higher sensitivity but lower precision, while others align very conservatively resulting in more precise but rather insensitive alignments. As an overall performance measure of alignment quality we therefore calculate a *balanced score* that is the geometric mean of alignment sensitivity and alignment precision $S_{\text{bal}} = \sqrt{S_{\text{sens}} S_{\text{prec}}}$. For each of the four performance measures, we calculate the weighted average over all predicted alignments in a set, with weights chosen proportional to the size of the reference alignment. Thereby, we adequately account for any lengths differences in the ancestral intergenic sequences.

5.3.3. Benchmark results

Figure 5.1 shows the benchmark results of CS-ALIGN and other current multiple alignment programs, including FSA (Bradley et al., 2009), PECAN (Paten et al., 2008), PRANK (Löytynoja and Goldman, 2008), and MUSCLE (Edgar, 2004), on all 8-taxon datasets, all programs being run with their default parameter settings. The figure shows separate histogram plots for average balanced score, motif sensitivity, alignment sensitivity, and alignment precision (top to bottom) on symmetric and leftist trees (left and right column, respectively). CS-ALIGN consistently shows higher balanced scores, motif sensitivities, and overall alignment sensitivities than any other program in all datasets but the leftist, distant. As expected, the performance differences between tools become more pronounced with increasing sequence divergence. This is particularly the case for the datasets with evolutionary distant sequences, in which CS-ALIGN is the only tool that aligns more than 80% of all binding site residue pairs correctly, despite the high sequence divergence.

The benchmark results also reveal that FSA produces by far the most precise alignments, although at the cost of attaining the lowest alignment and motif sensitivity values of of all tools. Moreover, on the symmetric, distant dataset the amount of missed binding site residues in FSA's alignments is disproportionately high, as can be deduced from the rather low ratio of motif sensitivity to overall alignment sensitivity. The profile-based aligners CS-ALIGN and especially MUSCLE miss fewer binding site residues, even taking into account their overall higher alignment sensitivity.

Table 5.1 summarizes the benchmark results for *all* 12 datasets including the sequence sets simulated on 16-taxon trees. As expected, the alignment quality not only improves with increasingly similar sequences (distant to close) but also with denser sequence sampling (8- to 16-taxon trees). The consistency-based algorithm utilized by PECAN seems to profit most from the denser sequence sampling with a balanced score improvement of 11% on the symmetric, distant dataset. This is likely due to the fact that the increased number of sequences allows for better transitive alignment, which in turn results in more

