MULTAN: a program to align multiple DNA sequences

William Bains

MEDIGEN Project, Department of Medicine, Stanford Medical School, and Veterans Administration Medical Center 151M, 3801 Miranda Ave., Stanford, CA 94305, USA

Received 5 August 1985

ABSTRACT. I describe a computer program which can align a large number of nucleic acid sequences with one another. The program uses an heuristic, iterative algorithm which has been tested extensively, and is found to produce useful alignments of a variety of sequence families. The algorithm is fast enough to be practical for the analysis of large numbers of sequences, and is implemented in a program which contains a variety of other functions to facilitate the analysis of the aligned result.

INTRODUCTION

The development of rapid methods for the determination of nucleic acid sequences has resulted in an explosive increase in the amount of sequence information published in the last 10 years (8, 11). The quantity of sequence data available has necessitated the development of computer programs to analyse nucleic acid sequences, and in particular to determine how sequences may be related to each other (10, 18). Several algorithms for aligning two sequences with each other have been published: these either rigorously find the optimal alignment for a given set of constraints (10, 13), or rapidly find an excellent approximation to the rigorously optimal alignment (18). Recently the Needleman-Wunsch algorithm for finding the rigorously optimal alignment has been generalised to aligning three sequences (19). However, it has proven impractical to extend these methods to aligning more than three sequences (13), as the dynamic programming procedures used would take unacceptable amounts of time. Several approximate approaches to aligning multiple DNA sequences have been published. Generally, these work by finding sub-sequences shared between all the data sequences (7, 11, 17). Such programs are useful for finding short elements common to a collection of diverse sequences, but are not designed to perform a global alignment on a set

MULTAN is available to all academic researchers on JANET, ARPANET, BIONET and other media. Enquiries should be directed to the author.
of sequences, that is to take a set of longer, related sequences and align all of each data sequence optimally with each other sequence.

The sub-sequence approach has been adapted to this latter problem by compiling the sequences found to be common to all the data set into a complete alignment (7, 17, Martinez per comm). However, this approach has three potential drawbacks: It presupposes that each of the data sequences contains at least one of the subsequences used (which is not necessarily the case with sets of short, divergent sequences). It can fail to align the terminal regions of two sequences if they are more divergent than the bulk of the sequence, and there is a strong requirement for good matching between the subsequences found in the initial stage of the algorithm for an effective alignment (18). Two programs which do not use this methodology have been published (3, 15): however, these are designed to align the overlapping segments of sequence derived from a single gene clone during sequence-determination procedures, and so are incapable of aligning sequences more than ~5% diverged from each other.

In this paper I describe a program which uses an heuristic algorithm to perform a global alignment on any number of sequences, and perform some basic analyses on the result. In the present implementation the program, called MULTAN, can analyse up to 50 related sequences each about 1000 bases long. The program should be generally applicable to the analysis of any group of related DNA or RNA sequences.

**METHODOLOGY**

**Algorithm.**

The algorithm used is purely heuristic. Initially it was an implementation of descriptions of how a scientist might align several related sequences; subsequent modifications were included if they were found empirically to improve the repeatability or the operating speed of the algorithm.

**Multiple alignments.** The essence of the algorithm is that the data sequences are aligned one at a time with a consensus sequence derived from the entire data set, but not with each other. This process is circular, as the consensus sequence derived from a set of sequences depends on how they are aligned with each other, so one of the data sequences is chosen as a "Trial Consensus", or "Seed sequence", for an initial alignment. The alignment routine in MULTAN makes no distinction between a Trial Consensus and a consensus sequence generated from the aligned data sequences. Each data sequence is aligned with the consensus sequence (or Trial Consensus) introducing gaps into either the consensus or the data sequence as required by the pairwise alignment algorithm.
Allignjnant Step

1) Data sequencea. 2) Pairwlse alignment.

Sequence 1: GCTTGCTCA
Sequence 2: GCTTGCTCA (trial consensus)
Sequence 3: GCTTGCTCA

3) Introduction of gaps into all sequencea. 4) Compilation of gapped sequencea.

Sequence 1: TGCTTTCGTCA
Sequence 2: -GC-TTC-GT-A
Sequence 3: -GC-TTC-GT-A

5) New consensus: -GC-TTCGGTCA

Figure 1 Illustration of an Alignment Step, and the subsequent generation of a new consensus. Three oligonucleotides are aligned using the MULTAN methodology, starting with sequence 2 as the Trial Consensus. In order to make the procedure for introducing gaps clearer, gaps have been shown by a '-' throughout.

