Algorithms for bioinformatics

Giacomo Fantoni
telegram: @GiacomoFantoni
Github: https://github.com/giacThePhantom/algorithms-for-bioinformatics

June 22, 2022

Contents

Chapter 1

Needleman Wunsch

1.1 Introduction

Direct comparison of two sequences based on the presence in both of the corresponding amino acids in an identical array is insufficient to establish the full genetic relationship between two proteins. Allowance for gaps multiplies the number of comparisons that can be made but introduces unnecessary and partial comparisons.

1.2 A general method for sequence comparison

The maximum match can be defined as the largest number of amino acids of one protein that can be matched with those of another protein while allowing for all possible deletions. It can be determined by representing in a matrix all possible pair combinations that can be constructed from the amino acid sequences of the protein being compared. So A_j is the jth amino acids of protein A and B_i is the ith amino acids of protein B. A_j are the columns and B_i all the rows of the matrix MAT. Then A_{ij} represent a pair combination with amino acids A_j and B_i. Every possible comparison can be represented by pathway through the matrix. A pathway is signified by a line connecting cells of the array. Complete diagonals contain no gaps. A necessary pathway begins at a cell in the first column of row. Either i or j must increase by only one, while the other may increase by one or more, leading to the next cell in a pathway. This is repeated until i, j or both reach their limiting value. Every partial or unnecessary pathway will be contained in at least one necessary pathway. The values in the matrix are computed as:

$$MAT_{ij} = \max(MAT_{i-1,j-1} + \alpha\delta(A_j, B_i), MAT_{i-i,j} + d, MAT_{i,j-1} + d)$$

Where d is the penalty factor, a number subtracted for every gap made, may be defined as a barrier for allowing the gap. And α can be a function that can represent any theory with the significance of a pair of amino acids. No gap would be allowed in the operation unless the benefit from allowing that gap would exceed the barrier. This method can be expanded for allowing the comparison of n sequences through and n-dimensional matrix. The maximum-match pathway can be obtained by beginning at the terminals of the sequences and proceeding towards the origin, first by adding to the value of each cell possessing indices $i = y - 1$ and or $j = z - 1$. The process is iterated until all cells in the matrix have been operated upon. Each cell in the outer row or column will contain the maximum number of matches that can be obtained by originating any pathway at

that cell and the largest number in that row or column is equal to the maximum match. The cells of the array which contributed to the maximum match may be determined by recording the origin of the number that was added to each cell when the array was operated upon.

1.3 Evaluating the significance of the maximum match

To accomplish the estimate of if a result found differs significantly from a match between random sequences two sets of random sequences can be constructed, each one from the set of amino acid composition of each of the proteins. If the value found for the real proteins is significantly different the difference a function of of the sequences alone and not of the composition.

1.4 Cell values and weighting factors

Cells can be weighted in accordance with the maximum number of corresponding bases in codons of the represented amino acids, to make the comparison more accurate. Also the significance of the maximum match is enhanced by decreasing the weight of those pathways containing a large number of gaps through the penalty factor.

Chapter 2

Smith Watermann

2.1 Introduction

The Smith Watermann algorithm extends the one of Needleman and Wunsch to find a pair of segment, one from each of two long sequences, such that there is no other pair of segments with greater similarity. This similarity measure allows for deletion and insertion of arbitrary length.

2.2 Algorithm

Consider two molecular sequences $A = a_1a_2 \dots a_n$ and $B = b_1b_2 \dots b_m$. Given a similarity $s(a, b)$ of elements of the sequence and W_k the weight of deletions of length k, to find pairs of segments with high degrees of similarity, a matrix H is set up such that:

$$H_{k0} = H_{0l} = 0 \qquad \forall 0 \leq k \leq n \wedge 0 \leq l \leq m$$

H_{ij} si the maximum similarity of two segments ending in a_i and b_j. H_{ij} is computed such that:

$$H_{ij} = \max(H_{i-1,j-1} + s(a_i, b_j), \max_{k \geq 1}(H_{i-k,j} - W_k), \max_{l \geq 1}(H_{i,j-l} - W_k), 0)$$

With $1 \leq i \leq n$ and $1 \leq j \leq m$. So H_{ij} is:

- $H_{i-1,j-1} + s(a_i, b_j)$ If a_i and b_j are associated.
- $H_{i-k,j} - w_k$ if a_i is at the end of a deletion of length k.
- $H_{i-k,j} - W_l$ if b_j is at the end of a deletion of length l.
- 0 is used to prevent calculated negative similarity, indicating no similarity up to a_i and b_j.

