© 1994 Oxford University Press 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 receive locally reduced gap penalties to encourage the opening up of new gaps at these positions. These modifications are incorporated into a new program, CLUSTAL W which is freely available.

INTRODUCTION

The simultaneous alignment of many nucleotide or amino acid sequences is now an essential tool in molecular biology. Multiple alignments are used to find diagnostic patterns to characterise protein families; to detect or demonstrate homology between new sequences and existing families of sequences; to help predict the secondary and tertiary structures of new sequences; to suggest oligonucleotide primers for PCR; as an essential prelude to molecular evolutionary analysis. The rate of appearance of new sequence data is steadily increasing and the development of efficient and accurate automatic methods for multiple alignment is, therefore, of major importance. The majority of automatic multiple alignments are now carried out using the 'progressive' approach of Feng and Doolittle (1). In this paper, we describe a number of improvements to the progressive multiple alignment method which greatly improve the sensitivity without sacrificing any of the speed and efficiency which makes this approach so 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 the methods capable of handling larger problems in practical timescales make use of heuristics. Currently, the most widely used approach is to exploit the fact that homologous sequences are evolutionarily related. One can build up a multiple alignment progressively by a series of pairwise alignments, following the branching order in a phylogenetic tree (1). One first aligns the most closely related sequences, gradually adding in the more distant ones. This approach is sufficiently fast to allow alignments of virtually any size. Further, in simple cases, the quality of the alignments is excellent, as judged by the ability to correctly align corresponding domains from sequences of known secondary or tertiary structure (6). In more difficult cases, the alignments give good starting points for further automatic or manual refinement.

This approach works well when the data set consists of sequences of different degrees of divergence. Pairwise alignment of very closely related sequences can be carried out very accurately. The correct answer may often be obtained using a wide range of parameter values (gap penalties and weight matrix). By the time the most distantly related sequences are aligned, one already has a sample of aligned sequences which gives important information about the variability at each position. The positions of the gaps that were introduced during the early alignments of the closely related sequences are not changed as new sequences are added. This is justified because the placement of gaps in

*To whom correspondence should be addressed

+Present address: European Bioinformatics Institute, Hinxton Hall, Hinxton, Cambridge CB10 1RQ, UK alignments between closely related sequences is much more accurate than between distantly related ones. When all of the sequences are highly divergent (e.g. less than ~25–30% identity between any pair of sequences), this progressive approach becomes much less reliable.

There are two major problems with the progressive approach: the local minimum problem and the choice of alignment parameters. The local minimum problem stems from the 'greedy' nature of the alignment strategy. The algorithm greedily adds sequences together, following the initial tree. There is no guarantee that the global optimal solution, as defined by some overall measure of multiple alignment quality (7,8), or anything close to it, will be found. More specifically, any mistakes (misaligned regions) made early in the alignment process cannot be corrected later as new information from other sequences is added. This problem is frequently thought of as mainly resulting from an incorrect branching order in the initial tree. The initial trees are derived from a matrix of distances between separately aligned pairs of sequences and are much less reliable than trees from complete multiple alignments. In our experience, however, the real problem is caused simply by errors in the initial alignments. Even if the topology of the guide tree is correct, each alignment step in the multiple alignment process may have some percentage of the residues misaligned. This percentage will be very low on average for very closely related sequences but will increase as sequences diverge. It is these misalignments which carry through from the early alignment steps that cause the local minimum problem. The only way to correct this is to use an iterative or stochastic sampling procedure (e.g. 7,9,10). We do not directly address this problem in this paper.

The alignment parameter choice problem is, in our view, at least as serious as the local minimum problem. Stochastic or iterative algorithms will be just as badly affected as progressive ones if the parameters are inappropriate: they will arrive at a false global minimum. Traditionally, one chooses one weight matrix and two gap penalties (one for opening a new gap and one for extending an existing gap) and hope that these will work well over all parts of all the sequences in the data set. When the sequences are all closely related, this works. The first reason is that virtually all residue weight matrices give most weight to identities. When identities dominate an alignment, almost any weight matrix will find approximately the correct solution. With very divergent sequences, however, the scores given to non-identical residues will become critically important; there will be more mismatches than identities. Different weight matrices will be optimal at different evolutionary distances or for different classes of proteins.

The second reason is that the range of gap penalty values that will find the correct or best possible solution can be very broad for highly similar sequences (11). As more and more divergent sequences are used, however, the exact values of the gap penalties become important for success. In each case, there may be a very narrow range of values which will deliver the best alignment. Further, in protein alignments, gaps do not occur randomly (i.e. with equal probability at all positions). They occur far more often between the major secondary structural elements of α-helices and β-strands than within (12).

