Volume 14 Number 22 1986 Nucleic Acids Research

Multiple sequence alignment by consensus

Michael S.Waterman

Departments of Mathematics and Molecular Biology, University of Southern California, Los Angeles, CA 90089-1113, USA

Received 16 September 1986; Revised and Accepted 20 October 1986

ABSTRACT

An algorithm for multiple sequence alignment is given that matches words of length and degree of mismatch chosen by the user. The alignment maximizes an alignment scoring function. The method is based on a novel extension of our consensus sequence methods. The algorithm works for both DNA and protein sequences, and from earlier work on consensus sequences, it is possible to estimate statistical significance.

INTRODUCTION

Sequence alignment is an important problem motivated by molecular biology and many computer algorithms have been devised to accomplish alignments. Most of the results have been for two sequences. The analytical work began with the paper of Needleman and Wunsch(1) who solved the problem of maximum similarity alignment for two sequences. Later Sellers(2) gave a related dynamic programming algorithm for minimum distance alignment of two sequences. These algorithms were extended to cover multiple insertions and deletions by Waterman et al.(3). In Waterman (1984)(4) the subject of sequence comparisons is reviewed. It is clear that, while the two sequence case has been adequately solved for many cases, the situation for alignment of more than two sequences is quite different. Next we review the history of approaches to multiple sequence alignment. Then we give a new algorithm for multiple sequence alignment that is based on a novel extension of our consensus sequence methods.

Sankoff(5) gave the first treatment of which we are aware that considered multiple sequence alignment. His method requires a tree relating the sequences and employs dynamic

programming as well as parsimony. For comparing R sequences of length N, the method takes time proportional to $2^R N^R$ ($O(2^R N^R)$) and storage $O(N^R)$. Only small sequences are practical even when R=3. Sankoff et al.(6) apply the algorithm, modified somewhat, to 5S sequences. In Waterman et al.(3) a similar algorithm is presented which does not assume a tree. The time and space requirements are similar.

More recently, Waterman and Perlwitz(7), apply the geometry of geodesics to sequence alignment. The idea of sequence is extended so that a "nucleotide" is a mixture of A,C,G,T and ϕ (deletions). The basic algorithm is based on dynamic programming, and the sequences are aligned and merged two at a time. When the relationships between the sequences are understood, as when a correct evolutionary tree is available, the method works very well. Still it is a pairwise algorithm and the final alignment is dependent on the order in which the sequences are processed.

Computer science has studied what they call string matching problems, beginning with Wagner and Fischer (they prefer "string" to sequence)(8). In Itoga(1981)(9) the string merging problem is studied, and Hsu and Du(1984)(10) consider the longest common subsequence of a set of strings. None of these authors seem aware of the earlier results described above.

Sobel and Martinez(11) approach sequence matching as a regions problem, where their algorithm is based on locating all exact repeats of patterns which occur in the sequence set. The method of finding repeats takes O(NlogN) operations(12). The best set of regions making up the alignment is found by longest path methods from computer science. This is perhaps the only practical method available for more than three sequences until the present paper.

Some interesting results have been obtained for protein sequences. There it is thought that gaps should receive a constant penalty, regardless of length. Fredman(13) improved the algorithms to $O(N^3)$ for aligning 3 sequences of length n with the constant gap penalty, whereas the direct extension of Waterman et al. is $O(N^6)$. Murata et al.(14) give a similar

improvement. Johnson and Doolittle(15) give a method, not guaranteed to be globally optimal, which is based on the progressive evaluation of selected segments from each sequence.

There are two recent methods designed to align multiple sequences that require more discussion. That of Sobel and Martinez(11) is based on aligning segments common to the sequences; it is frequently of interest to align patterns with some degree of mismatch, insertion, and/or deletion. There may be no segment common to all the sequences. The method of Johnson and Doolittle(15) is based on aligning segments found within a window of width W and for R sequences of length N has running time proportional to $O(R(N-W)W^{R-1})$. For small R this is a practical method, but is not practical for a larger number of sequences. Next we present our method to overcome some of these difficulties.

