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 biologically significant similarities may only be detected by aligning a set of sequences (7, 8). Likewise, patterns or motifs common to a set of functionally related proteins may only be apparent from analysis of a multiple alignment of these sequences (9).

To align a pair of sequences, one must have a notion of what makes one possible alignment better than another: a measure of the quality of an alignment. Although there are many programs available for pairwise sequence alignment, the most widely accepted tools use variations of the dynamic programming method (10–13). These methods use an explicit measure of alignment quality, consisting of defined costs for aligned pairs of residues, or residues with gaps, and use an algorithm for finding an alignment with minimum total cost. Extending these methods to multiple sequences poses a number of problems, among which are how to measure the cost of a multiple alignment and how to choose gap costs consistent with the measure chosen (14).

The biggest obstacle to using dynamic programming for multiple sequence alignment, however, has been the computational requirements of the method; those tools that do use dynamic programming have been limited to aligning no more than three sequences (15, 16). Given these difficulties, most alternative multiple alignment programs use heuristics or minimize alignment costs that are not clearly tied to models of molecular evolution (17–22). As such, they lack an explicit overall measure of alignment quality. Recently, however, methods have been proposed to greatly reduce the computational demands of dynamic programming applied to multiple sequence alignment (23, 24). We describe the design and application of a tool for multiple sequence alignment that implements these methods.

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

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 (14, 23, 24).

Basic Strategy of the Carrillo–Lipman Algorithm

The basic dynamic programming algorithm for aligning two sequences finds an optimal (minimal cost) path through a rectangular path graph or lattice (11, 12). There is a one-to-one correspondence between alignments and paths in the path graph. Each edge of a path corresponds to the alignment of two letters or of a letter in one sequence with a null (missing element) in the other. The algorithm involves a fixed number of operations for each cell of the lattice, so that its time complexity is proportional to the product of the lengths of the two sequences.

The application of dynamic programming to the alignment of n sequences involves a fixed number of operations for each cell of an n-dimensional lattice (13–16). The number of cells is the product of the lengths of the sequences to be aligned. When these sequences are the length of an average protein ($\approx$200), examining every cell of such a lattice is impractical for $n > 3$.

Recently, however, Carrillo and Lipman showed that one can drastically reduce the number of cells examined while still guaranteeing the discovery of an optimal alignment (23). Their central idea was that every multiple alignment imposes a pairwise alignment on each pair of sequences. Treating an n-sequence alignment as a path through a lattice in n space, this imposed pairwise alignment can be viewed simply as a projected path in a standard two-dimensional path graph. It is possible to calculate an upper bound for the cost of the projection of an optimal multiple alignment onto a given pair of sequences (23). In the corresponding two-dimensional path graph, this upper bound limits the points through which the

†Present address: National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Building 38A, 8th Floor, Bethesda, MD 20894.

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 cost greater than the corresponding upper bound. When this happens, the specified upper bounds were not great enough to encompass all optimal alignments. Increasing the upper bounds that have been exceeded and rerunning MSA frequently produces an improved alignment. Once the alignment found satisfies the specified constraints, increasing the bounds further rarely leads to any improvement; note that such an alignment is optimal given these constraints. In the extreme case, rigorous upper bounds can be used, effectively performing an unconstrained minimization and therefore guaranteeing an optimal alignment. Using rigorous bounds, the practical limit of MSA is four or five sequences.

Gap Costs for Multiple Alignments

Generally, to find biologically reasonable pairwise alignments, costs must be charged for gaps (runs of nulls) as well as for aligning individual elements with nulls (25). Algorithms that charge a fixed cost for each gap are widely used and have been studied extensively (25, 28–30). It is not trivial to extend these gap costs to multiple alignments and a variety of methods have been proposed (14–16, 31). The most natural method is to define gap costs by using the same rationale used to define substitution costs. For SP alignments, these natural gap costs are equal to the sum of the gap costs in all the imposed pairwise alignments. Altschul (14) has shown that 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.

Forcing Special Alignment Positions

Sometimes, prior to the construction of a multiple alignment, information will be available about the correspondence of specific residues in several or all of the sequences—e.g., active site residues. The MSA program permits the user to take advantage of this information. Specifically, any residues from two or more of the sequences may be forced into alignment, so long as the forced alignment positions are mutually consistent. The program operates as before, except that all pairwise and multiple alignments considered are subject to the imposed constraints.

We have found that occasionally with even four or five sequences, the optimal pairwise alignments are so inconsistent that MSA requires unacceptable amounts of time on personal work stations. If certain positions can be fixed then these problems are usually rendered tractable. Alternatively, by splicing out regions of relative certainty, effectively dividing the alignment problem into several smaller ones, the range of the tool can be considerably extended.

Example

We have described above the measure used to evaluate the quality of an alignment and the algorithm used to compute an optimal alignment using this measure. When structural in-

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 to evaluate our sequence-based approach. In Fig. 1 is an 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 alignment is unclear.

