CHAPTER THREE

Pairwise Sequence Alignment

Sequence comparison lies at the heart of bioinformatics analysis. It is an important first step toward structural and functional analysis of newly determined sequences. As new biological sequences are being generated at exponential rates, sequence comparison is becoming increasingly important to draw functional and evolutionary inference of a new protein with proteins already existing in the database. The most fundamental process in this type of comparison is sequence alignment. This is the process by which sequences are compared by searching for common character patterns and establishing residue–residue correspondence among related sequences. Pairwise sequence alignment is the process of aligning two sequences and is the basis of database similarity searching (see Chapter 4) and multiple sequence alignment (see Chapter 5). This chapter introduces the basics of pairwise alignment.

EVOLUTIONARY BASIS

DNA and proteins are products of evolution. The building blocks of these biological macromolecules, nucleotide bases, and amino acids form linear sequences that determine the primary structure of the molecules. These molecules can be considered molecular fossils that encode the history of millions of years of evolution. During this time period, the molecular sequences undergo random changes, some of which are selected during the process of evolution. As the selected sequences gradually accumulate mutations and diverge over time, traces of evolution may still remain in certain portions of the sequences to allow identification of the common ancestry. The presence of evolutionary traces is because some of the residues that perform key functional and structural roles tend to be preserved by natural selection; other residues that may be less crucial for structure and function tend to mutate more frequently. For example, active site residues of an enzyme family tend to be conserved because they are responsible for catalytic functions. Therefore, by comparing sequences through alignment, patterns of conservation and variation can be identified. The degree of sequence conservation in the alignment reveals evolutionary relatedness of different sequences, whereas the variation between sequences reflects the changes that have occurred during evolution in the form of substitutions, insertions, and deletions.

Identifying the evolutionary relationships between sequences helps to characterize the function of unknown sequences. When a sequence alignment reveals *significant*

similarity among a group of sequences, they can be considered as belonging to the same family (protein families will be further described in Chapter 7). If one member within the family has a known structure and function, then that information can be transferred to those that have not yet been experimentally characterized. Therefore, sequence alignment can be used as basis for prediction of structure and function of uncharacterized sequences.

Sequence alignment provides inference for the relatedness of two sequences under study. If the two sequences share significant similarity, it is extremely unlikely that the extensive similarity between the two sequences has been acquired randomly, meaning that the two sequences must have derived from a common evolutionary origin. When a sequence alignment is generated correctly, it reflects the evolutionary relationship of the two sequences: regions that are aligned but not identical represent residue substitutions; regions where residues from one sequence correspond to nothing in the other represent insertions or deletions that have taken place on one of the sequences during evolution. It is also possible that two sequences have derived from a common ancestor, but may have diverged to such an extent that the common ancestral relationships are not recognizable at the sequence level. In that case, the distant evolutionary relationships have to be detected using other methods (see Chapter 15).

SEQUENCE HOMOLOGY VERSUS SEQUENCE SIMILARITY

An important concept in sequence analysis is sequence homology. When two sequences are descended from a common evolutionary origin, they are said to have a *homologous relationship* or share *homology*. A related but different term is *sequence similarity*, which is the percentage of aligned residues that are similar in physiochemical properties such as size, charge, and hydrophobicity.

It is important to distinguish sequence homology from the related term sequence similarity because the two terms are often confused by some researchers who use them interchangeably in scientific literature. To be clear, *sequence homology* is an inference or a conclusion about a common ancestral relationship drawn from sequence similarity comparison when the two sequences share a high enough degree of similarity. On the other hand, *similarity* is a direct result of observation from the sequence alignment. Sequence similarity can be quantified using percentages; homology is a qualitative statement. For example, one may say that two sequences share 40% similarity. It is incorrect to say that the two sequences share 40% homology. They are either homologous or nonhomologous.

Generally, if the sequence similarity level is high enough, a common evolutionary relationship can be inferred. In dealing with real research problems, the issue of at what similarity level can one infer homologous relationships is not always clear. The answer depends on the type of sequences being examined and sequence lengths. Nucleotide sequences consist of only four characters, and therefore, unrelated sequences have

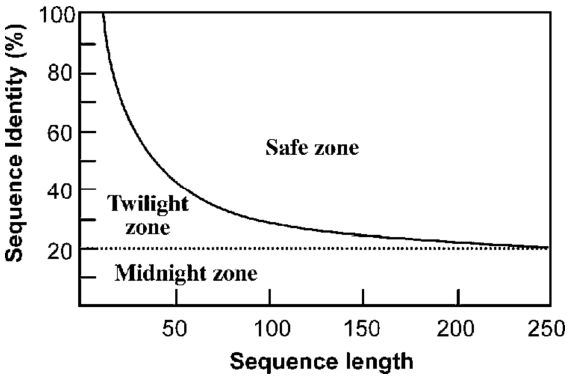


Figure 3.1: The three zones of protein sequence alignments. Two protein sequences can be regarded as homologous if the percentage sequence identity falls in the safe zone. Sequence identity values below the zone boundary, but above 20%, are considered to be in the twilight zone, where homologous relationships are less certain. The region below 20% is the midnight zone, where homologous relationships cannot be reliably determined. (*Source:* Modified from Rost 1999).

at least a 25% chance of being identical. For protein sequences, there are twenty possible amino acid residues, and so two unrelated sequences can match up 5% of the residues by random chance. If gaps are allowed, the percentage could increase to 10–20%. Sequence length is also a crucial factor. The shorter the sequence, the higher the chance that some alignment is attributable to random chance. The longer the sequence, the less likely the matching at the same level of similarity is attributable to random chance.

This suggests that shorter sequences require higher cutoffs for inferring homologous relationships than longer sequences. For determining a homology relationship of two protein sequences, for example, if both sequences are aligned at full length, which is 100 residues long, an identity of 30% or higher can be safely regarded as having close homology. They are sometimes referred to as being in the "safe zone" (Fig. 3.1). If their identity level falls between 20% and 30%, determination of homologous relationships in this range becomes less certain. This is the area often regarded as the "twilight zone," where remote homologs mix with randomly related sequences. Below 20% identity, where high proportions of nonrelated sequences are present, homologous relationships cannot be reliably determined and thus fall into the "midnight zone." It needs to be stressed that the percentage identity values only provide a tentative guidance for homology identification. This is not a precise rule for determining sequence relationships, especially for sequences in the twilight zone. A statistically more rigorous approach to determine homologous relationships is introduced in the section on the Statistical Significance of Sequence Alignment.

SEQUENCE SIMILARITY VERSUS SEQUENCE IDENTITY

Another set of related terms for sequence comparison are sequence similarity and sequence identity. Sequence similarity and sequence identity are synonymous for nucleotide sequences. For protein sequences, however, the two concepts are very

different. In a protein sequence alignment, *sequence identity* refers to the percentage of matches of the same amino acid residues between two aligned sequences. *Similarity* refers to the percentage of aligned residues that have similar physicochemical characteristics and can be more readily substituted for each other.

There are two ways to calculate the sequence similarity/identity. One involves the use of the overall sequence lengths of both sequences; the other normalizes by the size of the shorter sequence. The first method uses the following formula:

$$S = [(L_s \times 2)/(L_a + L_b)] \times 100 \quad \text{(Eq. 3.1)}$$

where S is the percentage sequence similarity, L_s is the number of aligned residues with similar characteristics, and L_a and L_b are the total lengths of each individual sequence. The sequence identity (I%) can be calculated in a similar fashion:

$$I = [(L_i \times 2)/(L_a + L_b)] \times 100 \quad \text{(Eq. 3.2)}$$

where L_i is the number of aligned identical residues.

The second method of calculation is to derive the percentage of identical/similar residues over the full length of the smaller sequence using the formula:

$$\text{I(S)\%} = L_{i(s)}/L_a\,\% \quad \text{(Eq. 3.3)}$$

where L_a is the length of the shorter of the two sequences.

METHODS

The overall goal of pairwise sequence alignment is to find the best pairing of two sequences, such that there is maximum correspondence among residues. To achieve this goal, one sequence needs to be shifted relative to the other to find the position where maximum matches are found. There are two different alignment strategies that are often used: global alignment and local alignment.

Global Alignment and Local Alignment

In *global alignment*, two sequences to be aligned are assumed to be generally similar over their entire length. Alignment is carried out from beginning to end of both sequences to find the best possible alignment across the entire length between the two sequences. This method is more applicable for aligning two closely related sequences of roughly the same length. For divergent sequences and sequences of variable lengths, this method may not be able to generate optimal results because it fails to recognize highly similar local regions between the two sequences.

Local alignment, on the other hand, does not assume that the two sequences in question have similarity over the entire length. It only finds local regions with the highest level of similarity between the two sequences and aligns these regions without regard for the alignment of the rest of the sequence regions. This approach can be

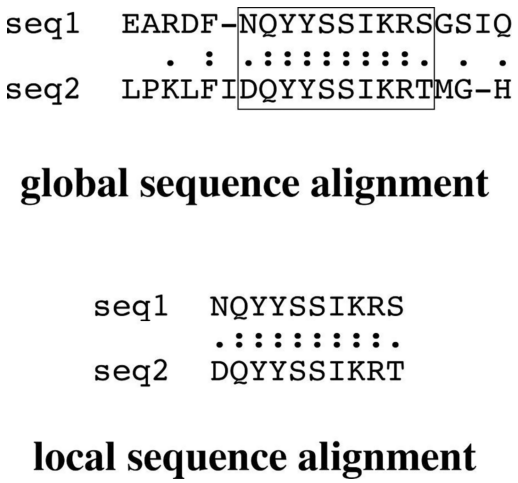


Figure 3.2: An example of pairwise sequence comparison showing the distinction between global and local alignment. The global alignment (*top*) includes all residues of both sequences. The region with the highest similarity is highlighted in a box. The local alignment only includes portions of the two sequences that have the highest regional similarity. In the line between the two sequences, ":" indicates identical residue matches and "." indicates similar residue matches.

used for aligning more divergent sequences with the goal of searching for conserved patterns in DNA or protein sequences. The two sequences to be aligned can be of different lengths. This approach is more appropriate for aligning divergent biological sequences containing only modules that are similar, which are referred to as *domains* or *motifs*. Figure 3.2 illustrates the differences between global and local pairwise alignment.

Alignment Algorithms

Alignment algorithms, both global and local, are fundamentally similar and only differ in the optimization strategy used in aligning similar residues. Both types of algorithms can be based on one of the three methods: the dot matrix method, the dynamic programming method, and the word method. The dot matrix and dynamic programming methods are discussed herein. The word method, which is used in fast database similarity searching, is introduced in Chapter 4.

Dot Matrix Method

The most basic sequence alignment method is the dot matrix method, also known as the *dot plot method*. It is a graphical way of comparing two sequences in a two-dimensional matrix. In a dot matrix, two sequences to be compared are written in the horizontal and vertical axes of the matrix. The comparison is done by scanning each residue of one sequence for similarity with all residues in the other sequence. If a residue match is found, a dot is placed within the graph. Otherwise, the matrix positions are left blank. When the two sequences have substantial regions of similarity, many dots line up to form contiguous diagonal lines, which reveal the sequence alignment. If there are interruptions in the middle of a diagonal line, they indicate insertions or deletions. Parallel diagonal lines within the matrix represent repetitive regions of the sequences (Fig. 3.3).

A problem exists when comparing large sequences using the dot matrix method, namely, the high noise level. In most dot plots, dots are plotted all over the graph,

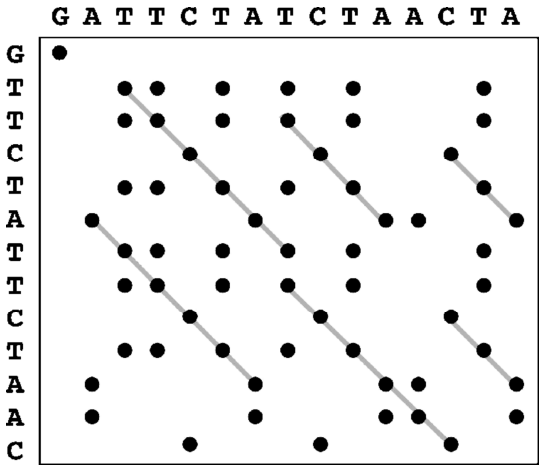


Figure 3.3: Example of comparing two sequences using dot plots. Lines linking the dots in diagonals indicate sequence alignment. Diagonal lines above or below the main diagonal represent internal repeats of either sequence.

obscuring identification of the true alignment. For DNA sequences, the problem is particularly acute because there are only four possible characters in DNA and each residue therefore has a one-in-four chance of matching a residue in another sequence. To reduce noise, instead of using a single residue to scan for similarity, a filtering technique has to be applied, which uses a "window" of fixed length covering a stretch of residue pairs. When applying filtering, windows slide across the two sequences to compare all possible stretches. Dots are only placed when a stretch of residues equal to the window size from one sequence matches completely with a stretch of another sequence. This method has been shown to be effective in reducing the noise level. The window is also called a *tuple*, the size of which can be manipulated so that a clear pattern of sequence match can be plotted. However, if the selected window size is too long, sensitivity of the alignment is lost.

There are many variations of using the dot plot method. For example, a sequence can be aligned with itself to identify internal repeat elements. In the self-comparison, there is a main diagonal for perfect matching of each residue. If repeats are present, short parallel lines are observed above and below the main diagonal. Self-complementarity of DNA sequences (also called *inverted repeats*) – for example, those that form the stems of a hairpin structure – can also be identified using a dot plot. In this case, a DNA sequence is compared with its reverse-complemented sequence. Parallel diagonals represent the inverted repeats. For comparing protein sequences, a weighting scheme has to be used to account for similarities of physicochemical properties of amino acid residues.

The dot matrix method gives a direct visual statement of the relationship between two sequences and helps easy identification of the regions of greatest similarities. One particular advantage of this method is in identification of sequence repeat regions based on the presence of parallel diagonals of the same size vertically or horizontally in the matrix. The method thus has some applications in genomics. It is useful in identifying chromosomal repeats and in comparing gene order conservation between two closely related genomes (see Chapter 17). It can also be used in identifying nucleic

acid secondary structures through detecting self-complementarity of a sequence (see Chapter 16).

