Protein Sequence Alignment and Database Scanning

Geoffrey J. Barton

Laboratory of Molecular Biophysics
University of Oxford
Rex Richards Building
South Parks Road
Oxford OX1 3QU
U.K.

Tel: 0865-275368
Fax: 0865-510454
e-mail: gjb@bioch.ox.ac.uk

This is a preprint of the chapter in: *Protein Structure prediction - a practical approach*, Edited by M. J. E. Sternberg, IRL Press at Oxford University Press, 1996, ISBN 0 19 963496 3. Please see that book for the final version.

Contents

1 Introduction **3**

2 Amino acid scoring schemes **3**
2.1 Identity scoring . . . 3
2.2 Genetic code scoring . . . 4
2.3 Chemical similarity scoring . . . 4
2.4 Observed substitutions . . . 4
2.4.1 The Dayhoff mutation data matrix . . . 5
2.4.2 PET91 - An updated Dayhoff matrix . . . 6
2.4.3 BLOSUM - matrix from ungapped alignments. . . . 6
2.4.4 Matrices derived from tertiary structure alignments . . . 6
2.5 Which matrix should I use? . . . 7

3 Comparison of two sequences **8**
3.1 Sequence comparison without gaps - fixed length segments . . . 8
3.1.1 Correlation methods . . . 9
3.1.2 Variable length segments . . . 10
3.2 Sequence comparison with gaps . . . 10
3.2.1 Finding the best alignment with dynamic programming . . 10
3.2.2 Alternative weighting for gaps . . . 12
3.3 Identification of local similarities . . . 12
3.3.1 Finding second and subsequent best local alignments . . . 12

4 Evaluation of alignment accuracy **13**
4.1 Predicting overall alignment accuracy . . . 13
4.1.1 Predicting quality using percentage identity . . . 14
4.2 Predicting the reliable regions of an alignment . . . 14
4.3 Incorporating non-sequence information into alignment . . . 16

5 Multiple sequence alignment **16**
5.1 Extension of dynamic programming to more than two sequences . 17
5.2 Tree or hierarchical methods using dynamic programming . . . 17
5.3 Extension of segment methods to multiple alignment . . . 18
5.4 Representation and analysis of multiple alignments . . . 19

6 Database scanning **20**
6.1 Basic principles of database searching 20
6.2 Time considerations 21
6.3 Which database should I search? Local or network? 22
6.4 Searching with dynamic programming 23
6.4.1 Scanning with parallel computers Prosrch, MPsrch and others 23
6.5 Index methods 24
6.5.1 Simple index for identical matching 24
6.5.2 Indexing with gaps - the FLASH algorithm 24
6.6 Approximations: The FASTP and FASTA algorithm 25
6.7 Approximations: BLAST Basic Local Alignment Search Tool . . 26
6.8 Guidelines for Database Scanning 27

7 Summary **29**

8 Figure Legends **30**
8.1 Figure 1 30
8.2 Figure 2 30
8.3 Figure 3 30
8.4 Figure 4 30
8.5 Figure 5 30
8.6 Figure 6 30
8.7 Figure 7 31
8.8 Figure 8 31
8.9 Figure 9 31
8.10 Figure 10 31

1 Introduction

In the context of protein structure prediction, there are two principle reasons for comparing and aligning protein sequences:

- To obtain an accurate alignment. This may be for protein modelling by comparison to proteins of known three-dimensional structure.
- To scan a database with a newly determined protein sequence and identify possible functions for the protein by analogy with well characterised proteins.

In this chapter I review the underlying principles and techniques for sequence comparison as applied to proteins and used to satisfy these two aims.

2 Amino acid scoring schemes

All algorithms to compare protein sequences rely on some scheme to score the equivalencing of each of the 210 possible pairs of amino acids. (i.e. 190 pairs of different amino acids + 20 pairs of identical amino acids). Most scoring schemes represent the 210 pairs of scores as a 20×20 matrix of similarities where identical amino acids and those of similar character (e.g. I, L) give higher scores compared to those of different character (e.g. I, D). Since the first protein sequences were obtained, many different types of scoring scheme have been devised. The most commonly used are those based on observed substitution and of these, the 1976 Dayhoff matrix for 250 PAMS [1] has until recently dominanted. This and other schemes are discussed in the following sections.

2.1 Identity scoring

This is the simplest scoring scheme; amino acid pairs are classified into two types: identical and non-identical. Non-identical pairs are scored 0 and identical pairs given a positive score (usually 1). The scoring scheme is generally considered less effective than schemes that weight non-identical pairs, particularly for the detection of weak similarities [2, 3]. The normalised sum of identity scores for an alignment is popularly quoted as "percentage identity", but this value can be useful to indicate the overall similarity between two sequences, there are pitfalls associated with the measure. These are discussed in section 4.1.1.

2.2 Genetic code scoring

Whereas the identity scoring scheme considers all amino acid transitions with equal weight, genetic code scoring as introduced by Fitch [4] considers the minimum number of DNA/RNA base changes (0,1,2 or 3) that would be required to interconvert the codons for the two amino acids. The scheme has been used both in the construction of phylogenetic trees and in the determination of homology between protein sequences having similar three dimensional structures [5]. However, today it is rarely the first choice for scoring alignments of protein sequences.

2.3 Chemical similarity scoring

The aim with chemical similarity scoring schemes is to give greater weight to the alignment of amino acids with similar physico-chemical properties. This is desirable since major changes in amino acid type could reduce the ability of the protein to perform its biological role and hence the protein would be selected against during the course of evolution. The intuitive scheme developed by McLachlan [6] classified amino acids on the basis of polar or non-polar character, size, shape and charge and gives a score of 6 to interconversions between identical rare amino acids (eg F, F) reducing to 0 for substitutions between amino acids of quite different character (eg. F, E). Feng *et al.* [3] encode features similar to McLachlan by combining information from the structural features of the amino acids and the redundancy of the genetic code.

2.4 Observed substitutions

Scoring schemes based on observed substitutions are derived by analysing the substitution frequencies seen in alignments of sequences. This is something of a chicken and egg problem, since in order to generate the alignments, one really needs a scoring scheme but in order to derive the scoring scheme one needs the alignments! Early schemes based on observed substitutions worked from closely related sequences that could easily be aligned by eye. More recent schemes have had the benefit of the earlier substitution matrices to generate alignments on which to build. Long experience with scoring schemes based on observed substitutions suggests that they are superior to simple identity, genetic code, or intuitive physico-chemical property schemes.

2.4.1 The Dayhoff mutation data matrix

Possibly the most widely used scheme for scoring amino acid pairs is that developed by Dayhoff and co-workers [1]. The system arose out of a general model for the evolution of proteins. Dayhoff and co workers examined alignments of closely similar sequences where the the likelihood of a particular mutation (e. A-D) being the result of a set of successive mutations (eg. A-x-y-D) was low. Since relatively few families were considered, the resulting matrix of accepted point mutations included a large number of entries equal to 0 or 1. A complete picture of the mutation process including those amino acids which did not change was determined by calculating the average ratio of the number of changes a particular amino acid type underwent to the total number of amino acids of that type present in the database. This was combined with the point mutation data to give the mutation probability matrix (M) where each element $M_{i,j}$ gives the probability of the amino acid in column j mutating to the amino acid in row i after a particular evolutionary time, for example after 2 PAM (Percentage of Acceptable point Mutations per 10^8 years).

The mutation probability matrix is specific for a particular evolutionary distance, but may be used to generate matrices for greater evolutionary distances by multiplying it repeatedly by itself. At the level of 2,000 PAM Schwartz and Dayhoff suggest that all the information present in the matrix has degenerated except that the matrix element for Cys-Cys is 10% higher than would be expected by chance. At the evolutionary distance of 256 PAMs one amino acid in five remains unchanged but the amino acids vary in their mutability; 48% of the tryptophans, 41% of the cysteines and 20% of the histidines would be unchanged, but only 7% of serines would remain.

