Nucleic Acids Research, 1994, Vol. 22, No. 22 4673–4680

CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice

Julie D.Thompson, Desmond G.Higgins+ and Toby J.Gibson*
European Molecular Biology Laboratory, Postfach 102209, Meyerhofstrasse 1, D-69012 Heidelberg, Germany

Received July 12, 1994; Revised and Accepted September 23, 1994

ABSTRACT

The sensitivity of the commonly used progressive multiple sequence alignment method has been greatly improved for the alignment of divergent protein sequences. Firstly, individual weights are assigned to each sequence in a partial alignment in order to down-weight near-duplicate sequences and up-weight the most divergent ones. Secondly, amino acid substitution matrices are varied at different alignment stages according to the divergence of the sequences to be aligned. Thirdly, residue-specific gap penalties and locally reduced gap penalties in hydrophilic regions encourage new gaps in potential loop regions rather than regular secondary structure. Fourthly, positions in early alignments where gaps have been opened

practical. The new methods are made available in a program called CLUSTAL W, which is freely available and portable to a wide variety of computers and operating systems.

In order to align just two sequences, it is standard practice to use dynamic programming (2). This guarantees a mathematically optimal alignment, given a table of scores for matches and mismatches between all amino acids or nucleotides [e.g. the PAM250 matrix (3) or BLOSUM62 matrix (4)] and penalties for insertions or deletions of different lengths. Attempts at generalising dynamic programming to multiple alignments are limited to small numbers of short sequences (5). For much more than eight or so proteins of average length, the problem is uncomputable given current computer power. Therefore, all of

alignments between closely related sequences is much more penalty after each residue. Short stretches of hydrophilic residues

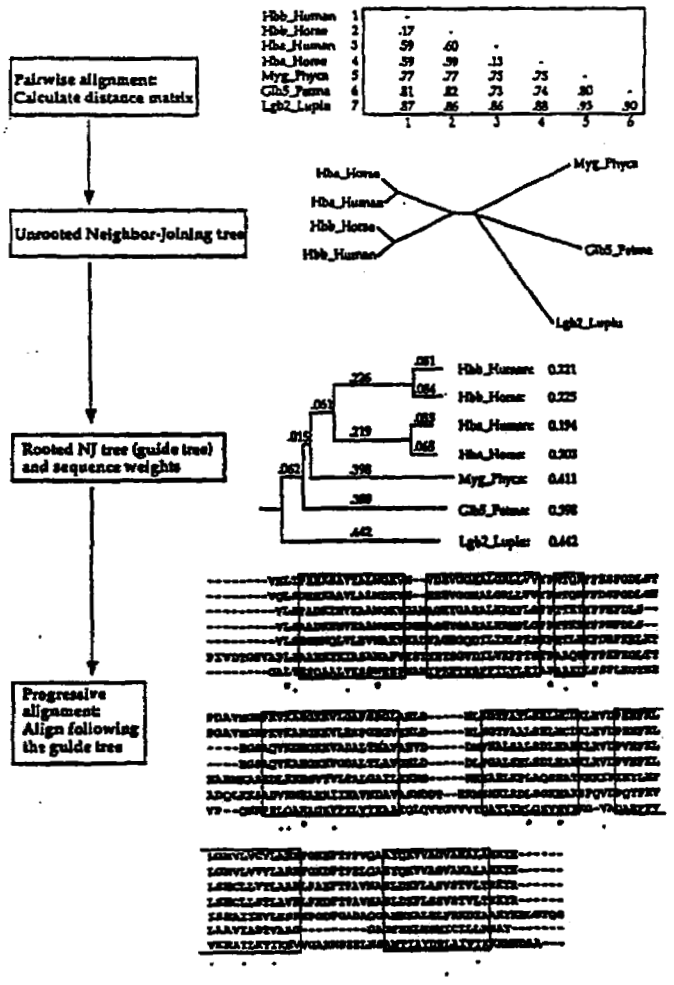

Figure 1. The basic progressive alignment procedure, illustrated using a set of 7 globins of known tertiary structure. The sequence names are from Swiss Prot (38): Hba_Horse: horse α-globin; Hba_Human: human α-globin; Hbb_Horse: horse β-globin; Hbb_Human: human β-globin; Myg_Phyca: sperm whale myoglobin; Glb5_Petma: lamprey cyanohaemoglobin; Lgb2_Luplu: lupin leghaemoglobin. In the distance matrix, the mean number of differences per residue is given. The unrooted tree shows all branch lengths drawn to scale. In the rooted tree, all branch lengths (mean number of differences per residue along each branch) are given as well as weights for each sequence. In the multiple alignment, the approximate positions of the 7 α-helices common to all 7 proteins are shown. This alignment was derived using CLUSTAL W with default parameters and the PAM (3) series of weight matrices.