A

```
*******   *********        ******************           ***         ******
IVNGEEAvpgSWPWQVSLQDktgF---HFCGGSLINENWVVTAAHCgvttSDVVVA-GEFdqgsssekiQKLKIA
IVGGYTCganTVPYQVSLN--SgY---HFCGGSLINSQWVVSAAHCyksGIQVRL--GEDninvvegneQFISAS
VVGGTEAqrnSWPSQISLQYrsgsswAHTCGGTLIRQNWVMTAAHCvdrelTFRVVVGEHnlnqnngteQYVGVQ

**********       ************* ********         ************       ********
KVFKNSKYNSltinn--DITLLKLSTAASFSqTVSAVCLPsasddfaagTTCVTTGWGLTRytnantPDRLQQAS
KSIVHPSYNSntlnn--DIMLIKLKSAASLNsRVASISLPtsca--sagTQCLISGWGNTKssgtsyPDVLKCLK
KIVVHPYWNTddvaagyDIALLRLAQSVTLNsYVQLGVLPragtilannSPCYITGWGLTR-tngqlAQTLQQAY

******                 *****      *************       ********          ****
LPLLSNtnckk--ywgtkikdaMICAG-asgV-SSCMGDSGGPLVCKngaWTLVGIVSWGss-tcs-tstPGVY
APILSDsscks--aypgqitsnMFCAGyleggKDSCQGDSGGPVVC--SGK--LQGIVSWGs--gcaqknkPGVY
LPTVDYaicssssywgstvknsMVCAG-gngvRSGCQGDSGGPLHCLvngqYAVHGVTSFVsrlgcnvtrkPTVF

*****************
ARVTALVNWVQQTLAAN
TKVCNYVSWIKQTIASN
TRVSAYISWINNVIASN
```

B

```
*******   *********        ******************           ******         ***
iiggvesiphsrpymahldivtekglrvicggflisrqfvltaahckgr-eit----vilgahdvrkrestqqki
ivggtnsswgewpwqvslqvkltaqr-hlcggslighqwvltaahcfdglplqdvwriysgilnlsditkdtpfs
IVNGEEAvpgSWPWQVSLQDk--tgF-HFCGGSLINENWVVTAAHCgvt-tSD---VVVAGEFdqgsssekiQKL
IVGGYTCganTVPYQVSLN----SgY-HFCGGSLINSQWVVSAAHCyks-GIQ----VRLGEDninvvegneQFI
VVGGTEAqrnSWPSQISLQYrsgsswAHTCGGTLIRQNWVMTAAHCvdr-elT--FRVVVGEHnlnqnngteQYV

*************       ************* ********         ************       ****
kvekqiihesyns--vpnlhdimllklekkveltpavnvvplpspsdfihpgamcwaagwgktg-vrdpt-sytl
qikeiiihqnykv--segnhdialiklqaplnytefqkpiclpskgdtstiytncwvtgwgfsk-ekgei-qnil
KIAKVFKNSKYNS--ltinnDITLLKLSTAASFSqTVSAVCLPsasddfaagTTCVTTGWGLTRytnant-PDRL
SASKSIVHPSYNS--ntlnnDIMLIKLKSAASLNsRVASISLPtsca--sagTQCLISGWGNTK-ssgtsyPDVL
GVQKIVVHPYWNTddvaagyDIALLRLAQSVTLNsYVQLGVLPragtilannSPCYITGWGLTR-tngql-AQTL

**********                 *****     **************      *********
revelrimdekacvdyryyeykfq---vcvgspttlraafmgdsggpllc--ag--vahgivsy----ghpdakp
qkvniplvtneecqk-ryqdykitqrmvcagykeggkdackgdsggplvckhngmwrlvgitswge--gcarreq
QQASLPLLSNtnck--kywgtkikdaMICAG--asgVSSCMGDSGGPLVCKngaWTLVGIVSWGs--stcstst
KCLKAPILSDsscks-aypg-qitsnMFCAGyleggKDSCQGDSGGPVVC--SG--KLQGIVSWGs--gcaqknk
QQAYLPTVDYaicssssywgstvknsMVCAGg-ngvRSGCQGDSGGPLHCLvngqYAVHGVTSFVsrlgcnvtrk

**********************
paiftrvstyvptinavin--
pgvytkvaeymdwilektqss
PGVYARVTALVNWVQQTLAAN
PGVYTKVCNYVSWIKQTIASN
PTVFTRVSAYISWINNVIASN
```

FIG. 1. (*A*) Alignment of bovine chymotrypsin (top line), bovine trypsin, and pig elastase. *, Residues in agreement with the structural alignment; underlining, residues not aligned in complete agreement with the structural alignment; lowercase letters, residues not aligned structurally. (*B*) Alignment of rat mast cell proteinase II (top line), human plasma kallikrein, bovine chymotrypsin, bovine trypsin, and pig elastase. *, Residues in agreement with the structural alignment; underlining, residues not aligned in complete agreement with the structural alignment; lowercase letters, residues not aligned structurally. The single-letter code for amino acids is used.