The major improvements described in this paper attempt to address the alignment parameter choice problem. We dynamically vary the gap penalties in a position- and residue-specific manner. The observed relative frequencies of gaps adjacent to each of the 20 amino acids (12) are used to locally adjust the gap opening penalty after each residue. Short stretches of hydrophilic residues (e.g. 5 or more) usually indicate loop or random coil regions and the gap opening penalties are locally reduced in these stretches. In addition, the locations of the gaps found in the early alignments are also given reduced gap opening penalties. It has been observed in alignments between sequences of known structure that gaps tend not to be closer than roughly eight residues on average (12). We increase the gap opening penalty within eight residues of exising gaps. The two main series of amino acid weight matrices that are used today are the PAM series (3) and the BLOSUM series (4). In each case, there is a range of matrices to choose from. Some matrices are appropriate for aligning very closely related sequences where most weight by far is given to identities, with only the most frequent conservative substitutions receiving high scores. Other matrices work better at greater evolutionary distances where less importance is attached to identities (13). We choose different weight matrices, as the alignment proceeds, depending on the estimated divergence of the sequences to be aligned at each stage.

Sequences are weighted to correct for unequal sampling across all evolutionary distances in the data set (14). This down-weights sequences that are very similar to other sequences in the data set and up-weights the most divergent ones. The weights are calculated directly from the branch lengths in the initial guide tree (15). Sequence weighting has already been shown to be effective in improving the sensitivity of profile searches (15,16). In the original CLUSTAL programs (17–19), the initial guide trees, used to guide the multiple alignment, were calculated using the UPGMA method (20). We now use the Neighbour-Joining method (21) which is more robust against the effects of unequal evolutionary rates in different lineages and which gives better estimates of individual branch lengths. This is useful because it is these branch lengths which are used to derive the sequence weights. We also allow users to choose between fast approximate alignments (22) or full dynamic programming for the distance calculations used to make the guide tree.

The new improvements dramatically improve the sensitivity of the progressive alignment method for difficult alignments involving highly diverged sequences. We show one very demanding test case of over 60 SH3 domains (23) which includes sequence pairs with as little as 12% identity and where there is only one exactly conserved residue across all of the sequences. Using default parameters, we can achieve an alignment that is almost exactly correct, according to available structural information (24). Using the program in a wide variety of situations, we find that it will normally find the correct alignment in all but the most difficult and pathological of cases.

MATERIAL AND METHODS

The basic alignment method

The basic multiple alignment algorithm consists of three main stages: (i) all pairs of sequences are aligned separately in order to calculate a distance matrix giving the divergence of each pair of sequences; (ii) a guide tree is calculated from the distance matrix; (iii) the sequences are progressively aligned according to the branching order in the guide tree. An example using 7 globin sequences of known tertiary structure (25) is given in Figure 1.

The distance matrix/pairwise alignments

In the original CLUSTAL programs, the pairwise distances were calculated using a fast approximate method (22). This allows very

