A fast homology program for aligning biological sequences

Philip Taylor

Institute of Virology, University of Glasgow, Church Street, Glasgow G11 5JR, UK

Received 7 June 1983

ABSTRACT

The algorithm of Gotoh\(^1\) computes in two passes of MN steps the alignment of a pair of sequences of lengths M and N, subject to a constraint on the form of the gap weighting function. This compares with the previous algorithm of Waterman et al.\(^2\) which runs in \(M^N\) steps. Gotoh\(^1\) also gave a method using two passes of \((L+2)MN\) steps in the case where gap weights remain constant for gaps of length greater than L.

Here we describe a procedure for computing the alignment (evolutionary distance and optimal path) in a single pass of MN steps for both cases.

INTRODUCTION

In modern molecular biology it is frequently of interest to investigate the degree of similarity of a pair of sequences of nucleotides, or of amino-acids. Most computer methods for aligning these biological sequences proceed essentially by inserting gaps in such a way as to maximise the number of matched pairs obtained in the resultant sequences. Gaps however are counted as unmatched regions, so that these are constrained to be as few and as short as possible. Thus the algorithms are designed to find the optimal alignment or alignments by balancing the introduced gaps against the remaining unmatched pairs in order to produce a minimum measure of dissimilarity, or divergence. The use of the term 'divergence' derives from a commonly made inference, namely that an optimal alignment of two similar sequences displays their homology. But as Smith et al.\(^3\) pointed out, similarity of sequence may in fact reflect analogy i.e. functional convergence, rather than descent from a common ancestor.

The principle of aligning two sequences by considering the
match obtained by all possible pairings was introduced by the iterative matrix process of Needleman and Wunsch. The method was developed by Sellers who modified the algorithm so that it measured the divergence between two sequences; and subsequently by Waterman et al. who extended the algorithm to handle gaps of varying length. This feature is especially valuable where there are known (long) insertions or deletions in the sequence data.

However the computation of Waterman et al. required iterative steps of the algorithm, thereby imposing a practical limitation on its usefulness for comparing moderately sized sequences on a small laboratory computer. The situation was markedly improved by a new algorithm of Gotoh, which requires that the gap weighting function be of the form \( w = uk + v \) where \( k \) is the gap length and \( u, v \) are predetermined constants. His method computes the divergence of two sequences in \( MN \) steps, and completes the alignment with a second pass, thus \( 2MN \) steps in all. He further showed that where gap weights are limited by \( w = uL + v \) for \( k > L \) in anticipation of long gaps, the computation could be completed in two passes each of \( (L+2)MN \) steps.

This paper presents several varieties of an algorithm which reduce the calculation to \( MN \) iterations both for limited and unlimited gap weights.

### EVOLUTIONARY DISTANCE

We assign point-mutation weights \( d(a,b) \) for the pairing of element \( a \) in the first sequence with element \( b \) in the second \( (d(a,b) > 0 \) unless \( a = b) \), and gap weights \( w_k \) increasing monotonically with size \( k \). Then the algorithm aligns two sequences \( A \) and \( B \) so as to minimise the sum of all such weights, denoted \( D(A,B) \). This sum is called Evolutionary Distance because it measures the (minimal) amount of mutational change from sequence \( A \) to sequence \( B \), and because it satisfies the following metric properties:
(i) \( D(A,B) = 0 \) iff \( A = B \). The distance from \( A \) to \( B \) equals zero if and only if they are identical.
(ii) \( D(A,B) = D(B,A) \). The distance from \( A \) to \( B \) equals the distance from \( B \) to \( A \), hence we may speak of the distance between \( A \) and \( B \). [Actually we often think of the evolutionary distance between two sequences as being the sum of their distances from a hypothetical intermediate common ancestor]. This property requires that point-mutation weights are symmetrical i.e. \( d(a,b) = d(b,a) \) for all \( a \) and \( b \).
(iii) \( D(A,B) \leq D(A,C) + D(C,B) \). Among three sequences the distance between any two cannot be more than the sum of the distances between the other two pairs. Usually it is less; it is equal only in the special case where the 'other two pairs' do not have any mutations in common. This property, sometimes called the Triangle Inequality, is important for comparing the genetic relatedness of a number of sequences, as in constructing an evolutionary tree for example. If, as proposed by Goad and Kanehisa\(^6\) and others, a contingency table of variable point-mutation weights \( d(a,b) \) is used in place of a single value, these weights must themselves conform to the triangle inequality viz. \( d(a,b) \leq d(a,c) + d(c,b) \) for all elements \( a, b \) and \( c \).

A further property of evolutionary distance is \( D(A^*,B^*) = D(A,B) \) where \( A^* \) denotes the reversal of \( A \). It is reasonable to expect the optimal alignment to be indifferent to the direction in which the algorithm processes the sequences.