(described below). If, during the pairwise alignment of a data sequence with the consensus, a gap was introduced into the data sequence at a particular point, that gap remains and all the other data sequences remain unaffected. If a gap was introduced in the consensus sequence at a given nucleotide, then a gap is introduced into all data sequences except the one currently being aligned with the consensus. Thus the presence of a gap in one sequence relative to the consensus causes that sequence and no other to contain a gap at that point, whereas the presence of a gap in the consensus relative to a data sequence causes all other sequences, including the consensus, to possess a gap at that point. This alignment process is illustrated in Figure 1, which shows an alignment of a small data set with a Trial Consensus, called one Alignment Step, and the subsequent generation of a new consensus. The generation of the new consensus is described below.

To find a potential optimal alignment, the program executes these two phases - an Alignment Step followed by the generation of a new consensus - alternately. After each new consensus has been generated, it is compared to previous consensuses to see if they are the same. If they are identical then the program has reached a limiting state, where the alignment of the data to a particular consensus is used to generate the same consensus, so that repetition of the iterative cycle can produce no further change in either the alignment or the consensus. The iterative loop is terminated, and the resulting consensus and alignment are called the Terminal Consensus and Terminal Alignment. If the new consensus is not identical to the previous one, then the new consensus is used as the basis for a new alignment in another Alignment Step.

The program usually operates in this iterative fashion; however, for reasons described below, MULTAN has the facility to operate one Alignment Step at a
time, with the potential for user interaction between iterative loops.

**Concensus Generation.** The concensus sequence must be able to hold information about the variability of the sequence data as well as representing the mode of the aligned sequences. In its simplest form a concensus sequence is simply the sequence of bases which are most common at each position in the data sequences (the 'modal sequence'). However, it has been found empirically that aligning sequences to such a 'modal sequence' gives poor results if the data sequences are moderately diverged. Instead, MULTAN includes some ambiguity into the concensus, so that some positions are not specified as being occupied by one base only, but rather are specified as being filled by one of two or more bases. In the present implementation this ambiguity is achieved by incorporating symbols which can stand for particular pairs of bases (such as P = A or G), or N = any base, into the concensus. Any pair of bases is allowed, and such ambiguous pairs are coded in the Stanford Ambiguity Code (2) (this code is listed at the bottom of table 1). The choice of which base or ambiguous symbol to insert is made on the basis of the data sequences' Adherence to the concensus, defined as the number of positions in the data sequences which are consistent with the concensus at a given position, expressed as a fraction of the number of sequences which have a base at that position. (To avoid the generation of a concensus consisting only of 'N's, the Adherence of a position in the concensus which is an N is defined as 0). A position is filled by an N only if the Adherence of the best choice of non-N character for that position is below a user-defined parameter NLIM. Two other rules determine whether a position in the concensus should be set blank, i.e. is a gap (see Table 1).

Chance ensures that any set of unrelated sequences can be aligned so that some bases occur with high frequencies at certain positions. This is especially true if gaps may be introduced to maximise the matching between sequences. Thus the ends of the concensus must be carefully defined, or spurious matches will appear to extend it indefinitely into sequences flanking those truly shared between the data sequences. In MULTAN the boundaries of the concensus are defined by requiring the terminal tetranucleotide to have a mean Adherence of at least TER, a user-defined variable (optimally set to 70%-80%), as well as being represented in a minimum number of sequences.

The rules used to generate a concensus are summarised in Table 1. **Pairwise alignment of sequences.** In MULTAN an heuristic pairwise algorithm has been used, which has been found to produce reliable alignments and to be acceptably fast. The algorithm aligns two sequences without introducing gaps to maximise
Table 1 Rules for generating a consensus. The rules are operated in the order listed, so that if Rule 4 generates a result incompatible with Rule 2, Rule 4 takes precedence.

1) For each position, sum the number of bases and gaps in the data sequence. If a data sequence is ambiguous, treat the ambiguous symbol as a mixture of the bases for which it can stand (eg J = 0.5A + 0.5C).

2) For each position, if the best Adherence to an unambiguous base is greater than 0.6 times the best Adherence to an ambiguous symbol. Position = unambiguous base otherwise position = ambiguous symbol.