The dot matrix method displays all possible sequence matches. However, it is often up to the user to construct a full alignment with insertions and deletions by linking nearby diagonals. Another limitation of this visual analysis method is that it lacks statistical rigor in assessing the quality of the alignment. The method is also restricted to pairwise alignment. It is difficult for the method to scale up to multiple alignment. The following are examples of web servers that provide pairwise sequence comparison using dot plots.

Dotmatcher (bioweb.pasteur.fr/seqanal/interfaces/dotmatcher.html) and Dottup (bioweb.pasteur.fr/seqanal/interfaces/dottup.html) are two programs of the EMBOSS package, which have been made available online. Dotmatcher aligns and displays dot plots of two input sequences (DNA or proteins) in FASTA format. A window of specified length and a scoring scheme are used. Diagonal lines are only plotted over the position of the windows if the similarity is above a certain threshold. Dottup aligns sequences using the word method (to be described in Chapter 4) and is capable of handling genome-length sequences. Diagonal lines are only drawn if exact matches of words of specified length are found.

Dothelix (www.genebee.msu.su/services/dhm/advanced.html) is a dot matrix program for DNA or protein sequences. The program has a number of options for length threshold (similar to window size) and implements scoring matrices for protein sequences. In addition to drawing diagonal lines with similarity scores above a certain threshold, the program displays actual pairwise alignment.

MatrixPlot (www.cbs.dtu.dk/services/MatrixPlot/) is a more sophisticated matrix plot program for alignment of protein and nucleic acid sequences. The user has the option of adding information such as sequence logo profiles (see Chapter 7) and distance matrices from known three-dimensional structures of proteins or nucleic acids. Instead of using dots and lines, the program uses colored grids to indicate alignment or other user-defined information.

Dynamic Programming Method

Dynamic programming is a method that determines optimal alignment by matching two sequences for all possible pairs of characters between the two sequences. It is fundamentally similar to the dot matrix method in that it also creates a two-dimensional alignment grid. However, it finds alignment in a more quantitative way by converting a dot matrix into a scoring matrix to account for matches and mismatches between sequences. By searching for the set of highest scores in this matrix, the best alignment can be accurately obtained.

Dynamic programming works by first constructing a two-dimensional matrix whose axes are the two sequences to be compared. The residue matching is according to a particular scoring matrix. The scores are calculated one row at a time. This starts with the first row of one sequence, which is used to scan through the entire length of the other sequence, followed by scanning of the second row. The matching scores

are calculated. The scanning of the second row takes into account the scores already obtained in the first round. The best score is put into the bottom right corner of an intermediate matrix (Fig. 3.4). This process is iterated until values for all the cells are filled. Thus, the scores are accumulated along the diagonal going from the upper left corner to the lower right corner. Once the scores have been accumulated in matrix, the next step is to find the path that represents the optimal alignment. This is done by tracing back through the matrix in reverse order from the lower right-hand corner of the matrix toward the origin of the matrix in the upper left-hand corner. The best matching path is the one that has the maximum total score (see Fig. 3.4). If two or more paths reach the same highest score, one is chosen arbitrarily to represent the best alignment. The path can also move horizontally or vertically at a certain point, which corresponds to introduction of a gap or an insertion or deletion for one of the two sequences.

Gap Penalties. Performing optimal alignment between sequences often involves applying gaps that represent insertions and deletions. Because in natural evolutionary processes insertion and deletions are relatively rare in comparison to substitutions, introducing gaps should be made more difficult computationally, reflecting the rarity of insertional and deletional events in evolution. However, assigning penalty values can be more or less arbitrary because there is no evolutionary theory to determine a precise cost for introducing insertions and deletions. If the penalty values are set too low, gaps can become too numerous to allow even nonrelated sequences to be matched up with high similarity scores. If the penalty values are set too high, gaps may become too difficult to appear, and reasonable alignment cannot be achieved, which is also unrealistic. Through empirical studies for globular proteins, a set of penalty values have been developed that appear to suit most alignment purposes. They are normally implemented as default values in most alignment programs.

Another factor to consider is the cost difference between opening a gap and extending an existing gap. It is known that it is easier to extend a gap that has already been started. Thus, gap opening should have a much higher penalty than gap extension. This is based on the rationale that if insertions and deletions ever occur, several adjacent residues are likely to have been inserted or deleted together. These differential gap penalties are also referred to as *affine gap penalties*. The normal strategy is to use preset gap penalty values for introducing and extending gaps. For example, one may use a $-12/-1$ scheme in which the gap opening penalty is -12 and the gap extension penalty -1. The total gap penalty (W) is a linear function of gap length, which is calculated using the formula:

$$W = \gamma + \delta \times (k - 1) \qquad \text{(Eq. 3.4)}$$

where γ is the gap opening penalty, δ is the gap extension penalty, and k is the length of the gap. Besides the affine gap penalty, a *constant gap penalty* is sometimes also used, which assigns the same score for each gap position regardless whether it is

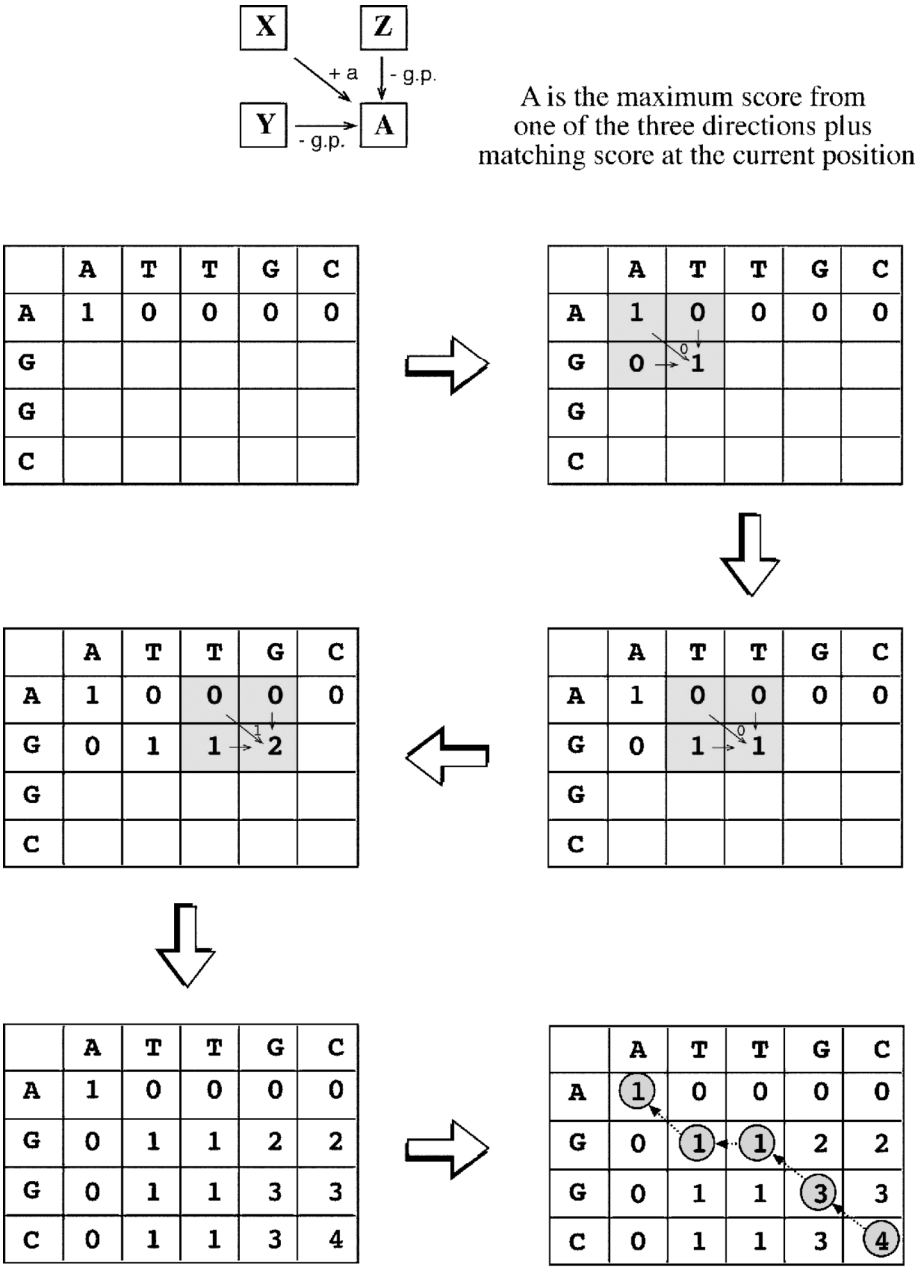


Figure 3.4: Example of pairwise alignment of two sequences using dynamic programming. The score for the lower right square **(A)** of a 2 × 2 matrix is the maximum score from the one of other three neighboring squares (X, Y, and Z) plus and minus the exact single residue match score (a) for the lower right corner and the gap penalty (g.p.), respectively. A matrix is set up for the two short sequences. A simple scoring system is applied in which an identical match is assigned a score of 1, a mismatch a score 0, and gap penalty (see below) is −1. The scores in the matrix are filled one row at a time and one cell at a time beginning from top to bottom. The best scores are filled to the lower right corner of a submatrix (grey boxes) according to this rule. When all the cells are filled with scores, a best alignment is determined through a trace-back procedure to search for the path with the best total score. When a path moves horizontally or vertically, a penalty is applied.

opening or extending. However, this penalty scheme has been found to be less realistic than the affine penalty.

Gaps at the terminal regions are often treated with no penalty because in reality many true homologous sequences are of different lengths. Consequently, end gaps can be allowed to be free to avoid getting unrealistic alignments.

Dynamic Programming for Global Alignment. The classical global pairwise alignment algorithm using dynamic programming is the Needleman–Wunsch algorithm. In this algorithm, an optimal alignment is obtained over the entire lengths of the two sequences. It must extend from the beginning to the end of both sequences to achieve the highest total score. In other words, the alignment path has to go from the bottom right corner of the matrix to the top left corner. The drawback of focusing on getting a maximum score for the full-length sequence alignment is the risk of missing the best local similarity. This strategy is only suitable for aligning two closely related sequences that are of the same length. For divergent sequences or sequences with different domain structures, the approach does not produce optimal alignment. One of the few web servers dedicated to global pairwise alignment is GAP.

GAP (http://bioinformatics.iastate.edu/aat/align/align.html) is a web-based pairwise global alignment program. It aligns two sequences without penalizing terminal gaps so similar sequences of unequal lengths can be aligned. To be able to insert long gaps in the alignment, such gaps are treated with a constant penalty. This feature is useful in aligning cDNA to exons in genomic DNA containing the same gene.

Dynamic Programming for Local Alignment. In regular sequence alignment, the divergence level between the two sequences to be aligned is not easily known. The sequence lengths of the two sequences may also be unequal. In such cases, identification of regional sequence similarity may be of greater significance than finding a match that includes all residues. The first application of dynamic programming in local alignment is the Smith–Waterman algorithm. In this algorithm, positive scores are assigned for matching residues and zeros for mismatches. No negative scores are used. A similar tracing-back procedure is used in dynamic programming. However, the alignment path may begin and end internally along the main diagonal. It starts with the highest scoring position and proceeds diagonally up to the left until reaching a cell with a zero. Gaps are inserted if necessary. In this case, affine gap penalty is often used. Occasionally, several optimally aligned segments with best scores are obtained. As in the global alignment, the final result is influenced by the choice of scoring systems (to be described next) used. The goal of local alignment is to get the highest alignment score locally, which may be at the expense of the highest possible overall score for a full-length alignment. This approach may be suitable for aligning divergent sequences or sequences with multiple domains that may be of different origins. Most commonly used pairwise alignment web servers apply the local alignment strategy, which include SIM, SSEARCH, and LALIGN.

SIM (http://bioinformatics.iastate.edu/aat/align/align.html) is a web-based program for pairwise alignment using the Smith–Waterman algorithm that finds the best scored nonoverlapping local alignments between two sequences. It is able to handle tens of kilobases of genomic sequence. The user has the option to set a scoring matrix and gap penalty scores. A specified number of best scored alignments are produced.

SSEARCH (http://pir.georgetown.edu/pirwww/search/pairwise.html) is a simple web-based programs that uses the Smith–Waterman algorithm for pairwise alignment of sequences. Only one best scored alignment is given. There is no option for scoring matrices or gap penalty scores.

LALIGN (www.ch.embnet.org/software/LALIGN_form.html) is a web-based program that uses a variant of the Smith–Waterman algorithm to align two sequences. Unlike SSEARCH, which returns the single best scored alignment, LALIGN gives a specified number of best scored alignments. The user has the option to set the scoring matrix and gap penalty scores. The same web interface also provides an option for global alignment performed by the ALIGN program.

SCORING MATRICES

In the dynamic programming algorithm presented, the alignment procedure has to make use of a scoring system, which is a set of values for quantifying the likelihood of one residue being substituted by another in an alignment. The scoring systems is called a *substitution matrix* and is derived from statistical analysis of residue substitution data from sets of reliable alignments of highly related sequences.

Scoring matrices for nucleotide sequences are relatively simple. A positive value or high score is given for a match and a negative value or low score for a mismatch. This assignment is based on the assumption that the frequencies of mutation are equal for all bases. However, this assumption may not be realistic; observations show that transitions (substitutions between purines and purines or between pyrimidines and pyrimidines) occur more frequently than transversions (substitutions between purines and pyrimidines). Therefore, a more sophisticated statistical model with different probability values to reflect the two types of mutations is needed (see the Kimura model in Chapter 10).