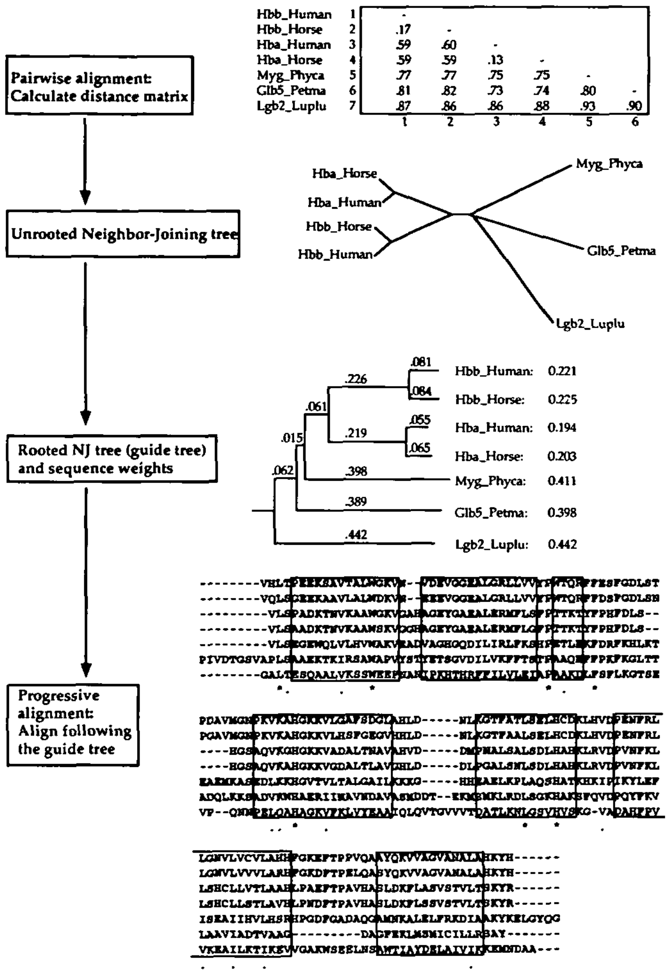

Pairwise alignment: Calculate distance matrix
Unrooted Neighbor-Joining tree
Rooted NJ tree (guide tree) and sequence weights
Progressive alignment: Align following the guide tree
Hbb_Human 1 -
Hbb_Horse 2 .17 -
Hba_Human 3 .59 .60 -
Hba_Horse 4 .59 .59 .13 -
Myg_Phyca 5 .77 .77 .75 .75 -
Glb5_Petma 6 .81 .82 .73 .74 .80 -
Lgb2_Luplu 7 .87 .86 .86 .88 .93 .90
1 2 3 4 5 6
Hba_Horse
Hba_Human
Hbb_Horse
Hbb_Human
Myg_Phyca
Glb5_Petma
Lgb2_Luplu
.081
.084
.226
.055
.065
.219
.061
.015
.062
.398
.389
.442
Hbb_Human: 0.221
Hbb_Horse: 0.225
Hba_Human: 0.194
Hba_Horse: 0.203
Myg_Phyca: 0.411
Glb5_Petma: 0.398
Lgb2_Luplu: 0.442

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 scores and are converted to distances by dividing by 100 and subtracting from 1.0 to give number of differences per site. We do not correct for multiple substitutions in these initial distances. 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 the score of a residue versus a gap as having the worst possible score. When sequences are weighted (see Improvements to progressive alignment, below), each weight matrix value is

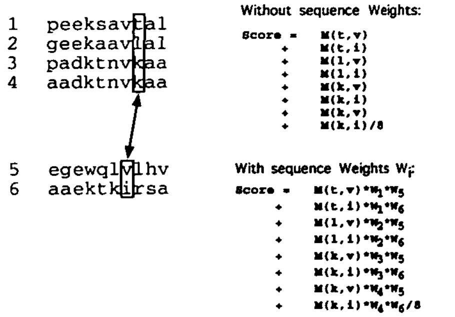


Figure 2. The scoring scheme for comparing two positions from two alignments. Two sections of alignment with 4 and 2 sequences respectively are shown. The score of the position with amino acids T,L,K,K versus the position with amino acids V and I is given with and without sequence weights. $M(X,Y)$ is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

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. Initial gap penalties are calculated depending on the weight matrix, the similarity of the sequences and the length of the sequences. Then, an attempt is made to derive sensible local gap opening penalties at every position in each prealigned group of sequences that will vary as new sequences are added. The use of different weight matrices as the alignment progresses is novel and largely by-passes the problem of initial choice of weight matrix. The final modification allows us to delay the addition of very divergent sequences until the end of the alignment process, when all of the more closely related sequences have already been aligned.

Sequence weighting

Sequence weights are calculated directly from the guide tree. The weights are normalised such that the biggest one is set to 1.0 and the rest are all less than 1.0. Groups of closely related sequences receive lowered weights because they contain much duplicated information. Highly divergent sequences without any close relatives receive high weights. These weights are used as simple multiplication factors for scoring positions from different sequences or prealigned groups of sequences. The method is illustrated in Figure 2. In the globin example in Figure 1, the two α-globins get down-weighted because they are almost duplicate sequences (as do the two β-globins); they receive a combined weight of only slightly more than if a single α-globin was used.

Initial gap penalties

Initially, two gap penalties are used: a gap opening penalty (GOP), which gives the cost of opening a new gap of any length, and a gap extension penalty (GEP), which gives the cost of every item in a gap. Initial values can be set by the user from a menu. The software then automatically attempts to choose appropriate gap penalties for each sequence alignment, depending on the following factors.

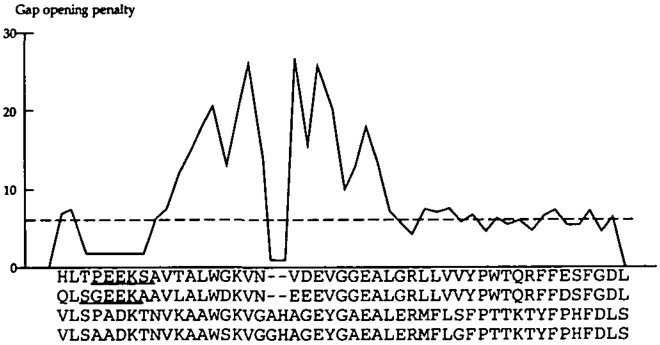


Figure 3. The variation in local gap opening penalty is plotted for a section of alignment. The inital gap opening penalty is indicated by a dotted line. Two hydrophilic stretches are underlined. The lowest penalties correspond to the ends of the alignment, the hydrophilic stretches and the two positions 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).