The pair of segments with maximum similarity is found first by locating the maximum elemnt of H. The other elements are determined sequentially with a traceback procedure ending with an element of H equal to 0. This procedure other than identifying the elements produces their alignment. The parameters where:

$$s(a_i, b_j) = \begin{cases} 1 & a_i = b_j \\ 0 & a_i \neq b_j \end{cases}$$

And

$$W_k = \frac{1}{3}k$$

This algorithm in particular allows for the alignment of sequences that contained both mismatches and internal deletions.

Chapter 3

PAM - a model of evolutionary change in proteins

3.1 Accepted point mutation

An accepted point mutation in a protein is a replacement of one amino acid by another accepted by natural selection. To be accepted the new amino acid usually must function in a similar way to the old one. The likelihood of amino acid X replacing Y is the same as Y replacing X is assumed the same because it depends on the product of the frequencies of occurrence and on their chemical and physical similarity. SO evolution is a vibration around given frequencies.

3.2 Mutability of amino acids

The relative mutability is the probability that each amino acid will change in a given small evolutionary intervall. To compute it the number of times that each amino acid has changed in an interval and the number of times that it has occurred in the sequences and thus has been subject to mutation. In calculating this number in for many trees, with sequences of different lengths and evolutionary distance is combined in relative mutabilities. Each relative mutability is a ration between the total number of changes on all branches of all protein trees considered and the total exposure of the amino acid to mutation, or the sum for all branches of its local frequency of occurrence multiplied by the total number of mutation per 100 links of that branch.

3.3 Mutation probability matrix for the evolutionary distance of one PAM

The individual kind of mutations and the relative mutability of the amino acids can be combined into a mutation probability matrix in which M_{ij} gives the probability that the amino acid in column j will be replaced by the amino acid in row i after a given evolutionary period. The non-diagonal elements are computed as:

$$M_{ij} = \frac{\lambda m_j A_{ij}}{\sum_i A_{ij}}$$

Where:

- A_{ij} is an element of the accepted point mutation matrix.
- λ is a proportionality constant.
- m_j is the mutability of the jth amino acid.

The diagonal elements are:

$$M_{jj} = 1 - \lambda m_j$$

The sum of all elements of each column or row is 1. The probability of observing a change is proportional to the mutability of the amino acid in that place. The same proportionality constant λ holds for all columns. $100 \cdot \sum f_i M_{ij}$ gives the number of amino acids that will remain unchanged when a protein 100 links long of average composition is exposed to the evolutionary change. This depends on λ. To change the evolutionary period the matrix is multiplied my itself n times, and with $n \to \infty$ each column approaches the asymptotic amino acid composition. The percentage of amino acids that will be observed to change on the average in the interval are found by:

$$100(1 - \sum_i f_i M_{ij})$$

The term of the relatedness odds matrix are:

$$R_{ij} = \frac{M_{ij}}{f_i}$$

Or the mutation probability of a change over the probability that i will occur in the second sequence by chance. Each term of this matrix gives the probability of replacement per occurrence of i per occurrence of j. Amino acids with score > 1 replace each other more often as alternative in related sequences that in random sequences.

Chapter 4

BLOSUM

4.1 Introduction

4.1.1 Abstrac

The most used substitution matrix with scores for all possible exchanges of one amino acid is based on the Dayhoff model of evolutionary rates. This work proposes a different approach from blocks of aligned sequence segments, leading to improvement in alignments and in searches.

4.1.2 Introduction

Sequence alignment of proteins provide important insights into gene and protein function. There are different types of alignments:

- Global alignments of pairs related by common ancestry.
- Multiple alignments of members of protein families.
- Alignments made during data base searches to detect homology.