3) If Adherence is less than NUM and the number of sequences possessing a base at that position is greater than 50%. Position = N.

4) If less than 25% of sequences have a base at a given position. Position = gap.

5) If less than 50% of sequences have no base at a given position and the Adherence of the base chosen by Rule 2 is less than 50%. Position = N.

6) If less than 50% of sequences have no base at a given position and the Adherence of the base chosen by Rule 2 is less than 50%. Position = gap.

7) The 5' and 3' terminal tertrnapolnucleotides must have a mean Adherence of at least TER, and the proportion of data sequences having a base at each position of the terminal tertrnapolnucleotides must be greater than 33%.

8) Ambiguous bases are coded thus:
   b) P = Purine (A or G), Y = Pyrimidine (C or T).
   c) J = (A or C), K = (G or T), L = (A or T), M = (G or C)
   d) N = Any base.

0.5

The matching in subsequent phases of the alignment routine is performed on 12 base blocks of sequence, not on individual bases. Thus the match between two sequences at a particular point is calculated as the mean of the 12 base block of sequence surrounding that point. In this regard this algorithm resembles both the fast pairwise alignment algorithm of Wilbur and Lipman (18), which aligns 'words' several bases long rather than individual bases, and the oligonucleotide-matching algorithms which compare many sequences mentioned in the Introduction (7, 11, 17). After the initial alignment, the region of best local matching is found, and the sequences are scanned from that point bidirectionally (i.e. from the point of best matching towards both 5' and 3' ends), searching for a drop in the local matching between the sequences below a threshold value. Such a drop is called a Discontinuity, and signifies the site for a potential gap. The sequences are then scanned distally to the Discontinuity to determine whether a gap inserted into either sequence would raise the local matching after the Discontinuity. This scanning does not continue indefinitely, but only up to a maximum of MAXDEL (another user-defined variable) number of bases along the sequence. MAXDEL is therefore the largest size of gap that MULTAN can introduce into any one sequence. Insertion of a gap is penalised by a weighting factor

Gap-Weight = BIAS + (DELWGT x length-of-gap)
where BIAS and DELWGT are user-defined. When the optimal arrangement has been achieved, scanning continues from the distal end of the gap inserted. The location of the ends of the gap are optimised to maximise local matching and, regardless of the direction in which the sequence was being scanned when the match was detected, the ends of the gap are shifted as far 5' ('Left') as is consistent with optimal matching. Thus if there is redundancy in the sequence in which the gap is being inserted, as would occur, for example, in matching the sub-sequences

...QAAAAQ... and ...QAAAQ...

the gap would be introduced at the 5' end of the redundant sequence thus

...G-AAAG...

regardless of whether this gap had been detected during 5'-to-3' or 3'-to-5' scanning. This ensures that gaps in such redundant sequences are not randomly distributed throughout the sequence, leading to unnecessary spaces in the compiled alignment.

In the matching routine used to determine the similarity between 12 base blocks, Identical bases are given a score of 100, Purine-Purine and Pyrimidine-Pyrimidine mismatches a score of 40 and other mismatches a score of 0. This favours Pu-Pu and Py-Py mismatches over other mismatches, reflecting two biological constraints. Firstly, for non-coding DNA, the rate of mutation of CG dinucleotides to TG or CA is observed to be much faster than other mutational changes, and so a CG dinucleotide is more likely to be 'correctly' matched with TG or CA than with other dinucleotides (1). It is likely that all Py-Py and Pu-Pu changes as faster than any Py-Pu change (6). In coding regions, the majority of Pu-Pu and Py-Py changes in third base positions In codons are 'silent', leading to no change in the coding capacity of the DNA, while about half of the third base Pu-Py changes cause changes in the protein coded by the DNA. Thus here again, Py-Py and Pu-Pu changes are more conservative than Py-Pu changes.

This algorithm was developed specifically for MULTAN, but in principle any pairwise alignment algorithm could be used in the program providing A) it aligns all of each pair of sequences, and not just separate segments of them and B) If there is any ambiguity to the position of a gap, i.e. if there is more than one position where a gap may be introduced to produce equal matching between two segments, then the algorithm consistently biases the position of that gap. Thus in MULTAN gaps are always shifted as far 'Left' (5') as possible. The first constraint is required because a single alignment of all the consensus with all the data sequence (excepting those sequences at the termini of the data sequences which may not overlap each other) is required by the multiple
Parameters

The algorithm described above contains 5 parameters which are user-defined, and affect the results which MULTAN will produce. Their definitions are summarized and their default values listed in Table 2.