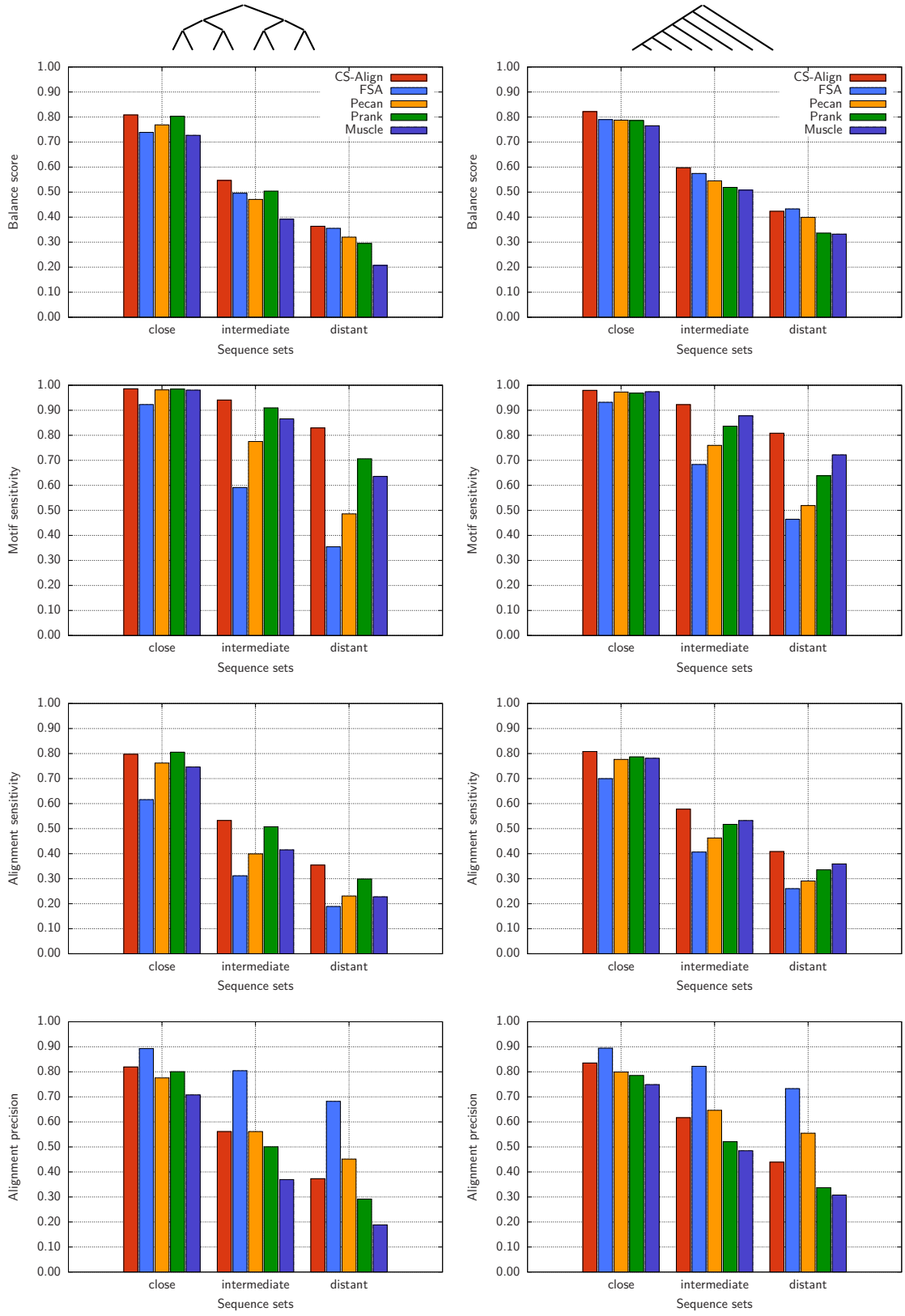


Figure 5.16.: Alignment benchmark results for sequence sets simulated on 8-taxon trees. The figure shows separate histogram plots for average balanced score, motif sensitivity, overall alignment sensitivity, and overall alignment precision (top to bottom), on symmetric (left column) and leftist (right column) trees. CS-ALIGN has higher (or equal) balanced scores, motif sensitivities, and overall sensitivities than any other program, except for the balanced score in the leftist, distant dataset.

Table 5.1.: Comparison of benchmark results on all datasets. For each alignment tool and parameter combination of 8/16 taxa, symmetric/leftist tree topology, and close/intermediate/distant evolutionary relationship, two values are given: the average balanced score S_b, and the average motif sensitivity S_m. In 9 out of 12 datasets CS-ALIGN has higher (or equal) balanced scores than any other program. CS-ALIGN consistently produces the most sensitive alignments with respect to binding site alignment, giving the highest motif sensitivity values with all datasets. The best value in each column-wise quintet is highlighted by gray background.

		symmetric						leftist					
		close		interm.		distant		close		interm.		distant	
	Tool	S_b	S_m	S_b	S_m	S_b	S_m	S_b	S_m	S_b	S_m	S_b	S_m
8	CS-ALIGN	0.81	0.99	0.55	0.94	0.36	0.83	0.82	0.98	0.60	0.92	0.42	0.81
	FSA	0.74	0.92	0.50	0.59	0.36	0.35	0.79	0.93	0.57	0.68	0.43	0.46
	PECAN	0.77	0.98	0.47	0.78	0.32	0.49	0.79	0.97	0.55	0.76	0.40	0.52
	PRANK	0.80	0.98	0.50	0.91	0.29	0.71	0.79	0.97	0.52	0.84	0.34	0.64
	MUSCLE	0.73	0.98	0.39	0.87	0.21	0.64	0.76	0.97	0.51	0.88	0.33	0.72
16	CS-ALIGN	0.85	0.99	0.60	0.97	0.37	0.85	0.85	0.98	0.65	0.95	0.47	0.83
	FSA	0.76	0.95	0.51	0.61	0.36	0.34	0.82	0.95	0.61	0.71	0.48	0.49
	PECAN	0.83	0.99	0.51	0.87	0.33	0.59	0.82	0.98	0.59	0.81	0.44	0.59
	PRANK	0.86	0.99	0.58	0.95	0.33	0.76	0.80	0.97	0.54	0.84	0.37	0.64
	MUSCLE	0.74	0.99	0.39	0.92	0.20	0.71	0.78	0.98	0.54	0.90	0.37	0.77