In each case a scoring scheme for estimating similarity is used. The mutation data matrices of Dayhoff are considered the default in alignment and searching programs. However the most common task is the detection of much more distant relationships, which are only inferred from substitution rates in the Dayhoff model.

4.2 Methods

4.2.1 Deriving a frequency table from a data base blocks

Local alignments can be represented as ungapped blocks with each row a different protein segment and each column an aligned residue position. Protomat can be used for obtaining a set of blocks given a group of related proteins. Considering a single block representing a conserved region of a protein family, for a new member a set of score for matches and mismatches that best favours a correct alignment with each of the other segments in the block. For each column of the block, the number of matches and mismatches of each type between the new sequence and every other are

counted. This is repeated for all columns of all blocks and the summed results are stored in a table. The new sequence is then added to the group. For another new sequence the procedure is repeated. Doing so the table in the end will consist of counts of all possible amino acid pairs in a column. Counts of all possible pairs in each column of each block in the data base are summed. If a block has a width w amino acids and a depth of s sequences, contributes $ws\frac{(s-1)}{2}$ amino acids pairs to the count. This results in a frequency table listing the number of times each of the different amino acid pairs occurs among the blocks. This table is used to compute the odds ratio matrix between the observed frequencies and the expected one.

4.2.2 Computing a logarithm of odds matrix

Let the total number of amino acid pairs i, j for each entry of the frequency table be $f - ij$. Then the observed probability of occurrence for each i, j pair is:

$$q_{ij} = \frac{f_{ij}}{\sum_{i=1}^{20}\sum_{j=1}^{i} f_{iJ}}$$

The expected probability of occurrence for each pair is computed following the occurrence of the ith amino acid in a i, j pair:

$$p_i = q_{ii} + \sum_{j \neq i} \frac{q_{ij}}{2}$$

The expected probability of occurrence e_{ij} for each i, j pair is then $p_i p_j$ for $i = j$ and $p_i p_j + p_j p_i = 2p_i p_j$ for $i \neq j$. An odds ratio matrix is computed such that each entry is $\frac{q_{ij}}{e_{ij}}$. A lod (logarithm of odds) is then calculated in bit units as $s = \log_2\left(\frac{q_{ij}}{e_{ij}}\right)$. If the observed frequencies are as expected $s_{ij} = 0$, if less $s_{ij} < 0<$ if more $s_{ij} > 0/$ Lod ratios are multiplied by a 2 scaling factor and rounded to the nearest integer value to produce the block substitution matrix BLOSUM. The relative entropy or the average mutual information per amino acid pair is computed:

$$H = \sum_{i=1}^{20} \sum -j = 1^{i} q_{ij} \times s_{ij}$$

And the expected score in bit units:

$$E = \sum_{i=1}^{20}\sum_{j=1}^{i} p_i \times p_j \times s_{ij}$$

4.2.3 Clustering segments within blocks

To reduce multiple contributions to amino acid pair frequencies from the most closely related members of a family, sequences are clustered within blocks and each cluster is weighted as a single sequence in counting pairs. A clustering percentage in which sequence segments identical for at least that value are clustered is used. The contribution of closely related segments to the frequency table is reduced. Varying the clustering percentage leads to a family of matrices.

4.2.4 Constructing blocks data bases

Protomat was used to build the block data base from 504 non redundant groups of proteins. Protomat uses an amino acid substitution matrix at two phases. THe motif program uses a substitution matrix when individual sequences are aligned against sequence segments containing a candidate motif. The Motomat program uses a substitution matrix when a block is extended to either side of the motif region. A unitary substitution matrix was used, next blosum was applied to the blocks and the resulting matrix was used to construct a second database. Then blousm was applied to the second data base and the resulting matrix was used to construct version of blocks data base. THe blosum program was applied to the final data base using a series of clustering percentages to obtain a family of lod substitution matrices. Similar matrices were obtained using PAM.