Implementation

The program which implements this algorithm is written in DEC FORTRAN (compatible with FORTRAN 66), and run on the SUMEX-AIM DEC 2060 computer under the TOPS-20 operating system. With the exception of a few file-handling commands the program should be easily portable onto other FORTRAN implementations. Data files are in Stanford SEQ format (2).

RESULTS

Verification

MULTAN is designed to align a set of data sequences with each other to minimize the number of mismatches between sequences, and to do so by introducing a minimum number of gaps. As this procedure is empirical, and cannot be rigorously proven to produce optimal results, I have performed a number of tests to show that it does produce an acceptable result when operating on a range of test sequences. The test data for which the program was optimised, and on which most of these tests were done, were the human repetitive sequence elements BLUR (4). As a set of ~300bp sequences with a mean mutual divergence of ~18%, including small deletions and insertions, these sequences were considered typical of the data which MULTAN would be called upon to align.

Alignment of the BLUR sequences. Figure 2 shown the alignment of the BLUR sequences produced by MULTAN. Two concensuses are shown: the ambiguous
<table>
<thead>
<tr>
<th>Unambiguous consensus</th>
<th>Ambiguous consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCCAGGGGAGGCCTGAGG</td>
<td>TGTTGCTCTCCTGGTACCC</td>
</tr>
<tr>
<td>AGAGCTGACTCCATCA</td>
<td>TGGTCCTCTGCTAGGAGCC</td>
</tr>
<tr>
<td>CACTCTCTACTAAAAATG</td>
<td>CAGACAGACTCCATCA</td>
</tr>
<tr>
<td>AAAAAA</td>
<td>AAAAAA</td>
</tr>
<tr>
<td>GGGG</td>
<td>GGGG</td>
</tr>
<tr>
<td>AAAA</td>
<td>AAAA</td>
</tr>
<tr>
<td>TTTT</td>
<td>TTTT</td>
</tr>
<tr>
<td>CTTT</td>
<td>CTTT</td>
</tr>
<tr>
<td>TGCAGG</td>
<td>TGCAGG</td>
</tr>
<tr>
<td>GGGG</td>
<td>GGGG</td>
</tr>
</tbody>
</table>

**Figure 2** Alignment of the ten BLUR sequences by MULTAN. The BLUR sequences were aligned by MULTAN operating iteratively, BIAS=5, DELWGT=1, MAXDEL=50, NUM=50, TER=80. The output is essentially as produced by MULTAN. Symbols other than those for the bases (listed in Table 1) have the following meaning: "." = base the same in this position as in the consensus, ":" = this base consistent with an ambiguous base in the consensus, "-" = this base deleted relative to the consensus.

The unambiguous consensus to which MULTAN aligned the data sequences, and an unambiguous consensus derived from the aligned sequences. Note that the unambiguous consensus and the alignment of the data sequences with it are essentially the
same as those derived by Deininger et al (4). (The alignment of Deininger et al (4) was probably performed 'by hand', without computer assistance.) More significantly, MULTAN derives the same alignment and consensus regardless of the order in which the sequences are input, and regardless of the Trial Consensus chosen to start the iterative alignment procedure. Thus for these user-defined parameters the result is independent of the starting conditions.

The program took 96 seconds CPU time (3 iterative cycles) to derive this alignment and the two consensus sequences starting with BLUR8 as the Trial Consensus.

The results are not invariant with variation of the alignment parameters: varying either BIAS or DELWGT can alter the alignment derived by favouring or disfavouring the insertion of gaps during the alignment of the data sequences with the consensus. To see if the alignment routine was robust to small changes in these parameters, the BLUR sequences were aligned under several different sets of alignment parameters. In each case ten alignments were performed, one using each of the ten BLUR sequences as a 'Seed' sequence in the iterative process. The results are summarised in Figure 3, and show that the result shown in Figure 2 is relatively insensitive to changes in BIAS and DELWGT.

Thus the derivation of the Deininger et al (4) consensus by MULTAN is not a feature of a very restricted set of parameters. Varying the consensus-generation parameters TER and NLIM also only has a small effect on the consensus generated, and insignificant effects on alignment.