The relationship of Evolutionary Distance to other measures used in homology matching, together with aspects of its interpretation and statistical & biological significances, were discussed by Smith et al.\(^3\)

**METHODS**

Four versions of the basic algorithm were programmed. They compute the distance matrix \( D_{m,n} \) for all subsequences of length \( m \) and \( n \) from the following three sequential inductive relationships:

\[
P_{m,n} = \min(D_{m-1,n} + v, P_{m-1,n}) + u \\
Q_{m,n} = \min(D_{m,n-1} + v, Q_{m,n-1}) + u
\]
These iterative formulae are adapted from those given by Gotoh\(^1\)

The present programs however process the path matrix \( E_{m,n} \) (Goad and Kanehisa\(^6\), Gotoh\(^1\)) in a novel way. In parallel with computation of \( D_{m,n} \), they count gap lengths, for both horizontal and vertical path segments, corresponding to the first (along the x-axis) and second (down the y-axis) sequence respectively. The several versions of the program differ in respect of the trade-off made between the limitations of the way in which this information is stored, and the program requirements of memory and (disk) file storage space.

1. In the first version, horizontal gap length is recorded in a single word variable, and when a horizontal path of length \( k \) is selected (\( D_{m,n} = P_{m,n} \)) the preceding \( k-1 \) elements of the path matrix \( E_{m,n} \), i.e. \( E_{j,n} \) for \( m-k < j < m-1 \), are retrospectively modified by adding the horizontal bit indicator. Vertical gap lengths on the other hand are encoded into higher-order bits of the path matrix. Thus for example when using byte storage for the matrix, and assuming that 3 bits are allocated to horizontal, diagonal and vertical direction indicators, there are 5 bits available for recording the (vertical) gap length. This accommodates gaps of up to 31 elements in the first sequence, while there is no limitation on gap length in the second sequence. In retracing a path from the \( E \) matrix, it is necessary to continue with the previous direction if possible, else to prefer a diagonal or vertical segment to a horizontal one. This implies that possibly not all of multiple optimal alignments can be recovered from the filed path matrix \( E_{m,n} \) with this version, c.f. version (4) below. [In this respect version (1) of the algorithm is similar to the procedure of Gotoh\(^1\)].

2. By using an auxiliary memory array to store vertical gap length, we may increase the permissible maximum to 63 by defining bits 0 and 1 for diagonal and horizontal indicators, and bits 2-7 for storing the current cumulative vertical gap length which is however masked off before filing the \( E \) array if the vertical segment is not chosen (\( D_{m,n} < Q_{m,n} \)). Hence this version of the program approximately doubles the possible gap
size in return for a modest increase in memory usage.

(3) If only one minimal path is desired, it is feasible to choose a segment at each point of the matrix and encode it as \( \emptyset \) for diagonal, -1 (255) for horizontal, and 1-254 for vertical (gap length). Again the vertical gaps must be maintained in an auxiliary memory array distinct from the array stored row-by-row in the E matrix file; and as before multiple horizontal gaps are marked retrospectively. Where disk space is at a premium, this implementation of the program will probably be the most attractive to small-computer users.

(4) The last method consists in using a word array to store/file the paths. With for example a 16-bit word and using one bit for a diagonal indicator, seven bits each may be allocated to horizontal and vertical path segments, thus permitting gaps up to 127 elements in either sequence. This scheme also allows all possible optimal paths to be retraced/enumerated where required.

In all the foregoing cases, because both horizontal and vertical gap lengths are known at every point in the matrix computation, it is straightforward to apply an upper bound to the formula for gap weight viz. \( w = \text{constant (uL+v)} \) for \( k \) greater than a prescribed \( L \). This is highly advantageous where long insertions or deletions are known in advance, as these might otherwise be suppressed by their large penalty weightings.

The first three of these versions of the algorithm were tested on a variety of sequences. As expected, they give identical results of sequence alignment and evolutionary distance, provided that their limits are respected. Because the sequences used in the next section were known to have only short gaps (less than 10 elements say), the reported results e.g. execution times were obtained with the first version of the program.

**RESULTS**

Figure 1 illustrates the operation of the algorithm, with the first sequence printed along the horizontal axis and the second sequence printed down the vertical axis. Distance and path elements for each point in the matrix computation are shown
FIG. 1. The Distance \( D_{mn} \) and Path \( E_{mn} \) matrices generated by the algorithm in comparing (a) two sequences A,B and (b) the same sequences B,A but in reverse order. Values of \( E_{mn} \) are printed below corresponding values of \( D_{mn} \). Bit assignments in \( E_{mn} \) are 1 = diagonal, 2 = horizontal and 4 = vertical path segments, with higher-order octal digits recording the vertical gap lengths (less 1). The circled value of \( E_{mn} \) is retrospectively modified. Chosen paths are shown in solid, alternate paths in broken arrows. Mutation weight = 4, and gap weights = 3\( k + 7 \) (up to a gap length of 24). Matching is 3/6 or 50%, and Evolutionary Distance = 18.3

at the row/column intersections. A gap-weight limiting length of 24 was specified for the program run, but this has no effect on the alignment of these particular short sample sequences.