When used for the comparison of protein sequences, the mutation probability matrix is usually normalised by dividing each element $M_{i,j}$ by the relative frequency of exposure to mutation of the amino acid i. This operation results in the symmetrical "relatedness odds matrix" with each element giving the probability of amino acid replacement per occurrence of i per occurrence of j. The logarithm of each element is taken to allow probabilities to be summed over a series of amino acids rather than requiring multiplication. The resulting matrix is the "log-odds matrix" which is frequently referred to as "Dayhoff's matrix" and often used at a distance of close to 256 PAM since this lies near to the limit of detection of distant relationships where approximately 80% of the amino acid positions are observed to have changed [2].

2.4.2 PET91 - An updated Dayhoff matrix

The 1978 family of Dayhoff matrices was derived from a comparatively small set of sequences. Many of the 190 possible substitutions were not observed at all and so suitable weights were determined indirectly. Recently, Jones *et al.* [7] have derived an updated substitution matrix by examining 2,621 families of of sequences in the SWISSPROT database release 15.0. The principal differences between the Jones *et al.* matrix (PET91) and the Dayhoff matrix are for substitutions that were poorly represented in the 1978 study. However, the overall character of the matrices is similar. Both reflect substitutions that conserve size and hydrophobicity, which are the principle properties of the amino aids [8]

2.4.3 BLOSUM - matrix from ungapped alignments.

Dayhoff-like matrices derive their initial substitution frequencies from global alignments of very similar sequences. An alternative approach has been developed by Henikoff and Henikoff using local multiple alignments of more distantly related sequences [9]. First a database of multiple alignments without gaps for short regions of related sequences was derived. Within each alignment in the database, the sequences were clustered into groups where the sequences are similar at some threshold value of percentage identity. Substitution frequencies for all pairs of amino acids were then calculated between the groups and this used to calculate a log odds BLOSUM (blocks substitution matrix) matrix. Different matrices are obtained by varying the clustering threshold. For example, the BLOSUM 80 matrix was derived using a threshold of 80% identity.

2.4.4 Matrices derived from tertiary structure alignments

The most reliable protein sequence alignments may be obtained when all the proteins have had their tertiary structures experimentally determined. Comparison of three dimensional structures also allows much more distantly related proteins to be aligned accurately. Analysis of such alignments should therefore give the best substitution matrices. Accordingly, Risler *et al.* [10] derived substitution frequencies from 32 proteins structurally aligned in 11 groups. On similar lines, Overington *et al.* [11] aligned 7 families for which 3 or more proteins of known three dimensional structure were known and derived a series of substitution matrices. Overington *et al.* also subdivided the substitution data by the secondary structure and environment of each amino acid, however this led to rather sparse matrices due to the lack of examples. Bowie *et al.* [12] have also derived substitution tables specific for different amino acid environments and secondary structures.

2.5 Which matrix should I use?

The general consensus is that matrices derived from observed substitution data (e.g. the Dayhoff or BLOSUM matrices) are superior to identity, genetic code or physical property matrices (e.g. see [3]). However, there are Dayhoff matrices of different PAM values and BLOSUM matrices of different percentage identity and which of these should be used?

Schwartz and Dayhoff [2] recommended a mutation data matrix for the distance of 250 PAMs as a result of a study using a dynamic programming procedure [13] to compare a variety of proteins known to be distantly related. The 250 PAM matrix was selected since in Monte Carlo studies (see Section 4.1) matrices reflecting this evolutionary distance gave a consistently higher significance score than other matrices in the range 0-750 PAM. The matrix also gave better scores when compared to McLachlan's substitution matrix [6], the genetic code matrix and identity scoring. Recently, Altschul [14] has examined Dayhoff style mutation data matrices from an information theoretical perspective. For alignments that do not include gaps he concluded, in broad agreement with Schwarz and Dayhoff, that a matrix of 200 PAMS was most appropriate when the sequences to be compared were thought to be related. However, when comparing sequences that were not known in advance to be related, for example when database scanning, a 120 PAM matrix was the best compromise. When using a local alignment method (Section 6.7) Altschul suggests that three matrices should ideally be used: PAM40, PAM120 and PAM250, the lower PAM matrices will tend to find short alignments of highly similar sequences, while higher PAM matrices will find longer, weaker local alignments. Similar conclusions were reached by Collins and Coulson [15] who advocate using a compromise PAM100 matrix, but also suggest the use of multiple PAM matrices to allow detection of local similarities of all types.

Henikoff and Henikoff [16] have compared the BLOSUM matrices to PAM, PET, Overington, Gonnet [17] and multiple PAM matrices by evaluating how effectively the matrices can detect known members of a protein family from a database when searching with the ungapped local alignment program BLAST [18]. They conclude that overall the BLOSUM 62 matrix is the most effective. However, all the substitution matrices investigated perform better than BLOSUM 62 for a proportion of the families. This suggests that no single matrix is the complete answer for all sequence comparisons. It is probably best to compliment the BLOSUM 62 matrix with comparisons using PET91 at 250 PAMS, and Overington structurally derived matrices. It seems likely that as more protein three dimensional structures are determined, substitution tables derived from structure comparison will give the most reliable data.

3 Comparison of two sequences

Given a scoring scheme, the next problem is how to compare the sequences, decide how similar they are and generate an alignment. This problem may be subdivided into alignment methods for two sequences, multiple alignment methods, and methods that incorporate additional non-sequence information, for example from the tertiary structure of the protein.

The simplest two-sequence comparison methods do not explicitly consider insertions and deletions (gaps). More sophisticated methods make use of dynamic programming to determine the best alignment including gaps (see Section 3.2)

3.1 Sequence comparison without gaps - fixed length segments

Given two sequences A and B of length m and n, all possible overlapping segments having a particular length (sometimes called a 'window length') from A are compared to all segments of B. This requires of the order of $m \times n$ comparisons to be made. For each pair of segments the amino acid pair scores are accumulated over the length of the segment. For example, consider the comparison of two 7-residue segments; ALGAWDE and ALATWDE using identity scoring. The total score for this pair would be 1 + 1 + 0 + 0 + 1 + 1 + 1 = 5.

In early studies of protein sequences, statistical analysis of segment comparison scores was used to infer homology between sequences. For example, Fitch [4] applied the genetic code scoring scheme to the comparison of $\alpha-$ and $\beta-$ haemoglobin and showed the score distribution to be non-random. Today, segment comparison methods are most commonly used in association with a "dot plot" or "diagram" [19] and can be a more effective method of finding repeats than using dynamic programming.

The scores obtained by comparing all pairs of segments from A and B may be represented as a comparison matrix R where each element $R_{i,j}$ represents the score for matching an odd length segment centred on residue A_i with one centred on residue B_j. This matrix can provide a graphic representation of the segment comparison data particularly if the scores are contoured at a series of probability levels to illustrate the most significantly similar regions. Collins and Coulson [20] have summarised the features of the “dot-plot”. The runs of similarity can be enhanced visually by placing a dot at all the contributing match points in a window rather than just at the centre.

McLachlan [6] introduced two further refinements into segment comparison methods. The first was the inclusion of weights in the comparison of two segments in order to improve the definition of the ends of regions of similarity. For example, the scores obtained at each position in a 5-residue segment comparison might be multiplied by 1,2,3,2,1 respectively before being summed. The second refinement was the development of probability distributions which agreed well with experimental comparisons on random and unrelated sequences and which could be used to estimate the significance of an observed comparison.

3.1.1 Correlation methods

Several experimental, and semi-empirical properties have been derived associated with amino acid types, for example hydrophobicity (e.g. [21]), and propensity to form an α – helix (eg. [22]). Correlation methods for the comparison of protein sequences exploit the large number of amino acid properties as an alternative to comparing the sequences on the basis of pair scoring schemes.