reliable alignments with fewer errors. The overall trend observed in the 8-taxon datasets also remains true for the 16-taxon sets: in 9 out of all 12 datasets tested, CS-ALIGN has higher (or equal) balanced scores than any other program and it consistently produces the most sensitive alignments with respect to binding site alignment, giving the highest motif sensitivity values of all programs on all datasets.

5.3.4. Analysis of CS-ALIGN components

We conduct an ablation analysis of the novel components in CS-ALIGN's alignment algorithm, namely context-specific pseudocounts, context-context match scoring, and enumeration of suboptimal alignments in PO-HMMs to test the effectiveness of each and to elucidate how they contribute to the overall alignment quality. For our first survey, we enable the enumeration of suboptimal alignments, but differentiate between four possible scoring regimes:

1. **Basic:** Pseudocounts for match state emission probabilities are calculated using a standard substitution matrix method employing the Tamura & Nei matrix. Context-context match scoring as given in equation (5.30) is disabled, i.e., w_{ctx} in equation (5.32) is set to zero.

2. **Context-context scoring:** Pseudocounts for match state emission probabilities are calculated using the standard substitution matrix method as in the basic regime.

Context-context match scoring as given in equation (5.30) is enabled with $w_{\text{ctx}} = 0.1$ (see equation 5.32).

3. **Context-specific pseudocounts:** Pseudocounts for match state emission probabilities are calculated using a library of $K = 4000$ context-profiles, generated by clustering all sequence profile windows from the draft progressive stage. Context-context match scoring is disabled as in the basic.

4. **Context-context scoring with context-specific pseudocounts:** The combination of context-context scoring and context-specific pseudocounts, i.e. both enabled.

Table 5.2 summarizes the ablation analysis of CS-ALIGN on the leftist, distant 8-taxon tree optimization set. The comparison of balanced scores S_{b} and motif sensitivities S_{m} for different scoring regimes reveals that context-context scoring as well as context-specific pseudocounts help to correctly align binding sites, while having only minor beneficial effects on overall alignment quality as indicated by the minor improvements in balanced score over the basic scoring regime. The best performing scoring regime is the one that uses context-specific pseudocounts but goes without context-context scoring. We therefore use it as default scoring regime in CS-ALIGN.

To quantify the contribution of suboptimal alignments on alignment quality, we run CS-ALIGN with and without enumeration of suboptimal alignments in progressive PO-PO alignment. The results given in Table 5.3 indicate that suboptimal alignments seem to have only marginal effect on alignment quality. This result surprised us. We had expected that the enumeration of suboptimal alignments would improve alignment quality to a larger extent, due to the prevention of alignment errors that get frozen at early stages of progressive alignment. To understand the reasons for the observed results, we inspected individual alignments in more detail. We found that even if the enumeration of suboptimal alignments is turned on, for the majority of all sequence sets (232 out of 258 in the optimization set) there is never a suboptimal alignment stored because their probabilities lie below the

Table 5.2.: Ablation analysis of CS-ALIGN on optimization set simulated on left, distant 8-taxon tree. Comparison of balanced scores S_{b} and motif sensitivities S_{m} for different scoring regimes reveals that context-context scoring as well as context-specific pseudocounts help to correctly align binding sites, while having only minor beneficial effects on overall alignment quality as indicated by the minor improvements in balanced score over the basic regime. CS-ALIGN's default scoring regime is highlighted by gray background.

