CABIOS COMMUNICATIONS
Vol.5. no.2. 1989
Page 151 – 153

Fast and sensitive multiple sequence alignments on a microcomputer

Desmond G.Higgins* and Paul M.Sharp

Abstract

A strategy is described for the rapid alignment of many long nucleic acid or protein sequences on a microcomputer. The program described can handle up to 100 sequences of 1200 residues each. The approach is based on progressively aligning sequences according to the branching order in an initial phylogenetic tree. The results obtained using the package appear to be as sensitive as those from any other available method.

Introduction

In the recent literature on biological sequence analysis, at least a dozen methods for performing multiple alignments of nucleic acid or protein sequences have been described [e.g. Bains (1986), Sobel and Martinez (1986), Barton and Sternberg (1987), Feng and Doolittle (1987), Santibanez and Rohde (1987), Taylor (1987)]. The motivation for this effort has been the need for the automatic alignment of three or more sequences for the purposes of evolutionary or structural comparisons or for attempting to demonstrate similarity between sets of sequences. In this paper, we describe a strategy which we believe offers the best combination of speed and sensitivity available for any multiple alignment method. We offer a program which can perform multiple alignments of up to 100 sequences of maximum length 1200 residues on a microcomputer in a reasonable amount of time. We judge the program to be 'sensitive' because the results obtained are very difficult to improve by eye.

The strategy we use is essentially that of Feng and Doolittle (1987) adapted for use on microcomputers. The general approach is to progressively align groups of sequences according to the branching order in a hypothetical phylogenetic tree, with gaps that occur in earlier alignments being preserved through later stages. At each alignment stage, a two-sequence alignment algorithm, such as the dynamic programming method of Needleman and Wunsch (1970), is used. For two sequences, the Needleman and Wunsch algorithm gives an alignment that is guaranteed to be optimal for a given set of scoring rules (i.e. weights for aligned residues and penalties for gaps). When this method is used to align two sets of sequences, the score at each position in the alignment is taken from the average score for each residue in one set compared against each residue in the second set. Any gaps introduced into either set of sequences are scored as single gaps. The main difficulty in using this approach on a microcomputer arises from the excessive memory requirements of the Needleman and Wunsch (1970) method—memory usage is proportional to the square of the average sequence length.

In a previous paper (Higgins and Sharp, 1988) we described a strategy for the very rapid multiple alignment of large numbers of sequences on a microcomputer. This method also comprised a progressive approach, using the fast, but approximate, two-sequence alignment method of Wilbur and Lipman (1983). While this approach is extremely rapid and economical with core memory, it works well only for closely related sequences. We did not consider using the exactly optimal method of Needleman and Wunsch (1970) for the progressive alignments because of the excessive memory requirements. However, a recent paper by Myers and Miller (1988) demonstrates how to achieve exactly optimal alignments of two sequences where memory usage varies only linearly with sequence length, without making use of bit packing or secondary disk storage. Thus, a progressive series of alignments of larger and larger groups of sequences, using the method of Myers and Miller (1988) for each alignment, is the key to the current approach.

System

The program described in this paper was written in standard FORTRAN 77 and compiled using the Microsoft FORTRAN compiler, version 4.0. Program performance was tested on an IBM AT compatible microcomputer, running at 10 MHz with no maths coprocessor, 640 kbytes of memory and a hard disk. This program (CLUSTAL4) is an extension to the package described in Higgins and Sharp (1988). Copies of the executable files, documentation and test data files will be sent on request. Please send three 5.25 inch floppies formatted to 360 kbytes, or one high density 5.25 inch floppy formatted to 1.2 Mbytes.

Algorithms

The program takes, as input, a dendrogram produced by applying the UPGMA method (Sneath and Sokal, 1973) to a matrix of similarity scores between all pairs of sequence to be aligned. The similarity scores are calculated as the number of exactly matched residues in a Wilbur and Lipman (1983) align-

Department of Genetics, Trinity College, Dublin 2, Ireland
**To whom correspondence should be addressed*

ment between two sequences, minus a fixed penalty for every gap. For short sequences, several similarity scores per second can be calculated on a microcomputer using the package described in Higgins and Sharp (1988).

The sequences are then aligned in groups corresponding to the branching order in the dendrogram. The alignments are carried out using the method of Myers and Miller (1988), adapted for use in a multiple alignment context. Myers and Miller took the distance minimizing algorithm of Gotoh (1982) and applied a 'divide and conquer' strategy (attributed to Hirschberg, 1975) to give alignments in linear space; as a consequence memory usage is linearly related to sequence length. Briefly, the method is based on finding the optimal mid-point of an alignment. When this is found, the matched symbols (two aligned residues, or one residue opposite a gap) are part of the final alignment. The rest of the alignment is found by recursively finding optimal mid-points on either side of the initial mid-point. In this context, the optimal mid-point can be defined as the aligned symbols at the centre of the optimal alignment. The centre is taken to be half way along one sequence.