Scoring matrices for amino acids are more complicated because scoring has to reflect the physicochemical properties of amino acid residues, as well as the likelihood of certain residues being substituted among true homologous sequences. Certain amino acids with similar physicochemical properties can be more easily substituted than those without similar characteristics. Substitutions among similar residues are likely to preserve the essential functional and structural features. However, substitutions between residues of different physicochemical properties are more likely to cause disruptions to the structure and function. This type of disruptive substitution is less likely to be selected in evolution because it renders nonfunctional proteins.

For example, phenylalanine, tyrosine, and tryptophan all share aromatic ring structures. Because of their chemical similarities, they are easily substituted for each other without perturbing the regular function and structure of the protein. Similarly, arginine, lysine, and histidine are all large basic residues and there is a high probability of them being substituted for each other. Aspartic acid, glutamic acid, asparagine, and glutamine belong to the acid and acid amide groups and can be associated with relatively high frequencies of substitution. The hydrophobic residue group includes methionine, isoleucine, leucine, and valine. Small and polar residues include serine, threonine, and cysteine. Residues within these groups have high likelihoods of being substituted for each other. However, cysteine contains a sulfhydryl group that plays a role in metal binding, active site, and disulfide bond formation. Substitution of cysteine with other residues therefore often abolishes the enzymatic activity or destabilizes the protein structure. It is thus a very infrequently substituted residue. The small and nonpolar residues such as glycine and proline are also unique in that their presence often disrupts regular protein secondary structures (see Chapter 12). Thus, substitutions with these residues do not frequently occur. For more information on grouping amino acids based on physicochemical properties, see Table 12.1.

Amino Acid Scoring Matrices

Amino acid substitution matrices, which are 20×20 matrices, have been devised to reflect the likelihood of residue substitutions. There are essentially two types of amino acid substitution matrices. One type is based on interchangeability of the genetic code or amino acid properties, and the other is derived from empirical studies of amino acid substitutions. Although the two different approaches coincide to a certain extent, the first approach, which is based on the genetic code or the physicochemical features of amino acids, has been shown to be less accurate than the second approach, which is based on surveys of actual amino acid substitutions among related proteins. Thus, the empirical approach has gained the most popularity in sequence alignment applications and is the focus of our next discussion.

The empirical matrices, which include PAM and BLOSUM matrices, are derived from actual alignments of highly similar sequences. By analyzing the probabilities of amino acid substitutions in these alignments, a scoring system can be developed by giving a high score for a more likely substitution and a low score for a rare substitution.

For a given substitution matrix, a positive score means that the frequency of amino acid substitutions found in a data set of homologous sequences is greater than would have occurred by random chance. They represent substitutions of very similar residues or identical residues. A zero score means that the frequency of amino acid substitutions found in the homologous sequence data set is equal to that expected by chance. In this case, the relationship between the amino acids is weakly similar at best in terms of physicochemical properties. A negative score means that the frequency of amino acid substitutions found in the homologous sequence data set is less than would

have occurred by random chance. This normally occurs with substitutions between dissimilar residues.

The substitution matrices apply logarithmic conversions to describe the probability of amino acid substitutions. The converted values are the so-called log-odds scores (or log-odds ratios), which are logarithmic ratios of the observed mutation frequency divided by the probability of substitution expected by random chance. The conversion can be either to the base of 10 or to the base of 2. For example, in an alignment that involves ten sequences, each having only one aligned position, nine of the sequences are F (phenylalanine) and the remaining one I (isoleucine). The observed frequency of I being substituted by F is one in ten (0.1), whereas the probability of I being substituted by F by random chance is one in twenty (0.05). Thus, the ratio of the two probabilities is 2 (0.1/0.05). After taking this ratio to the logarithm to the base of 2, this makes the log odds equal to 1. This value can then be interpreted as the likelihood of substitution between the two residues being 2^1, which is two times more frequently than by random chance.

PAM Matrices

The PAM matrices (also called Dayhoff PAM matrices) were first constructed by Margaret Dayhoff, who compiled alignments of seventy-one groups of very closely related protein sequences. PAM stands for "point accepted mutation" (although "accepted point mutation" or APM may be a more appropriate term, PAM is easier to pronounce). Because of the use of very closely related homologs, the observed mutations were not expected to significantly change the common function of the proteins. Thus, the observed amino acid mutations are considered to be accepted by natural selection.

These protein sequences were clustered based on phylogenetic reconstruction using maximum parsimony (see Chapter 11). The PAM matrices were subsequently derived based on the evolutionary divergence between sequences of the same cluster. One PAM unit is defined as 1% of the amino acid positions that have been changed. To construct a PAM1 substitution table, a group of closely related sequences with mutation frequencies corresponding to one PAM unit is chosen. Based on the collected mutational data from this group of sequences, a substitution matrix can be derived.

Construction of the PAM1 matrix involves alignment of full-length sequences and subsequent construction of phylogenetic trees using the parsimony principle. This allows computation of ancestral sequences for each internal node of the trees (see Chapter 11). Ancestral sequence information is used to count the number of substitutions along each branch of a tree. The PAM score for a particular residue pair is derived from a multistep procedure involving calculations of relative mutability (which is the number of mutational changes from a common ancestor for a particular amino acid residue divided by the total number of such residues occurring in an alignment), normalization of the expected residue substitution frequencies by random chance, and logarithmic transformation to the base of 10 of the normalized

TABLE 3.1. Correspondence of PAM Numbers with Observed Amino Acid Mutational Rates

PAM Number	Observed Mutation Rate (%)	Sequence Identity (%)
0	0	100
1	1	99
30	25	75
80	50	50
110	40	60
200	75	25
250	80	20

mutability value divided by the frequency of a particular residue. The resulting value is rounded to the nearest integer and entered into the substitution matrix, which reflects the likelihood of amino acid substitutions. This completes the log-odds score computation. After compiling all substitution probabilities of possible amino acid mutations, a 20×20 PAM matrix is established. Positive scores in the matrix denote substitutions occurring more frequently than expected among evolutionarily conserved replacements. Negative scores correspond to substitutions that occur less frequently than expected.

Other PAM matrices with increasing numbers for more divergent sequences are extrapolated from PAM1 through matrix multiplication. For example, PAM80 is produced by values of the PAM1 matrix multiplied by itself eighty times. The mathematical transformation accounts for multiple substitutions having occurred in an amino acid position during evolution. For example, when a mutation is observed as F replaced by I, the evolutionary changes may have actually undergone a number of intermediate steps before becoming I, such as in a scenario of $F \rightarrow M \rightarrow L \rightarrow I$. For that reason, a PAM80 matrix only corresponds to 50% of observed mutational rates.

A PAM unit is defined as 1% amino acid change or one mutation per 100 residues. The increasing PAM numbers correlate with increasing PAM units and thus evolutionary distances of protein sequences (Table 3.1). For example, PAM250, which corresponds to 20% amino acid identity, represents 250 mutations per 100 residues. In theory, the number of evolutionary changes approximately corresponds to an expected evolutionary span of 2,500 million years. Thus, the PAM250 matrix is normally used for divergent sequences. Accordingly, PAM matrices with lower serial numbers are more suitable for aligning more closely related sequences. The extrapolated values of the PAM250 amino acid substitution matrix are shown in Figure 3.5.

BLOSUM Matrices

In the PAM matrix construction, the only direct observation of residue substitutions is in PAM1, based on a relatively small set of extremely closely related sequences. Sequence alignment statistics for more divergent sequences are not available. To fill

C	12																		
S	0	2																	
T	-2	1	3																
P	-3	1	0	6															
A	-2	1	1	1	2														
G	-3	1	0	-1	1	5													
N	-4	1	0	-1	0	0	2												
D	-5	0	0	-1	0	1	2	4											
E	-5	0	0	-1	0	0	1	3	4										
Q	-5	-1	-1	0	0	-1	1	2	2	4									
H	-3	-1	-1	0	-1	-2	2	1	1	3	6								
R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6							
K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5						
M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6					
I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5				
L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-2	4	2	6			
V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4		
F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9	
Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10
W	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17
	C	**S**	**T**	**P**	**A**	**G**	**N**	**D**	**E**	**Q**	**H**	**R**	**K**	**M**	**I**	**L**	**V**	**F**	**Y**	**W**

Figure 3.5: PAM250 amino acid substitution matrix. Residues are grouped according to physicochemical similarities.

in the gap, a new set of substitution matrices have been developed. This is the series of blocks amino acid substitution matrices (BLOSUM), all of which are derived based on direct observation for every possible amino acid substitution in multiple sequence alignments. These were constructed based on more than 2,000 conserved amino acid patterns representing 500 groups of protein sequences. The sequence patterns, also called *blocks*, are ungapped alignments of less than sixty amino acid residues in length. The frequencies of amino acid substitutions of the residues in these blocks are calculated to produce a numerical table, or block substitution matrix.

Instead of using the extrapolation function, the BLOSUM matrices are actual percentage identity values of sequences selected for construction of the matrices. For example, BLOSUM62 indicates that the sequences selected for constructing the matrix share an average identity value of 62%. Other BLOSUM matrices based on sequence groups of various identity levels have also been constructed. In the reversing order as the PAM numbering system, the lower the BLOSUM number, the more divergent sequences they represent.

The BLOSUM score for a particular residue pair is derived from the log ratio of observed residue substitution frequency versus the expected probability of a particular residue. The log odds is taken to the base of 2 instead of 10 as in the PAM matrices. The resulting value is rounded to the nearest integer and entered into the substitution matrix. As in the PAM matrices, positive and negative values correspond to substitutions that occur more or less frequently than expected among evolutionarily

C	9																		
S	-1	4																	
T	-1	1	5																
P	-3	-1	-1	7															
A	0	1	0	-1	4														
G	-3	0	-2	-2	0	6													
N	-3	1	0	-2	-2	0	6												
D	-3	0	-1	-1	-2	-1	1	6											
E	-4	0	-1	-1	-1	-2	0	2	5										
Q	-3	0	-1	-1	-1	-2	0	0	2	5									
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8								
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5							
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5						
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5					
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4				
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4			
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4		
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6	
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11
	C	**S**	**T**	**P**	**A**	**G**	**N**	**D**	**E**	**Q**	**H**	**R**	**K**	**M**	**I**	**L**	**V**	**F**	**Y**	**W**

Figure 3.6: BLOSUM62 amino acid substitution matrix.

conserved replacements. The values of the BLOSUM62 matrix are shown in Figure 3.6.

Comparison between PAM and BLOSUM

There are a number of differences between PAM and BLOSUM. The principal difference is that the PAM matrices, except PAM1, are derived from an evolutionary model whereas the BLOSUM matrices consist of entirely direct observations. Thus, the BLOSUM matrices may have less evolutionary meaning than the PAM matrices. This is why the PAM matrices are used most often for reconstructing phylogenetic trees. However, because of the mathematical extrapolation procedure used, the PAM values may be less realistic for divergent sequences. The BLOSUM matrices are entirely derived from local sequence alignments of conserved sequence blocks, whereas the PAM1 matrix is based on the global alignment of full-length sequences composed of both conserved and variable regions. This is why the BLOSUM matrices may be more advantageous in searching databases and finding conserved domains in proteins.

Several empirical tests have shown that the BLOSUM matrices outperform the PAM matrices in terms of accuracy of local alignment. This could be largely because the BLOSUM matrices are derived from a much larger and more representative dataset than the one used to derive the PAM matrices. This renders the values for the BLOSUM matrices more reliable. To compensate for the deficiencies in the PAM system, newer matrices using the same approach have been devised based on much larger data sets. These include the Gonnet matrices and the Jones–Taylor–Thornton matrices, which

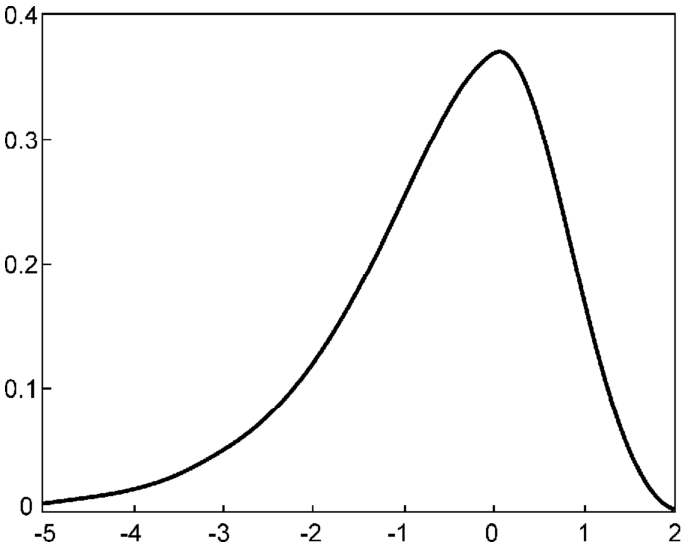


Figure 3.7: Gumble extreme value distribution for alignment scores. The distribution can be expressed as $P = 1 - e^{-Kmne^{-\lambda x}}$, where m and n are the sequence lengths, λ is a scaling factor for the scoring matrix used, and K is a constant that depends on the scoring matrix and gap penalty combination that is used. The x-axis of the curve indicates the standard deviation of the distribution; the y-axis indicates the alignment scores in arbitrary units.

have been shown to have equivalent performance to BLOSUM in regular alignment, but are particularly robust in phylogenetic tree construction.

STATISTICAL SIGNIFICANCE OF SEQUENCE ALIGNMENT

When given a sequence alignment showing a certain degree of similarity, it is often important to ask whether the observed sequence alignment can occur by random chance or the alignment is indeed statistically sound. The truly statistically significant sequence alignment will be able to provide evidence of homology between the sequences involved.

Solving this problem requires a statistical test of the alignment scores of two unrelated sequences of the same length. By calculating alignment scores of a large number of unrelated sequence pairs, a distribution model of the randomized sequence scores can be derived. From the distribution, a statistical test can be performed based on the number of standard deviations from the average score. Many studies have demonstrated that the distribution of similarity scores assumes a peculiar shape that resembles a highly skewed normal distribution with a long tail on one side (Fig. 3.7). The distribution matches the "Gumble extreme value distribution" for which a mathematical expression is available. This means that, given a sequence similarity value, by using the mathematical formula for the extreme distribution, the statistical significance can be accurately estimated.