Although the alignment of the ten BLUR sequences in Figure 2 is 'reasonable', such intuitive judgements are not always a reliable guide to the true optimality of an alignment. To compare this alignment with that reached by an established, pairwise algorithm, each pair of the BLUR sequences was aligned using the algorithm of Wilbur and Lipman (18) with ktuple=1, window size = 40 bases (which values ensure that the Wilbur and Lipman algorithm produces results almost identical to the rigorous alignment of Needleman and Wunsch (10, 13, 18)). The algorithm was implemented in IFIND, a database searching program provided by IntelliGenetics (Palo Alto, California). To compare the results, the Sellers-Waterman metric distance \( a \) (14), defined in this case as

\[
a = (\text{number of mismatches}) + (\text{number of gaps}) + \sum (\text{length of gaps})
\]

was calculated for each pair of sequences, both as aligned by MULTAN and as aligned by the Wilbur and Lipman algorithm. The results are shown in Table 3. The only cases in which the two methods give significantly different results are alignments involving BLUR 1. This is expected, since BLUR 1 contains only the first half of the \( Alu-I \) dimeric repeat. Several other BLUR sequences contain only
Figure 3 Summary of 240 alignment runs on the BLUR sequences, using 24 sets of parameters. MAXDEL=50, NLIM=45, TER=80 in all cases. Five plots are shown, corresponding to runs during which BIAS was 1, 5, 10, 15 or 20 as indicated. In each plot the 'x' axis represents the value of DELWGT used, the 'y' axis represents the number of the BLUR sequence used as the Trial Consensus during that run, and the 'z' axis the number of bases by which the consensus reached during that run varied from that shown in Figure 2. The exact heights of the columns on the 'z' axis are shown in each non-zero column. The shaded area represents alignments in which a 'P' appeared in one or two positions in the consensus generated during that run while a 'Q' occurred in the corresponding position in the consensus shown in Figure 2. This occurred at no more than three positions in all the alignments summarised in Figure 3.

part of this first repeat joined to a complete second repeat; BLUR 10 contains essentially only a second repeat. Thus, although BLUR 1 is best aligned to the first half of most BLUR sequences, it is best aligned with the second half of a few of them, and can be aligned only with the isolated second repeat of BLUR 10. So an optimal alignment of all BLUR sequences would align BLUR 1 with the first half of the consensus, and BLUR 10 with the second half, in which case they would not overlap at all, while a pairwise alignment between the two would result in significant overlap, and hence a different divergence figure. This is an inevitable difference between dual and multiple sequence alignments, and is a problem facing algorithms that rely on the identification of common sub-sequences in order to align several sequences.

Table 3 shows that, when the truncated BLUR 1 is omitted from consideration, the alignment produced by MULTAN is comparable to that produced by a rigorous pairwise alignment routine.
Table 3. Sellers–Waterman metric distances between BLUR sequences as aligned by MULTAN (Table 3A) and as aligned by the Wilbur and Lipman algorithm (Table 3B). Two measures of divergence are given. That in the top right region of each table is calculated assuming that unpaired terminal bases should not be treated as gaps ("No Terminals"), and the lower left half of each table assumes that such unpaired terminal regions are gaps ("With Terminals").

*N.O.* = No Overlap: the sequences do not overlap each other as aligned (Note that this is only relevant if the unpaired termini are ignored. If unpalred terminal bases are counted as gaps, then the total metric distance between non-overlapping sequences derives from considering both sequences to be matched to gaps of their own length). Calculation of the metric distance is described in the text.

---

**Other Sequence Tests.** A number of other sequences have been successfully and reproducibly aligned by MULTAN. Sets of gene coding regions have been aligned for H2A, H2B, H3 and H4 histone, and actin gene families from vertebrates, Drosophila, higer plants and fungi, as have a collection of eubacterial and archaebacterial f-Met-tRNAs and Drosophila satellite DNAs. In this latter case alignments produced by MULTAN were essentially the same as those produced by established pairwise algorithms (Brutlag per com). In all cases satisfactory alignments were achieved using iterative operation with the same alignment and consensus-generating parameters as were used to align the BLUR sequences for Figure 2. The first two exons and the first Intron of nine mammalian β-globin genes have also been aligned, indicating that
**Figure 4**