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.

Dependence on the similarity of the sequences. The per cent identity of the two (groups of) sequences to be aligned is used to increase the GOP for closely related sequences and decrease it for more divergent sequences on a linear scale.

Dependence on the lengths of the sequences. The scores for both true and false sequence alignments grow with the length of the sequences. We use the logarithm of the length of the shorter sequence to increase the GOP with sequence length. Using these three modifications, the initial GOP calculated by the program is:

GOP $\rightarrow$ {GOP + log[min(N,M)]} * (average residue mismatch score) * (per cent identity scaling factor)

where N, M are the lengths of the two sequences.

Dependence on the difference in the lengths of the sequences. The GEP is modified depending on the difference between the lengths of the two sequences to be aligned. If one sequence is much shorter than the other, the GEP is increased to inhibit too many long gaps in the shorter sequence. The initial GEP calculated by the program is:

GEP $\rightarrow$ GEP * [1.0 + $|\log(N/M)|$]

where N, M are the lengths of the two sequences.

Position-specific gap penalties

In most dynamic programming applications, the initial gap opening and extension penalties are applied equally at every position in the sequence, regardless of the location of a gap, except for terminal gaps which are usually allowed at no cost. In CLUSTAL W, before any pair of sequences or prealigned groups of sequences are aligned, we generate a table of gap opening penalties for every position in the two (sets of) sequences. An example is shown in Figure 3. We manipulate the initial gap opening penalty in a position-specific manner, in order to make gaps more or less likely at different positions.

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 of these rules from one part of the globin example in Figure 1 is given in Figure 3.

Lowered gap penalties at existing gaps. If there are already gaps at a position, then the GOP is reduced in proportion to the number of sequences with a gap at this position and the GEP is lowered by a half. The new gap opening penalty is calculated as:

GOP → GOP * 0.3 * (no. of sequences without a gap/no. of sequences).

Increased gap penalties near existing gaps. If a position does not have any gaps but is within 8 residues of an existing gap, the GOP is increased by:

GOP → GOP * {2 + [(8 − distance from gap) * 2]/8}

Reduced gap penalties in hydrophilic stretches. Any run of 5 hydrophilic residues is considered to be a hydrophilic stretch. The residues that are to be considered hydrophilic may be set by the user but are conservatively set to D, E, G, K, N, Q, P, R or S by default. If, at any position, there are no gaps and any of the sequences has such a stretch, the GOP is reduced by one third.

Residue-specific penalties. If there is no hydrophilic stretch and the position does not contain any gaps, then the GOP is multiplied by one of the 20 numbers in Table 1, depending on the residue. If there is a mixture of residues at a position, the multiplication factor is the average of all the contributions from each sequence.

Weight matrices

Two main series of weight matrices are offered to the user: the Dayhoff PAM series (3) and the BLOSUM series (4). The default is the BLOSUM series. In each case, there is a choice of matrix ranging from strict ones, useful for comparing very closely related sequences to very 'soft' ones that are useful for comparing very distantly related sequences. Depending on the distance between the two sequences or groups of sequences to be compared, we switch between 4 different matrices. The distances are measured directly from the guide tree. The ranges of distances and tables used with the PAM series of matrices are: 80−100%:PAM20, 60−80%:PAM60, 40−60%:PAM120, 0−40%:PAM350. The range used with the BLOSUM series is: 80−100%: BLOSUM80, 60−80%:BLOSUM62, 30−60%:BLOSUM45, 0−30%:BLOSUM30.

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 terms of computer processing and memory usage, is the alignment of two (groups of) sequences at each step in the final progressive alignment. To make it possible to align very long sequences (e.g. dynein heavy chains at ~5,000 residues) in a reasonable amount of memory, we use the memory efficient dynamic programming algorithm of Myers and Miller (26). This sacrifices some processing time but makes very large alignments practical in very little memory. One disadvantage of this algorithm is that it does not allow different gap opening and extension penalties at each position. We have modified the algorithm so as to allow this and the details are described in a separate paper (27).

Menus/file formats

Six different sequence input formats are detected automatically and read by the program: EMBL/Swiss Prot, NBRF/PIR, Pearson/FASTA (29), GCG/MSF (30), GDE (Steven Smith, Harvard University Genome Center) and CLUSTAL format alignments. The last three formats allow users to read in complete alignments (e.g. for calculating phylogenetic trees or for addition of new sequences to an existing alignment). Alignment output may be requested in standard CLUSTAL format (self-explanatory blocked alignments) or in formats compatible with the GDE, PHYLIP (31) or GCG (30) packages. The program offers the user the ability to calculate Neighbour-Joining phylogenetic trees from existing alignments with options to correct for multiple hits (32,33) and to estimate confidence levels using a bootstrap resampling procedure (34). The trees may be output in the 'New Hampshire' format that is compatible with the PHYLIP package (31).