4.2.5 Alignments and homology searches

Global multiple alignments were done using Multalin and to provide a positive matrix each entry was increased by 8. Pearson's RDF2 program was used to evaluate local pairwise alignments. Homology searches were done using Blastp, Fasta and Ssearch. The Swiss-Prot data bank was searched. The first of the longest and most distance sequences in the group was used as a searching query, inferring distance from Protomat results. The results of each search were analysed by considering the sequences used by Protomat to construct blocks for the protein group as the true positive sequences. The number of misses is the nubmer of true positive sequences not reported for blastp. For fasta and ssearch the empirical evaluation criteria of Pearson was used: the number of misses is the number of true positive scores which ranked below the 99.5 percentile of the true negative scores.

4.3 Results

4.3.1 Comparison to Dayhoff matrices

The blosum series based on percent clustering can be compared to the Dayhoff matrices using a measure of average information per residue pair in bit units or relative entropy, which is 0 when the target distribution of pair frequencies is the same as the expected one and increases as they become more different. In the Dayhoff matrices relative entropy decreased when increasing PAM, while in blosum it increased linearly when increasing clustering percentage. Matrices with comparable relative entropy have similar expected scores.

4.3.2 Performance in multiple alignment of known structures

To test sequence alignment accuracy the results obtained to alignments seen in three dimensional structures was used. A simultaneous multiple alignment program MSA was used as a standard. Multalin, a hierarchical multiple alignment program performed worse using PAM matrices, while using blosum better. Therefore blosum matrices produced accurate global alignments of these sequences.

4.3.3 Performance in searching for homology in sequence data banks

The number of misses when searching was averaged in order to assess the overall searching performance of different matrices using blast, fasta and smith-waterman. Blosum matrices performed better than the best PAM matrix. BLOSUM improved detection of members od this family. The test was repeated with similar result of PAM for other protein families.

Chapter 5

FASTA

Chapter 6

BLAST

Chapter 7

How to BLAST

Chapter 8

Gapped BLAST and PSI BLAST

Chapter 9

PSI BLAST Composition-Based Statistics

Chapter 10

CLUSTAL-W

Chapter 11

T-COFFEE

T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment We describe a new method (T-Coffee) for multiple sequence alignment that provides a dramatic improvement in accuracy with a modest sacrifice in speed as compared to the most commonly used alternatives.

11.1 Introduction

The simultaneous alignment of three or more nucleotide or amino acid sequences is one of the commonest tasks in bioinformatics. Multiple alignments are an essential prerequisite to many analyses of protein families such as homology modeling or phylogenetic reconstruction, or to illustrate conserved and variable sites within a family.

11.1.1 Reasons for the development

11.1.1.1 Once a gap, always a gap

The most commonly used heuristic methods are based on the **progressive alignment strategy**, with **ClustalW** 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. Although successful, this method suffers from its greediness. Errors made in the first alignments cannot be rectified later as the rest of the sequences are added in. Even though the paradigm "once a gap always a gap" remains true, misplacing gaps becomes much less likely.

11.1.1.2 Global and local alignment

Some methods attempt to carry out global alignments, where one tries to align the full lengths of the sequences with each other. Alternatively, one might wish to consider local similarity, as occurs when two proteins share only a domain or motif (Smith-Waterman, Lalign). In principle, a method able to combine the best properties of global and local multiple alignments might be very powerful. This is the second motivation for T-Coffee: the design of a method that provides a simple, flexible and, most importantly, accurate solution to the problem of how to combine information of this sort.