large numbers of sequences to be aligned, even on a microcomputer. The scores are calculated as the number of *k*-tuple matches (runs of identical residues, typically 1 or 2 long for proteins or 2–4 long for nucleotide sequences) in the best alignment between two sequences minus a fixed penalty for every gap. We now offer a choice between this method and the slower but more accurate scores from full dynamic programming alignments using two gap penalties (for opening or extending gaps) and a full amino acid weight matrix. These scores are calculated as the number of identities in the best alignment divided by the number of residues compared (gap positions are excluded). Both of these scores are initially calculated as per cent identity

In Figure 1 we give the 7×7 distance matrix between the 7 globin sequences calculated using the full dynamic programming method.

The guide tree

The trees used to guide the final multiple alignment process are calculated from the distance matrix of step 1 using the Neighbour-Joining method (21). This produces unrooted trees with branch lengths proportional to estimated divergence along each branch. The root is placed by a 'mid-point' method (15) at a position where the means of the branch lengths on either side of the root are equal. These trees are also used to derive a weight for each sequence (15). The weights are dependent upon the distance from the root of the tree but sequences which have a common branch with other sequences share the weight derived from the shared branch. In the example in Figure 1, the leghaemoglobin (Lgb2_Luplu) gets a weight of 0.442, which is equal to the length of the branch from the root to it. The human β-globin (Hbb_Human) gets a weight consisting of the length of the branch leading to it that is not shared with any other sequences (0.081) plus half the length of the branch shared with the horse β-globin (0.226/2) plus one quarter the length of the branch shared by all four haemoglobins (0.061/4) plus one fifth the branch shared between the haemoglobins and myoglobin (0.015/5) plus one sixth the branch leading to all the vertebrate globins (0.062). This sums to a total of 0.221. In contrast, in the normal progressive alignment algorithm, all sequences would be equally weighted. The rooted tree with branch lengths and sequence weights for the 7 globins is given in Figure 1.

Progressive alignment

The basic procedure at this stage is to use a series of pairwise alignments to align larger and larger groups of sequences, following the branching order in the guide tree. You proceed from the tips of the rooted tree towards the root. In the globin example in Figure 1 you align the sequences in the following order: human vs. horse β-globin; human vs. horse α-globin; the 2 α-globins vs. the 2 β-globins; the myoglobin vs. the haemoglobins; the cyanohaemoglobin vs. the haemoglobins plus myoglobin; the leghaemoglobin vs. all the rest. At each stage a full dynamic programming (26,27) algorithm is used with a residue weight matrix and penalties for opening and extending gaps. Each step consists of aligning two existing alignments or sequences. Gaps that are present in older alignments remain fixed. In the basic algorithm, new gaps that are introduced at each stage get full gap opening and extension penalties, even if they are introduced inside old gap positions (see the section on gap penalties below for modifications to this rule). In order to calculate the score between a position from one sequence or alignment and one from another, the average of all the pairwise weight matrix scores from the amino acids in the two sets of sequences is used, i.e. if you align 2 alignments with 2 and 4 sequences respectively, the score at each position is the average of 8 (2×4) comparisons. This is illustrated in Figure 2. If either set of sequences contains one or more gaps in one of the positions being considered, each gap versus a residue is scored as zero. The default amino acid weight matrices we use are rescored to have only positive values. Therefore, this treatment of gaps treats

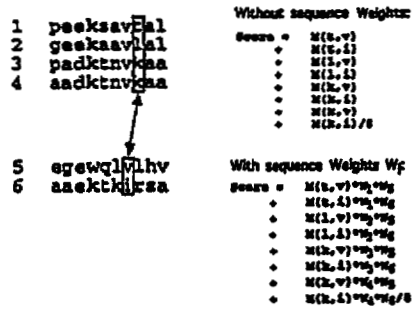

Figure 2. The scoring scheme for comparing two positions from two alignments. Two sections of alignment with 4 and 2 sequences respectively are shown. The score of the position with amino acids T,L,K,K versus the position with amino acids V and I is given with and without sequence weights. $M(X,Y)$ is the weight matrix entry for amino acid X versus amino acid Y. W_n is the weight for sequence n.