Alignment to an alignment

Profile alignment is used to align two existing alignments (either of which may consist of just one sequence) or to add a series of new sequences to an existing alignment. This is useful because one may wish to build up a multiple alignment gradually, choosing different parameters manually or correcting intermediate errors as the alignment proceeds. Often, just a few sequences cause misalignments in the progressive algorithm and these can be removed from the process and then added at the end by profile alignment. A second use is where one has a high quality reference alignment and wishes to keep it fixed while adding new sequences automatically.

Portability/availability

The full source code of the package is provided free to academic users. The program will run on any machine with a full ANSI conforming C compiler. It has been tested on the following 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.

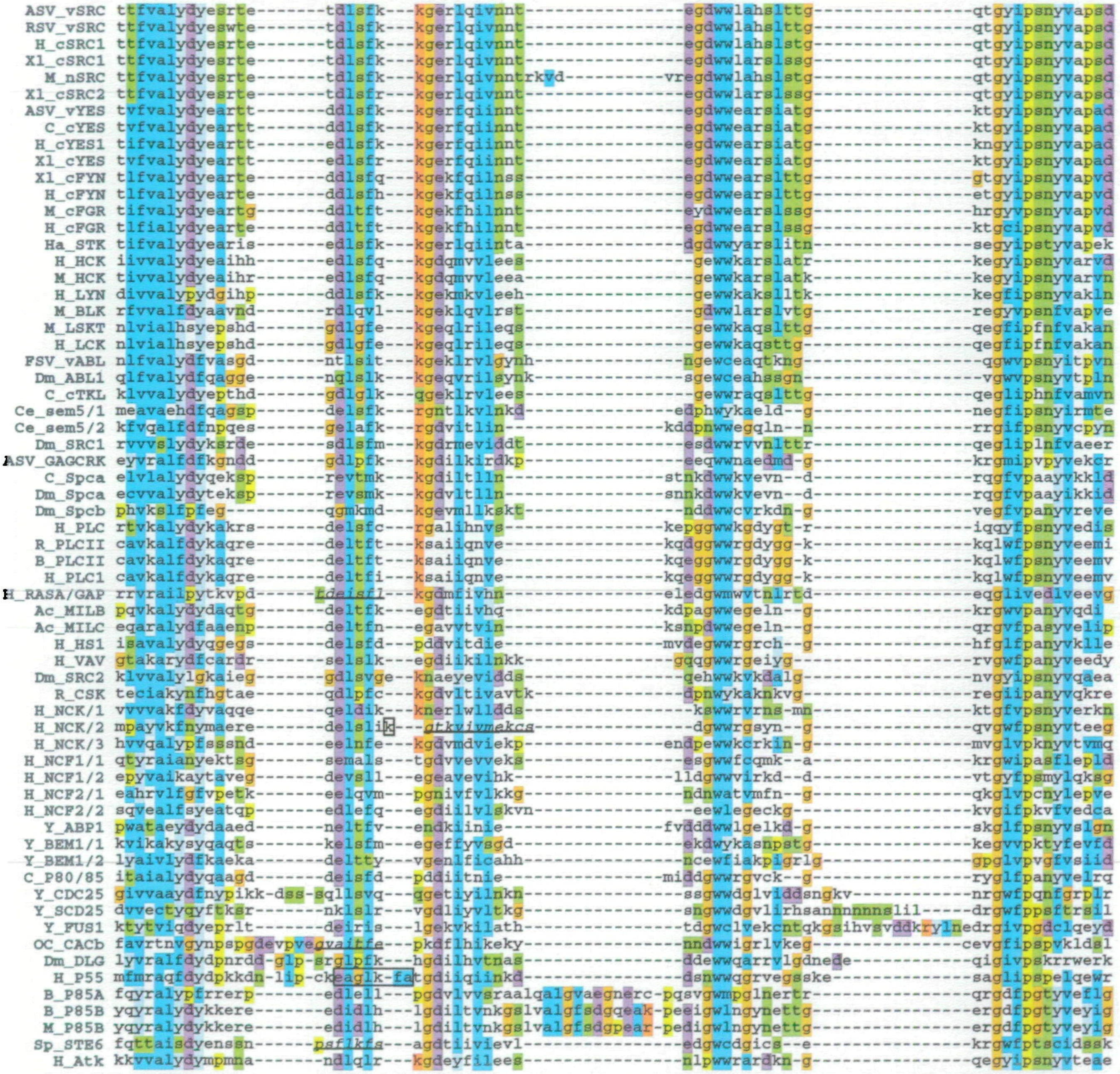
Finding the best alignment

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



```
  ASV_vSRC ttfvalydyesrte-------tdlsfk---kgerlqivnnt----------------egdwwlahslttg----------------qtgyipsnyvapsd
  RSV_vSRC ttfvalydyeswte-------tdlsfk---kgerlqivnnt----------------egdwwlahslttg----------------qtgyipsnyvapsd
   H_cSRC1 ttfvalydyesrte-------tdlsfk---kgerlqivnnt----------------egdwwlahslstg----------------qtgyipsnyvapsd
  Xl_cSRC1 ttfvalydyesrte-------tdlsfk---kgerlqivnnt----------------egdwwlarslssg----------------qtgyipsnyvapsd
    M_nSRC ttfvalydyesrte-------tdlsfk---kgerlqivnntrkvd----------vregdwwlahslstg----------------qtgyipsnyvapsd
  Xl_cSRC2 ttfvalydyesrte-------tdlsfr---kgerlqivnnt----------------egdwwlarslssg----------------qtgyipsnyvapsd
  ASV_vYES tvfvalydyeartt-------ddlsfk---kgerfqiinnt----------------egdwwearsiatg----------------ktgyipsnyvapad
    C_cYES tvfvalydyeartt-------ddlsfk---kgerfqiinnt----------------egdwwearsiatg----------------ktgyipsnyvapad
   H_cYES1 tifvalydyeartt-------edlsfk---kgerfqiinnt----------------egdwwearsiatg----------------kngyipsnyvapad
   Xl_cYES tvfvalydyeartt-------edlsfr---kgerfqiinnt----------------egdwwearsiatg----------------ktgyipsnyvapad
   Xl_cFYN tlfvalydyearte-------ddlsfq---kgekfqilnss----------------egdwwearslttg----------------gtgyipsnyvapvd
    H_cFYN tlfvalydyearte-------ddlsfh---kgekfqilnss----------------egdwwearslttg----------------etgyipsnyvapvd
    M_cFGR tifvalydyeartg-------ddltft---kgekfhilnnt----------------eydwwearslssg----------------hrgyvpsnyvapvd
    H_cFGR tlfialydyearte-------ddltft---kgekfhilnnt----------------egdwwearslssg----------------ktgcipsnyvapvd
    Ha_STK tifvalydyearis-------edlsfk---kgerlqiinta----------------dgdwwyarslitn----------------segyipstyvapek