Scoring regime	S_{b}	S_{m}
Basic	0.41	0.74
Context-context scoring	0.41	0.78
Context-specific pseudocounts	0.42	0.79
Context-context scoring with context-specific pseudocounts	0.41	0.78

Table 5.3.: Alignment quality measures for PO-HMM and HMM alignment for *basic* scoring. Partial order HMMs, which keep track of significant suboptimal alignments in progressive PO-PO alignment, seem to have only a marginal effect on alignment quality in comparison to simple HMMs. S_b and S_m as in table 5.2.

Suboptimal alignments	S_b	S_m
Off	0.40	0.72
On	0.41	0.74

inclusion threshold $P_{min} = 0.01$. Moreover, for the 26 cases in which at least one suboptimal alignment is actually stored in the PO-HMM, the suboptimal paths often differ in only a few positions from the optimal alignment. Of course, these local variants have a smaller upside potential with respect to the balanced score than a suboptimal alignment that differs substantially from the optimal alignment. We speculate that the absence of substantially different suboptimal alignments in the posterior probability matrices could be a result of the utilized sequence simulation process. CIS-EVOLVER does not model the duplication of DNA segments nor does it allow for binding site turnover, both of which would lead to simulated sequence sets with more ambiguous alignment alternatives.

5.4. Discussion

The diversity of mammalian promoter sequences is comparable to the distant dataset in our multiple alignment benchmark. Despite the conserved sequence signature of regulatory motifs, the number of correctly aligned TFBSs falls well below 50% for some of the tested multiple alignment programs. To improve the alignment quality of regulatory sequences, we have combined partial order graphs, profile HMMs, and the concept of context-specific pseudocounts into a novel progressive alignment method. Our multiple alignment tool CS-ALIGN achieves considerable improvements with respect to correct binding site alignment as well as overall alignment quality compared to the best current multiple alignment programs (see, e.g. Jayaraman and Siddharthan (2010)). However, for sequence diversities comparable to mammalian promoter sequences, even CS-ALIGN can only align about 80% of all TFBS correctly. This result confirms that the alignment of noncoding regulatory DNA at mammalian level diversity remains a very challenging task.

The alignment problem becomes even more challenging when we account for the gain and loss of TFBS, as it often occurs in regulatory sequences (Moses et al., 2006). Because the CIS-EVOLVER software assumes that a TFBS is conserved across all taxa in the mutation guide tree, we were unfortunately unable to test the effect of TFBS change in our alignment benchmark. Thus, to compute multiple alignments of mammalian regulatory sequences with close to 100% correctly aligned TFBS, further efforts will be necessary.

CS-ALIGN offers still room for improvements, especially with regard to the treatment

of TFBSs during alignment. With a share of about 7% of all residues in yeast, TFBS are relatively infrequent, which in turn requires a large number K of context profiles for them to be represented in the context library. As a possible solution one could provide the context-profiles of functional sites *a priori* to CS-ALIGN. Motif finding tools such as MEME (Bailey and Elkan, 1994), PRIORITY (Gordân et al., 2010), or MADONA (Hartmann and Söding *et al.*, to be published) should be better suited to detect the sequence signature of regulatory sites than the expectation-maximization clustering that aims to obtain a set of K context profiles which recur frequently among all training profiles, not necessarily only among TFBS. As an added benefit, the motif finding tool MADONA would be able to detect lineage specific TFBS as resulting from binding site gain and loss

Another area for improvement in the CS-ALIGN algorithm is the estimation of gap penalties, i.e., $M \to I$ and $I \to I$ transition probabilities. Currently, CS-ALIGN utilizes the same gap penalties at each PO-HMM node, but we could easily learn $M \to I$ and $I \to I$ transition probabilities dependent on tree branch lengths.

Finally, our method still contains several heuristic, non-probabilistic elements, such as the column match score, the context-context score, or the generation of suboptimal alignments. For an ideal alignment algorithm, however "the study of biological sequence data should not be divorced from the process that created it", as aptly expressed by Thorne et al. (1991). As a future objective, we would like to combine the concept of progressive partial order alignment with an explicit and completely probabilistic model of regulatory sequence evolution. With our method CS-ALIGN for context-specific multiple alignment of regulatory sequences using partial order HMMs, we have laid the foundation for the development of such an alignment program that can perform the multiple alignment and TFBS annotation of regulatory noncoding sequences in one integrated framework.