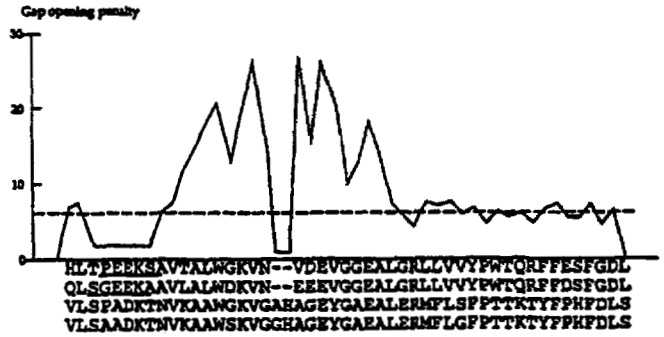

Figure 3. The variation in local gap opening penalty is plotted for a section of alignment. The inital gap opening penalty is indicated by a dotted line. Two hydrophilic stretches are underlined. The lowest penalties correspond to the ends of the alignment, the hydrophilic stretches and the two positions with gaps. The highest values are within 8 residues of the two gap positions. The rest of the variation is caused by the residue specific gap penalties (12).

multiplied by the weights from the 2 sequences, as illustrated in Figure 2.

Improvements to progressive alignment

All of the remaining modifications apply only to the final progressive alignment stage. Sequence weighting is relatively straightforward and is already widely used in profile searches (15,16). The treatment of gap penalties is more complicated

Dependence on the weight matrix. It has been shown (16,28) that varying the gap penalties used with different weight matrices can improve the accuracy of sequence alignments. Here, we use the average score for two mismatched residues (i.e. off-diagonal values in the matrix) as a scaling factor for the GOP.

[redacted]

The local gap penalty modification rules are applied in a hierarchical manner. The exact details of each rule are given below. Firstly, if there is a gap at a position, the gap opening and gap extension penalties are lowered; the other rules do not apply. This makes gaps more likely at positions where there are already gaps. If there is no gap at a position, then the gap opening penalty is increased if the position is within 8 residues of an existing gap. This discourages gaps that are too close together. Finally, at any position within a run of hydrophilic residues, the penalty is decreased. These runs usually indicate loop regions in protein structures. If there is no run of hydrophilic residues, the penalty is modified using a table of residue-specific gap propensities (12). These propensities were derived by counting the frequency of each residue at either end of gaps in alignments of proteins of known structure. An illustration of the application

Divergent sequences

The most divergent sequences (most different on average from all of the other sequences) are usually the most difficult to align correctly. It is sometimes better to delay the incorporation of these sequences until all of the more easily aligned sequences are merged first. This may give a better chance of correctly placing the gaps and matching weakly conserved positions against the rest of the sequences. A choice is offered to set a cut-off (default is 40% identity or less with any other sequence) that will delay the alignment of the divergent sequences until all of the rest have been aligned.

Software and algorithms

Dynamic programming

The most demanding part of the multiple alignment strategy, in

hardware/software combinations: Decstation/Ultrix, Vax or ALPHA/VMS, Silicon Graphics/IRIX. The source code and documentation are available by E-mail from the EMBL file server (send the words HELP and HELP SOFTWARE on two lines to the internet address: Netserv@EMBL-Heidelberg.DE) or by anonymous FTP from FTP.EMBL-Heidelberg.DE. Queries may be addressed by E-mail to Des.Higgins@EBI.AC.UK or Gibson@EMBL-Heidelberg.DE.

RESULTS AND DISCUSSION

Alignment of SH3 domains

The ~60 residue SH3 domain was chosen to illustrate the performance of CLUSTAL W, as there is a reference manual alignment (23) and the fold is known (24). SH3 domains, with a minimum similarity below 12% identity, are poorly aligned by progressive alignment programs such as CLUSTAL V and PILEUP: neither program can generate the correct blocks corresponding to the secondary structure elements.

