UWBC Biotechnical Resource Series

Richard R. Burgess, Series Editor
University of Wisconsin Biotechnology Center
Madison, Wisconsin

Sequence Analysis Primer

Edited by

Michael Gribskov
and
John Devereux

G. Grant, ed. **Synthetic Peptides: A User's**

M. Gribskov and J. Devereux, eds. **Sequence Analysis Primer**

In Preparation:

the same type used to compute a dot matrix) and note the average displacements of the highest-scoring segments. This is conveniently done using

A brute force approach of aligning sequences with the automatic insertion of gaps shows that the problem is very difficult. Simply comparing two

might say that, for sequence 1 and sequence 2 numbered 1 to i and 1 to j respectively

$$S_{ij} = s_{ij} + \max_{\substack{1 \le k < i \\ 1 \le l < j}} S_{kl} \qquad (3)$$

where S_{ij} is the score for the alignment ending at i in sequence 1 and j in sequence 2

Simple Example of Dynamic Programming Alignment

Actual alignments are calculated in two stages. First, the two sequences are arranged on a lattice in much the same way as in dot matrix methods. For each point in the lattice, the alignment score, S_{ij}, is calculated. At the same time, the position of the best alignment in the previous row or column, i.e., the score of the best previous alignment which was used to calculate S_{ij}, is stored. This stored value is called a pointer and is represented by an arrow. In the second stage, the alignment is produced by starting at the highest alignment score in the lattice, and building up the alignment from right to left by following the

containing the S_{ij} values, is filled in from left to right and top to bottom. Figure 20a shows the score matrix after filling the first row (i=1). Two positions, indicated by circled scores, are matches and receive a score of 1. All other positions receive scores of zero. The alignment score, S_{ij}, is the sum of the score for comparing the bases at i and j, plus the best previous alignment. Since all of these elements correspond to the first base in sequence 1, there are no best previous alignments and no pointers are saved at this point.

Figure 20b shows a later stage in the calculation of the score matrix. $S_{2,1}$ is an edge and therefore there is no best previous alignment to consider. $S_{2,2}$ has only one position that could contain a previous alignment, $S_{1,1}$, and this is therefore the position used for the pointer. To calculate $S_{2,2}$ we add the score

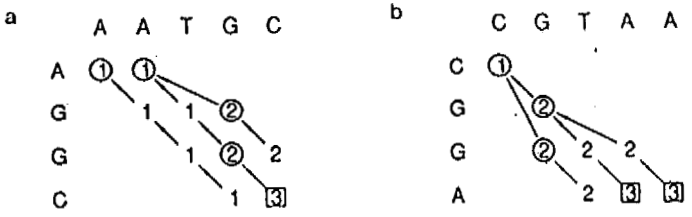

Figure 21: Effect of not saving all path pointers. The alignment in Figure 20 is repeated, but only one path pointer, the one which would introduce the shortest gap, is saved for each position, S_{ij}, in the score matrix. a. forward alignment b. reverse alignment - the same

alignment 1 (Figure 20c). The highroad/lowroad options will always give different alignments if there are equivalent alignments. For long sequences there could be hundreds of equivalent alignments and it is prohibitive and confusing to list them all. The highroad/lowroad procedure can be thought of as establishing an upper and lower bound for the variation of the alignments.

Another way of detecting possible equivalent alignments is shown in Figure 21b. Simply perform the alignment a second time, keeping all

most programs have relied on gap penalties with both a length-dependent and a length-independent term (equation 5).

$$w_x = g + lx \tag{5}$$

where w_x is the penalty for a gap of length x
g is the length-independent term (gap opening penalty)
l is the length-dependent term (gap extension penalty)

Figure 23: Score matrices and path graphs for global and local alignments. The alignment of part of the promoter regions from Phi-X174 *A* gene and the *Escherichia coli lac* gene. For both alignments, matches receive a score of 1, the gap opening penalty is 1.2 and the gap extension penalty is 0.3. In the local alignment, mismatches receive a score of -0.6. The best paths are shaded, and the highest scoring positions shown in a square. The corresponding alignments are shown below the score/path matrices.

Extensions

The primary drawback to dynamic programming methods is that they require a considerable amount of computation. This limits their usefulness for tasks such as database searching. One simple way to speed up the alignment is to calculate only part of the score matrix, usually a diagonal band down the center (e.g., Sankoff and Kruskal, 1983). This can be safely done, for instance, if you know the sequences are homologous and do not require large gaps in their alignment, or if you have information from a faster method, such as hashing (see Hashing and Neighborhood Algorithms) that tells you where the most similar regions of the sequences are. Several methods that perform a banded alignment and iteratively increase the width of the band until the optimal alignment is found have been presented (Ukkonen, 1983).

A further great increase in alignment speed can be achieved through subdivision. If segments in each sequence can be identified, for instance by hashing methods (see Hashing and Neighborhood Algorithms), that are so similar they are unlikely to match with anything else, the alignment can be broken down into two smaller alignments, separated by the matching segment. Each equal subdivision increases the speed of the alignment by a factor of two.

Once a cDNA clone is sequenced, one usually wishes to identify the protein encoded by the message. One approach is to translate all three (or six) reading frames of the nucleic acid sequence and use the resulting protein sequences as probes in a fast database search (e.g., TFASTA - Lipman and Pearson, 1985; TBLASTN - Gish et al., in preparation). Unfortunately, this approach can be quite sensitive to frameshift errors in the cDNA sequence

this approach. In the absence of a single appropriate scoring table, most nucleic acid sequence alignments continue to be based on identity scoring systems.

Identity-based scoring systems often do not give the desired sensitivity when comparing distantly related sequences, especially for protein sequences. There is a strong consensus that, for proteins, scoring systems based on the chemical or mutational similarity of the amino acid residues are much better than identity scoring systems (Schwartz and Dayhoff, 1978; Feng and Doolittle, 1987). One early method of scoring the similarity of amino acid residues is known as the minimum base change or genetic code matrix. This scoring system calculates the similarity between residues as the minimum number of base changes required to change a codon for one residue to a codon for another. This system seemed especially plausible for evolutionary studies because it allowed the difference in amino acid residues to be stated in terms of the minimum number of mutational events needed to convert one residue to another.

The most commonly used scoring systems for protein sequences are based on the MDM_{78} table (mutation data matrix, 1978) of Dayhoff and coworkers (Schwartz and Dayhoff, 1979; George et al., 1990; see Appendix IV). Often called simply the "Dayhoff" table, this scoring table is derived using the "accepted point mutation" model of evolution (Dayhoff et al., 1978). A dataset was compiled from a group of closely related proteins (less than 15% amino acid differences), that could be unambiguously aligned. From these aligned sequences, Dayhoff and coworkers calculated a matrix describing the probability, for each residue, that a mutation would change the

although it has been argued that it may be better to use an equivalent log-odds matrix calculated for a lower PAM for alignments of unknown sequences (Altschul, personal communication). The log of the probability of two sequences being evolutionarily related can, in principle, be calculated as the sum of the scores for each aligned pair of residues, i.e., the alignment score if a log-odds matrix is used as the scoring system for the alignment. However, this is overly simplistic since it ignores the effect of insertions and deletions

- the scoring system as a whole obeys the triangle inequality; if you consider any three distances, the distance from A to C must be less than or equal to the distance from A to B plus the distance from B to C

Metric distances have received a great deal of attention by mathematicians