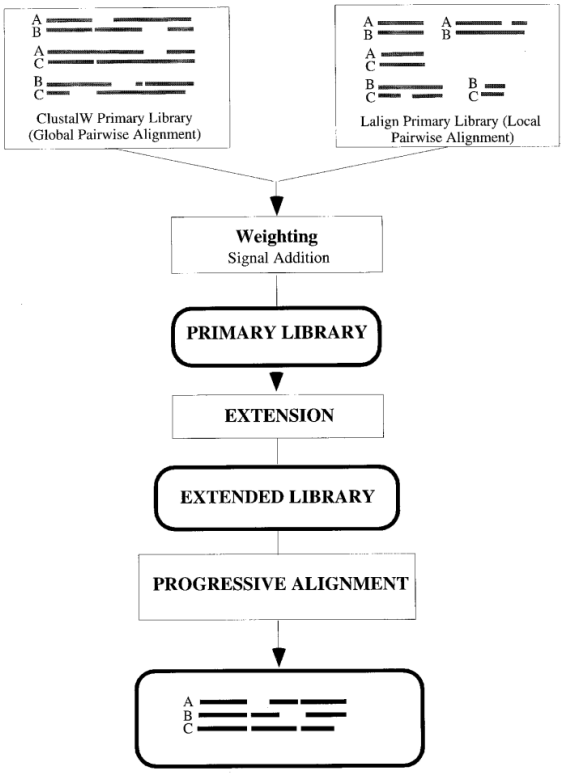

Figure 11.1: Layout of the T-Coffee strategy; the main steps required to compute a multiple sequence alignment using the T-Coffee method. Square blocks designate procedures while rounded blocks indicate data structures.

11.2 T-Coffee Algorithm

T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation) has two main features.
First, it provides a simple and flexible means of generating multiple alignments, using heterogeneous data sources. The data from these sources are provided to T-Coffee via a **library** of pair-wise alignments.
The main feature of T-Coffee is the **optimization** method, which is used to find the multiple alignment that best fits the pair-wise alignments in the input library. A so-called **progressive strategy** is implemented, which is similar to that used in ClustalW. The difference from CustalW is that T-Coffe makes use of the information in the library to carry out progressive alignment in a manner that allows us to consider the alignments between all the pairs while we carry out each step of the progressive multiple alignment. This gives us progressive alignment, with all its advantages of speed and simplicity, but with a far lesser tendency to make errors like the one shown in Figure 11.2(a), i.e. misalignment of the word CAT. T-Coffee is a progressive alignment with an ability to consider information from all of the sequences during each alignment step, not just those being aligned at that stage.