4A) Alignment of nine mammalian globin genes, from the start of the first exon to the end of the second exon. Only part of the alignment is shown here. Sequences were aligned using BIAS=4, DELW=1, MAXDEL=50, TER=80, NLIM=45. The human β-globin protein sequence ("h.b.";) has been written above the sections of the consensus which refer to coding regions of the genes. The sequences were: (con.) - Terminal consensus to which the sequences were aligned. (1) = Human 0. (2) - Human 0. (3) - Human 0. (4) - Human γ A. (5) - Mouse β major. (6) - Mouse γ 2. (7) - Rabbit β A1. (8) - Rabbit β A2. (9) - Goat β-like pseudogene. Sequences were retrieved from the GenBank database (Bolt, Beranek and Newman for NIH): sequence names are HUMHBB, HUMHBE, HUMHBD, HUMHGBA, MUSHBBM, MUSHBQ2, RABHBB1A1, RABHBB1A2, GOTHBBP respectively.

4B) Detail of the 'splice site' of the consensus sequence from 4A. Top line: MULTAN consensus sequence. Bottom line: consensus splice site sequence from all known eukaryotic splice sites after Mount (1982).
MULTAN can handle sequences in which segments (the 100 base first intron) are highly divergent between data elements. Part of the alignment of the β-globin genes is shown in Figure 4A: the consensus within the first intron contains a number of short strings on ‘N’, signifying a very high degree of polymorphism at these sites, because few sequence features have been preserved in this intron in all mammals. In Figure 4B the sequences flanking the splice sites of the introns (shown by two periods above the cleaved bases in Figure 4A) are compared to the consensus of Mount (9). Again, the same alignment- and consensus-generating parameters were used in this alignment as in the BLUR alignment.

**Capacity and Speed.**

As currently implemented, MULTAN can align up to 50 sequences of ~1kb length. The array holding the sequence data has 1500 base spaces per sequence, but each gap introduced into the consensus sequence causes it, and all but one of the rest of the data sequences, to be shifted towards the ends of the array by an amount equal to the length of the gap. Thus if a total of 1000 bases of inserts is expected in a Terminal Alignment, there will only be room for 500 bases of actual sequence information in the program.

The speed of the program depends largely on the speed of the pairwise alignment routine used. The routine used in MULTAN is fairly fast, and furthermore the time taken to align two sequences increases approximately linearly with sequence length, so that aligning long sequences is not prohibitively time-consuming. The reason for this surprising linearity is that the most time-consuming part of the alignment algorithm used by MULTAN is the determination of the optimal gap to introduce when a Discontinuity is detected. This time is dependant on the number of possible gaps which are studied and is limited by MAXDEL. If MAXDEL is much smaller than the length of the sequence, so that the program will nearly always have scanned up to MAXDEL bases from the Discontinuity before it reaches the end of the sequence, then the time taken to process a Discontinuity will be effectively limited by MAXDEL, and so will be a constant. Thus the time taken to process an entire sequence will be proportional to the number of Discontinuities, a value that is approximately proportional to the sequence length for sequences of equal mutual divergence. The speed of the program is illustrated in Figure 5, where the results of some alignments performed on random sequences to test the length, number and divergence dependence of MULTAN’s operating time are shown. These figures should not be regarded as definitive, but rather as guides: the time taken depends on the type of difference between the sequences as well as the amount of difference because, as mentioned in the section on.
Figure 5. CPU time required by MULTAN to align random sequences. Two sets of values are shown: in all cases the vertical (\(y\)) axis is in units of seconds of CPU time taken to achieve a Terminal Alignment. Panels A, C and E are the times taken by a single alignment step. Panels B, D and F are the times taken to reach a Terminal Consensus during iterative analysis. Panels A and B show the effect of increasing sequence length on analysis time: each data set contained 10 random sequences having a mutual divergence of \(\sim15\%\). Panels C and D show the effect of increasing numbers of sequences on the analysis time: each data set consisted of sequences 500 bases long and having a mutual divergence of \(\sim15\%\). Panels E and F show the effect of sequence divergence on alignment time: each data set contained ten sequences 300 bases long. All sequences were random with respect to base order and dinucleotide composition, and contained 80% A+T. Base changes and small deletions and insertions were made randomly with respect both to the local sequence being altered and the position of the alteration in the overall sequence. Dashed lines are drawn solely to indicate trends in the results, and do not necessarily represent lines of 'best fit' to the data.