Bibliography

Altschul, S. F. Amino acid substitution matrices from an information theoretic perspective. *J Mol Biol* **1991**;*219*(3):555–565.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. *J Mol Biol* **1990**;*215*(3):403–410.

Altschul, S. F., Madden, T. L., Schaeffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **1997**;*25*(17):3389–3402.

Bailey, T. L. and Elkan, C. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol* **1994**;*2*:28–36.

Baussand, J., Deremble, C., and Carbone, A. Periodic distributions of hydrophobic amino acids allows the definition of fundamental building blocks to align distantly related proteins. *Proteins* **2007**;*67*(3):695–708.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Wheeler, D. L. GenBank. *Nucleic Acids Res* **2008**;*36*(Database issue):25–30.

Berman, B. P., Pfeiffer, B. D., Laverty, T. R., Salzberg, S. L., Rubin, G. M., Eisen, M. B., and Celniker, S. E. Computational identification of developmental enhancers: conservation and function of transcription factor binding-site clusters in drosophila melanogaster and drosophila pseudoobscura. *Genome Biol* **2004**;*5*(9):R61.

Biegert, A. and Söding, J. De novo identification of highly diverged protein repeats by probabilistic consistency. *Bioinformatics* **2008**;*24*(6):807–814.

Bradley, R. K., Roberts, A., Smoot, M., Juvekar, S., Do, J., Dewey, C., Holmes, I., and Pachter, L. Fast statistical alignment. *PLoS Comput Biol* **2009**;*5*(5):e1000392.

Cliften, P., Sudarsanam, P., Desikan, A., Fulton, L., Fulton, B., Majors, J., Waterston, R., Cohen, B. A., and Johnston, M. Finding functional features in saccharomyces genomes by phylogenetic footprinting. *Science* **2003**;*301*(5629):71–76.

Consortium, D. . G. Evolution of genes and genomes on the drosophila phylogeny. *Nature* **2007**a; *450*(7167):203–218.

Consortium, E. N. C. O. D. E. P. Identification and analysis of functional elements in 1genome by the encode pilot project. *Nature* **2007**b;*447*(7146):799–816.

Crooks, G. E., Green, R. E., and Brenner, S. E. Pairwise alignment incorporating dipeptide covariation. *Bioinformatics* **2005**;*21*(19):3704–3710.

Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure* **1978**;*5*(3):345–352.

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* **1977**;*39*(1):1–38.

Do, C. B., Mahabhashyam, M. S. P., Brudno, M., and Batzoglou, S. Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome Res* **2005**;*15*(2):330–340.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge, **1998**.

Eddy, S. R. Profile hidden Markov models. *Bioinformatics* **1998**;*14*(9):755–763.

Edgar, R. C. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res* **2004**;*32*(5):1792–1797.

Elias, I. Settling the intractability of multiple alignment. *J Comput Biol* **2006**;*13*(7):1323–1339.

Farrar, M. Striped smith-waterman speeds database searches six times over other simd implementations. *Bioinformatics* **2007**;*23*(2):156–161.

Feng, D. F. and Doolittle, R. F. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol* **1987**;*25*(4):351–360.

Fischer, J. D., Mayer, C. E., and Söding, J. Prediction of protein functional residues from sequence by probability density estimation. *Bioinformatics* **2008**;*24*(5):613–620.

Gambin, A., Lasota, S., Szklarczyk, R., Tiuryn, J., and Tyszkiewicz, J. Contextual alignment of biological sequences (Extended abstract). *Bioinformatics* **2002**;*18 Suppl 2*:116–127.

Gambin, A. and Otto, R. Contextual multiple sequence alignment. *J Biomed Biotechnol* **2005**; *2005*(2):124–131.

Gelly, J. C., Chiche, L., and Gracy, J. EvDTree: structure-dependent substitution profiles based on decision tree classification of 3D environments. *BMC Bioinformatics* **2005**;*6*:4–4.

Gonnet, G. H., Cohen, M. A., and Benner, S. A. Exhaustive matching of the entire protein sequence database. *Science* **1992**;*256*(5062):1443–1445.

Goonesekere, N. C. and Lee, B. Context-specific amino acid substitution matrices and their use in the detection of protein homologs. *Proteins* **2008**;*71*(2):910–919.