DESCRIPTION OF THE METHOD

We begin with a set of R sequences of length N

$$\begin{array}{l} a_{1,1}a_{1,2} \cdots a_{1,N} \\ a_{2,1}a_{2,2} \cdots a_{2,N} \\ \cdots\cdots\cdots \cdots \cdots \\ a_{R,1}a_{R,2} \cdots a_{R,N} \end{array}$$

These sequences can be taken to be initially aligned on some biologically determined feature. The alignment is, of course, unknown except approximately. It is the purpose of this paper to give an algorithm aligning the sequences by matching or aligning on words of a given size. The usual methods of sequence analysis align on single letters, that is words of length 1.

A concept basic to our algorithm is that of consensus word. The definition has been given in earlier work (Waterman et al. (1984)(16) and Galas et al.(1985)(17)) and will be briefly given here. First, take a fixed word size(length) k and a word w of length k. There are 4^k such words in DNA and 20^k in proteins. Next, define the window width W. This parameter gives the width of sequence in which a word can be found and thus defines the

amount of shifting allowed in matching consensus words. The sequences starting at column j+1 with window width W appear as

$$\begin{array}{l} a_{1,j+1}a_{1,j+2} \cdots a_{1,j+W} \\ a_{2,j+1}a_{2,j+2} \cdots a_{2,j+W} \\ \cdots\cdots \quad \cdots\cdots \quad \cdots\cdots \\ a_{R,j+1}a_{R,j+2} \cdots a_{R,j+W} \end{array}$$

First, we search the first sequence of the window for matches to our word w. An exact match to w is called a d=0 neighbor while a 1-letter mismatch from w is called a d=1 neighbor, and so on. It is possible to include insertions and deletions in this list of neighbors. We may decide, e.g., to limit the amount of mismatch to d=0,1,2 and not find w in a portion of sequence unless it is within this neighborhood. Let $q_{w,d}$ equal the number of lines that the best occurrence of w is as a d-th neighbor. Each of these occurrences receives weight λ_d. The score of word w in this window is

$$s_{j+1,j+W}(w) = \sum_d \lambda_d q_{w,d}.$$

A best word is word w^* satisfying

$$s_{j+1,j+W}(w^*) = \max_w s_{j+1,j+W}(w).$$

The idea of the algorithm is to align on consensus words, attempting to maximize the sum of the scores of the words. Before the practical algorithms are presented, a more general concept of alignment on words is presented.

Now we define a partial order on words. The words w_1 and w_2 satisfy $w_1 \prec w_2$ if the occurrences of w_1 in sequence i are to the left of the occurrences of w_2 in sequence i (and do not intersect) for i=1 to R. It is not necessary for w_1 or w_2 to have occurrences in all sequences. Implicit in the definition is a window width W and neighborhood specification. The goal of an optimal alignment is to find words w_i which satisfy

$$\max\{\Sigma_{i\geq 1} s(w_i) : w_1 \prec w_2 \prec \ldots\}.$$

(It is frequently desirable to require $s(w_i) \geq c$ for all i, where c is some cutoff value.) It is not possible to accomplish this

goal in reasonable time, but it is possible to come quite close. We now define two practical algorithms.

Next $w_1|w_2$ means that consensus words w_1 and w_2 can be found in non-overlapping windows, each word satisfying as usual the window width and neighborhood constraints. The modified optimization problem is to satisfy

$$T = \max\{\Sigma_{i\geq 1} s(w_i) : w_1|w_2|\ldots\}.$$

There is a straightforward recursion to find T. Let T_i be the maximum sum for the sequences from base 1 to base i:

$$\begin{matrix} a_{1,1} & \cdots & a_{1,i} \\ a_{2,1} & \cdots & a_{2,i} \\ \cdots & \cdots & \cdots \\ a_{R,1} & \cdots & a_{R,i} \end{matrix}$$

Then T_i satisfies

$$T_i = \max\{T_j + s_{j+1,i} : i-W+1 \leq j \leq i-k\}$$

and $T_{-W} = T_{-W+1} = \ldots = T_0 = T_1 = \ldots = T_{k-1} = 0$. Also $s_{x,y} = 0$ if $y-x+1<k$. This algorithm runs in time approximately proportional to NW^2RB where B = neighborhood size. Here the factor of WRB accounts for the consensus word algorithm with a window width W. (This is an overestimate since the actual windows vary from k to W in width.)