The pairwise alignment routine, the program assumes a biological pattern of sequence divergence not found in random sequences. Thus 10 random 300 base sequences with a mutual divergence of \(\sim18\%\) take 279 seconds to align, while the 10 BLUR sequences, also 300 base sequences with a mean mutual divergence of \(\sim18\%\), take only 98 seconds.

Limitations of the Multiple Alignment Algorithm.

Because the alignment routine described here has been optimised to align a
particular type of sequence. It is possible that there will be other sequences which it cannot align. This has been found to be the case. Several sets of sequences have been found with which MULTAN produces less than optimal results. They fall into two classes.

**Failures of Iterative Operation.** The major weakness in the iterative method of alignment is the method used to determine when an acceptable alignment has been reached, so that a Terminal Alignment may be recognised (as already mentioned, it is impractical to show that this is an optimal alignment). In some cases it is found that, after a few iterative cycles, the program becomes locked in a circular pattern whereby alignment 1 gives rise to consensus 1', to which the sequences are aligned to give alignment 2 and consensus 2', to which sequences may again be aligned to give alignment 1 again. Very rarely, loops containing 3 consensus sequences have been seen. The program detects such cycles by testing the present consensus against the previous three, so that this cyclic terminal state does not prevent the arrival at a Terminal Consensus. However, which of the consensus sequences in the final loop is chosen as the 'Terminal' one will depend on where MULTAN enters the final loop, and not on which consensus sequence is 'optimal' for the data set and alignment parameters.

A more serious failure occurs when no terminal state is reached at all. This is found to occur if a large number of fairly diverged sequences, or a small number of very diverged sequences, are aligned. Thus while MULTAN may rapidly align the 10 BLUR sequences, iterative analysis of 50 genomic Alu-I sequences can only reach a Terminal state if the starting parameters are significantly altered from those used to align the BLUR sequences. Although MULTAN can align the first two exons and the first intron of nine mammalian \( \beta \)-globin genes, addition of two chicken \( \beta \)-globin genes to the list causes failure of iterative operation. The reason for this is that regions of high polymorphism, such as the introns of \( \beta \)-globin genes or the flanking regions of Alu-I elements (the latter share essentially no homology), can give rise only to very ambiguous consensus sequences, to which the data sequences may be aligned in a large number of ways, each with an essentially identical number of matches between the sequences and the consensus. Thus each alignment gives rise to a new consensus which, although different from the previous consensus, does not fit the data sequences any better. This problem can be overcome by setting both TER and NLIM to high values, thus eliminating terminal polymorphism from consideration and setting the consensus for all internal polymorphic regions to strings of 'N'. For example, 50 Alu-I sequences can be iteratively analysed if NLIM is set to 75%. but 15% of the bases in the resulting consensus are 'N'.
Table 4 Other functions of MULTAN. Functions other than those associated with iterative or single-step alignment of the sequence data are summarised.

1) **Comparison of sequences.**
   a) Sequences may be compared by base difference, with or without weighting for multiple hits or unequal mutation rates.
   b) Sequences may also be compared using the Sellers-Waterman metric distance (11).
   c) The comparison figures generated above may be used to construct a Distance-Wagner tree using the algorithm of Farris (4).

2) **Sequence Display.** Sequences may be displayed on their own or as aligned to each other, or the aligned data may be set to a holding file for storage against system failure or as data for other programs.

3) **Concensus Generation.** The concensus may be re-generated to specified levels of ambiguity, the Adherence to the concensus may be examined, and the concensus itself listed to the screen or output to a file in Stanford SEQ format (2).

4) **Sequence selection.** Sequences may be selectively removed from consideration for any of the above functions. The names and comments (if any) associated with sequences may be examined.

5) **Parameter alteration.** User-defined parameters may be altered from their default values interactively.

6) **Self-documentation.** Interactive 'Help' is available at all prompts to explain what is required at that prompt and what options are available at the major prompts, together with the basics of how the algorithms operate and how they are implemented in the program.

Thus MULTAN can analyse such sequences, but only at the cost of losing some fine structure in the resulting alignment.

Both causes of failure of iterative operation may also be avoided by using the program non-iteratively. In this mode the user executes an Alignment Step and then generates a new concensus, examining the result at each stage to see if an acceptable alignment has been achieved. Although this might appear to be less 'rigorous' than iterative operation, it must be remembered that the iterative algorithm does not produce a 'rigorous' result either, although it does rely on different criteria to determine the acceptability of an alignment and a concensus than would a user executing single-step analysis, and one uninfluenced by a priori knowledge of biological or other constraints.