     H_HCK iivvalydyeaihh-------edlsfq---kgdqmvvlees-----------------gewwkarslatr----------------kegyipsnyvarvd
     M_HCK tivvalydyeaihr-------edlsfq---kgdqmvvleea-----------------gewwkarslatk----------------kegyipsnyvarvn
     H_LYN divvalypydgihp-------ddlsfk---kgekmkvleeh-----------------gewwkakslltk----------------kegfipsnyvakln
     M_BLK rfvvalfdyaavnd-------rdlqvl---kgeklqvlrst-----------------gdwwlarslvtg----------------regyvpsnfvapve
    M_LSKT nlvialhsyepshd-------gdlgfe---kgeqlrileqs-----------------gewwkaqslttg----------------qegfipfnfvakan
     H_LCK nlvialhsyepshd-------gdlgfe---kgeqlrileqs-----------------gewwkaqsttg-----------------qegfipfnfvakan
  FSV_vABL nlfvalydfvasgd-------ntlsit---kgeklrvlgynh---------------ngewceaqtkng------------------qgwvpsnyitpvn
   Dm_ABL1 qlfvalydfqagge-------nqlslk---kgeqvrilsynk---------------sgewceahssgn------------------vgwvpsnyvtpln
    C_cTKL klvvalydyepthd-------gdlglk---qgeklrvlees-----------------gewwraqslttg----------------qegliphnfvamvn
 Ce_sem5/1 meavaehdfqagsp-------delsfk---rgntlkvlnkd---------------edphwykaeld--g----------------negfipsnyirmte
 Ce_sem5/2 kfvqalfdfnpqes-------gelafk---rgdvitlin----------------kddpnwwegqln--n----------------rrgifpsnyvcpyn
   Dm_SRC1 rvvvslydyksrde-------sdlsfm---kgdrmeviddt----------------esdwwrvvnlttr----------------qegliplnfvaeer
ASV_GAGCRK eyvralfdfkgndd-------gdlpfk---kgdilkirdkp----------------eeqwwnaedmd-g----------------krgmipvpyvekcr
    C_Spca elvlalydyqeksp-------revtmk---kgdiltlln----------------stnkdwwkvevn--d----------------rqgfvpaayvkkld
   Dm_Spca ecvvalydyteksp-------revsmk---kgdvltlln----------------snnkdwwkvevn--d----------------rqgfvpaayikkid
   Dm_Spcb phvkslfpfeg----------qgmkmd---kgevmllkskt----------------nddwwcvrkdn-g----------------vegfvpanyvreve
     H_PLC rtvkalydykakrs-------delsfc---rgalihnvs----------------kepggwwkgdygt-r----------------iqqyfpsnyvedis
   R_PLCII cavkalfdykaqre-------deltft---ksaiiqnve----------------kqdggwwrgdygg-k----------------kqlwfpsnyveemi
   B_PLCII cavkalfdykaqre-------deltft---ksaiiqnve----------------kqeggwwrgdygg-k----------------kqlwfpsnyveemv
    H_PLC1 cavkalfdykaqre-------deltfi---ksaiiqnve----------------kqeggwwrgdygg-k----------------kqlwfpsnyveemv
H_RASA/GAP rrvrailpytkvpd------tdeisfl---kgdmfivhn----------------eledgwmwvtnlrtd----------------eqglivedlveevg
   Ac_MILB pqvkalydydaqtg-------deltfk---egdtiivhq----------------kdpagwwegeln--g----------------krgwvpanyvqdi
   Ac_MILC eqaralydfaaenp-------deltfn---egavvtvin----------------ksnpdwwegeln--g----------------qrgvfpasyvelip
     H_HS1 isavalydyqgegs-------delsfd---pddvitdie----------------mvdegwwrgrch--g----------------hfglfpanyvklle