Gordân, R., Narlikar, L., and Hartemink, A. J. Finding regulatory dna motifs using alignment-free evolutionary conservation information. *Nucleic Acids Res* **2010**;*38*(6):e90.

Gotoh, O. An improved algorithm for matching biological sequences. *J Mol Biol* **1982**;*162*(3):705–708.

Gotoh, O. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol* **1996**;*264*(4):823–838.

Grasso, C. and Lee, C. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics* **2004**;*20*(10):1546–1556.

Gribskov, M., McLachlan, A. D., and Eisenberg, D. Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci USA* **1987**;*84*:4355–4358.

Halpern, A. L. and Bruno, W. J. Evolutionary distances for protein-coding sequences: modeling site-specific residue frequencies. *Mol Biol Evol* **1998**;*15*(7):910–917.

Han, K. F. and Baker, D. Global properties of the mapping between local amino acid sequence and local structure in proteins. *Proc Natl Acad Sci USA* **1996**;*93*(12):5814–5818.

Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., Hannett, N. M., Tagne, J.-B., Reynolds, D. B., Yoo, J., Jennings, E. G., Zeitlinger, J., Pokholok, D. K., Kellis, M., Rolfe, P. A., Takusagawa, K. T., Lander, E. S., Gifford, D. K., Fraenkel, E., and Young, R. A. Transcriptional regulatory code of a eukaryotic genome. *Nature* **2004**;*431*(7004):99–104.

Hasegawa, M., Kishino, H., and Yano, T. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *J Mol Evol* **1985**;*22*(2):160–174.

He, X., Ling, X., and Sinha, S. Alignment and prediction of cis-regulatory modules based on a probabilistic model of evolution. *PLoS Comput Biol* **2009**;*5*(3):e1000299.

Henikoff, J. G. and Henikoff, S. Using substitution probabilities to improve position-specific scoring matrices. *Comput Appl Biosci* **1996**;*12*(2):135–143.

Henikoff, S. and Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci USA* **1992**;*89*(22):10915–10919.

Henikoff, S. and Henikoff, J. G. Position-based sequence weights. *J Mol Biol* **1994**;*243*(4):574–578.

Hogeweg, P. and Hesper, B. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *J Mol Evol* **1984**;*20*(2):175–186.

Holm, L. and Sander, C. Mapping the protein universe. *Science* **1996**;*273*(5275):595–603.

Holmes, I. and Durbin, R. Dynamic programming alignment accuracy. *J Comput Biol* **1998**; *5*(3):493–504.

Huang, Y. M. and Bystroff, C. Improved pairwise alignments of proteins in the Twilight Zone using local structure predictions. *Bioinformatics* **2006**;*22*(4):413–422.

Jayaraman, G. and Siddharthan, R. Sigma-2: Multiple sequence alignment of non-coding dna via an evolutionary model. *BMC Bioinformatics* **2010**;*11*(1):464.

Jones, D. T., Taylor, W. R., and Thornton, J. M. A mutation data matrix for transmembrane proteins. *FEBS Lett* **1994**;*339*(3):269–275.

Jung, J. and Lee, B. Use of residue pairs in protein sequence-sequence and sequence-structure alignments. *Protein Sci* **2000**;*9*(8):1576–1588.

Katoh, K., Misawa, K., ichi Kuma, K., and Miyata, T. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Res* **2002**;*30*(14):3059–3066.

Kellis, M., Patterson, N., Endrizzi, M., Birren, B., and Lander, E. S. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature* **2003**;*423*(6937):241–254.

Kim, J. and Sinha, S. Towards realistic benchmarks for multiple alignments of non-coding sequences. *BMC Bioinformatics* **2010**;*11*:54.

Kimura, M. The neutral theory of molecular evolution: a review of recent evidence. *Jpn J Genet* **1991**;*66*(4):367–386.

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. Hidden markov models in computational biology. applications to protein modeling. *J Mol Biol* **1994**;*235*(5):1501–1531.

Kschischo, M. and Lässig, M. Finite-temperature sequence alignment. *Pac Symp Biocomput* **2000**; pages 624–635.

Lee, C., Grasso, C., and Sharlow, M. F. Multiple sequence alignment using partial order graphs. *Bioinformatics* **2002**;*18*(3):452–464.