Figure 4 shows an alignment generated by CLUSTAL W of the example set of SH3 domains. The alignment was generated in two steps. After progressive alignment, five blocks were produced, corresponding to structural elements, with gaps inserted exclusively in the known loop regions. The β-strands in blocks 1, 4 and 5 were all correctly superposed. However, four sequences in block 2 and one sequence in block 3 were misaligned by 1–2 residues (underlined in Figure 4). A second progressive alignment of the aligned sequences, including the gaps, improved this alignment: A single misaligned sequence, H__P55, remains in block 2 (boxed in Figure 4), while block 3 is now completely aligned. This alignment corrects several errors (e.g. P85A, P85B and FUS1) in the manual alignment (23).

The SH3 alignment illustrates several features of CLUSTAL W usage. Firstly, in a practical application involving divergent sequences, the initial progressive alignment is likely to be a good but not perfect approximation to the correct alignment. The alignment quality can be improved in a number of ways. If the block structure of the alignment appears to be correct, realignment of the alignment will usually improve most of the misaligned blocks: the existing gaps allow the blocks to 'float' cheaply to a locally optimal position without disturbing the rest of the alignment. Remaining sequences which are doubtfully aligned can then be individually tested by profile alignment to the remainder: the misaligned H__P55 SH3 domain can be correctly aligned by profile (with GOP ≤8). The indel regions in the final alignment can then be manually cleaned up: usually the exact alignment in the loop regions is not determinable, and may have no meaning in structural terms. It is then desirable to have a single gap per structural loop. CLUSTAL W achieved this for two of the four SH3 loop regions (Figure 4).

If the block structure of the alignment appears suspect, greater intervention by the user may be required. The most divergent sequences, especially if they have large insertions (which can be discerned with the aid of dot matrix plots), should be left out of the progressive alignment. If there are sets of closely related sequences that are deeply diverged from other sets, these can be separately aligned and then merged by profile alignment. Incorrectly determined sequences, containing frameshifts, can also confound regions of an alignment: these can be hard to detect but sometimes they have been grouped within the excluded divergent sequences: then they may be revealed when they are individually compared to the alignment as having apparently nonsense segments with respect to the other sequences.

Table 1. Pascarella and Argos residue specific gap modification factors

A	1.13	M	1.29
C	1.13	N	0.63
D	0.96	P	0.74
E	1.31	Q	1.07
F	1.20	R	0.72
G	0.61	S	0.76
H	1.00	T	0.89
I	1.32	V	1.25
K	0.96	Y	1.00
L	1.21	W	1.23

The values are normalised around a mean value of 1.0 for H. The lower the value, the greater the chance of having an adjacent gap. These are derived from the original table of relative frequencies of gaps adjacent to each residue (12) by subtraction from 2.0.

Finding the best alignment

In cases where all of the sequences in a data set are very similar (e.g. no pair less than 35% identical), CLUSTAL W will find an alignment which is difficult to improve by eye. In this sense, the alignment is optimal with regard to the alternative of manual alignment. Mathematically, this is vague and can only be put on a more systematic footing by finding an objective function (a measure of multiple alignment quality) that exactly mirrors the information used by an 'expert' to evaluate an alignment. Nonetheless, if an alignment is impossible to improve by eye, then the program has achieved a very useful result.

In more difficult cases, as more divergent sequences are included, it becomes increasingly difficult to find good alignments and to evaluate them. What we find with CLUSTAL W is that the basic block-like structure of the alignment (corresponding to the major secondary structure elements) is usually recovered, with some of the most divergent sequences misaligned in small regions. This is a very useful starting point for manual refinement, as it helps define the major blocks of similarity. The problem sequences can be removed from the analysis and realigned to the rest of the sequences automatically or with different parameter settings. An examination of the tree used to guide the alignment will usually show which sequences will be most unreliably placed (those that branch off closest to the root and/or those that align to other single sequences at a very low level of sequence identity rather than align to a group of prealigned sequences). Finally, one can simply iterate the multiple alignment process by feeding an output alignment back into CLUSTAL W and repeating the multiple alignment process (using the same or different parameters). The SH3 domain alignment in Figure 4 was derived in this way by 2 passes using default parameters. In the second pass, the local gap penalties are dominated by the placement of the initial major gap positions. The alignment will either remain unchanged or will converge rapidly (after 1 or 2 extra passes) on a better solution. If the placement of the initial gaps is approximately correct but some of the sequences are locally misaligned, this works well.

Comparison with other methods

Recently, several papers have addressed the problem of position-specific parameters for multiple alignment. In one case (35), local gap penalties are increased in α-helical and β-strand regions when

[redacted]

sensitivity of the progressive multiple alignment approach. This is achieved with almost no sacrifice in speed and efficiency.