The statistical test for the relatedness of two sequences can be performed using the following procedure. An optimal alignment between two given sequences is first obtained. Unrelated sequences of the same length are then generated through a randomization process in which one of the two sequences is randomly shuffled. A new

alignment score is then computed for the shuffled sequence pair. More such scores are similarly obtained through repeated shuffling. The pool of alignment scores from the shuffled sequences is used to generate parameters for the extreme distribution. The original alignment score is then compared against the distribution of random alignments to determine whether the score is beyond random chance. If the score is located in the extreme margin of the distribution, that means that the alignment between the two sequences is unlikely due to random chance and is thus considered significant. A *P*-value is given to indicate the probability that the original alignment is due to random chance.

A *P*-value resulting from the test provides a much more reliable indicator of possible homologous relationships than using percent identity values. It is thus important to know how to interpret the *P*-values. It has been shown that if a *P*-value is smaller than 10^{-100}, it indicates an exact match between the two sequences. If the *P*-value is in the range of 10^{-50} to 10^{-100}, it is considered to be a nearly identical match. A *P*-value in the range of 10^{-5} to 10^{-50} is interpreted as sequences having clear homology. A *P*-value in the range of 10^{-1} to 10^{-5} indicates possible distant homologs. If *P* is larger than 10^{-1}, the two sequence may be randomly related. However, the caveat is that sometimes truly related protein sequences may lack the statistical significance at the sequence level owing to fast divergence rates. Their evolutionary relationships can nonetheless be revealed at the three-dimensional structural level (see Chapter 15).

These statistics were derived from ungapped local sequence alignments. It is not known whether the Gumble distribution applies equally well to gapped alignments. However, for all practical purposes, it is reasonable to assume that scores for gapped alignments essentially fit the same distribution. A frequently used software program for assessing statistical significance of a pairwise alignment is the PRSS program.

PRSS (Probability of Random Shuffles; www.ch.embnet.org/software/PRSS_form.html) is a web-based program that can be used to evaluate the statistical significance of DNA or protein sequence alignment. It first aligns two sequences using the Smith–Waterman algorithm and calculates the score. It then holds one sequence in its original form and randomizes the order of residues in the other sequence. The shuffled sequence is realigned with the unshuffled sequence. The resulting alignment score is recorded. This process is iterated many (normally 1,000) times to help generate data for fitting the Gumble distribution. The original alignment score is then compared against the overall score distribution to derive a *P*-value. The major feature of the program is that it allows partial shuffling. For example, shuffling can be restricted to residues within a local window of 25–40, whereas the residues outside the window remain unchanged.

SUMMARY

Pairwise sequence alignment is the fundamental component of many bioinformatics applications. It is extremely useful in structural, functional, and evolutionary analyses

of sequences. Pairwise sequence alignment provides inference for the relatedness of two sequences. Strongly similar sequences are often homologous. However, a distinction needs to be made between sequence homology and similarity. The former is inference drawn from sequence comparison, whereas the latter relates to actual observation after sequence alignment. For protein sequences, identity values from pairwise alignment are often used to infer homology, although this approach can be rather imprecise.

There are two sequence alignment strategies, local alignment and global alignment, and three types of algorithm that perform both local and global alignments. They are the dot matrix method, dynamic programming method, and word method. The dot matrix method is useful in visually identifying similar regions, but lacks the sophistication of the other two methods. Dynamic programming is an exhaustive and quantitative method to find optimal alignments. This method effectively works in three steps. It first produces a sequence versus sequence matrix. The second step is to accumulate scores in the matrix. The last step is to trace back through the matrix in reverse order to identify the highest scoring path. This scoring step involves the use of scoring matrices and gap penalties.

Scoring matrices describe the statistical probabilities of one residue being substituted by another. PAM and BLOSUM are the two most commonly used matrices for aligning protein sequences. The PAM matrices involve the use of evolutionary models and extrapolation of probability values from alignment of close homologs to more divergent ones. In contrast, the BLOSUM matrices are derived from actual alignment. The PAM and BLOSUM serial numbers also have opposite meanings. Matrices of high PAM numbers are used to align divergent sequences and low PAM numbers for aligning closely related sequences. In practice, if one is uncertain about which matrix to use, it is advisable to test several matrices and choose the one that gives the best alignment result. Statistical significance of pairwise sequence similarity can be tested using a randomization test where score distribution follows an extreme value distribution.

FURTHER READING

Batzoglou, S. 2005. The many faces of sequence alignment. *Brief. Bioinformatics* 6:6–22.

Brenner, S. E., Chothia, C., and Hubbard, T. J. 1998. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proc. Natl. Acad. Sci. U S A* 95:6073–8.

Chao, K.-M., Pearson, W. R., and Miller, W. 1992. Aligning two sequences within a specified diagonal band. *Comput. Appl. Biosci.* 8:481–7.

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

Huang, X. 1994. On global sequence alignment. *Comput. Appl. Biosci.* 10:227–35.

Pagni, M., and Jongeneel, V. 2001. Making sense of score statistics for sequence alignments. *Brief. Bioinformatics* 2:51–67.

Pearson, W. R. 1996. Effective protein sequence comparison. *Methods Enzymol.* 266:227–58.

Rost, B. 1999. Twilight zone of protein sequence alignments. *Protein Eng.* 12:85–94.
States, D. J., Gish, W., and Altschul, S. F. 1991. Improved sensitivity of nucleic acid database searches using application-specific scoring matrices. *Methods* 3:66–70.
Valdar, W. S. 2002. Scoring residue conservation. *Proteins*. 48:227–41.
Vingron, M., and Waterman, M. S. 1994. Sequence alignment and penalty scores. *J. Mol. Biol.* 235:1–12.

CHAPTER FOUR

Database Similarity Searching

A main application of pairwise alignment is retrieving biological sequences in databases based on similarity. This process involves submission of a query sequence and performing a pairwise comparison of the query sequence with all individual sequences in a database. Thus, database similarity searching is pairwise alignment on a large scale. This type of searching is one of the most effective ways to assign putative functions to newly determined sequences. However, the dynamic programming method described in Chapter 3 is slow and impractical to use in most cases. Special search methods are needed to speed up the computational process of sequence comparison. The theory and applications of the database searching methods are discussed in this chapter.

UNIQUE REQUIREMENTS OF DATABASE SEARCHING

There are unique requirements for implementing algorithms for sequence database searching. The first criterion is *sensitivity*, which refers to the ability to find as many correct hits as possible. It is measured by the extent of inclusion of correctly identified sequence members of the same family. These correct hits are considered "true positives" in the database searching exercise. The second criterion is *selectivity*, also called *specificity*, which refers to the ability to exclude incorrect hits. These incorrect hits are unrelated sequences mistakenly identified in database searching and are considered "false positives." The third criterion is *speed*, which is the time it takes to get results from database searches. Depending on the size of the database, speed sometimes can be a primary concern.

Ideally, one wants to have the greatest sensitivity, selectivity, and speed in database searches. However, satisfying all three requirements is difficult in reality. What generally happens is that an increase in sensitivity is associated with decrease in selectivity. A very inclusive search tends to include many false positives. Similarly, an improvement in speed often comes at the cost of lowered sensitivity and selectivity. A compromise between the three criteria often has to be made.

In database searching, as well as in many other areas in bioinformatics, are two fundamental types of algorithms. One is the *exhaustive type*, which uses a rigorous algorithm to find the best or exact solution for a particular problem by examining all mathematical combinations. Dynamic programming is an example of the exhaustive method and is computationally very intensive. Another is the *heuristic type*, which is a computational strategy to find an empirical or near optimal solution by using rules of

thumb. Essentially, this type of algorithms take shortcuts by reducing the search space according to some criteria. However, the shortcut strategy is not guaranteed to find the best or most accurate solution. It is often used because of the need for obtaining results within a realistic time frame without significantly sacrificing the accuracy of the computational output.

HEURISTIC DATABASE SEARCHING

Searching a large database using the dynamic programming methods, such as the Smith–Waterman algorithm, although accurate and reliable, is too slow and impractical when computational resources are limited. An estimate conducted nearly a decade ago had shown that querying a database of 300,000 sequences using a query sequence of 100 residues took 2–3 hours to complete with a regular computer system at the time. Thus, speed of searching became an important issue. To speed up the comparison, heuristic methods have to be used. The heuristic algorithms perform faster searches because they examine only a fraction of the possible alignments examined in regular dynamic programming.

Currently, there are two major heuristic algorithms for performing database searches: BLAST and FASTA. These methods are not guaranteed to find the optimal alignment or true homologs, but are 50–100 times faster than dynamic programming. The increased computational speed comes at a moderate expense of sensitivity and specificity of the search, which is easily tolerated by working molecular biologists. Both programs can provide a reasonably good indication of sequence similarity by identifying similar sequence segments.

Both BLAST and FASTA use a heuristic *word method* for fast pairwise sequence alignment. This is the third method of pairwise sequence alignment. It works by finding short stretches of identical or nearly identical letters in two sequences. These short strings of characters are called *words*, which are similar to the windows used in the dot matrix method (see Chapter 3). The basic assumption is that two related sequences must have at least one word in common. By first identifying word matches, a longer alignment can be obtained by extending similarity regions from the words. Once regions of high sequence similarity are found, adjacent high-scoring regions can be joined into a full alignment.

BASIC LOCAL ALIGNMENT SEARCH TOOL (BLAST)

The BLAST program was developed by Stephen Altschul of NCBI in 1990 and has since become one of the most popular programs for sequence analysis. BLAST uses heuristics to align a query sequence with all sequences in a database. The objective is to find high-scoring ungapped segments among related sequences. The existence of such segments above a given threshold indicates pairwise similarity beyond random chance, which helps to discriminate related sequences from unrelated sequences in a database.

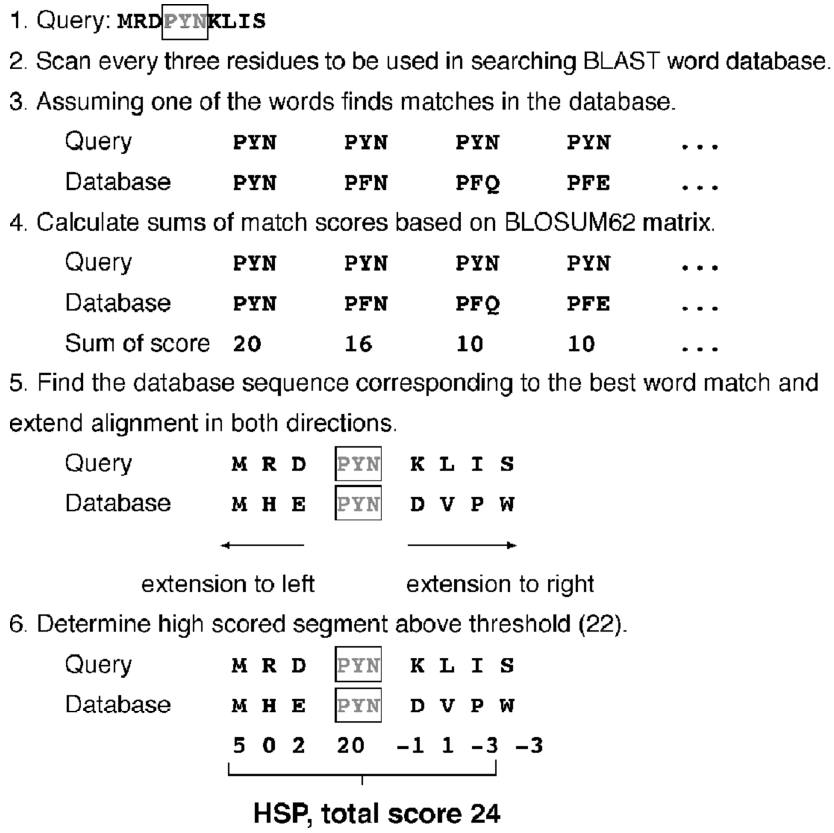


Figure 4.1: Illustration of the BLAST procedure using a hypothetical query sequence matching with a hypothetical database sequence. The alignment scoring is based on the BLOSUM62 matrix (see Chapter 3). The example of the word match is highlighted in the box.

BLAST performs sequence alignment through the following steps. The first step is to create a list of words from the query sequence. Each word is typically three residues for protein sequences and eleven residues for DNA sequences. The list includes every possible word extracted from the query sequence. This step is also called *seeding*. The second step is to search a sequence database for the occurrence of these words. This step is to identify database sequences containing the matching words. The matching of the words is scored by a given substitution matrix. A word is considered a match if it is above a threshold. The fourth step involves pairwise alignment by extending from the words in both directions while counting the alignment score using the same substitution matrix. The extension continues until the score of the alignment drops below a threshold due to mismatches (the drop threshold is twenty-two for proteins and twenty for DNA). The resulting contiguous aligned segment pair without gaps is called *high-scoring segment pair* (HSP; see working example in Fig. 4.1). In the original version of BLAST, the highest scored HSPs are presented as the final report. They are also called maximum scoring pairs.

A recent improvement in the implementation of BLAST is the ability to provide gapped alignment. In gapped BLAST, the highest scored segment is chosen to be extended in both directions using dynamic programming where gaps may be introduced. The extension continues if the alignment score is above a certain threshold; otherwise it is terminated. However, the overall score is allowed to drop below the

threshold only if it is temporary and rises again to attain above threshold values. Final trimming of terminal regions is needed before producing a report of the final alignment.

Variants

BLAST is a family of programs that includes BLASTN, BLASTP, BLASTX TBLASTN, and TBLASTX. BLASTN queries nucleotide sequences with a nucleotide sequence database. BLASTP uses protein sequences as queries to search against a protein sequence database. BLASTX uses nucleotide sequences as queries and translates them in all six reading frames to produce translated protein sequences, which are used to query a protein sequence database. TBLASTN queries protein sequences to a nucleotide sequence database with the sequences translated in all six reading frames. TBLASTX uses nucleotide sequences, which are translated in all six frames, to search against a nucleotide sequence database that has all the sequences translated in six frames. In addition, there is also a bl2seq program that performs local alignment of two user-provided input sequences. The graphical output includes horizontal bars and a diagonal in a two-dimensional diagram showing the overall extent of matching between the two sequences.