Kubota *et al.* [23] gathered 32 property scales from the literature and through application of factor analysis selected 6 properties which for carp parvalbumin gave good correlation for the comparison of the structurally similar CE- and EF-hand region Ca^{2+}binding sites and poor correlation in other regions. They expressed their sequence comparisons in the form of a comparison matrix similar to that of McLachlan [6] and demonstrated that their method could identify an alignment of α – lytic protease and Streptomyces Griseus protease A which agrees with that determined from comparison of the available crystal structures.

Argos [24] determined the most discriminating properties from a set of 55 by calculating correlation coefficients for all pairs of sequences within 30 families of proteins that had been aligned on the basis of their three-dimensional structures. The correlation coefficients for each property were then averaged over all the families to leave 5 representative properties. Unlike Kubota *et al.* [23], Argos applied the correlation coefficients from the five properties in addition to a more conventional segment comparison method using the Dayhoff matrix scoring scheme. He also combined the result of using more than one segment length on a single diagram such that the most significant scores for a particular length always prevail.

3.1.2 Variable length segments

The best local ungapped alignments of variable length may be found either by dynamic programming with a high gap-penalty, or using heuristic methods. Since the heuristic methods are primarily used for database searching they are described in Section 6.

3.2 Sequence comparison with gaps

The segment based techniques described in section 3.1 do not consider explicitly insertions and deletions. Deletions are often referred to as "gaps", while insertions and deletions are collectively referred to as "indels". Insertions and deletions are usually needed to align accurately even quite closely related sequences such as the α – and β – globins. The naive approach to finding the best alignment of two sequences including gaps is to generate all possible alignments, add up the scores for equivalencing each amino acid pair in each alignment then select the highest scoring alignment. However, for two sequences of 100 residues there are $> 10^{75}$ alternative alignments so such an approach would be time consuming and infeasible for longer sequences. Fortunately, there is a group of algorithms that can calculate the best score and alignment in the order of mn steps. These *dynamic programming* algorithms were first developed for protein sequence comparison by Needleman and Wunsch [13], though similar methods were independently devised during the late 1960's and early 1970's for use in the fields of speech processing and computer science [25].

3.2.1 Finding the best alignment with dynamic programming

Dynamic programming algorithms may be divided into those that find a global alignment of the sequences and those that find local alignments. The difference between global and local alignment is illustrated in Figure 1[1]. Global alignment is appropriate for sequences that are known to share similarity over their whole length. Local alignment is appropriate when the sequences may show isolated

[1] Figure1.ps

regions of similarity, for example multiple domains or repeats. Local alignment is best applied when scanning a database to find similarities or when there is no *a priori* knowledge that the protein sequences are similar.

There are many variations on the theme of dynamic programming applied to protein comparisons. Here I give a brief account of a basic method for finding the *global* best score for aligning two sequences. For a clear and detailed explanation of dynamic programming see Sankoff and Kruskal [26].

Let the two sequences of length m and n be $A = (A_1, A_2, \ldots A_m), B = (B_1, B_2, \ldots B_n)$ and the symbol for a single gap be Δ. At each aligned position there are three possible events.

$w(A_i, B_j)$ substitution of A_i by B_j.
$w(A_i, \Delta)$ deletion of A_i.
$w(\Delta, B_j)$ deletion of B_j.

The substitution weight $w(A_i, B_j)$ is derived from the chosen scoring scheme - perhaps Dayhoff's matrix. Gaps Δ are normally given a negative weight often referred to as the "gap penalty" since insertions and deletions are usually less common than substitutions.

The maximum score M for the alignment of A with B may be represented as $s(A_{1\ldots m}, B_{1\ldots n})$. This may be found by working forward along each sequence sucessively finding the best score for aligning $A_{1\ldots i}$ with $B_{1\ldots j}$ for all i, j where $1 \leq i \leq m$ and $1 \leq j \leq n$. The values of $s(A_{1\ldots i}, B_{1\ldots j})$ are stored in a matrix H where each element of H is calculated as follows:

$$H_{i,j} = \max \left\{ \begin{array}{c} H_{i-1,j-1} + w_{A_i,B_j} \\ H_{i,j-1} + w_{A_i,\Delta} \\ H_{i-1,j} + w_{\Delta,B_j} \end{array} \right\}$$

The element $H_{m,n}$ contains the best score for the alignment of the complete sequences.

If the alignment is required as well as the best score, then the alignment path may be determined by tracing back through the H matrix. Alternatively a matrix of pointers is recorded to indicate which of the three possibilities was the maximum at each value $H_{i,j}$.

3.2.2 Alternative weighting for gaps

The above scheme showed a simple length-dependent weighting for gaps. Thus two isolated gaps give the same score as two consecutive gaps. It is possible to generalise the algorithm to allow gaps of length greater than 1 to carry weights other than the simple sum of single gap weights [27]. Such gap weighting can give a more biologically meaningful model of transitions from one sequence to another since insertions and deletions of more than one residue are not uncommon events between homologous protein sequences. Most computer programs that implement dynamic programming allow gaps to be weighted with the form $v + uk$ where k is the gap length and v and u are constants ≥ 0, since this can be computed efficiently [28].

3.3 Identification of local similarities

Although segment based comparison methods (see section 3.1) rely on local comparisons, if insertions and deletions have occurred, the match may be disrupted for a region of the order of the length of the segment. In order to circumvent these difficulties algorithms which are modifications of the basic global alignment methods have been developed to locate common subsequences including a consideration of gaps (e.g.[29, 30, 31]). For protein sequences, the most commonly used local alignment algorithm that allows gaps is that described by Smith and Waterman [30]. This is essentially the same as the global alignment algorithm described in section 3.2.1, except that a zero is added to the recurrence equation.

$$H_{i,j} = \max \left\{ \begin{array}{c} H_{i-1,j-1} + w_{A_i,B_j} \\ H_{i,j-1} + w_{A_i,\Delta} \\ H_{i-1,j} + w_{\Delta,B_j} \\ 0 \end{array} \right\}$$

Thus all $H_{i,j}$ must have a value ≥ 0. The score for the best local alignment is simply the largest value of H and the coresponding alignment is obtained by tracing back from this cell.

3.3.1 Finding second and subsequent best local alignments

The Smith-Waterman algorithm returns the single best local alignment, but two proteins may share more than one common region. Waterman and Eggert [32] have shown how all local alignments may be obtained for a pair of sequences with minimal recalculation. Recently, Barton [33] has described how for a simple length dependent gap-penalty, all locally optimal alignments may be determined in the order of mn steps without recalculation.

4 Evaluation of alignment accuracy

What is a good alignment? The amino acid sequence codes for the protein three dimensional structure. Accordingly, when an alignment of two or more sequences is made, the implication is that the equivalenced resdues are preforming similar structural roles in the native folded protein. The best judge of alignment accuracy is thus obtained by comparing alignments resulting from sequence comparison with those derived from protein three dimensional structures. There are now many families of proteins for which two or more members have been determined to atomic resolution by X-ray crystallography or NMR. Accurate alignment of these proteins by consideration of their tertiary structures [34, 35, 36] provides a set of test alignments against which to compare sequence-only alignment methods. Care must be taken when performing the comparison since within protein families, some regions show greater similairty than others. For example, the core β – strands and α – helices are normally well conserved, but surface loops vary in structure and alignments in these regions may be ambiguous, or if the three-dimensional structures are very different in a region, alignment may be meaningless. Accordingly, evaluation of alignment accuracy is best concentrated on the core secondary structures of the protein and other conserved features [37]; such regions may automatically be identified by the algorithm of Russell and Barton [36].

4.1 Predicting overall alignment accuracy

It is important to know in advance what the likely accuracy of an alignment will be. A common method for assessing the significance of a global alignment score is to compare the score to the distribution of scores for alignment of random sequences of the same length and composition. The result (the S.D. score) is normally expressed in Standard Deviation units above the mean of the distribution.