Two modifications were needed to adapt the Myers and Miller algorithm for our program. Firstly, all real number operations were converted to using 2-byte integers. On a 16-bit microcomputer without a maths chip, this increases the speed of each alignment by a factor of 30. Indeed the speed approaches that described in Myers and Miller for their program running on a VAX 11/780. Secondly, the scoring system was modified to allow all residues at a given position in each group of sequences to contribute to the alignment scores. For proteins, we use the log-odds amino acid similarity matrix of Dayhoff (1978) to score aligned residues. The similarity matrix was rescaled to give positive integer weights between 0 and 25 and then converted to a difference matrix by subtracting each value from 25. Thus two aligned tryptophans have the lowest distance (0) while a cysteine aligned with a tryptophan has the largest distance (25). For nucleic acid sequences we use a three-tier weighting system where identical residues have zero distance, transitions have a distance of 5 and transversions have a distance of 10. A variable gap penalty is used; a fixed penalty is added to the alignment distance for every gap and an extra penalty is added for every item in the gap. Gaps that are introduced into a pre-aligned group of sequences are scored as single gaps. Both of these penalties can be specified at run time.

In order to calculate the alignment scores between two clusters of sequences, the gaps that are already inserted in the two clusters (from earlier alignment stages) are treated as being fixed. Thus, each cluster may be thought of as a single sequence,

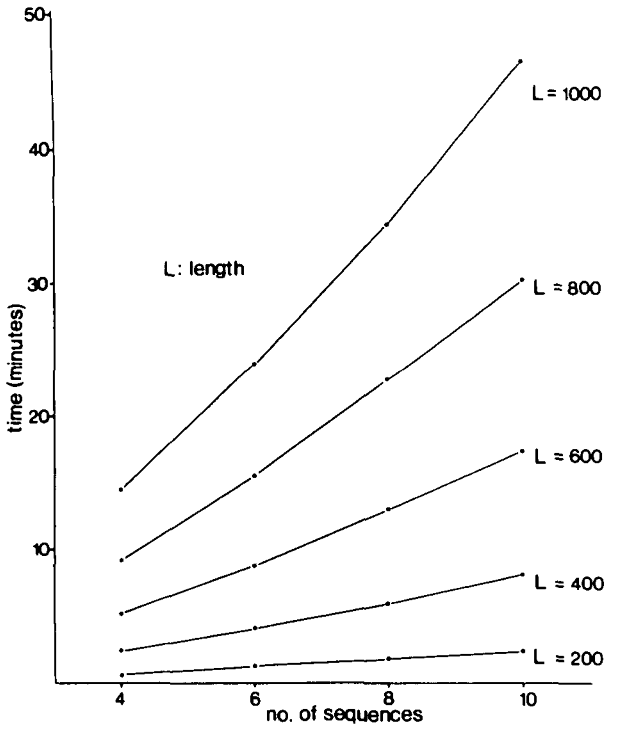

Fig. 1. Times required for the multiple alignment of different numbers of sequences of different lengths. Each curve represents the times for truncated fragments of a given length, L; this example used the HIV pol protein. Times for calculating the similarity matrices or dendrograms (maximum of <3 min) are not included.

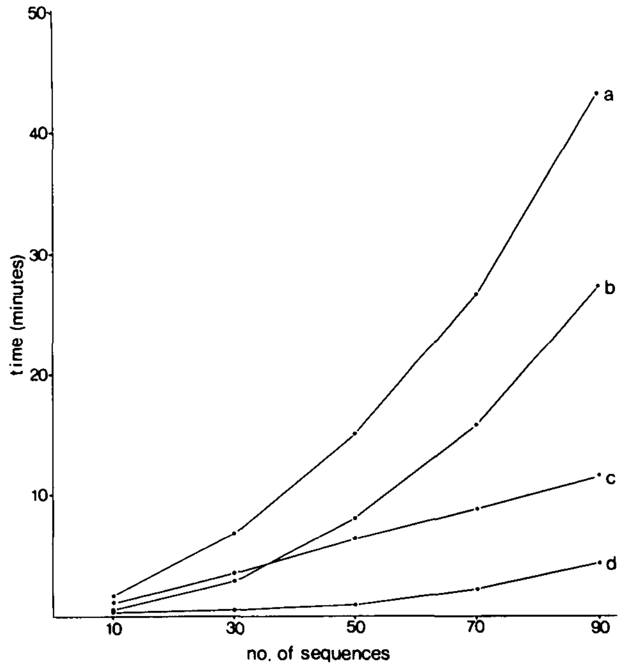

Fig. 2. Times for aligning different numbers of sequences. This example used globin sequences (alphaglobins, betaglobins and myoglobins) each truncated to 140 residues. The four curves show times for different parts of the multiple alignment process: (**a**) total time, including calculation of similarity matrix and dendrogram, (**b**) calculation of similarity matrix, (**c**) multiple alignment, (**d**) construction of UPGMA dendrogram.