The BLAST web server (www.ncbi.nlm.nih.gov/BLAST/) has been designed in such a way as to simplify the task of program selection. The programs are organized based on the type of query sequences, protein sequences, nucleotide sequences, or nucleotide sequence to be translated. In addition, programs for special purposes are grouped separately; for example, bl2seq, immunoglobulin BLAST, and VecScreen, a program for removing contaminating vector sequences. The BLAST programs specially designed for searching individual genome databases are also listed in a separate category.

The choice of the type of sequences also influences the sensitivity of the search. Generally speaking, there is a clear advantage of using protein sequences in detecting homologs. This is because DNA sequences only comprise four nucleotides, whereas protein sequences contain twenty amino acids. This means that there is at least a five-fold increase in statistical complexity for protein sequences. More importantly, amino acid substitution matrices incorporate subtle differences in physicochemical properties between amino acids, meaning that protein sequences are far more informative and sensitive in detection of homologs. This is why searches using protein sequences can yield more significant matches than using DNA sequences. For that reason, if the input sequence is a protein-encoding DNA sequence, it is preferable to use BLASTX, which translates it in six open reading frames before sequence comparisons are carried out.

If one is looking for protein homologs encoded in newly sequenced genomes, one may use TBLASTN, which translates nucleotide database sequences in all six open reading frames. This may help to identify protein coding genes that have not yet been annotated. If a DNA sequence is to be used as the query, a protein-level comparison can be done with TBLASTX. However, both programs are very computationally intensive and the search process can be very slow.

Statistical Significance

The BLAST output provides a list of pairwise sequence matches ranked by statistical significance. The significance scores help to distinguish evolutionarily related sequences from unrelated ones. Generally, only hits above a certain threshold are displayed.

Deriving the statistical measure is slightly different from that for single pairwise sequence alignment; the larger the database, the more unrelated sequence alignments there are. This necessitates a new parameter that takes into account the total number of sequence alignments conducted, which is proportional to the size of the database. In BLAST searches, this statistical indicator is known as the *E*-value (expectation value), and it indicates the probability that the resulting alignments from a database search are caused by random chance. The *E*-value is related to the *P*-value used to assess significance of single pairwise alignment (see Chapter 3). BLAST compares a query sequence against all database sequences, and so the *E*-value is determined by the following formula:

$$E = m \times n \times P \qquad \text{(Eq. 4.1)}$$

where m is the total number of residues in a database, n is the number of residues in the query sequence, and P is the probability that an HSP alignment is a result of random chance. For example, aligning a query sequence of 100 residues to a database containing a total of 10^{12} residues results in a *P*-value for the ungapped HSP region in one of the database matches of 1×1^{-20}. The *E*-value, which is the product of the three values, is $100 \times 10^{12} \times 10^{-20}$, which equals 10^{-6}. It is expressed as $1e-6$ in BLAST output. This indicates that the probability of this database sequence match occurring due to random chance is 10^{-6}.

The *E*-value provides information about the likelihood that a given sequence match is purely by chance. The lower the *E*-value, the less likely the database match is a result of random chance and therefore the more significant the match is. Empirical interpretation of the *E*-value is as follows. If $E < 1e-50$ (or 1×10^{-50}), there should be an extremely high confidence that the database match is a result of homologous relationships. If E is between 0.01 and $1e-50$, the match can be considered a result of homology. If E is between 0.01 and 10, the match is considered not significant, but may hint at a tentative remote homology relationship. Additional evidence is needed to confirm the tentative relationship. If $E > 10$, the sequences under consideration are either unrelated or related by extremely distant relationships that fall below the limit of detection with the current method.

Because the *E*-value is proportionally affected by the database size, an obvious problem is that as the database grows, the *E*-value for a given sequence match also increases. Because the genuine evolutionary relationship between the two sequences remains constant, the decrease in credibility of the sequence match as the database grows means that one may "lose" previously detected homologs as the database enlarges. Thus, an alternative to *E*-value calculations is needed.

A bit score is another prominent statistical indicator used in addition to the *E*-value in a BLAST output. The *bit score* measures sequence similarity independent of query sequence length and database size and is normalized based on the raw pairwise alignment score. The bit score (S') is determined by the following formula:

$$S' = (\lambda \times S - \ln K)/\ln 2 \qquad \text{(Eq. 4.2)}$$

where λ is the Gumble distribution constant, S is the raw alignment score, and K is a constant associated with the scoring matrix used. Clearly, the bit score (S') is linearly related to the raw alignment score (S). Thus, the higher the bit score, the more highly significant the match is. The bit score provides a constant statistical indicator for searching different databases of different sizes or for searching the same database at different times as the database enlarges.

Low Complexity Regions

For both protein and DNA sequences, there may be regions that contain highly repetitive residues, such as short segments of repeats, or segments that are overrepresented by a small number of residues. These sequence regions are referred to as *low complexity regions* (LCRs). LCRs are rather prevalent in database sequences; estimates indicate that LCRs account for about 15% of the total protein sequences in public databases. These elements in query sequences can cause spurious database matches and lead to artificially high alignment scores with unrelated sequences.

To avoid the problem of high similarity scores owing to matching of LCRs that obscure the real similarities, it is important to filter out the problematic regions in both the query and database sequences to improve the signal-to-noise ratio, a process known as *masking*. There are two types of masking: hard and soft. *Hard masking* involves replacing LCR sequences with an ambiguity character such as *N* for nucleotide residues or *X* for amino acid residues. The ambiguity characters are then ignored by the BLAST program, preventing the use of such regions in alignments and thus avoiding false positives. However, the drawback is that matching scores with true homologs may be lowered because of shortened alignments. *Soft masking* involves converting the problematic sequences to lower case letters, which are ignored in constructing the word dictionary, but are used in word extension and optimization of alignments.

SEG is a program that is able to detect and mask repetitive elements before executing database searches. It identifies LCRs by comparing residue frequencies of a certain region with average residue frequencies in the database. If the residue frequencies of a sequence region of the query sequence are significantly higher than the database average, the region is declared an LCR. SEG has been integrated into the BLAST web-based program. An option box for this low complexity filter needs to be selected to mask LCRs (either hard or soft masking).

RepeatMasker (http://woody.embl-heidelberg.de/repeatmask/) is an independent masking program that detects repetitive elements by comparing the query

sequence with a built-in library of repetitive elements using the Smith–Waterman algorithm. If the alignment score for a sequence region is above a certain threshold, the region is declared an LCR. The corresponding residues are then masked with *N*'s or *X*'s.

BLAST Output Format

The BLAST output includes a graphical overview box, a matching list and a text description of the alignment (Fig. 4.2). The graphical overview box contains colored horizontal bars that allow quick identification of the number of database hits and the degrees of similarity of the hits. The color coding of the horizontal bars corresponds to the ranking of similarities of the sequence hits (red: most related; green and blue: moderately related; black: unrelated). The length of the bars represents the spans of sequence alignments relative to the query sequence. Each bar is hyperlinked to the actual pairwise alignment in the text portion of the report. Below the graphical box is a list of matching hits ranked by the *E*-values in ascending order. Each hit includes the accession number, title (usually partial) of the database record, bit score, and *E*-value.

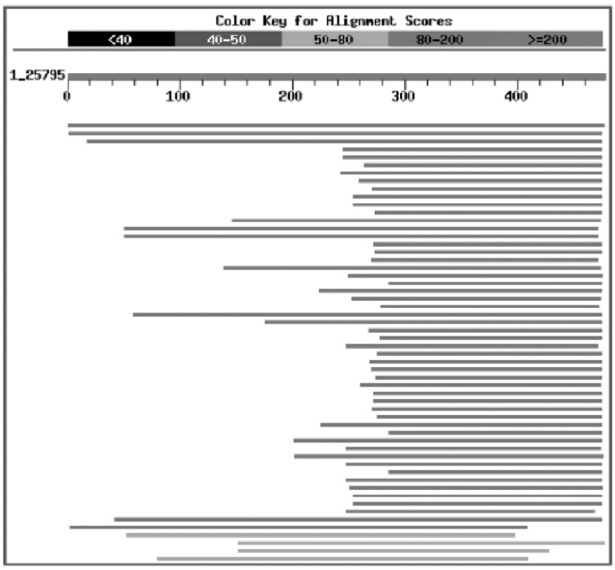
This list is followed by the text description, which may be divided into three sections: the header, statistics, and alignment. The header section contains the gene index number or the reference number of the database hit plus a one-line description of the database sequence. This is followed by the summary of the statistics of the search output, which includes the bit score, *E*-value, percentages of identity, similarity ("Positives"), and gaps. In the actual alignment section, the query sequence is on the top of the pair and the database sequence is at the bottom of the pair labeled as *Subject*. In between the two sequences, matching identical residues are written out at their corresponding positions, whereas nonidentical but similar residues are labeled with "+". Any residues identified as LCRs in the query sequence are masked with *X*s or *N*s so that no alignment is represented in those regions.

FASTA

FASTA (FAST ALL, www.ebi.ac.uk/fasta33/) was in fact the first database similarity search tool developed, preceding the development of BLAST. FASTA uses a "hashing" strategy to find matches for a short stretch of identical residues with a length of *k*. The string of residues is known as *ktuples* or *ktups*, which are equivalent to words in BLAST, but are normally shorter than the words. Typically, a ktup is composed of two residues for protein sequences and six residues for DNA sequences.

The first step in FASTA alignment is to identify ktups between two sequences by using the hashing strategy. This strategy works by constructing a lookup table that shows the position of each ktup for the two sequences under consideration. The positional difference for each word between the two sequences is obtained by subtracting the position of the first sequence from that of the second sequence and is expressed as the offset. The ktups that have the same offset values are then linked to reveal a

Graphical overview



Matching list

```
                                                                    Score    E
Sequences producing significant alignments:                         (bits) Value

gi|22958938|ref|ZP_00006599.1|  COG3920: Signal transduction...     896   0.0
gi|22968827|ref|ZP_00016409.1|  COG3920: Signal transduction...     390   e-107
gi|39933087|ref|NP_945363.1|  putative signal transduction h...     365   e-100
gi|17935877|ref|NP_532667.1|  two component sensor kinase [A...     175   2e-42
gi|15889280|ref|NP_354961.1|  AGR_C_3616p [Agrobacterium tum...     175   2e-42
gi|31322739|gb|AAP22926.1|  CheS3 [Rhodospirillum centenum]         158   2e-37
gi|16126793|ref|NP_421357.1|  sensor histidine kinase, putat...     157   5e-37
gi|16127400|ref|NP_421964.1|  sensor histidine kinase, putat...     155   1e-36
gi|15966187|ref|NP_386540.1|  HYPOTHETICAL PROTEIN [Sinorhiz...     155   2e-36
gi|16264804|ref|NP_437596.1|  putative two-component sensor ...     152   2e-35
gi|2808506|emb|CAA12536.1|  ExsG protein [Sinorhizobium meli...     151   2e-35
gi|13476692|ref|NP_108261.1|  two-component, sensor histidin...     149   9e-35
gi|16127278|ref|NP_421842.1|  sensor histidine kinase, putat...     149   1e-34
gi|17939110|ref|NP_535898.1|  two component sensor kinase [A...     147   4e-34
gi|13473179|ref|NP_104746.1|  hypothetical protein [Mesorhiz...     147   6e-34
gi|16119758|ref|NP_396464.1|  AGR_pAT_788p [Agrobacterium tu...     147   6e-34
gi|13488521|ref|NP_109528.1|  sensory transduction histidine...     146   1e-33
gi|16125089|ref|NP_419653.1|  sensor histidine kinase, putat...     145   1e-33
gi|22957499|ref|ZP_00005199.1|  COG3920: Signal transduction...     145   2e-33
```

Alignment output

header

statistics

alignment

```
>gi|22968827|ref|ZP_00016409.1|  COG3920: Signal transduction histidine kinase [Rhodospirillum
          rubrum]
         Length = 489

 Score =  377 bits (968), Expect = e-103
 Identities = 235/484 (48%), Positives = 306/484 (63%), Gaps = 14/484 (2%)

Query: 3   PAEIDELRRRLHEAEETLKAIRQGDVDALVVGASDDTDVYVIGGDPDICRSFLDMMEIGA 62
            P  + ELRRRL EAEETL AIR+G+VDALV+G     +V+ IGGD +  R+F++ M+ GA
Sbjct: 4   PVVLSELRRRLAEAEETLNAIREGEVDALVIGEGGVDEVFAIGGDTESYRTFMEAMDTGA 63

Query: 63  AALDNTGRVLYANAVLADLVGRPLPELEGMRL-----SELTGDPAXXXXXXXXXXXXXI 117
            AA+D  GRVLYAN+ L  L+  PLP L+G  L     +    +                I
Sbjct: 64  AAVDEDGRVLYANSALCRLIDHPLPTLQGKPLVSFFDARAAAEIGQMVGKTANQREKVEI 123

Query: 118 PLGVAGAER-QVMLSCGK-LRLGTVSGHAVTFTDFTEQLAAERSRQNEKAALAIIACANE 175
             L  A  +  QV L    K +RLG V GHAVTFTD TE++ +E + + E+ A AIIA ANE
Sbjct: 124 SLKDAATKMAQVFLVSAKPVRLGLVQGHAVTFTDLTERVRSETAERAERIAAAIIASANE 183

Query: 176 PVFVCDTLGLITHXXXXXXXXXXXXXRPLSEVMDLSVGDGTGLLTLGEIVAQATEGIP 235
             V VCD +G+ITH               +   + + L+  D   L++ G ++   A  G
Sbjct: 184 IVVVCDRVGMITHANSAASAIYDGDLIGKMPEDAIPLTFTDAPDLMSGGALIDLALNGQA 243

Query: 236 VQGIEAVAAEGTPF--YLISAAPLQVPGEAVSGCVITMVDLSQRKAAERHQQLLLRELDH 293
             QGIEA+A         YLISAAPLQV  + +SGCV+TMVDLSQRKAAE  Q LL+RELDH
Sbjct: 244 RQGIEAIATRAPKVKDYLISAAPLQVTEDQISGCVLTMVDLSQRKAAEHQQLLLMRELDH 303

Query: 294 RVKNTLALVMSISRRTMHSEETLEGYQKAFTARIQALAATHNLLADKSWSDISIRDVLVR 353
             RV+NTLALV+SIS RT+ +E+TL+G+ +AFT RI  LAATH+LLA + W+ +S+ D++
Sbjct: 304 RVRNTLALVLSISNRTLSNEDTLQGFHQAFTQRIHGLAATHSLLAKQGWTKLSLHDIVRA 363

Query: 354 ELAPYNEGFSQRILVEVPDVEIEPRSAIALGLVIHELATNATKYGSLSTPEGQ--VRVRG 411
             ELAPY E    R+ +E  +V + PR+AIALGL+ HELATNA KYG+LS   G   V VRG
Sbjct: 364 ELAPYVETDGTRLRLEGGEVALIPRAAIALGLIFHELATNAVKYGALSREGGHVLVAVRG 423

Query: 412 LPGADEPADVVCLEWLERGGPPVSEPTRSGFGQTVIRHAFAYAEGGAEVSFEPDGVRCR 471
              P AD  A  V  +W+E GGP VS P R GFG TVI H+ AY+  GG ++SF P+GV C
Sbjct: 424 -PTADGAAMRV--DWVESGGPMVSPPQRKGFGHTVISHSLAYSSKGGTDLSFPPEGVICA 480

Query: 472 VSVP 475
             + +P
Sbjct: 481 LRIP 484
```

Figure 4.2: An example of a BLAST output showing three portions: the graphical overview box, the list of matching hits, and the text portion containing header, statistics, and the actual alignment.