**Failures of Non-Iterative operation.** MULTAN is written around the assumption that all the sequences in a data set can be aligned; thus MULTAN will align all sequences presented to it, and so in a sense cannot fail to complete an isolated Alignment Step. However, extremely divergent sequences will not give satisfactory alignments. For example, the promoter regions of 18 eukaryotic genes (the first 10 bases of the mRNA and the preceding 120 bases) were not satisfactorily aligned. A 'TATA'-like concensus was identified, but the 'CCAAT'
box and the first base of the mRNA (nearly always 'A') were not. Thus these sequences, sharing 10 bases in 130, must be regarded as beyond MULTAN's ability. However such problems are well suited to the sub-sequence identification programs mentioned in the Introduction, and indeed are the type of problem for which these programs were designed.

Other Functions of MULTAN.

As well as aligning sequences, MULTAN has a number of other functions which analyse the aligned sequences and makes the program easier to operate. Several methods of sequence comparison are available, including the Sellers-Waterman metric measurement (14) (see above: Table 2A was calculated directly from the aligned BLUR sequences by MULTAN). These divergence figures may be used to construct a dendrogram of the sequences using the Distance-Wagner algorithm of Farris (5). Thus these analyses are performed without the need to reformat the extensive MULTAN output for re-entry into other programs. The program can also generate a concensus containing varying degrees of ambiguity (both the concensus sequences in Figure 2 are MULTAN output), and provide statistics on the adherence of the data sequences to the concensus. These subsidiary functions of MULTAN are summarised in Table 4. The program is self-documenting with extensive on-line 'help' for the user, and robust to erroneous user-entered data. The parameters TER, NUM, BIAS, DELWGT and MAXDEL are interactively alterable, as are a variety of other parameters concerning the format and quantity of results that MULTAN displays to the user.

DISCUSSION

I have presented a program which can align a number of related nucleic acid sequences and perform some basic analyses on the result. The algorithm used is heuristic, and is limited with respect to the types of data that it can handle. However most sequences tested can be aligned successfully by the program with minimal user interference, and only the most diverged (a set of eukaryotic promoters) fail to produce an acceptable alignment. As it is clear that eukaryotic promoters cannot be identified unambiguously from sequence data alone without experimental evidence assigning the site of mRNA initiation, this is not a surprising failure. The program aligns the ten BLUR sequences essentially as well as an established pairwise-matching algorithm, and in an acceptably short time.

Although at present MULTAN is restricted to analysing DNA (or RNA) sequences, the algorithm should be generalisable to other types of sequential data in which sequences of elements may be related to each other by
transformations involving small deletions, insertions and interchanges of elements. To make use of this potential, the program is presently being modified to align protein data, and to align protein-coding DNA sequences by aligning triplets rather than individual bases, so that gaps are only introduced into the data sequences or into the consensus between codons. These added facilities will complement the existing program, which is designed to align non-coding DNA sequences in which there is no a priori knowledge of probable or improbable sites for gaps to be introduced into a consensus.

The rapid rise in the number of published sequences of repetitive elements like the \textit{Alu-1} family, of multi-gene families and of single sequences from many related species makes it increasingly important that workers handling large numbers of related sequences have access to methods for examining their relationships. This paper demonstrates that MULTAN can fill that role for a large range of DNA sequence types, and thus make extensive analysis of large sequence families a possibility.

Acknowledgements.

I wish to thank Drs Doug Brutlag, Larry Kedes, Hugo Martinez and Dan Wells for their encouragement and suggestions during this work. This work was performed during the tenure of a Postdoctoral Fellowship from the Muscular Dystrophy Association of the USA, and was supported by grants from the Veterans Administration and the MDA to Laurence H. Kedes, in whose laboratory MULTAN was developed. The author acknowledges the generous use of the facilities of the SUMEX-AIM computer, and the hospitality of the MOLGEN project and the Heuristic Programming Project of the Department of Computer Science at Stanford University, and the generosity of Intelligenetics (Palo Alto, California) in providing DNA sequence databases and database searching programs for this work.

Current address and address for correspondence: Department of Biochemistry, University of Bath, Claverton Down, Bath BA2 7AY, UK

References