     H_VAV gtakarydfcardr-------selslk---egdiikilnkk---------------gqqgwwrgeiyg------------------rvgwfpanyveedy
   Dm_SRC2 klvvalylgkaieg-------gdlsvge--knaeyevidds----------------qehwwkvkdalg-----------------nvgyipsnyvqaea
     R_CSK teciakynfhgtae-------qdlpfc---kgdvltivavtk---------------dpnwykaknkvg-----------------regiipanyvqkre
   H_NCK/1 vvvvakfdyvaqqe-------qeldik---knerlwlldds-----------------kswwrvrns-mn----------------ktgfvpsnyverkn
   H_NCK/2 mpayvkfnymaere-------delslik---gtkviymekcs----------------dgwwrgsyn--g----------------qvgwfpsnyvteeg
   H_NCK/3 hvvqalypfsssnd-------eelnfe---kgdvmdviekp--------------endpewwkcrkin-g----------------mvglvpknyvtvmq
  H_NCF1/1 qtyraianyektsg-------semals---tgdvvevveks----------------esgwwfcqmk--a----------------krgwipasflepld
  H_NCF1/2 epyvaikaytaveg-------devsll---egeavevihk----------------lldgwwvirkd--d----------------vtgyfpsmylqksg
  H_NCF2/1 eahrvlfgfvpetk-------eelqvm---pgnivfvlkkg----------------ndnwatvmfn--g----------------qkglvpcnylepve
  H_NCF2/2 sqvealfsyeatqp-------edlefq---egdiilvlskvn----------------eewlegeckg------------------kvgifpkvfvedca
    Y_ABP1 pwataeydydaaed-------neltfv---endkiinie----------------fvdddwwlgelkd-g----------------skglfpsnyvslgn
  Y_BEM1/1 kvikakysyqaqts-------kelsfm---egeffyvsgd-----------------ekdwykasnpstg----------------kegvvpktyfevfd
  Y_BEM1/2 lyaivlydfkaeka-------deltty---vgenlficahh----------------ncewfiakpigrlg---------------gpglvpvgfvsiid
  C_P80/85 itaialydyqaagd-------deisfd---pddiitnie----------------middgwwrgvck--g----------------ryglfpanyvelrq
   Y_CDC25 givvaaydfnypikk-dss-sqllsvq---qgetiyilnkn----------------ssgwwdglviddsngkv------------nrgwfpqnfgrplr
   Y_SCD25 dvvectyqyftksr-------nklslr---vgdliyvltkg----------------sngwwdgvlirhsannnnnnslil-----drgwfppsftrsil
    Y_FUS1 ktytviqdyeprlt-------deiris---lgekvkilath----------------tdgwclvekcntqkgsihvsvddkrylnedrgivpgdclqeyd
   OC_CACb favrtnvgynpspgdevpvegvaitfe---pkdflhikeky----------------nndwwigrlvkeg---------------cevgfipspvkldsl
    Dm_DLG lyvralfdydpnrdd-glp-srglpfk---hgdilhvtnas----------------ddewwqarrvlgdnede------------qigivpskrrwerk
     H_P55 mfmraqfdydpkkdn-lip-ckeaglk-fatgdiiqiinkd----------------dsnwwqgrvegsske--------------saglipspelqewr
    B_P85A fqyralypfrrerp-------edlell---pgdvlvvsraalqalgvaegnerc-pqsvgwmpglnertr----------------qrgdfpgtyveflg
    B_P85B yqyralydykkere-------edidlh---lgdiltvnkgslvalgfsdgqeak-peeigwlngynettg----------------ergdfpgtyveyig
    M_P85B yqyralydykkere-------edidlh---lgdiltvnkgslvalgfsdgpear-pedigwlngynettg----------------ergdfpgtyveyig
   Sp_STE6 fqttaisdyenssn------psflkfs---agdtiivievl----------------edgwcdgics--e----------------krgwfptscidssk
     H_Atk kkvvalydympmna-------ndlqlr---kgdeyfilees----------------nlpwwrardkn-g----------------qegyipsnyvteae
```