Li, X. and Wong, W. H. Sampling motifs on phylogenetic trees. *Proc Natl Acad Sci U S A* **2005**; *102*(27):9481–9486.

Ludwig, M. Z., Patel, N. H., and Kreitman, M. Functional analysis of eve stripe 2 enhancer evolution in drosophila: rules governing conservation and change. *Development* **1998**;*125*(5):949–958.

Lunter, G., Rocco, A., Mimouni, N., Heger, A., Caldeira, A., and Hein, J. Uncertainty in homology inferences: assessing and improving genomic sequence alignment. *Genome Res* **2008**;*18*(2):298–309.

Löytynoja, A. and Goldman, N. An algorithm for progressive multiple alignment of sequences with insertions. *Proc Natl Acad Sci U S A* **2005**;*102*(30):10557–10562.

Löytynoja, A. and Goldman, N. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science* **2008**;*320*(5883):1632–1635.

Madera, M. Profile Comparer (PRC): a program for scoring and aligning profile hidden Markov models. *Bioinformatics* **2008**;pages –504.

Miyazawa, S. A reliable sequence alignment method based on probabilities of residue correspondences. *Protein Eng* **1995**;*8*(10):999–1009.

Moses, A. M., Pollard, D. A., Nix, D. A., Iyer, V. N., Li, X.-Y., Biggin, M. D., and Eisen, M. B. Large-scale turnover of functional transcription factor binding sites in drosophila. *PLoS Comput Biol* **2006**;*2*(10):e130.

Mueller, T., Rahmann, S., and Rehmsmeier, M. Non-symmetric score matrices and the detection of homologous transmembrane proteins. *Bioinformatics* **2001**;*17*(suppl1):182–189.

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol* **1995**;*247*:536–540.

Myers, G., Selznick, S., Zhang, Z., and Miller, W. Progressive multiple alignment with constraints. *J Comput Biol* **1996**;*3*(4):563–572.

Mückstein, U., Hofacker, I. L., and Stadler, P. F. Stochastic pairwise alignments. *Bioinformatics* **2002**;*18 Suppl 2*:153–160.

Needleman, S. B. and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* **1970**;*48*(3):443–453.

Ng, P. C., Henikoff, J. G., and Henikoff, S. PHAT: a transmembrane-specific substitution matrix. *Bioinformatics* **2000**;*16*(9):760–766.

Notredame, C. Recent Evolutions of Multiple Sequence Alignment Algorithms. *PLoS Computational Biology* **2007**;*3*(8):–123.

Notredame, C., Higgins, D. G., and Heringa, J. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol* **2000**;*302*(1):205–217.

Overington, J., Donnelly, D., Johnson, M. S., Sali, A., and Blundell, T. L. Environment-specific amino acid substitution tables: tertiary templates and prediction of protein folds. *Protein Sci* **1992**;*1*(2):216–226.

Paten, B., Herrero, J., Beal, K., Fitzgerald, S., and Birney, E. Enredo and pecan: genome-wide mammalian consistency-based multiple alignment with paralogs. *Genome Res* **2008**;*18*(11):1814–1828.

Pearson, W. R. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics* **1991**;*11*(3):635–650.

Pollard, D. A., Bergman, C. M., Stoye, J., Celniker, S. E., and Eisen, M. B. Benchmarking tools for the alignment of functional noncoding DNA. *BMC Bioinformatics* **2004**;*5*:6–6.

Pollard, D. A., Moses, A. M., Iyer, V. N., and Eisen, M. B. Detecting the limits of regulatory element conservation and divergence estimation using pairwise and multiple alignments. *BMC Bioinformatics* **2006**;*7*:376–376.

Rice, D. W. and Eisenberg, D. A 3D-1D substitution matrix for protein fold recognition that includes predicted secondary structure of the sequence. *J Mol Biol* **1997**;*267*(4):1026–1038.

Rosenberg, M. S. Multiple sequence alignment accuracy and evolutionary distance estimation. *BMC Bioinformatics* **2005**;*6*:278.