1. Given two amino acid sequences for comparision:

sequence 1 **AMPSDGL**
sequence 2 **GPSDNAT**

2. Construct a hashing table:

amino acid	sequence position		offset
	seq 1	seq 2	
A	1	6	-5
D	5	4	1
G	6	1	5
L	7	-	-
M	2	-	-
N	-	5	-
P	3	2	1
S	4	3	1
T	-	7	-

3. Identify residues with the same offset values (highlighted in grey).

4. Find the matching word of three residues in the order of 3, 4 and 5 in one sequence and 2, 3,and 4 in the other.

5. This allows establishment of alignment between the two sequences.

sequence 1 **AMPSDGL-**
sequence 2 **-GPSDNAT**

Figure 4.3: The procedure of ktup identification using the hashing strategy by FASTA. Identical offset values between residues of the two sequences allow the formation of ktups.

contiguous identical sequence region that corresponds to a stretch of diagonal in a two-dimensional matrix (Fig. 4.3).

The second step is to narrow down the high similarity regions between the two sequences. Normally, many diagonals between the two sequences can be identified in the hashing step. The top ten regions with the highest density of diagonals are identified as high similarity regions. The diagonals in these regions are scored using a substitution matrix. Neighboring high-scoring segments along the same diagonal are selected and joined to form a single alignment. This step allows introducing gaps between the diagonals while applying gap penalties. The score of the gapped alignment is calculated again. In step 3, the gapped alignment is refined further using the Smith–Waterman algorithm to produce a final alignment (Fig. 4.4). The last step is to perform a statistical evaluation of the final alignment as in BLAST, which produces the *E*-value.

Similar to BLAST, FASTA has a number of subprograms. The web-based FASTA program offered by the European Bioinformatics Institute (www.ebi.ac.uk/) allows the use of either DNA or protein sequences as the query to search against a protein database or nucleotide database. Some available variants of the program are FASTX, which translates a DNA sequence and uses the translated protein sequence to query a protein database, and TFASTX, which compares a protein query sequence to a translated DNA database.

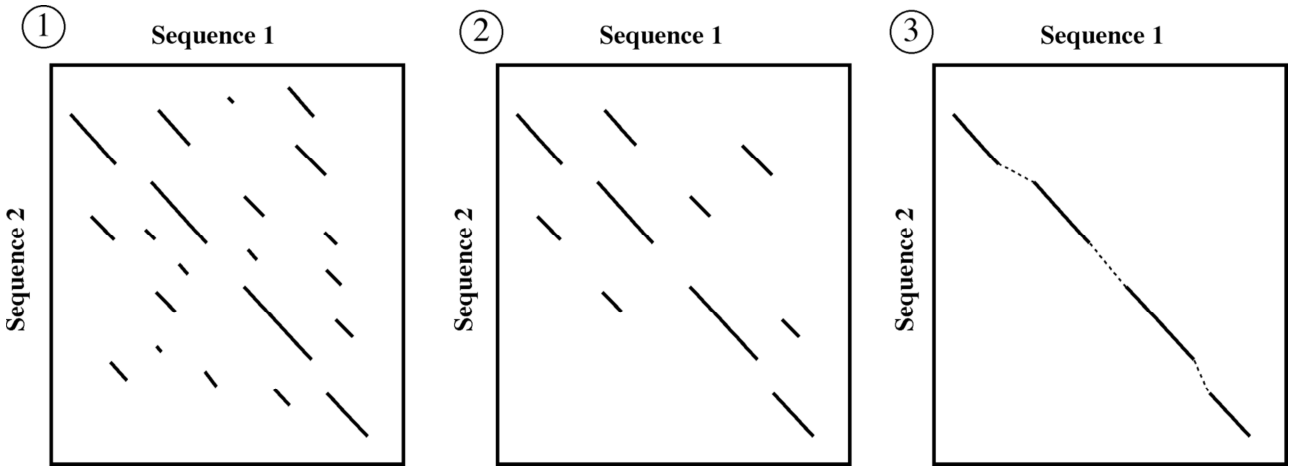


Figure 4.4: Steps of the FASTA alignment procedure. In step 1 (*left*), all possible ungapped alignments are found between two sequences with the hashing method. In step 2 (*middle*), the alignments are scored according to a particular scoring matrix. Only the ten best alignments are selected. In step 3 (*right*), the alignments in the same diagonal are selected and joined to form a single gapped alignment, which is optimized using the dynamic programming approach.

Statistical Significance

FASTA also uses *E*-values and bit scores. Estimation of the two parameters in FASTA is essentially the same as in BLAST. However, the FASTA output provides one more statistical parameter, the *Z*-score. This describes the number of standard deviations from the mean score for the database search. Because most of the alignments with the query sequence are with unrelated sequences, the higher the *Z*-score for a reported match, the further away from the mean of the score distribution, hence, the more significant the match. For a *Z*-score > 15, the match can be considered extremely significant, with certainty of a homologous relationship. If *Z* is in the range of 5 to 15, the sequence pair can be described as highly probable homologs. If $Z < 5$, their relationships is described as less certain.

COMPARISON OF FASTA AND BLAST

BLAST and FASTA have been shown to perform almost equally well in regular database searching. However, there are some notable differences between the two approaches. The major difference is in the seeding step; BLAST uses a substitution matrix to find matching words, whereas FASTA identifies identical matching words using the hashing procedure. By default, FASTA scans smaller window sizes. Thus, it gives more sensitive results than BLAST, with a better coverage rate for homologs. However, it is usually slower than BLAST. The use of low-complexity masking in the BLAST procedure means that it may have higher specificity than FASTA because potential false positives are reduced. BLAST sometimes gives multiple best-scoring alignments from the same sequence; FASTA returns only one final alignment.

DATABASE SEARCHING WITH THE SMITH–WATERMAN METHOD

As mentioned, the rigorous dynamic programming method is normally not used for database searching, because it is slow and computationally expensive. Heuristics such as BLAST and FASTA are developed for faster speed. However, the heuristic methods are limited in sensitivity and are not guaranteed to find the optimal alignment. They often fail to find alignment for distantly related sequences. It has been estimated that for some families of protein sequences, BLAST can miss 30% of truly significant hits. Recent developments in computation technologies, such as parallel processing supercomputers, have made dynamic programming a feasible approach to database searches to fill the performance gap.

For this purpose, the computer codes for the Needleman–Wunsch and Smith–Waterman algorithms have to be modified to run in a parallel processing environment so that searches can be completed within reasonable time periods. Currently, the search speed is still slower than the popular heuristic programs. Therefore, the method is not intended for routine use. Nevertheless, the availability of dynamic programming allows the maximum sensitivity for finding homologs at the sequence level. Empirical tests have indeed shown that the exhaustive method produces superior results over the heuristic methods. Below is a list of dynamic programming-based web servers for sequence database searches.

ScanPS (Scan Protein Sequence, www.ebi.ac.uk/scanps/) is a web-based program that implements a modified version of the Smith–Waterman algorithm optimized for parallel processing. The major feature is that the program allows iterative searching similar to PSI-BLAST (see Chapter 5), which builds profiles from one round of search results and uses them for the second round of database searching. Full dynamic programming is used in each cycle for added sensitivity.

ParAlign (www.paralign.org/) is a web-based server that uses parallel processors to perform exhaustive sequence comparisons using either a parallelized version of the Smith–Waterman algorithm or a heuristic program for further speed gains. The heuristic subprogram first finds exact ungapped alignments and uses them as anchors for extension into gapped alignments by combining the scores of several diagonals in the alignment matrix. The search speed of ParAlign approaches to that of BLAST, but with higher sensitivity.

SUMMARY

Database similarity searching is an essential first step in the functional characterization of novel gene or protein sequences. The major issues in database searching are sensitivity, selectivity, and speed. Speed is a particular concern in searching large databases. Thus, heuristic methods have been developed for efficient database similarity searches. The major heuristic database searching algorithms are BLAST and FASTA. They both use a word method for pairwise alignment. BLAST looks for HSPs

in a database. FASTA uses a hashing scheme to identify words. The major statistical measures for significance of database matches are *E*-values and bit scores. A caveat for sequence database searching is to filter the LCRs using masking programs. Another caveat is to use protein sequences as the query in database searching, because they produce much more sensitive matches. In addition, it is important to keep in mind that both BLAST and FASTA are heuristic programs and are not guaranteed to find all the homologous sequences. For significant matches automatically generated by these programs, it is recommended to follow up the leads by checking the alignment using more rigorous and independent alignment programs. Advances in computational technology have also made it possible to use full dynamic programming in database searching with increased sensitivity and selectivity.

FURTHER READING

Altschul, S. F., Boguski, M. S., Gish, W., and Wootton, J. C. 1994. Issues in searching molecular sequences databases. *Nat. Genet*. 6:119–29.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res*. 25:3389–402.

Chen, Z. 2003. Assessing sequence comparison methods with the average precision criterion. *Bioinformatics* 19:2456–60.

Karlin, S., and Altschul, S. F. 1993. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc. Natl. Acad. Sci. U S A* 90:5873–7.

Mullan, L. J., and Williams, G. W. 2002. BLAST and go? *Brief. Bioinform*. 3:200–2.

Sansom, C. 2000. Database searching with DNA and protein sequences: An introduction. *Brief. Bioinform*. 1:22–32.

Spang, R., and Vingron, M. 1998. Statistics of large-scale sequence searching. *Bioinformatics* 14:279–84.

CHAPTER FIVE

Multiple Sequence Alignment

A natural extension of pairwise alignment is multiple sequence alignment, which is to align multiple related sequences to achieve optimal matching of the sequences. Related sequences are identified through the database similarity searching described in Chapter 4. As the process generates multiple matching sequence pairs, it is often necessary to convert the numerous pairwise alignments into a single alignment, which arranges sequences in such a way that evolutionarily equivalent positions across all sequences are matched.

There is a unique advantage of multiple sequence alignment because it reveals more biological information than many pairwise alignments can. For example, it allows the identification of conserved sequence patterns and motifs in the whole sequence family, which are not obvious to detect by comparing only two sequences. Many conserved and functionally critical amino acid residues can be identified in a protein multiple alignment. Multiple sequence alignment is also an essential prerequisite to carrying out phylogenetic analysis of sequence families and prediction of protein secondary and tertiary structures. Multiple sequence alignment also has applications in designing degenerate polymerase chain reaction (PCR) primers based on multiple related sequences.

It is theoretically possible to use dynamic programming to align any number of sequences as for pairwise alignment. However, the amount of computing time and memory it requires increases exponentially as the number of sequences increases. As a consequence, full dynamic programming cannot be applied for datasets of more than ten sequences. In practice, heuristic approaches are most often used. In this chapter, methodologies and applications of multiple sequence alignment are discussed.

SCORING FUNCTION

Multiple sequence alignment is to arrange sequences in such a way that a maximum number of residues from each sequence are matched up according to a particular scoring function. The scoring function for multiple sequence alignment is based on the concept of sum of pairs (SP). As the name suggests, it is the sum of the scores of all possible pairs of sequences in a multiple alignment based on a particular scoring matrix. In calculating the SP scores, each column is scored by summing the scores for all possible pairwise matches, mismatches and gap costs. The score of the entire

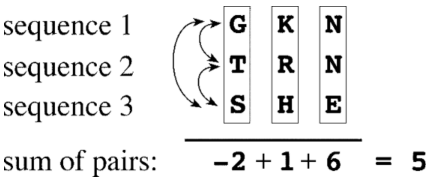


Figure 5.1: Given a multiple alignment of three sequences, the sum of scores is calculated as the sum of the similarity scores of every pair of sequences at each position. The scoring is based on the BLOSUM62 matrix (see Chapter 3). The total score for the alignment is 5, which means that the alignment is $2^5 = 32$ times more likely to occur among homologous sequences than by random chance.

alignment is the sum of all of the column scores (Fig. 5.1). The purpose of most multiple sequence alignment algorithms is to achieve maximum SP scores.