where the residues at each position are the 'average' residues at that position in each of the sequences represented by the cluster. In order to calculate the weight between a position in one cluster and another position in the second cluster, one takes the arithmetic average of all the pairwise weights between each residue in one cluster versus all of those in the second. For two clusters with K and L sequences each, this involves taking the average of K times L weights at each alignment position. This becomes very time-consuming with many sequences, but can be speeded up by pre-calculating the weight of each position in each cluster versus each possible residue.

Results and discussion

The speed of the program can be demonstrated by aligning different numbers of sequences of different sizes. We find that the speed is almost totally independent of the characteristics of the sequences, apart from length. This was also noted by Myers and Miller (1988) for their two-sequence alignment program. Figure 1 shows the times required for a series of multiple alignments of sequences from 200 to 1000 amino acids in length. One expects the alignment times to vary with the square of the sequence lengths and a visual inspection of the figure confirms this. The time required to align different numbers of sequences varies approximately linearly. A slight departure from linearity is evident with the longer sequences. This confirms the effectiveness of our stategy of pre-calculating the weights at different positions in each cluster. The times required for calculating the initial similarity matrices and construction of the dendrograms are not shown. These only need to be calculated once for any multiple alignment. The slowest dendrogram to construct was that for the ten 1000 residue sequences. This took under 3 min.

Figure 2 shows the times required to align from 10 to 90 sequences of 140 amino acids each. In this case the times for each of the various calculations are shown. The similarity matrices and dendrograms were constructed using the programs CLUSTAL1 and CLUSTAL2 (Higgins and Sharp, 1988) respectively. For large numbers of sequences, the calculation of the initial similarity matrix is the dominant time-consuming factor. For 90 sequences, this requires the calculation of 4005 values. Nonetheless, the times involved are quite practical on a microcomputer.

The sensitivity of the program is more difficult to demonstrate. Our basic criterion in determining sensitivity is to assess the ease with which the resulting alignments can be improved by manual adjustment. By this criterion, we find the results of our program to be excellent. In this respect, the program can be used confidently to replace the usual manual alignment of sets of closely related sequences for publication. Of greater scientific importance is the usefulness of the program for aligning regions of homologous secondary structure or in reconstructing evolutionary events between distantly related sequences. This is more difficult to demonstrate. Barton and Sternberg (1987), Feng and Doolittle (1987) and Taylor (1987) discuss these questions in detail. It is possible that no single method will be ideal for these purposes. As a general observation, we find the alignments produced by our program to be at least as good as those produced by each of the above authors.

Acknowledgements

This is a publication from the Irish National Centre for Bioinformatics. This work was supported by grant number BAP-0137-IRL from the European Community Biotechnology Action Programme. We are particularly grateful to Gene Myers for sending us the source code of his linear space alignment program.

References

Bains,W. (1986) MULTAN: a program to align multiple DNA sequences. *Nucleic Acids Res.*, **14**, 159–177.

Barton,G.J. and Sternberg,M.J.E. (1987) A strategy for the rapid multiple alignment of protein sequences. *J. Mol. Biol.*, **198**, 327–337.

Dayhoff,M.O. (1978) A model of evolutionary change in proteins. Matrices for detecting distant relationships. In Dayhoff,M.O. (ed.), *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Washington DC, Vol. 5 Suppl. 3, pp. 345–358.

Feng,D.-F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Gotoh,O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Higgins,D.G. and Sharp,P.M. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**, 237–244.

Hirschberg,D.S. (1975) A linear space algorithm for computing longest common subsequences. *Commun. Assoc. Comput. Mach.*, **18**, 341–343.

Myers,E.W. and Miller,W. (1988) Optimal alignments in linear space. *CABIOS*, **4**, 11–17.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.

Santibanez,M. and Rohde,K. (1987) A multiple alignment program for protein sequences. *CABIOS*, **3**, 111–114.

Sneath,P.H.A. and Sokal,R.R. (1973) *Numerical Taxonomy*. Freeman, San Francisco.

Sobel,E. and Martinez,H.M. (1986) A multiple sequence alignment program. *Nucleic Acids Res.*, **14**, 363–374.

Taylor,W.R. (1987) Multiple sequence alignment by a pairwise algorithm. *CABIOS*, **3**, 81–87.

Wilbur,W.J. and Lipman,D.J. (1983) Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA*, **80**, 726–730.

Received on October 7, 1988; accepted on December 6, 1988

Circle No. 10 on Reader Enquiry Card