Comparison of the S.D. score for alignment to alignment accuracy obtained by comparison of the core secondary structures, suggests that for proteins of 100-200 amino acids in length, a score above 15.0 S.D. indicates a near ideal alignment, scores above 5.0 S.D. a “good” alignment where $\geq 70\%$ of the residues in core secondary structures will be correctly equivalenced, while alignments with scores below 5.0 S.D. should be treated with caution [37, 38].

Figure 2[2]shows the distribution of S.D. scores for 100,000 optimal alignments of length ≥ 20 between proteins of unrelated three dimensional structure. From Figure 2[3], the mean S.D. score expected for the comparison of unrelated protein sequences is 3.2 S.D. with a S.D. of 0.9. However, the distribution is skewed with a tail of high S.D. scores. In any large collection of alignments it is possible to have a rare, high scoring alignment that actually shares no structural similarity. For example, Figure 3[4] illustrates an optimal local alignment between regions of citrate synthase (2cts) and transthyritin (2paba) which gives 7.55 S.D. though the secondary structure of these two protein segments are completely different.

4.1.1 Predicting quality using percentage identity

Percentage identity is a frequently quoted statistic for an alignment of two sequences. However, the expected value of percentage identity is strongly dependent upon the length of alignment [39] and this is frequently overlooked. Figure 4[5] shows the percentage identities found for a large number of locally optimal alignments of differing length between proteins known to be of unrelated three dimensional structure. Clearly, an alignment of length 200 showing 30% identity is more significant than an alignment of length 50 with the same identity. Applying this to the alignment shown in Figure 3[6] shows that although the alignment scores over 7.0 S.D. it has a percentage identity that one would often see by chance between unrelated proteins.

4.2 Predicting the reliable regions of an alignment

Although the overall accuracy of an alignment may be estimated from the SD score (see section 4.1) this value does not indicate which regions of the alignment are correct. Experience suggests that the reliable regions of an alignment are those that do not change when small changes are made to the gap-penalty and matrix

[2] Figure2.ps
[3] Figure2.ps
[4] Figure3.ps
[5] Figure4.ps
[6] Figure3.ps

parameters. An alternative strategy is to examine the sub-optimal alignments of the sequences to find the regions that are shared by sub optimal alignments within a scoring interval of the best alignment. For any two sequences, there usually many alternative alignments with scores similar to the best. These alignments share common regions and it is these regions that are deemed to be the most reliable. For example, the simple alignment of ALLIM with ALLM scoring 2 for identites, 1 for mismatch and -1 for a gap gives:

A	L	L	I	M
A	L	L		M

with a score of $2 + 2 + 2 - 1 + 2 = 7$. The sub optimal alignment:

A	L	L	I	M
A	L		L	M

gives a score of $2 + 2 - 1 + 1 + 2 = 6$ but shares the alignment of AL and M with the optimal alignment. Rather than calculate all sub-optimal alignments, Vingron and Argos [40] use an elegant and simple method to identify the reliable regions in an alignment by calculating the comparison matrix H both forwards and backwards and summing the two matrices. The cells in $H_{i,j}$ that are equal to the best score for the alignment delineate the optimal alignment path. Cells within a selected value of the best score are flagged and reliable regions defined as those for which there is no other cell $H_{i,k}$ or $H_{l,j}$ with $k \neq j$ and $l \neq i$. The results of the analysis are displayed in the form of a dot-plot with larger dots identifying the reliable regions.

Although the details of his calculation differ from Vingron and Argos, Zuker [41], produces a dot plot that highlights the regions where there are few alternative local alignments. He also caters for optimal local alignments with gaps. Zuker shows that the alignment of distantly related sequences such as *Streptomyces griseus* proteinase A and porcine elastase may be clearly seen to be unstable with many sub-optimal alignments close to the optimal.

Rather than use the dot-plot representation, Saqi and Sternberg [42] directly determine alternative sub-optimal alignments. They first calculate the H matrix and best path, then identify the cells that contributed to the best path and reduce these by a preset value (usually 10% of the typical scoring matrix value). A new H matrix is calculated and a new best path and alignment. This process is repeated iteratively to generate a series of global sub-optimal alignments.

Investigating sub-optimal alignments by one or more of these methods allows:

- The most reliable regions of an alignment to be identified and by inference the overall quality of the alignment.

- Alternative alignments close to the optimum to be generated. These can be useful when building three dimensional models of proteins by homology.

4.3 Incorporating non-sequence information into alignment

If the three dimensional structure of one of the proteins to be aligned is known, then this information may be encoded in the form of a modified gap penalty [37, 38, 43]. The penalty reduces the likelihood of insertions/deletions occurring in known secondary structure regions, or conversely increases the likelihood of placing gaps in known loop regions. This approach increases the usual accuracy of alignment and has the additional bonus of reducing the sensitivity of the alignments to changes in gap penalty [37].

A stricter constraint on the alignment is possible if specific residues are known to be equivalent in the two proteins. The weight for aligning these specific residues may be increased to force them to align. However, if this type of treatment is really necessary, then it is likely that the alignment will have a low significance score and must be treated with caution.

5 Multiple sequence alignment

So far I have only considered methods to align two sequences. However, when the sequence data is available, a multiple alignment is always preferable to pairwise alignment.

Techniques for the alignment of three or more sequences may be divided into four categories.

- Extensions of pairwise dynamic programming algorithms.

- Hierarchical extensions of pairwise methods.

- Segment methods.

- Consensus or 'regions' methods.

Of these, the second is by far the most practical and widely used method. Consensus methods are not greatly used for protein sequence alignment and so are not discussed further.

5.1 Extension of dynamic programming to more than two sequences

Needleman and Wunsch [13] suggested that their dynamic programming algorithm could be extended to the comparison of many sequences. Waterman *et al.* [27] also described how dynamic programming could be used to align more than two sequences. In practice, the need to store an N dimensional array (where N is the number of sequences) limits these extensions to three-sequence applications. In addition, the time required to perform the comparison of three sequences is proportional to N^5. Murata *et al.* [44] described a modification of the Needleman and Wunsch procedure for three sequences which ran in time proportional to N^3; unfortunately this approach required an additional three dimensional array thus further limiting its application to short sequences. One of the earliest practical applications of dynamic programming to multiple alignment was the work of Sankoff *et al.* [45] who aligned nine 5S RNA sequences that were linked by an evolutionary tree. Their algorithm which also constructed the protosequences at the interior nodes of the tree was made computationally feasible by decomposing the nine-sequence problem into seven three sequence alignments. The alignments were repeatedly performed working in from the periphery of the tree until no further change occurred to the protosequences.

5.2 Tree or hierarchical methods using dynamic programming

Practical methods for multiple sequence alignment based on a tree have been developed in several laboratories [38, 46, 47, 48, 49]. The principle is that since the alignment of two sequences can be achieved very easily, multiple alignments should be built by the successive application of pairwise methods.

The steps are summarised here and illustrated in Figure 5[7]:

- Compare all sequences pairwise. For N sequences there are $N \times (N-1)/2$ pairs.

[7] Figure5.ps

- Perform cluster analysis on the pairwise data to generate a hierarchy for alignment. This may be in the form of a binary tree, or a simple ordering.
- Build the multiple alignment by first aligning the most similar pair of sequences, then the next most similar pair and so on. Once an alignment of two sequences has been made, then this is fixed. Thus for a set of sequences A, B, C, D having aligned A with C and B with D the alignment of A, B, C, D is obtained by comparing the alignments of A and C with that of B and D using averaged scores at each aligned position.

This family of methods gives good usable alignments with gaps it can be applied to large numbers of sequences, and with the exception of the initial pairwise comparison step is very fast.

Although based on the successive application of pairwise methods, multiple alignment will often yield better alignments than any pair of sequences taken in isolation. This effect was illustrated by Barton and Sternberg [46] for the alignment of immunoglobulin and globin domains. Figure 6[8] shows that for some alignment pairs there is a marked improvement in accuracy over optimal pairwise aligment (e.g. variable *versus* constant domains).