EXHAUSTIVE ALGORITHMS

As mentioned, there are exhaustive and heuristic approaches used in multiple sequence alignment. The exhaustive alignment method involves examining all possible aligned positions simultaneously. Similar to dynamic programming in pairwise alignment, which involves the use of a two-dimensional matrix to search for an optimal alignment, to use dynamic programming for multiple sequence alignment, extra dimensions are needed to take all possible ways of sequence matching into consideration. This means to establish a multidimensional search matrix. For instance, for three sequences, a three-dimensional matrix is required to account for all possible alignment scores. Back-tracking is applied through the three-dimensional matrix to find the highest scored path that represents the optimal alignment. For aligning N sequences, an N-dimensional matrix is needed to be filled with alignment scores. As the amount of computational time and memory space required increases exponentially with the number of sequences, it makes the method computationally prohibitive to use for a large data set. For this reason, full dynamic programming is limited to small datasets of less than ten short sequences. For the same reason, few multiple alignment programs employing this "brute force" approach are publicly available. A program called DCA, which uses some exhaustive components, is described below.

DCA (Divide-and-Conquer Alignment, http://bibiserv.techfak.uni-bielefeld.de/dca/) is a web-based program that is in fact semiexhaustive because certain steps of computation are reduced to heuristics. It works by breaking each of the sequences into two smaller sections. The breaking points are determined based on regional similarity of the sequences. If the sections are not short enough, further divisions are carried out. When the lengths of the sequences reach a predefined threshold, dynamic programming is applied for aligning each set of subsequences. The resulting short alignments are joined together head to tail to yield a multiple alignment of the entire length of all sequences. This algorithm provides an option of using a more heuristic procedure (fastDCA) to choose optimal cutting points so it can more rapidly handle a greater number of sequences. It performs global alignment and requires the input sequences to be of similar lengths and domain structures. Despite the use of heuristics, the program is still extremely computationally intensive and can handle only datasets of a very limited number of sequences.

HEURISTIC ALGORITHMS

Because the use of dynamic programming is not feasible for routine multiple sequence alignment, faster and heuristic algorithms have been developed. The heuristic algorithms fall into three categories: progressive alignment type, iterative alignment type, and block-based alignment type. Each type of algorithm is described in turn.

Progressive Alignment Method

Progressive alignment depends on the stepwise assembly of multiple alignment and is heuristic in nature. It speeds up the alignment of multiple sequences through a multistep process. It first conducts pairwise alignments for each possible pair of sequences using the Needleman–Wunsch global alignment method and records these similarity scores from the pairwise comparisons. The scores can either be percent identity or similarity scores based on a particular substitution matrix. Both scores correlate with the evolutionary distances between sequences. The scores are then converted into evolutionary distances to generate a distance matrix for all the sequences involved. A simple phylogenetic analysis is then performed based on the distance matrix to group sequences based on pairwise distance scores. As a result, a phylogenetic tree is generated using the neighbor-joining method (see Chapter 11). The tree reflects evolutionary proximity among all the sequences.

It needs to be emphasized that the resulting tree is an approximate tree and does not have the rigor of a formally constructed phylogenetic tree (see Chapter 11). Nonetheless, the tree can be used as a guide for directing realignment of the sequences. For that reason, it is often referred to as a *guide tree*. According to the guide tree, the two most closely related sequences are first re-aligned using the Needleman–Wunsch algorithm. To align additional sequences, the two already aligned sequences are converted to a consensus sequence with gap positions fixed. The consensus is then treated as a single sequence in the subsequent step.

In the next step, the next closest sequence based on the guide tree is aligned with the consensus sequence using dynamic programming. More distant sequences or sequence profiles are subsequently added one at a time in accordance with their relative positions on the guide tree. After realignment with a new sequence using dynamic programming, a new consensus is derived, which is then used for the next round of alignment. The process is repeated until all the sequences are aligned (Fig. 5.2).

Probably the most well-known progressive alignment program is Clustal. Some of its important features are introduced next.

Clustal (www.ebi.ac.uk/clustalw/) is a progressive multiple alignment program available either as a stand-alone or on-line program. The stand-alone program, which runs on UNIX and Macintosh, has two variants, ClustalW and ClustalX. The W version provides a simple text-based interface and the X version provides a more user-friendly graphical interface.

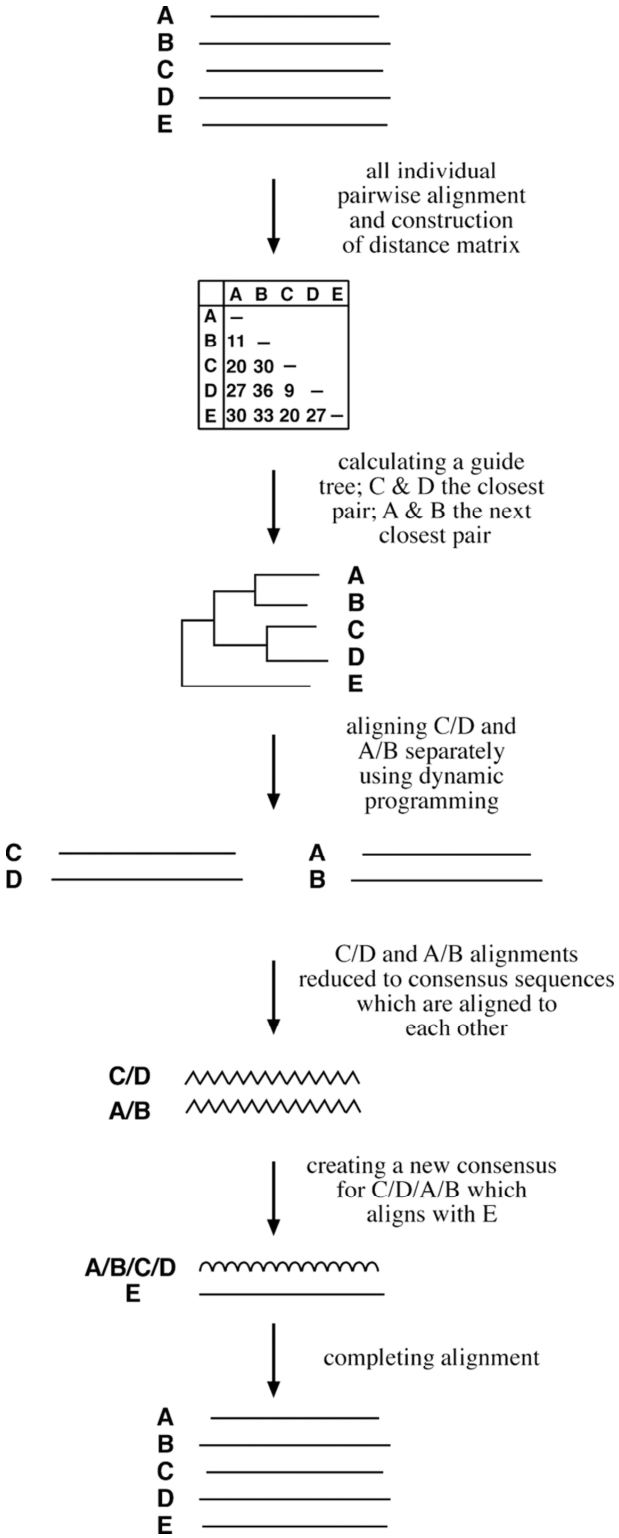


Figure 5.2: Schematic of a typical progressive alignment procedure (e.g., Clustal). Angled wavy lines represent consensus sequences for sequence pairs A/B and C/D. Curved wavy lines represent a consensus for A/B/C/D.

One of the most important features of this program is the flexibility of using substitution matrices. Clustal does not rely on a single substitution matrix. Instead, it applies different scoring matrices when aligning sequences, depending on degrees of similarity. The choice of a matrix depends on the evolutionary distances measured from the guide tree. For example, for closely related sequences that are aligned in the initial steps, Clustal automatically uses the BLOSUM62 or PAM120 matrix. When more divergent sequences are aligned in later steps of the progressive alignment, the BLOSUM45 or PAM250 matrices may be used instead.

Another feature of Clustal is the use of adjustable gap penalties that allow more insertions and deletions in regions that are outside the conserved domains, but fewer in conserved regions. For example, a gap near a series of hydrophobic residues carries more penalties than the one next to a series of hydrophilic or glycine residues, which are common in loop regions. In addition, gaps that are too close to one another can be penalized more than gaps occurring in isolated loci.

The program also applies a weighting scheme to increase the reliability of aligning divergent sequences (sequences with less than 25% identity). This is done by down-weighting redundant and closely related groups of sequences in the alignment by a certain factor. This scheme is useful in preventing similar sequences from dominating the alignment. The weight factor for each sequence is determined by its branch length on the guide tree. The branch lengths are normalized by how many times sequences share a basal branch from the root of the tree. The obtained value for each sequence is subsequently used to multiply the raw alignment scores of residues from that sequence so to achieve the goal of decreasing the matching scores of frequent characters in a multiple alignment and thereby increasing the ones of infrequent characters.

Drawbacks and Solutions

The progressive alignment method is not suitable for comparing sequences of different lengths because it is a global alignment–based method. As a result of the use of affine gap penalties (see Chapter 3), long gaps are not allowed, and, in some cases, this may limit the accuracy of the method. The final alignment result is also influenced by the order of sequence addition. Another major limitation is the "greedy" nature of the algorithm: it depends on initial pairwise alignment. Once gaps introduced in the early steps of alignment, they are fixed. Any errors made in these steps cannot be corrected. This problem of "once an error, always an error" can propagate throughout the entire alignment. In other words, the final alignment could be far from optimal. The problem can be more glaring when dealing with divergent sequences. To alleviate some of the limitations, a new generation of algorithms have been developed, which specifically target some of the problems of the Clustal program.

T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation; www.ch.embnet.org/software/TCoffee.html) performs progressive sequence alignments as in Clustal. The main difference is that, in processing a query, T-Coffee performs both global and local pairwise alignment for all possible pairs involved. The global pairwise alignment is performed using the Clustal program. The local pairwise

alignment is generated by the Lalign program, from which the top ten scored alignments are selected. The collection of local and global sequence alignments are pooled to form a library. The consistency of the alignments is evaluated. For every pair of residues in a pair of sequences, a consistency score is calculated for both global and local alignments. Each pairwise alignment is further aligned with a possible third sequence. The result is used to refine the original pairwise alignment based on a consistency criterion in a process known as *library extension*. Based on the refined pairwise alignments, a distance matrix is built to derive a guide tree, which is then used to direct a full multiple alignment using the progressive approach.

Because an optimal initial alignment is chosen from many alternative alignments, T-Coffee avoids the problem of getting stuck in the suboptimal alignment regions, which minimizes errors in the early stages of alignment assembly. Benchmark assessment has shown that T-Coffee indeed outperforms Clustal when aligning moderately divergent sequences. However, it is also slower than Clustal because of the extra computing time necessary for the calculation of consistency scores. T-Coffee provides a graphical output of the alignment results, with colored boxes to display degree of agreement in the alignment library for various sequence regions.

DbClustal (http://igbmc.u-strasbg.fr:8080/DbClustal/dbclustal.html) is a Clustal-based database search algorithm for protein sequences that combines local and global alignment features. It first performs a BLASTP search for a query sequence. The resulting sequence alignment pairs above a certain threshold are analyzed to obtain *anchor points*, which are common conserved regions, by using a program called Ballast. A global alignment is subsequently generated by Clustal, which is weighted toward the anchor points. Since the anchor points are derived from local alignments, this strategy minimizes errors caused by the global alignment. The resulting multiple alignment is further evaluated by NorMD, which removes unrelated or badly aligned sequences from the multiple alignment. Thus, the final alignment should be more accurate than using Clustal alone. It also allows the incorporation of very long gaps for insertions and terminal extensions.

Poa (Partial order alignments, www.bioinformatics.ucla.edu/poa/) is a progressive alignment program that does not rely on guide trees. Instead, the multiple alignment is assembled by adding sequences in the order they are given. Instead of using regular sequence consensus, a partial order graph is used to represent a growing multiple alignment, in which identical residues in a column are condensed to a node resembling a knot on a rope and divergent residues are allowed to remain as such, allowing the rope to "bubble" (Fig. 5.3). The graph profile preserves all the information from the original alignment. Each time a new sequence is added, it is aligned with every sequence within the partial order graph individually using the Smith–Waterman algorithm. This allows the formation of a modified graph model, which is then used for the next cycle of pairwise alignment. By building such a graph profile, the algorithm maintains the information of the original sequences and eliminates the problem of error fixation as in the Clustal alignment. Poa is local alignment-based and has been shown to produce more accurate alignments than Clustal. Another advantage of this

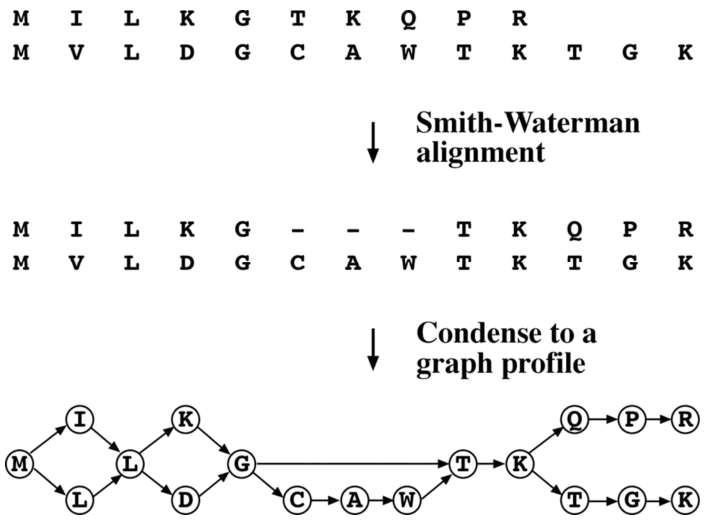


Figure 5.3: Conversion of a sequence alignment into a graphical profile in the Poa algorithm. Identical residues in the alignment are condensed as nodes in the partial order graph.

algorithm is its speed. It is reported to be able to align 5,000 sequences in 4 hours using a regular PC workstation. It is available both as an online program and as a stand-alone UNIX program.

