Proc. Natl. Acad. Sci. USA
Vol. 86, pp. 4412–4415, June 1989
Biochemistry

A tool for multiple sequence alignment

(proteins/structure/evolution/dynamic programming)

DAVID J. LIPMAN*†, STEPHEN F. ALTSCHUL*†, AND JOHN D. KECECIOGLU‡

*Mathematical Research Branch, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD 20894; and ‡Department of Computer Science, University of Arizona, Tucson, AZ 85721

Communicated by David R. Davies, March 16, 1989 (received for review November 28, 1988)

ABSTRACT **Multiple sequence alignment can be a useful technique for studying molecular evolution and analyzing sequence–structure relationships. Until recently, it has been impractical to apply dynamic programming, the most widely accepted method for producing pairwise alignments, to comparisons of more than three sequences. We describe the design and application of a tool for multiple alignment of amino acid sequences that implements a new algorithm that greatly reduces the computational demands of dynamic programming. This tool is able to align in reasonable time as many as eight sequences the length of an average protein.**

Comparative analysis of DNA and amino acid sequences is an increasingly important component of biological research. Sequence alignment, in particular, has been helpful in the study of molecular evolution (1), RNA folding (2), gene regulation (3), and protein structure–function relationships (4). Although pairwise sequence comparisons have proven useful, for example, in data base searches (5, 6), some

Alignment Cost

The dynamic programming method for aligning two sequences computes an optimal alignment—i.e., one whose replacement and gap costs have minimal sum. As with the PAM-250 matrix (1), which was derived from a study of amino acid replacements in homologous proteins, replacements may differ in cost. Gap costs may reflect the fact that a single mutational event can insert or delete several residues (25). The most biologically realistic approach for generalizing such costs to an alignment of several sequences involves minimizing the pairwise costs associated with the branches of an evolutionary tree whose leaves are the input sequences (24, 26, 27). Another measure of cost for multiple alignments is the sum of the alignment costs imposed on each pair of sequences in the multiple alignment (8, 14–16); this is called the SP measure (for sum of the pairs). We have used this measure because finding an optimal SP alignment requires much less time than does minimizing the branch lengths of a tree, especially when biologically realistic gap costs are used

projection can possibly pass. This in turn limits the points in the original lattice through which the optimal alignment can pass. Each projection thus defines a subset of the original lattice that contains the paths of all optimal alignments. The intersection of these subsets still contains all such paths. It is only this intersection, therefore, that must be considered by a dynamic programming algorithm seeking optimal alignments. In practice, this approach so limits the number of lattice points that it becomes possible to find optimal multiple alignments for as many as six sequences.

The MSA (multiple sequence alignment) program described here implements the algorithm of Carrillo and Lipman (23). It incorporates several important features not described in their paper; these features are discussed below.

Upper Bounds

The MSA program allows the user to choose an upper bound on the cost of aligning each pair of sequences. For each such pair, MSA then calculates which cells of the corresponding path graph can be contained within a path with cost no greater than the given bound. In the dynamic programming step, MSA then examines only those cells of the n-dimensional lattice that project onto the allowed cells in each of the two-dimensional path graphs. The program returns an alignment with minimum cost whose path is contained within this region.

As described above, Carrillo and Lipman have shown how to choose upper bounds on the cost of pairwise alignments that guarantee finding an optimal SP alignment (23). In practice, these rigorous bounds are almost always greater than is necessary. MSA therefore uses a heuristic procedure to choose upper bounds for the pairs. First, using a progressive alignment strategy similar to those described by Waterman and Perlwitz (17), Feng and Doolittle (20), and Taylor (21), it constructs a heuristic multiple alignment. For each pair, it sets the upper bound equal to the cost of the imposed alignment. This heuristic procedure has proved quite effective, but better methods for choosing bounds certainly may be found. MSA allows the user to specify any set of bounds and override their automatic assignment.

Using heuristic bounds, MSA generally can align six to eight sequences of length 200–300 residues. It is possible that some of the pairwise projections of the alignment found, while lying within the allowed region, may nevertheless have

strictly adopting this definition leads to unacceptable algorithmic complications. The MSA program uses instead the quasi-natural gap costs he proposes, which are identical to natural gap costs except in certain rare cases. Specifically, when a null run in one sequence of a multiple alignment begins after and ends before a null run in a second sequence, these costs count one more gap than do natural gap costs.

Other multiple alignment gap costs have been defined *in vacuo*, with no connection to the accompanying definition of substitution cost (15, 16, 31). However, since both types of cost work in tandem to specify optimal alignments, they should have a common rationale. Such consistency eliminates the need to readjust the gap cost when aligning different numbers of sequences. The user may specify whether terminal gaps are to be counted.

Pair Weights

While permitting reasonably efficient algorithms, the SP measure of multiple alignment cost has certain undesirable properties. These are best illustrated by considering an alignment of three sequences—A, B, and C. Imagine including several sequences very similar to A in the multiple alignment. If all pairwise alignments are given equal weight, then the many pairs similar to A–B and A–C will outvote the single B–C pair. Sequence A will essentially dictate the multiple alignment simply because there are several copies of it in the data. Since most any set of related DNA or protein sequences will contain some sequences more closely related to one another than to the rest, this problem remains even if extra copies of virtually identical sequences are removed.

The basic problem with the SP measure is that while all pairwise alignments are treated equally, some of these alignments are highly correlated; a way is needed to discount redundant information. By weighting the pairwise alignments this problem can be circumvented (32, 33). The MSA program implements either of the two methods for assigning pair weights proposed by Altschul *et al.* (32). Both methods require knowledge of an evolutionary tree relating the sequences to be aligned; MSA estimates this tree by the neighbor joining method of Saitou and Nei (34). The user may choose to use either a weighted or unweighted SP measure of multiple alignment cost.

formation is available, a different approach is possible in which one essentially superimposes the α carbon backbones of the proteins of interest. Greer (35) aligned three serine proteases in this manner—chymotrypsin, trypsin, and elastase—and found a number of positions in which all three aligned α carbons were within 1 Å of each other. Because structural homology is often evident when sequence homology is undetectable (36), we shall use this structural alignment of the chymotrypsin, trypsin, and elastase sequences; 149 of 161 positions are in complete agreement with the structural alignment of Greer; adding two additional serine proteases improves the agreement with the structural alignment to 155 consistent positions. In the latter case, all but one of the discrepancies involve residues whose side chain positions differ markedly and thus the evolutionary and functional significance of these details of the structural ali-

Implementation

MSA is written in the C programming language and has been tested on several machines using the UNIX system V operating system. It should run with little or no modification on any computer with a standard C compiler. Memory and

5. Wilbur, W. J. & Lipman, D. J. (1983) *Proc. Natl. Acad. Sci. USA* **80**, 726–730.
6. Lipman, D. J. & Pearson, W. R. (1985) *Science* **227**, 1435–1441.
7. Fitch, W. M. (1970) *J. Mol. Biol.* **49**, 1–14.
8. Bacon, D. J. & Anderson, W. F. (1986) *J. Mol. Biol.* **191**,