5.3 Extension of segment methods to multiple alignment

A naive extension of the segment comparison methods described in section 3.1 to N sequences would require a number of comparisons in the order of the product of the sequence lengths. Clearly, as with dynamic programming methods, such an approach is not practical. Bacon and Anderson [50] reduced the magnitude of this problem by considering the alignment in one specific order. Firstly sequence one is compared to sequence two and the top M scoring pairs of segments are stored. The next sequence is then compared to these top scoring segments, and the top scoring segments from the three sequences are kept. This process is continued and leads to a list of M alignments of top scoring segments from N sequences. Bacon and Anderson also extended the statistical models of McLachlan [6] to N sequences, and used this model as well as one based on random sequences to assess the significance of the highest scoring segment alignment found. They suggested that these techniques allow sequences to be objectively grouped, even when most of the pairwise interrelationships are weak, and cite examples of

[8]Figure6.ps

applications to five Ribonucleases, three FAD-binding enzymes and five -cro like DNA binding proteins. The Bacon and Anderson (1986) algorithm shows considerable promise for the location of significant short sequence similarities. However, the method does not provide an overall alignment of the sequences and does not explicitly consider gaps. Johnson and Doolittle [51] reduce the number of segment comparisons that must be performed by progressively evaluating selected segments from each sequence within a specified 'window'. Their method generates a complete alignment of the sequences with a consideration of gaps. Unfortunately, time constraints limit its application to 4-way alignments whilst 5-way alignments become unreasonably expensive for sequence lengths above fifty residues.

A variation on segment methods is employed by the alignment tool Macaw [52]. Macaw applies the BLAST algorithm (see Section 6.7) to locate the most significant ungapped similarities irrespective of length. This facility is coupled with a flexible alignment display tool under Microsoft Windows. The program works well for small numbers of sequences, but lacks the convenience of the hierarchical dynamic programming methods (see section 5.2).

5.4 Representation and analysis of multiple alignments

How do we extract the maximum information from a multiple protein sequence alignment?

When making a multiple sequence alignment a crude tree is normally generated. The tree shows the gross relationships between the sequences. It may show that sequences A, D and C are more similar to each other than they are to B and E. However, it does not show which individual residues have changed in order to make A, D and C different from B and E. These residues may be the most important ones to investigate by site-directed mutagenesis. Livingstone and Barton [53] have described a set-based strategy to identify such differences by comparing pairs of groups of aligned residues. Their method automatically provides a text summary of the similarities and a boxed and shaded or coloured alignment. An example or the graphical output of this analysis is illustrated in Figure 7[9] for the SH2 domain family.

Providing the alignment is *accurate* then the following may be inferred about the secondary structure of the protein family:

[9] Figure7.ps

- The position of insertions and deletions suggests regions where surface loops exist in the protein.
- Conserved glycine or proline suggests a β – turn.
- Residues with hydrophobic properties conserved at $i, i+2, i+4$ separated by unconserved or hydrophilic residues suggest a surface β – strand.
- A short run of hydrophobic amino acids (4 residues) suggests a buried β – strand.
- Pairs of conserved hydrophobic amino acids separated by pairs of unconserved, or hydrophilic residues suggests an α – helix with one face packing in the protein core. Likewise, an $i, i+3, i+4, i+7$ pattern of conserved hydrophobic residues.

These patterns are not always easy to see in a single sequence, but given a multiple alignment, they often stand out and allow secondary structure to be assigned with degree of confidence. For example, patterns were used to aid the accurate prediction of the secondary structure and position of buried residues for the annexins and SH2 domains prior to knowledge of their tertiary structures [54, 55, 56].

6 Database scanning

The techniques described in the previous sections all assume that we already have two or more sequences to align. However, if we have just determined a new sequence, then our first task is to find out whether it shares similarities with other proteins that have already been sequenced. To do this we must compare our sequence to the sequence database(s) using some computer algorithm. Any of the methods described in the previous sections may be used, but database scanning presents special problems that have led to the development of specialist algorithms. In this section I will review the options and goals of these methods.

6.1 Basic principles of database searching

When scanning a database we take a query sequence, and use an algorithm to compare the query to each sequence in the database. Every pair comparison yields a score where larger scores usually indicate a higher degree of similarity. Thus, a scan of a database containing 60,000 sequences will typically provide 60,000 scores for analysis. If a local alignment method is used, then the total number of scores may be much larger since more than one "hit" may occur with each sequence. Figure 8[10] illustrates three score distributions from such a scan. The dark shaded bars show scores with sequences known to be structurally related to the query sequence whereas the light shaded bars show scores with proteins that are thought not to be related to the query. A perfect database scanning method would completely separate these two distributions as shown in Figure 8c[11]. Normally, there is some overlap between the genuinely related and unrelated sequence distributions as shown in Figures 8c and 8b[12]. There are a number of methods for ranking and re-scaling the scores to improve separation and remove artefacts due to different sequence lengths and compositions. In their most highly developed form, these methods provide an estimate of the probability of seeing a score by chance given a database of the size used and the query length. However, regardless of the method of ranking, there are nearly always some proteins giving scores in the overlap region that in fact are structurally related to the query. In practice, since no method succeeds for all protein queries, the aim is to minimise the overlap and ensure that potentially interesting similarities are scored high enough that they will be noticed by the user. Of course, what constitutes an "interesting" match is dependent upon the subjective biological context of the query.

6.2 Time considerations

In the early days of database scanning, the computer time required to execute the scan was a major consideration. Today, the ready availability of cheap, high performance computers means that computer resources are rarely a limiting factor. In the early 1980s Computers with sufficient memory and processor speed to compare a query to a database using dynamic programming were expensive shared resources. Over the last 10 years, the speed of typical institutional computers has increased by a factor of 70 while the sequence database has only grown by a factor of 9. This disparity coupled with the dramatic fall in the cost of computing means that it is currently feasible to perform protein database scans in a few hours on a

[10]Figure8.ps
[11]Figure8.ps
[12]Figure8.ps

personal computer using dynamic programming algorithms [57].

For occasional use, high scanning speed is not essential. After all, if it has taken months to obtain the sequence data, what is an few hours to check for similarities? However, much greater speed is helpful when providing a national or regional database scanning service and when carrying out analyses that require very large numbers of sequences to be compared. For example, the comparison of a 25,000 sequence database to itself would require 4.5 months using dynamic programming on a typical workstation [57]. The algorithms discussed in Sections 6.6 and 6.7 that make approximations, or implementations on specialist hardware may reduce this time by a factor of 10-100.

6.3 Which database should I search? Local or network?

The answer to this question depends a little on why you are searching performing the search. If you have just determined a new sequence then it is essential that you search the most recent and up to date databases available to test if your new protein is unique. Having the most up to date database is less important if the aim is to gather a well known family of proteins together for multiple alignment as an aid to modelling.

The nucleic acid and protein sequence databases are collated by EMBL in Europe, the NCBI in the USA and the DDBJ in Japan. In addition, the NBRF in the USA also provide a database of nucleic acid and protein sequences. The databases are distributed on CD-ROM by EMBL, NCBI and NBRF organisations and if you require a local database to scan, this is the preferred method of obtaining it. Some of the database distributions include software for searching the databases (e.g. NBRF-ATLAS program). The disks are normally updated every three months but since over 1,000 new protein sequences are deposited per month, even the current disk is out of date as soon as it arrives! To overcome this problem, the database providers also maintain daily or weekly updates to the databases since the last CD release. If searching with a newly determined sequence one should ideally scan a database that includes all available sequences up to today and if nothing is found, periodically rescan the updated database. Maintaining the regular updates of the sequence databases is usually beyond the scope of an individual investigator, however major data centres do maintain such updated databases and software for searching them. Indeed, providing you have e-mail access to the Internet and are prepared to accept the scanning tools provided by the database centre, then there is no compelling reason for maintaining the databases locally. However, while network access to a database may provide the most up to date version of the data, it does not necessarily give the most effective scanning method for your sequence.