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 computational requirements are a function of the number of sequences, their length, and the size of the upper bounds for pairwise costs. Aligning five sequences, each from different families in the globin superfamily, on a commonly available 32-bit personal computer, took <2 min of computation time and required <1.3 megabytes of memory. The program is available from the authors on request.

Conclusion

The primary difference between MSA and previous multiple alignment programs is its capability to align more than three sequences using an explicitly defined measure of overall alignment quality. The default measure used by MSA considers some replacements more costly than others and penalizes for gaps in a manner reflecting the fact that a single mutational event can insert or delete several residues (14, 25). With pairwise alignment costs so defined, MSA computes a multiple alignment that minimizes the sum of the pairwise costs, weighting the pairs using information derived from an evolutionary tree.

This approach is most effective in aligning sequences that share a global, but perhaps quite distant, relationship. Other methods may be more appropriate for the analysis of the statistical significance of sequence similarities (8), detection of sequence motifs or consensus sequences (3, 9, 22), or aligning large numbers of sequences (17–21). Although MSA will not always produce alignments in such good agreement with structural superpositions as seen here, we believe it can be a powerful sequence analysis tool for molecular biologists.

The authors acknowledge contributions to this project from G. Myers and L. Fitzpatrick and suggestions on the manuscript from M. Boguski.

1. Dayhoff, M. O., ed. (1978) *Atlas of Protein Sequence and Structure* (Natl. Biomed. Res. Found., Washington, DC), Vol. 5, Suppl. 3.
2. Trifonov, E. N. & Bolshoi, G. (1983) *J. Mol. Biol.* **169,** 1–13.
3. Galas, D. J., Eggert, M. & Waterman, M. S. (1985) *J. Mol. Biol.* **186,** 117–128.
4. Wu, T. T. & Kabat, E. A. (1970) *J. Exp. Med.* **132,** 211–250.
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,** 153–161.
9. Bashford, D., Chothia, C. & Lesk, A. M. (1987) *J. Mol. Biol.* **196,** 199–216.
10. Waterman, M. S. (1984) *Bull. Math. Biol.* **46,** 473–500.
11. Needleman, S. B. & Wunsch, C. D. (1970) *J. Mol. Biol.* **48,** 443–453.
12. Sellers, P. H. (1974) *SIAM J. Appl. Math.* **26,** 787–793.
13. Sankoff, D. & Kruskal, J. B., eds. (1983) *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison* (Addison-Wesley, Reading, MA).
14. Altschul, S. F. (1989) *J. Theor. Biol.*, in press.
15. Murata, M., Richardson, J. S. & Sussman, J. L. (1985) *Proc. Natl. Acad. Sci. USA* **82,** 7657–7661.
16. Gotoh, O. (1986) *J. Theor. Biol.* **121,** 327–337.
17. Waterman, M. S. & Perlwitz, M. D. (1984) *Bull. Math. Biol.* **46,** 567–577.
18. Johnson, M. S. & Doolittle, R. F. (1986) *J. Mol. Evol.* **23,** 267–278.
19. Sobel, E. & Martinez, H. (1986) *Nucleic Acids Res.* **14,** 363–374.
20. Feng, D. & Doolittle, R. F. (1987) *J. Mol. Evol.* **25,** 351–360.
21. Taylor, W. R. (1987) *CABIOS* **3,** 81–87.
22. Taylor, W. R. (1986) *J. Mol. Biol.* **188,** 233–258.
23. Carrillo, H. & Lipman, D. (1988) *SIAM J. Appl. Math.* **48,** 1073–1082.
24. Altschul, S. F. & Lipman, D. J. (1989) *SIAM J. Appl. Math.* **49,** 197–209.
25. Fitch, W. M. & Smith, T. F. (1983) *Proc. Natl. Acad. Sci. USA* **80,** 1382–1386.
26. Sankoff, D. (1975) *SIAM J. Appl. Math.* **28,** 35–42.
27. Sankoff, D. & Cedergren, R. J. (1983) in *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, eds. Sankoff, D. & Kruskal, J. B. (Addison-Wesley, Reading, MA), pp. 253–263.
28. Gotoh, O. (1982) *J. Mol. Biol.* **162,** 705–708.
29. Altschul, S. F. & Erickson, B. W. (1986) *Bull. Math. Biol.* **48,** 603–616.
30. Myers, E. W. & Miller, W. (1988) *CABIOS* **4,** 11–17.
31. Fredman, M. L. (1984) *Bull. Math. Biol.* **46,** 553–566.
32. Altschul, S. F., Carroll, R. J. & Lipman, D. J. (1989) *J. Mol. Biol.* **208,** in press.
33. Altschul, S. F. (1989) *SIAM J. Discrete Math.* **2,** in press.
34. Saitou, N. & Nei, M. (1987) *Mol. Biol. Evol.* **4,** 406–425.
35. Greer, J. (1981) *J. Mol. Biol.* **153,** 1027–1042.
36. Bajaj, M. & Blundell, T. (1984) *Annu. Rev. Biophys. Bioeng.* **13,** 453–492.