There are several areas where further improvements in sensitivity and accuracy can be made. Firstly, the residue weight matrices and gap settings can be made more accurate as more and more data accumulate, while matrices for specific sequence types can be derived [e.g. for transmembrane regions (37)]. Secondly, stochastic or iterative optimisation methods can be used to refine initial alignments (7,9,10). CLUSTAL W could be run with several sets of starting parameters and in each case, the alignments refined according to an objective function. The search for a good objective function that takes into account the sequence- and position-specific information used in CLUSTAL W is a key area of research. Finally, the average number of examples of each protein domain or family is growing steadily. It is not only important that programs can cope with the large volumes of data that are being generated, they should be able to exploit the new information to make the alignments more and more accurate. Globally optimal alignments (according to an objective function) may not always be possible, but the problem may be avoided if sufficiently large volumes of data become available. CLUSTAL W is a step in this direction.

ACKNOWLEDGEMENTS

Numerous people have offered advice and suggestions for improvements to earlier versions of the CLUSTAL programs. D.H. wishes to apologise to all of the irate CLUSTAL V users who had to live with the bugs and lack of facilities for getting trees in the New Hampshire format. We wish to specifically thank Jeroen Coppieters who suggested using a series of weight matrices and Steven Henikoff for advice on using the BLOSUM matrices. We are grateful to Rein Aasland, Peer Bork, Ariel Blocker and Bértrand Seraphin for providing challenging alignment problems. T.G. and J.T. thank Kevin Leonard for support and encouragement. Finally, we thank all of the people who have been involved

16. Lüthy, R., Xenarios, I. and Bucher, P. (1994) Protein Sci. 3, 139–146.
17. Higgins, D.G. and Sharp, P.M. (1988) Gene 73, 237–244.
18. Higgins, D.G. and Sharp, P.M. (1989) CABIOS 5, 151–153.
19. Higgins, D.G., Bleasby, A.J. and Fuchs, R. (1992) CABIOS 8, 189–191.
20. Sneath, P.H.A. and Sokal, R.R. (1973) Numerical Taxonomy. W.H. Freeman, San Francisco.
21. Saitou, N. and Nei, M. (1987) Mol. Biol. Evol. 4, 406–425.
22. Bashford, D., Chothia, C. and Lesk, A.M. (1987) J. Mol. Biol. 196, 199–216.
23. Musacchio, A., Gibson, T., Lehto, V.-P. and Saraste, M. (1992). FEBS Lett. 307, 55–61.
24. Musacchio, A., Noble, M., Pauptit, R., Wierenga, R. and Saraste, M. (1992). Nature, 359, 851–855.
25. Bashford, D., Chothia, C. and Lesk, A.M. (1987). J. Mol. Biol. 196, 199–216.
26. Myers, E.W. and Miller, W. (1988). CABIOS 4, 11–17.
27. Thompson, J.D. (1994). CABIOS submitted for publication.
28. Smith, T.F., Waterman, M.S. and Fitch, W.M. (1981) J. Mol. Evol. 18, 38–46.
29. Pearson, W.R. and Lipman, D.J. (1988) Proc. Natl. Acad. Sci. USA. 85, 2444–2448.
30. Devereux, J., Haeberli, P. and Smithies, O. (1984) Nucleic Acids Res. 12, 387–395.
31. Felsenstein, J. (1989) Cladistics 5, 164–166.
32. Kimura, M. (1980) J. Mol. Evol. 16, 111–120.
33. Kimura, M. (1983) The Neutral Theory of Molecular Evolution. Cambridge University Press, Cambridge.
34. Felsenstein, J. (1985) Evolution 39, 783–791.
35. Smith, R.F. and Smith, T.F. (1992) Protein Engng 5, 35–41.
36. Krogh, A., Brown, M., Mian, S., Sjölander, K. and Haussler, D. (1994) J. Mol. Biol. 235–1501–1531.
37. Jones, D.T., Taylor, W.R. and Thornton, J.M. (1994) FEBS Lett. 339, 269–275.
38. Bairoch, A. and Böckmann, B. (1992) Nucleic Acids Res. 20, 2019–2022.
39. Noble, M.E.M., Musacchio, A., Saraste, M., Courtneidge, S.A. and Wierenga, R.K. (1993) EMBO J. 12, 2617–2624.
40. Kabsch, W. and Sander, C. (1983) Biopolymers 22, 2577–2637.