6.4 Searching with dynamic programming

Dynamic programming requires a matrix of pair scores and a gap penalty and will return the best score for aligning the two sequences (See section 3.2). Both local and global alignment methods may be applied to database scanning, but local alignment methods are more useful since they do not make the assumption that the query protein and database sequence are of similar length.

Although it is feasible to use dynamic programming to search databases on a desktop computer (Section 6.2) the technique has not generally been adopted for database searching. This is mainly because fast implementations of the Smith-Waterman and similar algorithms (e.g. see [33]) have not been widely available until recently.

6.4.1 Scanning with parallel computers Prosrch, MPsrch and others

Collins *et al.* [15, 58] are responsible for much of the early work on scanning sequence databases with dynamic programming. They implemented a variant of the Smith and Waterman [30] algorithm on the parallel AMT-DAP computer. This provided them with sufficient processing speed to not only record the top scoring local alignment between the query and each sequence, but also to record alternative local alignments. As such, the DAP implementation of local alignment with gaps is currently the only program to provide this service on the Internet (Send the text HELP to dapmail@biocomp.ed.ac.uk). Collins *et al.* [15, 59] also provide a method to estimate the statistical significance of their alignments. They fit a straight line to $\log(N)$, where N is the number of aligments with a given score, versus score to the lower 97% of the top 16,384 alignments, then express the score for alignment as a probability derived from the distance from this line. This scoring method was used since there is currently no formal statistical method of estimating the expected score for a local alignment with gaps. The Collins *et al* approach provides a convenient way of correcting for changes in the score distribution for unrelated alignments due to differences in composition and length of the query sequence database.

A development of Collins' work is a parallel Smith-Waterman implementation for the MasPar range of massively parallel computers [60]. Scans can be made using this program using a service at EMBL Heidelberg (Send the text HELP to BLITZ@embl-heidelberg.de). Unfortunately, the BLITZ service currently only returns a single top scoring alignment, but like the DAP program it gives an estimate of the alignment significance. A further Smith-Waterman implementation (BLAZE) has been developed by Intelligenetics and is commercially available for the MasPar. The GenQuest system at Oak Ridge National Laboratory USA, also supports database searching with the Smith-Waterman algorithm using a specialised parallel computing environment (send the text HELP to grail@ornl.gov for instructions).

6.5 Index methods

6.5.1 Simple index for identical matching

Indexing has long been used for identifying identical ungapped regions in sequences. For example the SCAN facility in the PSQ and ATLAS programs distributed with the NBRF-PIR databank allows the rapid identification of short identical strings [61]. This is achieved by pre-processing the entire databank once to identify the locations of all unique tripeptides. These data are stored in a direct access file together with pointers to the sequence identifier codes. The query peptide is also divided into a series of tripeptides and identification of the sequence in the databank then becomes a simple matter of looking up the starting positions of each peptide in the list held on file. There is a tradeoff with indexing methods between the time and space taken to build and store the index and the number of queries expected. Search times are usually very fast and involve a few disk accesses, the drawback with simple indexes is that they are restricted to exact matching without gaps.

6.5.2 Indexing with gaps - the FLASH algorithm

Recently, Rigoutsos and Califano of the IBM T. J. Watson Research Center have extended the idea of indexing to allow for gaps and mismatches [62]. The indexes, or lookup tables are highly redundant and based on a probabilistic model. As a consequence the index files are very large and the problem is less one of absolute CPU speed, and more a question of fast disk access. For example, the index for SWISS-PROT release 25 requires 2.8 GBytes of disk space (I. Rigoutsos personal communication). However, the Rigoutsos and Califano FLASH algorithm permits very rapid scans to be performed on protein databases with claimed sensitivity and accuracy close to dynamic programming. The algorithm has been implemented on a network of 7 non dedicated RISC workstations which provides sufficient speed to service a searching facility *via* email (send the text SEND HELP to dflash@watson.ibm.com for information). As databases grow in size with a large amount unchanging, and the cost of disk storage falls, it seems likely that indexing techniques will become increasingly important methods of searching.

6.6 Approximations: The FASTP and FASTA algorithm

The early personal computers had insufficient memory and were too slow to carry out a database scan using dynamic programming. Accordingly, Wilbur and Lipman [63] developed a fast procedure for DNA scans that in concept searches for the most significant diagonals in a dot-plot. The initial step in the algorithm is to identify all exact matches of length k (k–tuples) or greater between the two sequences. Speed is achieved by employing a look up procedure. For example, for proteins, if $k = 3$ then there are 8,000 (20^3) possible k-tuples and each element of an array C of length 8,000 is set to represent one of these k-tuples. Sequence A is scanned once and the location of each k-tuple in A is recorded in the corresponding element of C. Sequence B is then scanned and by reference to C the location of all k-tuple matches common to A and B may be identified. If two k-tuples are present on the same diagonal then the difference between their starting position (offset) is also the same, thus the diagonals with the most significant number of matches may be identified. Since runs of identity are relatively rare even between related proteins, Lipman and Pearson [64] first identified the five diagonals of highest similarity with k set to 1, or 2. They then applied Dayhoff's scoring scheme (Section 2.4.1) to the amino acid pairs over these regions. The region giving the highest score for the protein comparison was used to rank order the sequences located in the databank for further study by more rigorous procedures. Pearson and Lipman [65] have refined these ideas in the program FASTA. FASTA saves the 10 highest regions of identity which are then re-scored with the PAM250 matrix (see Section 2.4.1). If there are several initial regions above a pre set cutoff score then those that could form a longer alignment are joined, allowing for gaps and a score *initn* is calculated by subtracting a penalty for each gap. *initn* is used to rank the database sequences by similarity. Finally, dynamic programming is used over a narrow region of the high scoring diagonal to produce an alignment with score *opt*. These steps are illustrated in Figure 9[13].

FASTA only shows the top scoring region, it does not locate all high scoring alignments between two sequences. As a consequence FASTA may not identify directly repeats or multiple domains that are shared between two proteins. The FASTA software can be obtained by anonymous ftp from virginia.edu and a number of sites offer searching facilities with FASTA (e.g. EMBL).

6.7 Approximations: BLAST Basic Local Alignment Search Tool

BLAST [18] is a heuristic method to find the highest scoring locally optimal alignments between a query sequence and a database. The important simplification that BLAST makes is not to allow gaps, but the algorithm does allow multiple hits to the same sequence. The BLAST algorithm and family of programs rely on work on the statistics of ungapped sequence alignments by Karlin and Altschul [66]. The statistics allow the probability of obtaining an ungapped alignment (MSP - Maximal Segment Pair) with a particular score to be estimated. The BLAST algorithm permits nearly all MSP's above a cutoff to be located efficiently in a database.

The algorithm operates in three steps:

1. For a given word length w (usually 3 for proteins) and score matrix (see Section 2) a list of all words (w-mers) that can can score $> T$ when compared to w-mers from the query is created.

2. The database is searched using the list of w-mers to to find the corresponding w-mers in the database (hits).

3. Each hit is extended to determine if an MSP that includes the w-mer scores $> S$, the preset threshold score for an MSP. Since pair score matrices typically include negative values, extension of the initial w-mer hit may increase or decrease the score. Accordingly, a parameter X defines how great an extension will be tried in an attempt to raise the score above S.

   The steps involved in the BLAST algorithm are illustrated in Figure 10[14].

[13] Figure9.ps
[14] Figure10.ps

A low value for T reduces the possibility of missing MSPs with the required S score, however lower T values also increase the size of the hit list generated in step 2 and hence the execution time and memory required. In practice, the BLASTP program used for protein searches sets compromise values of T and X to balance the processor requirements and sensitivity.