PRALINE (http://ibivu.cs.vu.nl/programs/pralinewww/) is a web-based progressive alignment program. It first performs preprocessing of the input sequences by building profiles for each sequence. Profiles (see Chapter 6) can be interpreted as probability description of a multiple alignment. By default, the profiles are automatically generated using PSI-BLAST database searching (see Chapter 6). Each preprocessed profile is then used for multiple alignment using the progressive approach. However, this method does not use a guide tree in the successive enlargement of the alignment, but rather considers the closest neighbor to be joined to a larger alignment by comparing the profile scores. Because the profiles already incorporate information of distant relatives of each input sequence, this approach allows more accurate alignment of distantly related sequences in the original dataset. In addition, the program also has the feature to incorporate protein secondary structure information which is derived from state-of-the-art secondary structure prediction programs, such as PROF or SSPRO (see Chapter 14). The secondary structure information is used to modify the profile scores to help constrain sequence matching to the structured regions. PRALINE is perhaps the most sophisticated and accurate alignment program available. Because of the high complexity of the algorithm, its obvious drawback is the extremely slow computation.

Iterative Alignment

The iterative approach is based on the idea that an optimal solution can be found by repeatedly modifying existing suboptimal solutions. The procedure starts by producing a low-quality alignment and gradually improves it by iterative realignment through well-defined procedures until no more improvements in the alignment scores

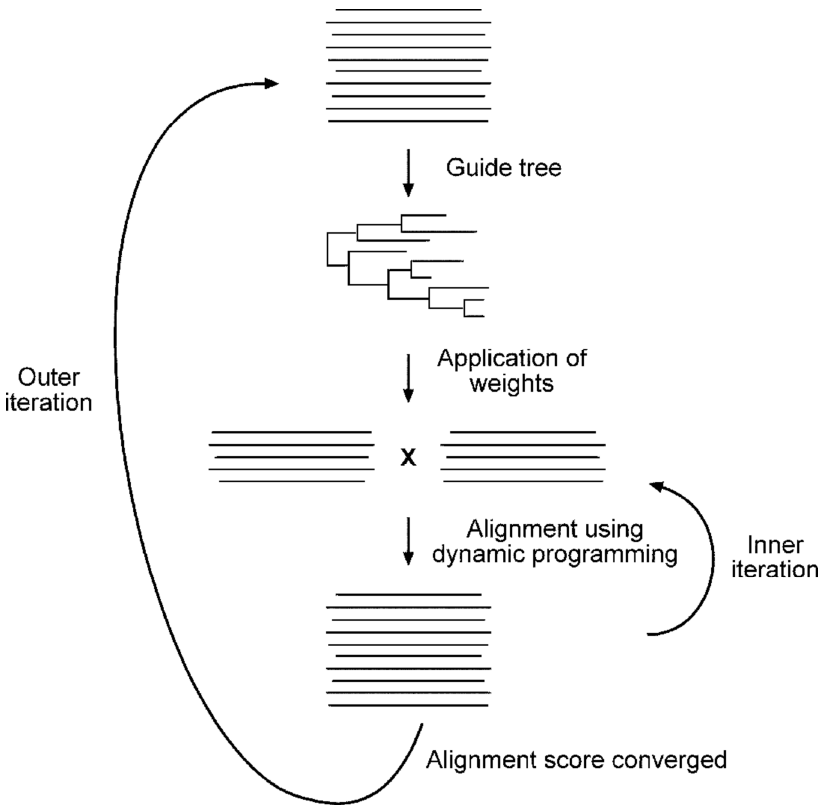


Figure 5.4: Schematic of iterative alignment procedure for PRRN, which involves two sets of iterations.

can be achieved. Because the order of the sequences used for alignment is different in each iteration, this method may alleviate the "greedy" problem of the progressive strategy. However, this method is also heuristic in nature and does not have guarantees for finding the optimal alignment. An example of iterative alignment is given below.

PRRN (http://prrn.ims.u-tokyo.ac.jp/) is a web-based program that uses a double-nested iterative strategy for multiple alignment. It performs multiple alignment through two sets of iterations: inner iteration and outer iteration. In the *outer iteration*, an initial random alignment is generated that is used to derive a UPGMA tree (see Chapter 11). Weights are subsequently applied to optimize the alignment. In the *inner iteration*, the sequences are randomly divided into two groups. Randomized alignment is used for each group in the initial cycle, after which the alignment positions in each group are fixed. The two groups, each treated as a single sequence, are then aligned to each other using global dynamic programming. The process is repeated through many cycles until the total SP score no longer increases. At this point, the resulting alignment is used to construct a new UPGMA tree. New weights are applied to optimize alignment scores. The newly optimized alignment is subject to further realignment in the inner iteration. This process is repeated over many cycles until there is no further improvement in the overall alignment scores (Fig. 5.4).

Block-Based Alignment

The progressive and iterative alignment strategies are largely global alignment based and may therefore fail to recognize conserved domains and motifs (see Chapter 7) among highly divergent sequences of varying lengths. For such divergent sequences that share only regional similarities, a local alignment based approach has to be used. The strategy identifies a block of ungapped alignment shared by all the sequences, hence, the block-based local alignment strategy. Two block-based alignment programs are introduced below.

DIALIGN2 (http://bioweb.pasteur.fr/seqanal/interfaces/dialign2.html) is a web-based program designed to detect local similarities. It does not apply gap penalties and thus is not sensitive to long gaps. The method breaks each of the sequences down to smaller segments and performs all possible pairwise alignments between the segments. High-scoring segments, called *blocks*, among different sequences are then compiled in a progressive manner to assemble a full multiple alignment. It places emphasis on block-to-block comparison rather than residue-to-residue comparison. The sequence regions between the blocks are left unaligned. The program has been shown to be especially suitable for aligning divergent sequences with only local similarity.

Match-Box (www.sciences.fundp.ac.be/biologie/bms/matchbox_submit.shtml) is a web-based server that also aims to identify conserved blocks (or boxes) among sequences. The program compares segments of every nine residues of all possible pairwise alignments. If the similarity of particular segments is above a certain threshold across all sequences, they are used as an anchor to assemble multiple alignments; residues between blocks are unaligned. The server requires the user to submit a set of sequences in the FASTA format and the results are returned by e-mail.

PRACTICAL ISSUES

Protein-Coding DNA Sequences

As mentioned in the Chapter 4, alignment at the protein level is more sensitive than at the DNA level. Sequence alignment directly at the DNA level can often result in frameshift errors because in DNA alignment gaps are introduced irrespective of codon boundaries. Therefore, in the process of achieving maximum sequence similarity at the DNA level, mismatches of genetic codons occur that violate the accepted evolutionary scenario that insertions or deletions occur in units of codons. The resulting alignment can thus be biologically unrealistic. The example in Figure 5.5 shows how such errors can occur when two sequences are being compared at the protein and DNA levels.

For that reason, sequence alignment at the protein level is much more informative for functional and evolutionary analysis. However, there are occasions when sequence

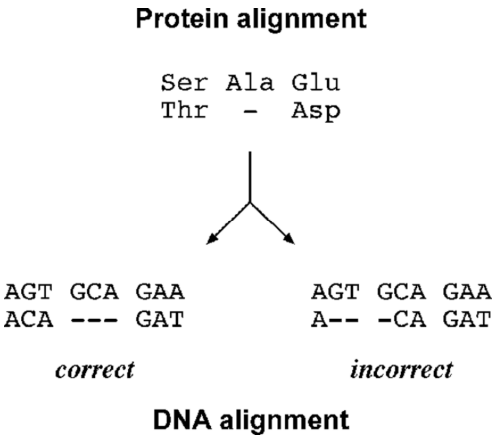


Figure 5.5: Comparison of alignment at the protein level and DNA level. The DNA alignment on the left is the correct one and consistent with amino acid sequence alignment, whereas the DNA alignment on the right, albeit more optimal in matching similar residues, is incorrect because it disregards the codon boundaries.

alignment at the DNA level is often necessary, for example, in designing PCR primers and in constructing DNA-based molecular phylogenetic trees.

Because conducting alignment directly at the DNA level often leads to errors, DNA can be translated into an amino acid sequence before carrying out alignment to avoid the errors of inserting gaps within codon boundaries. After alignment of the protein sequences, the alignment can be converted back to DNA alignment while ensuring that codons of the DNA sequences line up based on corresponding amino acids. The following are two web-based programs that allow easy conversion from protein alignment to DNA alignment.

RevTrans (www.cbs.dtu.dk/services/RevTrans/) takes a set of DNA sequences, translates them, aligns the resulting protein sequences, and uses the protein alignment as a scaffold for constructing the corresponding DNA multiple alignment. It also allows the user to provide multiple protein alignment for greater control of the alignment process. PROTA2DNA (http://bioweb.pasteur.fr/seqanal/interfaces/protal2dna.html) aligns DNA sequences corresponding to a protein multiple alignment.

Editing

No matter how good an alignment program seems, the automated alignment often contains misaligned regions. It is imperative that the user check the alignment carefully for biological relevance and edit the alignment if necessary. This involves introducing or removing gaps to maximize biologically meaningful matches. Sometimes, portions that are ambiguously aligned and deemed to be incorrect have to deleted. In manual editing, empirical evidence or mere experience is needed to make corrections on an alignment. One can simply use a word processor to edit the text-based alignment. There are also dedicated software programs that assist in the process.

BioEdit (www.mbio.ncsu.edu/BioEdit/bioedit.html) is a multifunctional sequence alignment editor for Windows. It has a coloring scheme for nucleotide or amino acid residues that facilitates manual editing. In addition, it is able to do BLAST searches, plasmid drawing, and restriction mapping.

Rascal (http://igbmc.u-strasbg.fr/PipeAlign/Rascal/rascal.html) is a web-based program that automatically refines a multiple sequence alignment. It is part of the PipeAlign package. It is able to identify misaligned regions and realign them to improve the quality of the alignment. It works by dividing the input alignment into several regions horizontally and vertically to identify well-aligned and poorly aligned regions using an internal scoring scheme. Regions below certain thresholds are considered misaligned and are subsequently realigned using the progressive approach. The overall quality of the alignment is then reassessed. If necessary, certain regions are further realigned. The program also works in conjunction with NorMD, which validates the refined alignment and identifies potentially unrelated sequences for removal.

Format Conversion

In many bioinformatics analyses, in particular, phylogenetic analysis, it is often necessary to convert various formats of sequence alignments to the one acceptable by an application program. The task of format conversion requires a program to be able to read a multiple alignment in one format and rewrite it into another while maintaining the original alignment. This is a different task from simply converting the format of individual unaligned sequences. The BioEdit program mentioned is able to save an alignment in a variety of different formats. In addition, the Readseq program mentioned in Chapter 2 is able to perform format conversion of multiple alignment.

Readseq (http://iubio.bio.indiana.edu/cgi-bin/readseq.cgi/) is a web-based program that is able to do both simple sequence format conversion as well as alignment format conversions. The program can handle formats such as MSF, Phylip, Clustal, PAUP, and Pretty.

SUMMARY

Multiple sequence alignment is an essential technique in many bioinformatics applications. Many algorithms have been developed to achieve optimal alignment. Some programs are exhaustive in nature; some are heuristic. Because exhaustive programs are not feasible in most cases, heuristic programs are commonly used. These include progressive, iterative, and block-based approaches. The progressive method is a stepwise assembly of multiple alignment according to pairwise similarity. A prominent example is Clustal, which is characterized by adjustable scoring matrices and gap penalties as well as by the application of weighting schemes. The major shortcoming of the program is its "greediness," which relates to error fixation in the early steps of computation. To remedy the problem, T-Coffee and DbClustal have been developed that combine both global and local alignment to generate more sensitive alignment. Another improvement on the traditional progressive approach is to use graphic profiles, as in Poa, which eliminate the problem of error fixation. Praline is profile based and has the capacity to restrict alignment based on protein structure information and is thus much more accurate than Clustal. The iterative approach works by repetitive

refinement of suboptimal alignments. The block-based method focuses on identifying regional similarities. It is important to keep in mind that no alignment program is absolutely guaranteed to find correct alignment, especially when the number of sequences is large and the divergence level is high. The alignment resulting from automated alignment programs often contains errors. The best approach is to perform alignment using a combination of multiple alignment programs. The alignment result can be further refined manually or using Rascal. Protein-encoding DNA sequences should preferably be aligned at the protein level first, after which the alignment can be converted back to DNA alignment.

FURTHER READING

Apostolico, A., and Giancarlo, R. 1998. Sequence alignment in molecular biology. *J. Comput. Biol*. 5:173–96.

Gaskell, G. J. 2000. Multiple sequence alignment tools on the Web. *Biotechniques* 29:60–2.

Gotoh, O. 1999. Multiple sequence alignment: Algorithms and applications. *Adv. Biophys*. 36:159–206.

Lecompte, O., Thompson, J. D., Plewniak, F., Thierry, J., and Poch, O. 2001. Multiple alignment of complete sequences (MACS) in the post-genomic era. *Gene* 270:17–30.

Morgenstern, B. 1999. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics* 15:211–8.

Morgenstern, B., Dress, A., and Werner T. 1996. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. U S A* 93:12098–103.

Mullan, L. J. 2002. Multiple sequence alignment – The gateway to further analysis. *Brief. Bioinform*. 3:303–5.

Nicholas, H. B. Jr., Ropelewski, A. J., and Deerfield, D. W. II. 2002. Strategies for multiple sequence alignment. *Biotechniques* 32:572–91.

Notredame, C. 2002. Recent progress in multiple sequence alignment: A survey. *Pharmacogenomics* 3:131–44.

Notredame, C., Higgins, D. G., and Heringa, J. 2000. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.* 302:205–17.

Thompson, J. D., Higgins, D. G., and Gibson, T. J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*. 22:4673–80.

Thompson, J. D., Plewniak, F., and Poch, O. 1999. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res*. 27:2682–90.