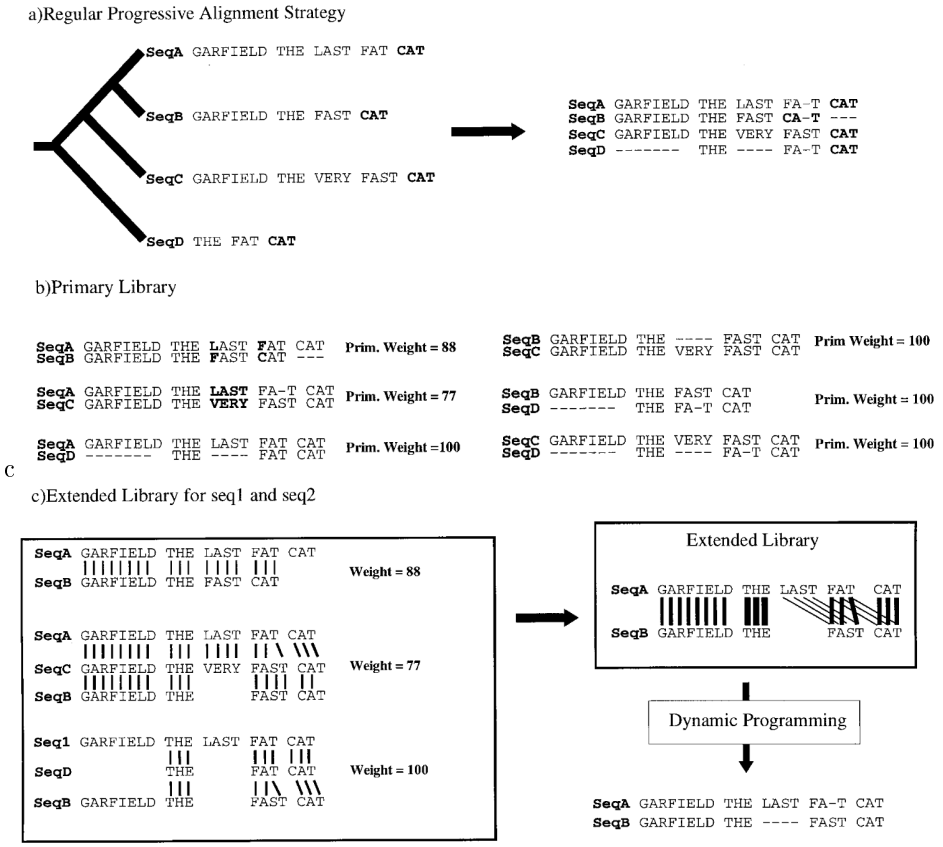

Figure 11.2: The library extension. **(a) Progressive alignment**. Four sequences have been designed. The tree indicates the order in which the sequences are aligned when using a progressive method such as ClustalW. The resulting alignment is shown, with the word CAT misaligned. **(b) Primary library**. Each pair of sequences is aligned using ClustalW. In these alignments, each pair of aligned residues is associated with a weight equal to the average identity among matched residues within the complete alignment (mismatches are indicated in bold type). **(c) Library extension for a pair of sequences**. The three possible alignments of sequence A and B are shown (A and B, A and B through C, A and B through D). These alignments are combined, as explained in the text, to produce the position-specific library. This library is resolved by dynamic programming to give the correct alignment. The thickness of the lines indicates the strength of the weight.

11.2.1 Generating a primary library of alignments

The primary library contains a set of pair-wise alignments between all of the sequences to be aligned. In the library, we include information on each of the $N(N-1)/2$ sequence pairs, where N is the number of sequences. Here, we use two alignment sources for each pair of sequences, one **local** and one **global**. The global alignments (Figures 11.1 and 11.2(b)) are constructed using ClustalW on the sequences, two at a time. The local alignments (Figure 11.1) are the ten top scoring non-intersecting local alignments, between each pair of sequences, gathered using the Lalign program.
In the library, each alignment is represented as a list of pair-wise residue matches (e.g. residue x of sequence A is aligned with residue y of sequence B). In effect, each of these pairs is a constraint. All

of these constraints are not equally important. Some may come from parts of alignments that are more likely to be correct. This is taken into account when computing the multiple alignment and give priority to the most reliable residue pairs. This is achieved by using a weighting scheme, which is described in subsection 11.2.2.

11.2.2 Derivation of the primary library weights

T-Coffee assigns a weight to each pair of aligned residues in the library (Figure 2(b)). The **sequence identity** weighting scheme is used, which has been prove to be effective and of great simplicity. Libraries are lists of weighted pair-wise constraints. Each constraint receives a weight equal to percent identity within the pair-wise alignment it comes from (Figure 11.2(b)). For each set of sequences, two primary libraries are computed along with their weights, one using ClustalW (global alignments; Figure 11.2(b)) and the second using Lalign (local).

11.2.3 Combination of the libraries

The aim is the efficient combination of local and global alignment information. This is achieved by pooling the global and local primary libraries in a simple process of **addition**. If any pair is duplicated between the two libraries, it is merged into a single entry that has a weight equal to the sum of the two weights. Otherwise, a new entry is created for the pair being considered (process called unofficially **"stacking" of the signal**). Pairs of residues that did not occur are not represented (weight of zero).
This primary library can be used directly to compute a multiple sequence alignment. However, we enormously increase the value of the information in the library by examining the consistency of each pair of residues with residue pairs from all of the other alignments. For each pair of aligned residues in the library, we can assign a weight that reflects the degree to which those residues align consistently with residues from all the other sequences. This process is called library extension (subsection 11.2.4).

11.2.4 Extending the library

Fitting a set of weighted constraints into a multiple alignment is a well known NP-complete problem. We circumvent the problem by using a heuristic algorithm that we call **library extension** (Figure 11.2(c)). The idea is to combine information in such a manner that the final weight, for any pair of residues, reflects some of the information contained in the whole library. To do so, a **triplet approach** is used, as summarized in Figure 11.2(c).
It is based on taking each aligned residue pair from the library and checking the alignment of the two residues with residues from the remaining sequences.