BLAST is unlikely to be as sensitive for all protein searches as a full dynamic programming algorithm. However, the underlying statistics provide a direct estimate of the significance of any match found. The program was developed at the NCBI and benefits from strong technical support and continuing refinement. For example, filters have recently been developed to exclude automatically regions of the query sequence that have low compositional complexity, or short periodicity internal repeats. The presence of such sequences can yield extremely large numbers of statistically significant but biologically uninteresting MSPs. For example, searching with a sequence that contains a long section of hydrophobic residues will find many proteins with transmembrane helices.

BLAST runs on virtually all types of computer (the program may be obtained by anonymous ftp from ncbi.nlm.nih.gov). Parallel processing is also supported on multi-processor computers such as the 4 or 8-processor Silicon Graphics POWER series. Searches using BLAST may be conducted by email at NCBI (Send the text HELP to blast@ncbi.nlm.nih.gov for instructions). Recently, specialist hardware has been developed to implement the BLAST algorithm at even higher speed. A network service based on these developments is available from the University of North Carolina at Chapel Hill (Send the text HELP on the subject line to bioscan@cs.unc.edu).

6.8 Guidelines for Database Scanning

Which is the best method for database scanning? Sadly, there is not a straightforward answer to this question. Attempts have been made to make comparisons but the process is complicated by the difficulty of designing suitable test cases and the number of adjustable parameters. The most effective method of assessing the success of a scanning technique is to test its ability to find all the members of a known protein family from the database of all known sequences (e.g. see [67, 68]). The principle is simple:

- Record the identifier codes of all proteins known to be in the family.
- Select a member to scan with (the query).

- Perform the scan using the method of choice.
- Count how many of the known members are found with higher scores than known non-members.

A less strict criterion is to count the number of members that score as high as the top 0.5% of the non-members in the databank [68]. The best scanning method will give the most members before non-members. i.e. will have the fewest false-positives. Of course, evaluation is not as simple as this appears. First one must choose well characterised protein families with which to test. Do we really know all the members? A high scoring non-member may in fact be a previously undiscovered family member. Further difficulites arise for scans where there are many false-negatives. If two methods both miss 30 known members, are they missing the same 30? Ideally, evaluation should also explore alternative parameter combinations, but this greatly increases the number of tests that need to be done and complicates the data analysis. For example, if we consider scanning with dynamic programming, then there is a choice of pair-score matrix and gap-penalty, local or global alignment. The best gap-penalty depends on the matrix in use. If both length-dependent and independent penalties are used, then the number of alternative combinations increases dramatically. The best combination of matrix and penalty may not be appropriate for other algorithms. BLAST does not consider gaps, so the situation is a little easier and this feature was exploited by Henikoff and Henikoff to evaluate different substitution matrices [16] however we still have the choice of other parameters special to the BLAST algorithm.

When given a newly determined sequence, a search with BLAST or FASTA will quickly tell you if a close homologue exists. Although a scan with full dynamic programming takes longer on a local workstation, the turn-round time from email servers such as BLITZ are similar to BLAST searches at NCBI. Accordingly, it is worth scanning using one of these services as well. If no similar sequences are found then alternative PAM matrices should be tried. Start with PAM120, then try PAM250 and in each case vary the gap penalty around the minimum value of the matrix. For PAM250 this is 8, values of 7-10 are worth trying. Care should always be taken to consider the likely significance of an apparent match. The methods for predicting the accuracy of alignment that are discussed in Section 4.1.

7 Summary

In the early years of sequence searching, only a few specialised centres had access to the necessary computing facilities and programming expertise to perform the scans. In the early-mid 1980s, the availability of personal computers and software that could perform useful analyses on them (e.g. FASTP) meant that it was normally most efficient for searches to be performed locally. Today, the optimum choice is again swinging towards databases maintained at a few centres, but now fast networks and windowing workstations allow the user to use software locally and be unaware that the search is being carried out on a computer in another country. Perhaps the best example of this to date is the Entrez software [69] available from the U.S. NCBI (Ask for information from info@ncbi.nlm.nih.gov). Entrez provides a windowing interface to a database that integrates the nucleotide and protein sequence databases with associated references and abstracts. Entrez will either use the database on CD-ROM or alternatively, with suitable network connection can interrogate the master database at the NCBI in Washington. While Entrez does not provide searching facilities for a new sequence it stores pre-computed similarities between pairs of sequences in the database. Thus, one can quickly navigate between a protein name, the sequence, its close homologues, the corresponding DNA sequence and all relevant publications. Network Entrez was heavily used when compiling this Chapter!

The advantages of centralised databases for the user are:

- That he need only have a comparatively low-powered computer and network connection.
- The database centres can keep the database up to date far more effectively than the individual investigator.
- The centres can provide access to a range of software to interrogate the data, either as a simple text search (e.g. find all entries with the word "kinase") or using sequence comparison algorithms. They can update the software as new algorithms become available.

The drawback with a centralised service is that one has to accept the service providers view of the best way to perform the search. However, with more database centres giving public access to search facilities every year there is an increasing choice of algorithms available.

8 Figure Legends

8.1 Figure 1

Comparison of global and local alignment. Global alignment optimizes the alignment over the full length of the sequences A and B. Local alignment locates the best alignment between sub-regions of A and B. There may be a large number of distinct local alignments.

8.2 Figure 2

Distribution of S.D. scores obtained for 100,000 alignments of length > 20 between unrelated proteins. The S.D. scores were calculated from 100 randomizations using a global alignment method [13], PAM250 matrix with 8 added to each element, and length independent gap-penalty of 8.

8.3 Figure 3

A local alignment found between citrate synthase and transthyritin. The S.D. score for this alignment is 7.55, its length is 54 residues and the identity is 25.9%. Despite this apparently high similarity, the sequences are of completely different secondary structure.

8.4 Figure 4

Plot of percentage identity versus alignment length for the 100,000 aligments from Figure 2.

8.5 Figure 5

The stages in generating a multiple sequence alignment using a hierarchical method (see text).

8.6 Figure 6

Comparison of alignment accuracy (as judged by comparison of alignments to those generated by tertiary structure comparison) for optimal pairwise and hierarchical multiple alignment by the method of Barton and Sternberg [46]. Figure adapted from [46].

8.7 Figure 7

SH2 domain analysis performed using the program AMAS [53]. The aim of this analysis is to locate patterns of amino acid conservation within each sub-group of related sequences and across all sequences in the set. The sequences are clustered by their overall similarity, then a set-based method is used to find the positions that have conserved physico-chemical properties within each group and between pairs of groups. The conservation is summarised by colour coding the alignment (shown here as grey shading). Pair conservation is summarised as a histogram. The histogram helps to locate conservation patterns characteristic of α – helix and β – strand. For full details see [53]. For details of how to obtain AMAS and other programs from the author's group please download the file README from geoff.biop.ox.ac.uk.

8.8 Figure 8

Schematic representation of typical alignment score distributions resulting from a database scan. The black bars represent proteins that are known to be similar to the query sequence, the white bars are not related to the query. (a) shows a scan that does not discriminate well, (c) shows perfect discrimination while (b) is the more usual intermediary result of database searching.

8.9 Figure 9

Summary of the steps in the FASTA sequence comparison program.

8.10 Figure 10

BLAST Algorithm.

References

[1] Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1978) A model of evolutionary change in proteins. matrices for detecting distant relationships In M. O. Dayhoff, (ed.), Atlas of protein sequence and structure, volume **5**, pp. 345–358 National biomedical research foundation Washington DC.

[2] Schwartz, R. M. and Dayhoff, M. O. (1978) In M. O. Dayhoff, (ed.), Atlas of protein sequence and structure, volume **5**, pp. 353–362 National biomedical research foundation Washington DC.

[3] Feng, D. F., Johnson, M. S., and Doolittle, R. F. (1985) *J. Mol. Evol.* **21**, 112–125.