Rychlewski, L., Jaroszewski, L., Li, W., and Godzik, A. Comparison of sequence profiles. Strategies for structural predictions using sequence information. *Protein Sci* **2000**;*9*(2):232–241.

Sadreyev, R. and Grishin, N. COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. *J Mol Biol* **2003**;*326*(1):317–336.

Satija, R., Pachter, L., and Hein, J. Combining statistical alignment and phylogenetic footprinting to detect regulatory elements. *Bioinformatics* **2008**;*24*(10):1236–1242.

Schwartz, A. S., Myers, E. W., and Pachter, L. Alignment metric accuracy. *arXiv:q-bioQM/0510052* **2007**;.

Schwartz, S. Multiple alignment by sequence annealing. *Bioinformatics* **2007**;*23*(2):–24.

Shi, J., Blundell, T. L., and Mizuguchi, K. FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J Mol Biol* **2001**; *310*(1):243–257.

Siddharthan, R., Siggia, E. D., and van Nimwegen, E. Phylogibbs: a gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput Biol* **2005**;*1*(7):e67.

Siepel, A. and Haussler, D. Phylogenetic estimation of context-dependent substitution rates by maximum likelihood. *Mol Biol Evol* **2004**;*21*(3):468–488.

Sinha, S. and He, X. Morph: probabilistic alignment combined with hidden markov models of cis-regulatory modules. *PLoS Comput Biol* **2007**;*3*(11):e216.

Sinha, S., van Nimwegen, E., and Siggia, E. D. A probabilistic method to detect regulatory modules. *Bioinformatics* **2003**;*19 Suppl 1*:i292–i301.

Sjoelander, K., Karplus, K., Brown, M., Hughey, R., Krogh, A., Mian, I. S., and Haussler, D. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Comput Appl Biosci* **1996**;*12*(4):327–345.

Smith, T. F. and Waterman, M. S. Identification of common molecular subsequences. *J Mol Biol* **1981**;*147*(1):195–197.

Stark et al., A. Discovery of functional elements in 12 Drosophila genomes using evolutionary signatures. *Nature* **2007**;*450*:219–232.

Söding, J. Protein homology detection by HMM-HMM comparison. *Bioinformatics* **2005**;*21*(7):951–960.

Tamura, K. and Nei, M. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Mol Biol Evol* **1993**;*10*(3):512–526.

Tatusov, R. L., Altschul, S. F., and Koonin, E. V. Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. *Proc Natl Acad Sci USA* **1994**; *91*(25):12091–12095.

Taylor, W. R. A flexible method to align large numbers of biological sequences. *J Mol Evol* **1988**; *28*(1-2):161–169.

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

Thorne, J. L., Kishino, H., and Felsenstein, J. An evolutionary model for maximum likelihood alignment of dna sequences. *J Mol Evol* **1991**;*33*(2):114–124.

Wang, L. and Jiang, T. On the complexity of multiple sequence alignment. *J Comput Biol* **1994**; *1*(4):337–348.

Wasserman, W. W., Palumbo, M., Thompson, W., Fickett, J. W., and Lawrence, C. E. Human-mouse genome comparisons to locate regulatory sites. *Nat Genet* **2000**;*26*(2):225–228.

Waterman, M. S. and Eggert, M. A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J Mol Biol* **1987**;*197*(4):723–728.

Wong, K. M., Suchard, M. A., and Huelsenbeck, J. P. Alignment uncertainty and genomic analysis. *Science* **2008**;*319*(5862):473–476.

Woolfe, A., Goodson, M., Goode, D. K., Snell, P., McEwen, G. K., Vavouri, T., Smith, S. F., North, P., Callaway, H., Kelly, K., Walter, K., Abnizova, I., Gilks, W., Edwards, Y. J. K., Cooke, J. E., and Elgar, G. Highly conserved non-coding sequences are associated with vertebrate development. *PLoS Biol* **2005**;*3*(1):e7.

Yona, G. and Levitt, M. Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. *J Mol Biol* **2002**;*315*(5):1257–1275.

Yu, Y. K., Wootton, J. C., and Altschul, S. F. The compositional adjustment of amino acid substitution matrices. *Proc Natl Acad Sci USA* **2003**;*100*(26):15688–15693.

Zhang, Y. and Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res* **2005**;*33*(7):2302–2309.