In addition to array space for the input sequence data, the program requires two (word) arrays of length \( M \) for transient storage of previous \( D \) and \( Q \) values, and (at least) one byte/word array for \( E \) path elements. Usually a more significant storage space limitation will be the size of the direct-access workfile on disk used for storing the path matrix. Some sizes are given in Table 1.

The program is heavily I/O bound in accessing the path matrix workfile. Compared to the number of logical steps in the algorithm, there are relatively few calculations and these can, with judicious choice of weights, be performed in integer arithmetic. Hence program execution time depends largely on the speed of disk operations. Table 1 gives some execution times using a disk with a transfer rate of 300 kilobytes/second.
TABLE 1. Program performance and organisation.

<table>
<thead>
<tr>
<th>Length of sequences</th>
<th>Size of direct-access workfile (kilobytes)</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10</td>
<td>9s</td>
</tr>
<tr>
<td>500</td>
<td>245</td>
<td>2m27s</td>
</tr>
<tr>
<td>1000</td>
<td>977</td>
<td>8m12s</td>
</tr>
<tr>
<td>2000</td>
<td>3,906</td>
<td>39m23s</td>
</tr>
<tr>
<td>3500</td>
<td>11,963</td>
<td>---</td>
</tr>
</tbody>
</table>

The file sizes were calculated assuming byte storage of the path matrix elements. If for example 16-bit word storage was substituted, these sizes would double. File-space is proportional to MN.

The measured program execution timings include traceback of one path from the matrix file, and construction of the aligned sequences therefrom. It can be seen that these timings approximately vary with MN. Test sequences consisted of pseudo-randomly shuffled elements of 4 kinds and present in equal proportions.

1 CACAAGGCAGAGATGTGGTCGAATGGCTCGGAGTTGAGATCAGAAAA
** **** **** **** ** ** ***** ** **** *** ** * *****

1 CAAAAGGAAGAAATGTGGACTGGCTTGGCTGGTATGATGACAACGGGGGAAAA

57 CCGACTCCTGATATGATGCAGTATGCGAAACGAGCAGTCATGTCACTGCAAGGCCT
** ** ** ************ ** * * * * * ********** **

57 CCCACACCAGATATGCTCAACTTCGCAAGAAGAGCAGTTAACTCTCTGCAGTCACT

113 AAGAGAGAAGACAATTGGCAAGTATGCTAAGTCAGAATTTGACAAATGAC CC
* ***************** ***** *** **** *** **** **** **

113 TCGTGAGAAGACAATTGGCAAATATGCCAAGGCAGAGTTTAACAAGTGACAAGCCT

165 TATAATTCAGACATCACCATTATATATATTATGCTACATATG
** ** ** ************ ** * * * * * **

169 TAAGATACCACATCA CTATTTATCTCATGTATATATATG

Mutation weight = 5 Gap weights = 6k + 2, k up to 21
MATCHED = 147 OUT OF 209 PERCENT 70
EVOLUTIONARY DISTANCE = 321

FIG. 2. Example of sequences aligned by the program. The limit on gap weight to that of a gap length of 21 has no effect in this instance. Data are VSV-Indiana and VSV-New Jersey sequences adjacent to the poly(A) tracts of N mRNA's, from McGeoch, Dolan and Pringle (1980).
Figure 2 shows a typical output from the program. The RNA nucleotide sequence data are portions of New Jersey and Indiana VSV genomes from McGeoch et al. 7

**DISCUSSION**

Computer generated alignments like fig.2 are now quite common in the literature. As a single 'solution' is normally presented, such alignments tend to be regarded as possessing a somewhat spurious 'validity' - it is well to bear always in mind that for a given set of weights, more than one alignment may be equivalent (i.e. yield the same evolutionary distance), and that different weights may produce sometimes very different alignments. The lower the homology percentage matching, the more numerous are these alternatives and variations likely to be. Altogether the number of ways in which two sequences may be aligned by the insertion of multiple gaps is typically very large, for example over 8 million for sequences of length 10. [A general formula has recently been submitted for publication elsewhere].

To some extent the experimenter can adjust the weights so as to give the 'desired' alignment, but there are limits both on the effectiveness of this balancing process and also on the range of weights which have a reasonable interpretation. A forthcoming publication will give detailed consideration to guidelines and criteria for selecting weights suited to sequences of a particular type or character.

Several algorithms were presented for computing the alignment of two biological sequences in MN iterative steps. These methods allow gap weights to be limited beyond a specified gap length. Some of the properties of the metric measure Evolutionary Distance were discussed.

The program was written in FORTRAN IV with generous commenting. Departures from Standard Fortran were kept to a minimum (e.g. manufacturer-specific file assignment statements) in order to facilitate implementation on different machines. Documentation and source code are available from the author.
ACKNOWLEDGEMENT

I thank Dr. D.J. McGeoch for critical reading of the manuscript, and Prof. J.H. Subak-Sharpe for encouragement and support. The work was supported by funds made available through the auspices of the Medical Research Council Virology Unit.

REFERENCES