[4] Fitch, W. M. (1966) *J. Mol. Biol.* **16**, 9–16.

[5] Cohen, F. E., Novotny, J., Sternberg, M. J. E., Campbell, D. G., and Williams, A. F. (1981) *Biochem. J.* **195**, 31–40.

[6] D., M. A. (1972) *J. Mol. Biol.* **64**, 417–37.

[7] Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992) *Comp. App. Biosci.* **8**, 275–82.

[8] Taylor, W. R. (1986) *J. Theor. Biol.* **119**, 205–218.

[9] Henikoff, S. and Henikoff, J. G. (1992) *Proc. Nat. Acad. Sci.* **89**, 10915–10919.

[10] Risler, J. L., Delorme, M. O., Delacroix, H., and Henaut, A. (1988) *J. Mol. Biol.* **204**, 1019–1029.

[11] Overington, J., Johnson, M. S., Sali, A., and Blundell, T. L. (1990) *Proc. R. Soc. Lond. B.* **241**, 132–145.

[12] Bowie, J. U., Luthy, R., and Eisenberg, D. (1991) *Science* **253**, 164–170.

[13] Needleman, S. B. and Wunsch, C. D. (1970) *J. Mol. Biol.* **48**, 443–453.

[14] Altschul, S. (1991) *J. Mol. Biol.* **1991**, 555–565.

[15] Collins, J. F., Coulson, A. F. W., and Lyall, A. (1988) *Comp. App. Biosci.* **4**, 67–71.

[16] Henikoff, S. and Henikoff, J. G. (1993) *Proteins* **17**, 49–61.

[17] Gonnet, G. H., Cohen, M. A., and Benner, S. A. (1992) *Science* **256**, 1443–1444.

[18] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) *J. Mol. Biol.* **215**, 403–410.

[19] Gibbs, A. J. and McIntyre, G. A. (1970) *European J. Biochem.* **16**, 1–11.

[20] Collins, J. F. and Coulson, A. F. W. (1987) In M. J. Bishop and C. J Rawlings, (ed.), Nucleic acid and protein sequence analysis - a practical approach, pp. 323–358 IRL Press.

[21] Jones, D. D. (1975) *J. Theor. Biol.* **50**, 167–183.

[22] Chou, P. Y. and Fasman, G. D. (1978) *Adv. Enz.* **47**, 45–148.

[23] Kubota, Y., Nishikawa, K., Takahashi, S., and Ooi, T. (1982) *Biochem.. Biophys. Acta* **701**, 242–252.

[24] Argos, P. (1987) *J. Mol. Biol.* **193**, 385–396.

[25] Kruskal, J. B. (1983) An overview of squence comparison In D. Sankoff and J. B. Kruskal, (ed.), Time warps, string edits and macromolecules: the theory and practice of sequence comparison, pp. 1–44 Addison Wesley.

[26] D. Sankoff and J. B. Kruskal, (ed.) (1983) Time warps, string edits and macromolecules: the theory and practice of sequence comparison, Addison Wesley, .

[27] Waterman, M. S., Smith, T. F., and Beyer, W. A. (1976) *Advances in Mathematics* **20**, 367–387.

[28] Gotoh, O. (1982) *J. Mol. Biol.* **162**, 705–708.

[29] Sellers, P. H. (1984) *Bull. Math. Biol.* **46**, 501–514.

[30] Smith, T. F. and Waterman, M. S. (1981) *J. Mol. Biol.* **147**, 195–197.

[31] Boswell, D. R. and McLachlan, A. D. (1984) *Nuc. Ac. Res.* **12**, 457–464.

[32] Waterman, M. S. and Eggert, M. (1987) *J. Mol. Biol.* **197**, 723–728.

[33] Barton, G. J. (1993) *Comp. App. Biosci.* **9**, 729–734.

[34] Taylor, W. and Orengo, C. (1989) *J. Mol. Biol.* **208**, 1–21.

[35] Sali, A. and Blundell, T. L. (1990) *J. Mol. Biol.* **212**, 403–428.

[36] Russell, R. B. and Barton, G. J. (1992) *Proteins* **14**, 309–323.

[37] Barton, G. J. and Sternberg, M. J. E. (1987) *Prot. Eng.* **1**, 89–94.

[38] Barton, G. J. (1990) *Meth. Enz.* **183**, 403–428.

[39] Sander, C. and Schneider, R. (1991) *Proteins* **9**, 56–68.

[40] Vingron, M. and Argos, P. (1990) *Prot. Eng.* **3**, 565–569.

[41] Zuker, M. (1991) *J. Mol. Biol.* **221**, 403–420.

[42] Saqi, M. A. and Sternberg, M. J. (1991) *J. Mol. Biol.* **219**, 727–732.

[43] Lesk, A. M., Levitt, M., and Chothia, C. (1986) *Prot. Eng.* **1**, 77–78.

[44] Murata, M., Richardson, J. S., and Sussman, J. L. (1985) *Proc. Nat. Acad. Sci.* **82**, 3073–3077.

[45] Sankoff, D., Cedergren, R. J., and Lapalme, G. (1976) *J. Mol. Evol.* **7**, 133–149.

[46] Barton, G. J. and Sternberg, M. J. E. (1987) *J. Mol. Biol.* **198**, 327–337.

[47] Feng, D. F. and Doolittle, R. F. (1987) *J. Mol. Evol.* **25**, 351–360.

[48] Taylor, W. R. (1990) *Meth. Enz.* **183**, 456–74.

[49] Higgins, D. G. and Sharp, P. M. (1989) *Comp. App. Biosci.* **5**, 151–153.

[50] Bacon, D. J. and Anderson, W. F. (1986) *J. Mol. Biol.* **191**, 153–161.

[51] Johnson, M. S. and Doolittle, R. F. (1986) *J. Mol. Evol.* **23**, 267–278.

[52] Schuler, G. D., Altschul, S. F., and Lipman, D. J. (1991) *Proteins* **9**, 180–190.

[53] Livingstone, C. D. and Barton, G. J. (1993) *Comp. App. Biosci.* **9**, 745–756.

[54] Barton, G. J., Newman, R. H., Freemont, P. F., and Crumpton, M. J. (1991) *European J. Biochem.* **198**, 749–760.

[55] Russell, R. B., Breed, J., and Barton, G. J. (1992) *FEBS Letters* **304**, 15–20.

[56] Barton, G. J. and Russell, R. B. (1993) *Nature* **361**, 505–506.

[57] Barton, G. J. (1992) *Science* **257**, 1609.

[58] Coulson, A. F. W., Collins, J. F., and Lyall, A. (1987) *The Computer Journal* **30**, 420–424.

[59] Collins, J. F. and W., C. A. F. (1990) *Meth. Enz.* **183**, 474–487.

[60] Sturrock, S. S. and Collins, J. F. MPsrch version 1.3 (1993).

[61] Orcott, B. C. and Barker, W. C. (1984) *Bull. Math. Biol.* **46**, 545–552.

[62] Rigoutsos, I. and Califano, A. (July, 1993) In Proceedings of the First International Conference on Intelligent Systems for Molecular Biology : .

[63] Wilbur, W. J. and Lipman, D. J. (1983) *Proc. Nat. Acad. Sci.* **80**, 726–730.

[64] Lipman, D. J. and Pearson, W. R. (1985) *Science* **227**, 1435–1441.

[65] Pearson, W. R. and Lipman, D. J. (1988) *Proc. Nat. Acad. Sci.* **85**, 2444–2448.

[66] Karlin, S. and Altschul, S. F. (1990) *Proc. Nat. Acad. Sci.* **87**, 2264–2268.

[67] Barton, G. J. and Sternberg, M. J. E. (1990) *J. Mol. Biol.* **212**, 389–402.

[68] Pearson, W. (1991) *Genomics* **11**, 635–650.

[69] Entrez: Sequences Bldg. 38A, NIH, Bethesda, MD 20894, USA (1992).