11.2.4.1 A quick example

For instance, let us consider the four sequences A, B, C and D of Figure 2. Let us call A(G) the G of GARFIELD in sequence A, B(G) the equivalent G in sequence B and W(A(G), B(G)) the weight associated with this pair of symbols in the primary library. In the direct alignment of A and B, A(G) and B(G) are matched (Figure 11.2(b) and (c)). Therefore, the initial weight for that pair of residues can be set to 88 (primary weight of the alignment of sequence A and B, which is the percent of identity of this pair). If we now look at the alignment of sequence A and sequence B through sequence C (Figure 11.2(c)), we can see that the A(G) and C(G) are aligned, as well as C(G) and A(G). We conclude that there is an alignment of A(G) with B(G) through sequence

C. We associate that alignment with a weight equal to the minimum of $W_1 = W(A(G), C(G))$ and $W_2 = W(C(G), B(G))$. Since $W_1 = 77$ and $W_2 = 100$, the resulting weight is set to 77. In the extended library, this new value is added to the previous one to give a total weight of 165 for the pair A(G), B(G). The complete extension will require an examination of all the remaining triplets. Not all of them bring information. For instance, the alignment of A and B through sequence D does not contain any information relative to A(G) or B(G), and, therefore, it has no influence on the weight associated with A(G) and B(G). In summary, the weight associated with a pair of residues will be the sum of all the weights gathered through the examination of all the triplets involving that pair.

11.2.4.2 Alignment

Weights will be zero for any residue pairs that never occur (this will be true of the majority of residue pairs). Otherwise, the weight will reflect a combination of the similarity of the pair of sequences or sequence segments that the residue pair comes from and the consistency of that residue pair with all other residue pairs in the primary library. These scores can then be used to align any two sequences from our data set using conventional dynamic programming. When one normally aligns a pair of sequences, one uses a set of scores derived from some general table of amino acid weights such as a Blosum matrix. In our case, we can replace that matrix with a set of scores that are specific to every possible pair of residues in our two sequences. This will allow an alignment to be carried out that will take account of the particular residues in the two sequences but will also be guided towards consistency with all of the other sequences in the data set.

11.2.5 Progressive alignment strategy

The *normal* progressive alignment strategy consists in creating a guide tree (a phylogenetic tree) using the neighbor-joining method. The closest two sequences on the tree are aligned first using normal dynamic programming. This pair of sequences is then fixed and any gaps that have been introduced cannot be shifted later. Then the next closest two sequences are aligned or a sequence is added to the existing alignment of the first two sequences, depending which is suggested by the guide tree.

As used here, the procedure does not require any additional parameters such as gap penalties. This stems, in part, from the fact that the substitution values (the library weights) were computed on alignments where such penalties had already been applied. Furthermore, high scoring segments that show consistency within the data set see their score enhanced by the extension to such a point that they become insensitive to gap penalties.

11.3 Biological validation

I chose not to report the comparisons with other tools and the complexity of the algorithm. If needed, exhaustive tabled can be found in the paper.

11.3.1 Application to serine/threonine kinases

A major application of any alignment algorithm will be the delineation of motifs or domains. 19 sequences from a sub-family of the serine/threonine kinases were provided. Each sequence in the alignment contains a nucleotide-binding site (NBS). In all these sequences, the NBS is followed by a second conserved motif toward the C terminus. T-Coffee was able to accurately align 18 of the 19

NBSs, ClustalW was only able to correctly align 16 of these NBSs. The second motif is more difficult because of the long indel in st11 yeast. Here as well, T-Coffee can properly align 18 of the motifs, while ClustalW get 15 correct. As a result of combining local and global alignment information, T-Coffee managed to align almost all of the motifs as in the BaliBase reference alignment. Moreover, T-Coffee was the only program that correctly aligned the second motif of kp68 human, which is an interferon-induced kinase.

Chapter 12

Protein profiles with HMMs

Chapter 13

Construction of phylogenetic trees

Chapter 14

Toward defining the course of evolution: minimum change for a specific tree topology

Chapter 15

Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach