CABIOS

Vol.7 no.4 1991
Pages 479–484

A novel randomized iterative strategy for aligning multiple protein sequences

M.P.Berger and P.J.Munson[1]

Abstract

The rigorous alignment of multiple protein sequences becomes impractical even with a modest number of sequences, since computer memory and time requirements increase as the product of the lengths of the sequences. We have devised a strategy to approach such an optimal alignment, which modifies the intensive computer storage and time requirements of dynamic programming. Our algorithm randomly divides a group of unaligned sequences into two subgroups, between which an optimal alignment is then obtained by a Needleman–Wunsch style of algorithm. Our algorithm uses a matrix with dimensions corresponding to the lengths of the two aligned sequence subgroups. The pairwise alignment process is repeated using different random divisions of the whole group into two subgroups. Compared with the rigorous approach of solving the n-dimensional lattice by dynamic programming, our iterative algorithm results in alignments that match or are close to the optimal solution, on a limited set of test problems. We have implemented this algorithm in a computer program that runs on the IBM PC class of machines, together with a user-friendly environment for interactively selecting sequences or groups of sequences to be aligned either simultaneously or progressively.

Introduction

The search for evolutionary, functional and structural relationships among protein sequences is becoming increasingly rewarding for researchers in molecular biology. Computer programs for the rapid search of databases to retrieve related sequences and for aligning sequences to show common relationships have established themselves as valuable tools. Obviously, by relating one's sequence of interest to previously characterized sequences, valuable insight can be obtained, whether one's interest is primarily in evolutionary relationships, in predicting secondary or tertiary structure or in locating segments with functional properties (e.g. ligand binding sites).

Various approaches can be taken for comparing biological sequences. Methods for global sequence alignment establish a residue-to-residue correspondence between the primary structures of proteins, by inserting 'gaps' in appropriate places, so that the resulting alignment highlights similar regions. These methods are mostly based on a relatively simple sequence similarity score, which is optimized by dynamic programming, as first shown by Needleman and Wunsch in 1970. Given a table that defines the cost of substituting one sequence residue with another (or of the similarity between two residues) and a penalty for inserting gaps (insertions or deletions) in either sequence, the Needleman–Wunsch algorithm is guaranteed to find an optimal alignment between two sequences in terms of minimal overall cost or maximal score. The relative ease of implementing the algorithm, the simplicity and consistency of the underlying scoring system, and the ready availability of suitable scoring matrices have contributed to the popularity of this approach.

The Needleman–Wunsch algorithm can also be extended to the simultaneous alignment of more than two sequences. In practice, however, this approach is restricted by an excessive cost in computer time and memory, and has only been implemented for the alignment of three sequences (Murata *et al.*, 1985), or by heuristically reducing the search space, for the alignment of up to 6–8 sequences of moderate length (Carillo and Lipman, 1988; Lipman *et al.*, 1989). Many existing computer programs use the pairwise Needleman–Wunsch algorithm to align multiple sequences by adding in each sequence in some predetermined order. For example, they start with the pair of sequences showing closest overall similarity and continue to add sequences in decreasing order of similarity. The model of sequence evolution implied in the order by which sequences are added can be strictly linear (Barton and Sternberg, 1987; Santibánez and Rohde, 1987; Taylor, 1987; Henneke, 1989) or allow for clusters of sequences to be aligned progressively (Feng and Doolittle, 1987; Taylor, 1988; Higgins and Sharp, 1989). In any cases, the final alignment will be determined by the alignment order, as substitutions, insertions and/or deletions introduced in previous steps remain unaffected by following alignments.

Both the simultaneous and the progressive type of alignment have their theoretical and practical advantages and shortcomings. The reader is referred to the extensive literature in this field (e.g. Hogeweg and Hesper, 1984; Feng and Doolittle, 1987; Carillo and Lipman, 1988; Altschul and Lipman, 1989).

We have developed an algorithm for the simultaneous alignment of multiple sequences that should closely approximate the results obtained by rigorous dynamic programming, e.g. as implemented by Murata *et al.* (1985), but without the same

Analytical Biostatistics Section, Division of Computer Research and Technology, (formerly, Laboratory of Theoretical and Physical Biology, National Institute of Child Health) National Institutes of Health, Building 12A, Room 2009, Bethesda, MD 20892, USA

[1]*To whom reprint requests should be sent*

limitation on the number of sequences or on their length. Our algorithm is novel in that it attempts to avoid the pitfall of methods that use a predetermined order of adding sequences, namely that they may easily get trapped by local optima resulting from the predetermined order of alignments. Instead, we use a random iterative approach for selecting sequences to be aligned. Our implementation of this new algorithm allows for the simultaneous alignment of subgroups of closely related sequences, which can be followed by a progressive alignment among less closely related sequences.

System and methods

The randomized pairwise multiple sequence alignment algorithm was developed on an IBM PS/2 Model 80 and was implemented in an interactive computer program, denoted MUSEQAL. The program MUSEQAL is written in Turbo C 2.0 (Borland Inc.) for the IBM PC class of machines running under the operating system MS-DOS. A minimum of 640K RAM is required. The program is available from the authors without charge.

Algorithm

Any possible alignment between two sequences S_1 and S_2 of length L_1 and L_2 can be depicted with a path through a matrix of dimensions L_1 and L_2, as shown in Figure 1. A straight path from cell i,j to the next diagonal cell $i+1,j+1$ represents an uninterrupted alignment of the ith and $(i+1)$th residue in S_1 with the jth and $(j+1)$th residue in S_2. Any move to the subrow $(i+1,j+y)$ $(1 < y \le L_2-j)$ or to the subcolumn $(i+x,j+1)$ $(1 < x \le L_1-i)$ denotes a gap in either S_1 or S_2 with length $y-1$ or $x-1$ respectively. Using a scoring table to define the similarity between any two residues $s(i,j)$, the original Needleman–Wunsch algorithm finds an optimal path through this matrix where the cost associated with each path is taken to be the sum of the similarity scores minus a penalty (gp) for any gap in either sequence. Starting at position L_1,L_2 (the lower right corner in Figure 1) and filling the matrix either row by row or column by column, the value of each cell $C(i,j)$ is calculated as follows:

$$C(i,j) = s(i,j) + \max[C(i+1,j+1),\ C(i+x,j+1) - gp,\ C(i+1,j+y) - gp] \quad (1)$$

$$(1 < x \le L_1-i,\ 1 < y \le L_2-j)$$

In essence, the value of each cell equals the similarity score $s(i,j)$ for aligning residue i in S_1 with residue j in S_2, as defined by the scoring matrix, plus the maximum value for one of three potential choices indicated in Figure 1 with arrows 1–3. Arrow 1 depicts a move to the next diagonal cell $C(i+1,j+1)$, arrows 2 and 3 show the insertion of gaps in either S_1 or S_2 by moving to the cell with the maximum value in the next subcolumn $C(i+x,j+1)$ or subrow $C(i+1,j+y)$. To restrict these moves, a gap penalty is subtracted from the value in these cells. With a matrix thus filled in, the globally optimal path and hence the global alignment can be reconstructed. If endgaps are not penalized, one will start with the cell with the maximum value (corresponding to the maximal score for the alignment) in the leftmost column or uppermost row. One then follows the path using, at each step, the 'optimal' arrow (1, 2 or 3 in Figure 1) corresponding to the argument of the maximum function in equation (1).

An extension of this dynamic programming algorithm to the alignment of more than two sequences can be done in several ways. We will follow the implementation chosen by Murata *et al.* (1985), who, for the simultaneous alignment of three sequences of length L_1,L_2,L_3, define a matrix with dimensions (L_1,L_2,L_3). Again, a path (alignment) is represented by a succession of matrix cells (i,j,k). As with the two-dimensional version, a move from one cell to the next follows certain constraints to guarantee a consistent alignment: cell (i,j,k) can be followed by cell $(i+1,j+1,k+1)$, which represents an uninterrupted alignment, or by any cell in the subregion defined by $(i+x,j+y,k+z)$, where at least one of x, y or z must equal 1 and one or two of x, y or z must be greater than 1. In the latter case, a single gap is opened in the alignment. Thus, an example alignment of S_1, S_2 and S_3 (Figure 2) would correspond to a path from (1,1,1) to (2,2,2) to (3,4,6) and to (4,5,7). A single gap is present.

The summation of matrix elements is done in all three dimensions, with similarity scores $s(i,j,k)$ calculated as the sum of all pairwise scores between residues at positions i,j,k. A gap

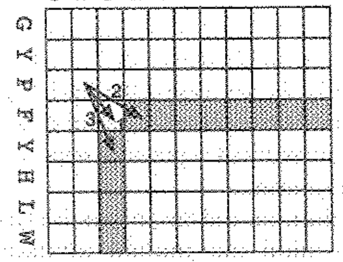

Fig. 1. Matrix used for finding an optimal alignment for two sequences with the Needleman–Wunsch algorithm. Any possible alignment can be traced with a path leading from one cell to the next, with moves restricted as shown by arrows 1–3. Arrow 1 denotes an uninterrupted alignment between residues W and T (upper sequence) and residues P and F (left sequence). Arrows 2 and 3 show the insertion of gaps in either the left or top sequence by moving to a cell in the shaded area. The Needleman–Wunsch algorithm finds the path associated with the highest score, as explained in the text.

Position:	1	2	3	4	5	6	7
Sequence S1:	L	A	–	–	–	F	P
Sequence S2:	P	C	E	–	–	L	W
Sequence S3:	L	Y	L	C	G	E	Y

Fig. 2. Example alignment to demonstrate the representation of alignments of three sequences in a three-dimensional lattice. A path representing this alignment would start at cell (1,1,1), corresponding to the first residue in all sequences. It would then continue to cell (2,2,2), followed by cell (3,4,6) and (4,5,7). Note that even though two null run(s) are present in this alignment, only one single gap is counted.

is initiated whenever the value C of a cell in any of the three subregions $(i+x,j+y,k+z)$ minus the gap penalty is higher than the next diagonal cell $(i+1,j+1,k+1)$. Thus, the optimal path and the resulting alignment score is completely defined by the succession of matrix elements (i,j,k), representing fully aligned columns. In regions of gaps, where nulls are present in any sequence (positions 3, 4, 5 in Figure 2), no pairwise scores and no alignments are defined. Furthermore, gap penalties are independent of the length of the gaps and no penalties for endgaps (overhanging residues) are applied.

This rigorous approach of simultaneously aligning sequences can easily be extended to more than three sequences. We would then consider an n-dimensional lattice with coordinates $(i,j,k, \ldots ,p)$ defined by the alignment of residues at positions i in S_1, j in S_2, . . . ,p in S_n. The optimal path through this multidimensional lattice returns the highest similarity score [sum of all pairwise similarity scores for residues corresponding to a cell in the path less a gap penalty for every 'break' in the path, i.e. a move from $(i,j,k, \ldots ,p)$ to something other than $(i+1,j+1,k+1, \ldots ,p+1)$].

Obviously, with the memory requirements and CPU time growing with the product of the lengths of the sequences, the full implementation of this approach is not feasible even for a moderate number of short sequences. It is, however, possible to divide the problem of finding an optimal alignment into a series of smaller steps that can be solved with only minor demands on computer time and memory. We have devised a strategy that iteratively approaches an optimal alignment.

The basic idea is as follows: given n initially aligned sequences, we randomly divide them into two groups. By 'freezing' the alignment of sequence members *within* each group we can optimize the alignment *between* the groups, using a two-dimensional Needleman–Wunsch type of algorithm with modifications to satisfy the rules and constraints of its multidimensional counterpart (e.g. the same scoring scheme and definition for gaps). Again, we can use equation (1) to calculate cell scores $C(i,j)$, but we must apply the following two extensions, illustrated by Figure 3.

(i) Similarity scores $s(i,j)$ are defined only for those cells representing fully occupied alignment positions. They are calculated as the sum of the pairwise scores between all residues aligned at position i in group S_1 and j in group S_2. In positions

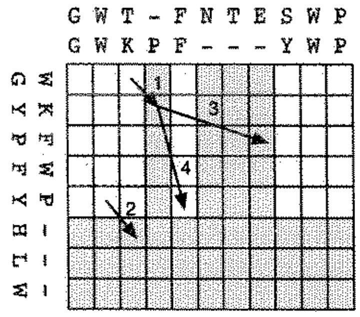

Fig. 3. Matrix used by our algorithm to find an optimal alignment between two groups of prealigned sequences. Again, each cell can be followed by cells as shown in Figure 1, but because of the presence of already established gaps in the two groups of prealigned sequences, the consequences of these moves on the gap penalty can be different. Thus, any path leading from a cell that denotes a complete alignment of residues in both groups (unshaded areas in Figure 3) to a cell denoting an incomplete alignment (null run) in either sequence group (shaded areas in Figure 3), will open a gap in the global alignment of both groups (arrows 1 or 2). On the other hand, any move from a gap region (shaded area) into or beyond a gap region (arrows 3 or 4) will not open any new gaps in the final alignment, but will simply add a null in either group of prealigned sequences.

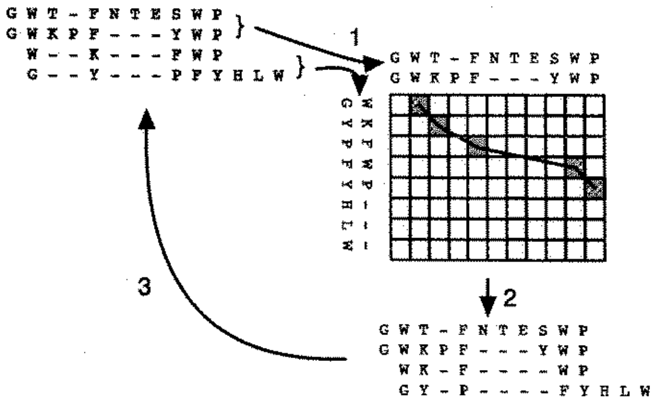

Fig. 4. Illustration of the iterative strategy used to improve the overall alignment of multiple sequences. The sequences are randomly divided into two groups (step 1), for which an optimal alignment is obtained by a modified Needleman–Wunsch type of algorithm. The resulting alignment (step 2), in turn, is the starting point for the next alignment of a different pair of groups (step 3). Each iteration that improves the alignment between two sequence groups will also improve the globally optimal alignment. The five shaded cells are elements of the optimized path in the matrix and correspond to the five fully occupied columns after step 2.

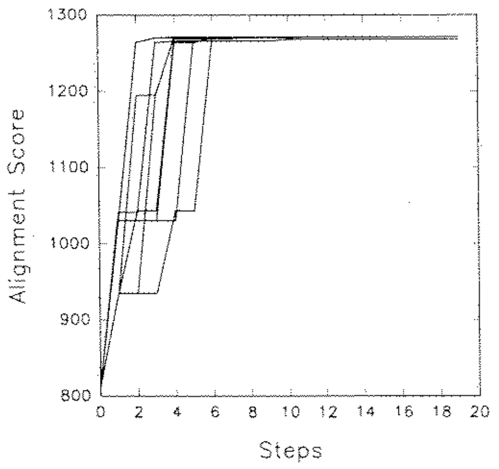

Fig. 5. Increase in alignment score of 10 tests runs performed with our algorithm, aligning the three copper-binding proteins used by Murata *et al.* (1985). Each run starts with a score of 833, corresponding to the gapless alignment of the three sequences at the N-terminal end, and calculated with a modified McLachlan scoring matrix (McLachlan, 1971). In all runs but one, the optimal alignment (Figure 6) with a score of 1271 was achieved. In one case, an alignment with two misplaced residues and a score of 1268 could not be improved. The time requirement for each step was < 1 s. An average of four steps was required to reach the final score.

```
CBP     AVYVVGGSGGW--TFNTES------WPKGKRFRAGDILLFNYNPSM-HNVVVVNQGGFSTCNTPAGA  58
SC      TVYTVGDSAGWKVPFFGDVDYDWKWASNKTFHIGDVLVFKYDRRF-HNVDKVTQKNYQSCNDTTPI  65
PC      IDVLLGADDGS-LAF----------VPSEFSISPGEKIVFKNNAGFPHNI-VFDEDSIPSGVDASKI  55

CBP     KVYTS------GRD-QIKLP-KGQSYFICNFPGHCQSGMKIAVNAL                       96
SC      ASYNT-------GNN-RINLKTVGQKYYICGVPKHCDLGQKVHINVTVRS                   107
PC      SMSEEDLLNAKGETFEVALSNKGEYSFYCS-P-HQGAGMVGKVTVN                       99
```

Fig. 6. Simultaneous alignment of the three copper-binding proteins, cucumber basic blue protein (CBP), stellacyanin (SC) and plastocyanin (PC). Using the same scoring matrix and gap penalty as Murata *et al.* (1985), this alignment was obtained with only three iterations of our randomized pairwise algorithm. The alignment is identical to the one presented by Murata *et al.*

where gaps (null runs in any sequence of either group) are present (shaded areas in Figure 3), $s(i,j)$ equals 0 and will not contribute to the summation of the running score, as explained above.

(ii) Any move from a non-gap region (unshaded areas in Figure 3) into or beyond a region with gaps in the prealigned sequences (shaded areas in Figure 3) opens a gap in the global alignment. This has two consequences for evaluating equation (1): a gap penalty has to be subtracted from the value of $C(i+1,j+1)$ whenever a diagonal move into a gap-region is considered (e.g. arrows 1 and 2 in Figure 3). On the other hand, any move from a cell (i,j) inside a gap region into or beyond a gap region in subrow $(i+1,j+1)$ $(1 < y \leq L_2-j)$ or subcolumn $(i+x,j+1)$ $(1 < x \leq L_1-i)$ (e.g. arrows 3 and 4 in Figure 3) must not be penalized ($gp = 0$ in equation 1), because the resulting alignment introduces no new gap, it simply adds a null run in either S_1 or S_2 to an already existing gap in S_1 or S_2. With these modifications, equation (1) allows us to find an optimal alignment for the two groups of prealigned sequences, which is bound to improve or at least equal the alignment of all n sequences in the (hypothetical) multidimensional lattice. Thus, our algorithm proceeds as follows (Figure 4).

Starting with a total of n (initially gapless) aligned sequences, we randomly divide the sequences into two subgroups, S_1 and S_2 with n_1 and n_2 sequences each $(n_1+n_2=n)$. The partitioning of the group sequences is chosen at random from the total of $(2^{n-1})-1$ possibilities (step 1 in Figure 4). Within the subgroups, the current alignment will be preserved, but any global gaps (null runs in all sequences) in a given subgroup will be lost. For example, in the alignment shown on the upper left corner of Figure 4, the two gaps initially present in the two lower sequences (WKFWP and GYPFYHLW) will be removed when the sequences form their own subgroup.

S_1 and S_2 are then aligned using our extended Needleman–Wunsch algorithm with modifications as outlined above and illustrated in Figure 3. The resulting alignment (step 2 in Figure 4), in turn, will replace the original alignment (step 3 in Figure 4) and will be the starting point for the next alignment of a different pair of subgroups.

An optimal overall alignment is thus approached iteratively. Each step is solved with a maximal memory requirement defined by the product of the lengths of S_1 and S_2, $L_1 \times L_2$. In practice, the size of the matrix can be reduced substantially by defining a diagonal cutoff and calculating equation (1) just within a band along the main diagonal, as suggested by Ukkonen (1985). Furthermore, we use Gotoh's (1982) method to compute the pairwise alignment in maximally length (S_1) * length (S_2) steps. The time needed to achieve an optimal alignment for all sequences then grows linearly with the number of steps.

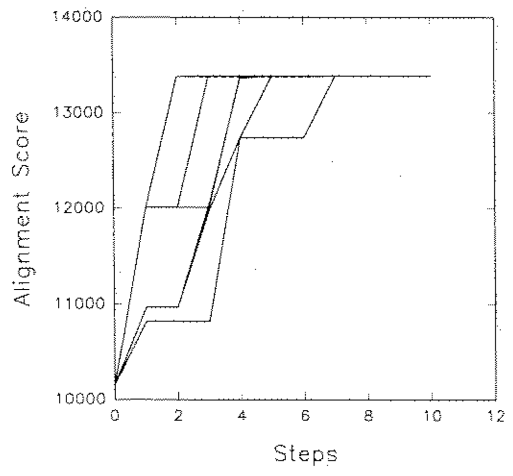

Fig. 7. Increase in alignment score for 10 test runs with the three ATPases (α and β subunits of bovine mitochondria and β subunit from *Escherichia coli*), for which Murata presents an optimal alignment (Murata, 1990). The identical alignment with the maximal score of 13 384 was reached in all 10 runs, in an average of 3.5 steps.

Results

Since our algorithm randomly divides the entire group of sequences, it does not follow the identical path towards the final alignment. Different paths are taken in successive runs of the program. Our iterative strategy does not obviously guarantee an optimal alignment. One way to evaluate the performance of our algorithm is to compare its results with those reported using the rigorous method of solving the entire multidimensional matrix. In their original paper, Murata *et al.* (1985) present the alignment of three copper-binding proteins, using their extended three-way Needleman–Wunsch algorithm with a scoring matrix derived from the work of McLachlan (1971). Murata *et al.* reports CPU time of 81 s for simultaneously aligning all three sequences on a VAX-11/780. Using the identical scoring matrix and gap penalty, the results obtained with repeated runs of our algorithm of an IBM PS/2 Model 80, using a diagonal cutoff of 15 [a diagonal band that intersects at row 15 with a matrix L_1,L_2 $(L_1 \leq L_2)$], are summarized in Figure 5. Each curve shows the progress in alignment score in a single run, from a starting value of 833 for the sequences initially aligned at their N-terminal end without gaps, to a maximum score of 1271. Table I illustrates a single run of our algorithm, and lists the random pairs of subgroups that are aligned at each step. While the alignment score generally increases, it is not required to do so at every step. Note that the identical partitioning of the group of sequences may recur

```
       ****      * * ****          * ******  ** *****            *  *  *
       +++++++   +++++++++++     -++++++++++++++++++++        +++++++++
CHT    IVNGEEAVPGSWPWQVSLQDKT---GFHFCGGSLINENWVVTAAHCGVT----TSDVVVAGEFDQG  59
TRP    IVGGYTCGANTVPYQVSLNSGY-----HFCGGSLINSQWVVSAAHCYKS-----GIQVRLGEDNIN  56
ELA    VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDREL---TFRVVVGEHNLN  63
MCP    IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHCKGR-----EITVILGAHDVR  61
KAL    IVGGTNSSWGEWPWQVSLQVKL-TAQRHLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGILNLS  65
TON    IVGGYKCEKNSQPWQVAVINE------YLCGGVLIDPSWVITAAHCYSN-----NYQVLLGRNNLF  55

              *     *   *  **                 **  ****         *    **
             +++++++++++++++++                ++++++++++++++ ++++++++
CHT    SSSEKIQKLKIAKVFKNSKYNSLTINN-----------DITLLKLSTAASFSQTVSAVCLPSASDD 114
TRP    VVEGNEQFISASKSIVHPSYNSNTLNN-----------DIMLIKLKSAASLNSRVASISLPTSCA- 110
ELA    QNNGTEQYVGVQKIVVHPYWNTDDVAAGY---------DIALLRLAQSVTLNSYVQLGVLPRAGTI 120
MCP    KRESTQQKIKVEKQIIHESYNSVPNLH-----------DIMLLKLEKKVELTPAVNVVPLPSPSDF 116
KAL    DITKDTPFSQIKEIIIHQNYKVSEGNH-----------DIALIKLQAPLNYTEFQKPICLPSKGDT 120
TON    KDEPFAQRRLVRQSFRHPDYIPLIVTNDTEQPVHDHSNDLMLLHLSEPADITGGVKVIDLPTKEP- 120

          *  *   ***  *          **    *      *    *        * ***   **   *
           +++++++++++++      ++++++++++++++                +++++     ++++
CHT    FAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLLSNTNC--KKYWGTKIKDAMICAG--ASGVSSC 176
TRP    -SAGTQCLISGWGNTKSSGTSYPDVLKCLKAPILSDSSC--KSAYPGQITSNMFCAGYLEGGKDSC 173
ELA    LANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTVDYAICSSSSYWGSTVKNSMVCAG-GNGVRSGC 184
MCP    IHPGAMSWAAGWGKTG-VRDPTSYTLREVELRIMDEKAC--VDYRYYEYKFQVCVGS-PTTLRAAF 178
KAL    STIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLVTNEECQ-KRYQDYKITQRMVCAGYKEGGKDAC 184
TON    -KVGSTCLASGWGSTNPSEMVVSHDLQCVNIHLLSNEKC--IETYKDNVTDVMLCAGEMEGGKDTC 183

        ******* *    *   * ** *     **     * *** *  *  **      *
       +++++++++++-   --++++++++           ++++++++++++++++++++++
CHT    MGDSGGPLVCKKNGAWTLVGIVSWGS--STCSTSTPGVYARVTALVNWVQQTLAAN           230
TRP    QGDSGGPVVC--SGK--LQGIVSWGS--GCAQKNKPGVYTKVCNYVSWIKQTIASN           223
ELA    QGDSGGPLHCLVNGQYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYISWINNVIASN           240
MCP    MGDSGGPLLC--AGV--AHGIVSY----GHPDAKPPAIFTRVSTYVPTINAVIN             224
KAL    KGDSGGPLVCKHNGMWRLVGITSWGE--GCARREQPGVYTKVAEYMDWILEKTQSS           238
TON    AGDSGGPLIC--DGV--LQGITSG-GATPCAKPKTPAIYAKLIKFTSWIKKVMKENP          235
```

Fig. 8. Alignment of six serine proteases. The first five sequences—bovine chymotrypsin (CHT), bovine trypsin (TRP), pig elastase (ELA), rat mast cell proteinase II (MCP) and human plasma kallikrein (KAL)—were aligned simultaneously, for comparison with an alignment of the same sequences reported by Lipman *et al.* (1989). Positions where all residues of CHT, TRP and ELA are correctly aligned within a structurally conserved region as defined by Greer (1981), are denoted with a plus sign (+). Missed columns are flagged with a minus sign (−). To demonstrate the additive or progressive type of alignment also possible with program MUSEQAL, we have aligned the sequence of rat tonin (TON) with the prealigned upper five sequences. An asterisk (*) denotes columns with at least four identical residues.

Table I. Sample run of an iterative alignment with the three copper-binding proteins CBP, SC and PC shown in Figure 6 (subgroups aligned with each step are S_1 and S_2)

Step	Score	S_1	S_2
0	811		
1	935	PC	CBP, SC
2	1043	CBP	SC, PC
3	1269	PC	CBP, SC
4	1269	CBP, PC	SC
5	1269	CBP, PC	SC
6	1271	SC,PC	CBP

several times, yet the overall score continues to increase, e.g. step 1 and step 3. The improvement in step 3 results from the fact that SC and CBP have been aligned to each other in step 2, thereby permitting a better pairwise alignment of PC to the group {CBP, SC}. Two successive alignments of the same partitioning of the sequences, of course, do not improve the score (e.g. step 4 versus step 5).

The optimal alignment corresponding to the score of 1271 is shown in Figure 6. It is identical to that reported by Murata *et al.* (1985). The time requirement for each step was <1 s. On average, 4 s were required to reach the maximal score. In only one of 10 test runs could the optimal score of 1271 not be reached and a slightly suboptimal alignment with a score of 1268, corresponding to two misaligned residues, remained.

Figure 7 shows the results obtained with 10 test runs for the iterative alignment of three ATPase sequences, for which Murata (1990) presents an optimal alignment, using the MDM78 scoring matrix (Dayhoff *et al.*, 1978) and a gap penalty of 16. He reports a CPU time requirement of 5 min 49 s on a VAX-11/750. Again using a diagonal cutoff of 15, each step with our algorithm took <3 s, so that the optimal alignment was achieved in all 10 runs in an average of 12 s.

The behavior of our algorithm cannot easily be predicted for the alignment of more than three sequences. Unfortunately, there are no full-scale implementations of our scoring system that could be used for comparison in these cases. Lipman *et al.* (1989), whose program MSA handles the simultaneous alignment of up to 6−8 sequences, use a different scoring scheme for matrix summation and a different definition for gaps. They evaluate the simultaneous alignment of five serine proteases with an alignment derived from information about the three-dimensional structure of chymotrypsin, trypsin and elastase, presented by Greer (1981). Lipman *et al.*'s alignment shows an agreement with Greer's structurally conserved regions in 154 aligned positions (columns), with seven columns missed or incorrectly aligned. The results of our iterative algorithm are shown in Figure 8, which is discussed in the next section. The alignment was reached with 48 steps of the algorithm in <2 min. One hundred and fifty-seven columns are aligned in agreement with Greer's structurally conserved regions (marked with a '+' sign in Figure 8); four columns were misaligned (marked with a '−' sign in Figure 8).

Implementation

Up to 100 sequences can be loaded into program MUSEQAL from a disk file in a format used also by the FASTP (Pearson and Lipman, 1988) and MACAW (Schuler *et al.*, 1990) programs and displayed in a resizable window on screen. The user can manually define groups of sequences to be aligned simultaneously or progressively. For example, the alignment in Figure 8 was produced by first simultaneously aligning the five upper sequences, bovine chymotrypsin (CHT), bovine trypsin (TRP), pig elastase (ELA), rat mast cell proteinase II (MCP) and human plasma kallikrein (KAL) (the same group of sequences aligned by Lipman *et al.*, 1989). The resulting aligned group, in turn, was aligned with rat tonin (TON). In this way, any combination of sequences or groups of prealigned sequences can be aligned in any order. With simultaneous alignments, the user can inspect the results of each step and stop or restart the iterations at any time. The display of the sequences is done using six background colors to highlight residues with similar physico-chemical properties (George *et al.*, 1988). This greatly facilitates the evaluation of alignments as columns or regions with similar residues are visually emphasized.

The current version of the program limits the total number of matrix cells at each step to 32 000. Thus, with (prealigned) sequences longer than 178 residues each, a diagonal cutoff value has to be selected. This will not only shorten the time needed for calculating each step, but might also potentially prevent the algorithm from finding an optimal alignment. However, with globally related sequences of not too dissimilar length, the optimal path will most often lie within a relatively narrow diagonal band.

Discussion

We present an algorithm for the simultaneous alignment of multiple protein sequences, according to a model of sequence similarity first implemented by Murata *et al.* (1985) for the case of aligning three sequences. Using an adaptation of the standard Needleman–Wunsch type of algorithm, we divide the n-dimensional problem of finding an optimal alignment for n sequences into a series of steps, which can be solved in two-dimensional space on a personal computer, requiring only a modest amount of memory. Compared with the rigorous approach of solving an n-dimensional lattice by dynamic programming, our iterative algorithm results in fairly reproducible alignments equal or close to the optimal solution, at least for the limited number of sequences, for which an optimal alignment is known. We have implemented this algorithm in a computer program running on an IBM PC, providing an environment for interactively selecting sequences or groups of sequences to be aligned simultaneously and/or progressively.