Figure 4. CLUSTAL W alignment of a set of SH3 domains taken from Musacchio *et al.* (23). Secondary structure assignments for the solved Spectrin (24) and Fyn (39) domains are according to DSSP (40). The alignment was generated in two steps using default parameters. After full multiple alignment, the aligned sequences were realigned. Segments which were correctly aligned in the second pass are underlined. The single misaligned segment in H__P55 and the misaligned residue in H__NCK/2 are boxed. The sequences are coloured to illustrate significant features. All G (orange) and P (yellow) are coloured. Other residues matching a frequent occurrence of a property in a column are coloured: hydrophobic = blue; hydrophobic tendency = light blue; basic = red; acidic = purple; hydrophilic = green; unconserved = white. The alignment figure was prepared with the GDE sequence editor (S.Smith, Harvard University) and COLORMASK (J.Thompson, EMBL).

the 3-D structures of one or more of the sequences are known. In a second case (36), a hidden Markov model was used to estimate position-specific gap penalties and residue substitution weight matrices when large numbers of examples of a protein domain were known. With CLUSTAL W, we attempt to derive the same information purely from the set of sequences to be aligned. Therefore, we can apply the method to any set of sequences. The success of this approach will depend on the number of available sequences and their evolutionary relationships. It will also depend on the decision making process during multiple alignment (e.g. when to change weight matrix) and the accuracy and appropriateness of our parameterisation. In the long term, this can only be evaluated by exhaustive testing of sets of sequences where the correct alignment (or parts of it) are known from structural information. What is clear, however, is that the modifications described here significantly improve the 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 with various CLUSTAL programs over the years, namely Paul Sharp, Rainer Fuchs and Alan Bleasby.

REFERENCES

1. Feng, D.-F. and Doolittle, R.F. (1987) J. Mol. Evol. 25, 351–360.
2. Needleman, S.B. and Wunsch, C.D. (1970) J. Mol. Biol. 48, 443–453.
3. Dayhoff, M.O., Schwartz, R.M. and Orcutt, B.C. (1978) In Atlas of Protein Sequence and Structure, vol. 5, suppl. 3 (Dayhoff, M.O., ed.), pp 345–352. NBRF, Washington.
4. Henikoff, S. and Henikoff, J.G. (1992) Proc. Natl. Acad. Sci. USA 89, 10915–10919.
5. Lipman, D.J., Altschul, S.F. and Kececioglu, J.D. (1989) Proc. Natl. Acad. Sci. USA 86, 4412–4415.
6. Barton, G.J. and Sternberg, M.J.E. (1987) J. Mol. Biol. 198, 327–337.
7. Gotoh, O. (1993) CABIOS 9, 361–370.
8. Altschul, S.F. (1989) J. Theor. Biol. 138, 297–309.
9. Lukashin, A.V., Engelbrecht, J. and Brunak, S. (1992) Nucleic Acids Res. 20, 2511–2516.
10. Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wooton, J.C. (1993) Science 262, 208–214.
11. Vingron, M. and Waterman, M.S. (1993) J. Mol. Biol. 234, 1–12.
12. Pascarella, S. and Argos, P. (1992) J. Mol. Biol. 224, 461–471.
13. Collins, J.F. and Coulson, A.F.W. (1987) In Nucleic Acid and Protein Sequence Analysis, A Practical Approach (Bishop, M.J. and Rawlings, C.J., eds), chapter 13, pp. 323–358.
14. Vingron, M. and Sibbald, P.R. (1993) Proc. Natl. Acad. Sci. USA 90, 8777–8781.
15. Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CABIOS 10, 19–29.
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.