In practice, simultaneous alignments will be restricted to sequences exhibiting a global and uniform degree of relatedness. In addition, the sequences must not be too dissimilar in length. This follows from the particular model of sequence similarity chosen, which gives equal weight to all residues over the total length of the aligned positions and does not align residues in regions with null runs. Thus, the alignment of a group of sequences that includes, for example, one distantly related member can be unfavorably biased, as explained in detail by Lipman *et al.* (1989). To accommodate such situations, Lipman's program can employ weights to calculate pairwise alignments according to a presumed evolutionary tree. With our program, it is the user's responsibility to select suitable subgroups of sequences to be aligned simultaneously and to determine the order of progressive alignments. In Murata's and our model of sequence similarity, gap penalties are defined as any number of null runs between complete columns of aligned residues. This implementation has been criticized on the grounds of having little relationship to substitution costs defined by the scoring matrix (Altschul, 1989; Lipman *et al.*, 1989). We felt, however, that the advantage of having a relatively simple but fast algorithm would outweigh any hypothetically adverse effects on the final alignment. Indeed, apart from the minor inconvenience of having to adjust gap penalties relative to the number of sequences to align, we did not experience any obvious misalignments caused by our definition of gaps. The sample alignment given in Figure 8, when compared with Lipman's alignment, appears to support this view.

It is conceivable that a more selective way of defining groups of sequences to be aligned with each step could substantially shorten the time required to reach the final alignment. This could be done heuristically or with a parallel implementation of the algorithm, whereby at each step, all $2^{(n-1)}-1$ possible alignments between pairs of subgroups of sequences could be evaluated in parallel and only the one yielding the highest score would be followed up.

References

Altschul,S.F. (1989) Gap costs for multiple sequence alignment. *J. Theor. Biol.*, **138**, 297–309.

Altschul,S.F. and Lipman,D.J. (1989) Trees, stars, and multiple biological sequence alignment. *SIAM J. Appl. Math.*, **49**, 197–209.

Barton,G.J. and Sternberg,M.J. (1987) A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, **198**, 327–337.

Carillo,H. and Lipman,D. (1988) The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, **49**, 197–209.

Dayhoff,M.O., Schwartz,R.M. and Orcutt,B.C. (1978) In Dayhoff,M.O. (ed.), *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Washington, DC.

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

George,D.G., Hunt,L.T. and Barker,W.C. (1988) In Schlesinger,D.H. (ed.), *Macromolecular Sequencing and Synthesis*. Alan R.Liss, New York.

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

Greer,J. (1981) Comparative model-building of the mammalian serine proteases. *J. Mol. Biol.*, **153**, 1027–1042.

Henneke,C.M. (1989) A multiple sequence alignment algorithm for homologous proteins using secondary structure information and optionally keying alignments to functionally important sites. *Comput. Applic. Biosci.*, **5**, 141–150.

Higgins,D.G. and Sharp,P. (1989) Fast and sensitive multiple sequence alignments on a microcomputer. *Comput. Applic. Biosci.*, **5**, 151–153.

Hogeweg,P. and Hesper,B. (1984) The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *J. Mol. Evol.*, **20**, 175–186.

Lipman,D.J., Altschul,S.F. and Kececioglu,J.D. (1989) A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA*, **86**, 4412–4415.

McLachlan,A.D. (1971) Tests for comparing related amino-acid sequences. Cytochrome c and cytochrome c-551. *J. Mol. Biol.*, **51**, 409–424.

Murata,M. (1990) Three-way Needleman–Wunsch algorithm. In Doolittle,R.F. (ed.), *Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences: Methods of Enzymology, Vol. 183*. Academic Press, San Diego, pp. 365–375.

Murata,M., Richardson,J.S. and Sussman,J.L. (1985) Simultaneous comparison of three protein sequences. *Proc. Natl. Acad. Sci. USA*, **82**, 3073–3077.

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

Pearson,W.R. and Lipman,D.J. (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, **85**, 2444–2448.

Santibánez,M. and Rohde,K. (1987) A multiple alignment program for protein sequences. *Comput. Applic. Biosci.*, **3**, 111–114.

Schuler,G.D., Altschul,S.F. and Lipman,D.J. (1991) A workbench for multiple alignment construction and analysis. *Proteins: Structure, Function and Genetics*, **9**, 180–190.

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

Taylor,W.R. (1988) A flexible method to align large numbers of biological sequences. *J. Mol. Evol.*, **28**, 161–169.

Ukkonen,E. (1985) Algorithms for approximate string matching. *Inf. Control*, **64**, 100–118.

Received on December 1, 1990; accepted on April 20, 1991

Circle No. 5 on Reader Enquiry Card