If much shifting is necessary to match the sequences, T is an underestimate and misses some of the relevant matching. To overcome this problem, the definition of T is modified to

$$S_i = \max\{S_j + \hat{s}_{j+1,i} : i-W+1 \leq j \leq i-k\}.$$

where $\hat{s}_{j+1,i}$ is the largest scoring consensus word in the window from j+1 to i such that all occurrences of the consensus word are to the right of the consensus words for S_j. This algorithm is not guaranteed to be equal to the global maximum, but it is much more useful than T.

EXAMPLE

The algorithm is now illustrated with an alignment of 34 5S sequences from E. coli and related organisms that were obtained

Figure 1. An alignment of 34 5S rRNA sequences by algorithm S with k=4, W=6, and up to 1 mismatch.

from the collection of Olsen and Pace. They can be found in GenBank or the review paper of Erdmann et al.(18).

An alignment of these sequences by algorithm S with k=4 and W=6, allowing up to 1 mismatch is given in Fig. 1. The parameter $\lambda_d = (k-d)/k$, where d equals the number of mismatches. Algorithm T which does not allow overlapping windows has score 304.75, while algorithm S has score 364.00. Some scores for various choices of the parameters are given next:

W = window width	k = word size	# mismatches	S
6	4	0	117.00
6	4	1	364.00
8	4	0	196.00
8	4	1	487.25
8	6	0	23.00
8	6	1	124.00
8	6	2	269.17
10	6	0	82.00
10	6	1	213.67
10	6	2	329.00

DISCUSSION

The existing multiple sequence alignment programs based on our algorithm are written for DNA sequences, but they can easily be extended to proteins. (We plan to do this.) The largest protein word possible is k=3, since 20^3=8,000, while words with k=8 or 9 is possible with DNA. Whenever the amino acid alphabet is decreased from the usual 20-letter alphabet, k can be increased.

Another feature we have not included in our program is a cost for unmatched letters (deletions/insertions). This could easily be done as Martinez does, for example, but we are not yet persuaded of the necessity here.

Statistical significance is always an issue in sequence alignment. Frequently the sequences are randomly permuted, and the optimal alignment scores obtained from the random sequences. Statistical significance is estimated from these scores. Here we have a more direct approach. It is possible to calculate the

probability that a word occurs in the specified neighborhood and in the window of one sequence. Then the binomial distribution or the theory of large deviations (for large R)(16) can be used to calculate the probability of ever seeing a matched word in L out of the R sequences. By requiring this probability to be small (e.g., .01), we can be assured that every matched word in an alignment is significant at that level of significance.

The implementation for DNA sequences is written in the C language and is available from the author. Please inquire for details.

ACKNOWLEDGMENTS

The 5S sequences were received from Gary Olsen and Norm Pace. Assistance from Mark Eggert and Felicitas Smith is greatly appreciated.

This work was supported by grants from the System Development Foundation and the National Instutes of Health(GM 36230).

REFERENCES

1. Needleman, S.B. and Wunsch, C.D. (1970). J. Mol. Biol. 48, 444.
2. Sellers, P. (1974). SIAM J. Appl. Math. 26, 787.
3. Waterman, M.S., Smith, T.F., and Beyer, W.A. (1976). Adv. Math. 20, 367.
4. Waterman, M.S. (1984). Bull. Math. Biol. 46, 4, 473-500.
5. Sankoff, D. (1975). SIAM J. Appl. Math. 78, 35-42.
6. Sankoff, D., Cedergren, R.J., and Lapalme, G. (1976). J. Mol. Evol. 7, 133-149.
7. Waterman, M.S. and Perlwitz, M. (1984). Bull. Math. Biol. 46, 4, 567-577.
8. Wagner and Fischer. (1976). J. of the Assoc. for Computing Machinery 23, 50-57.
9. Itoga, S.Y. (1981). BIT 21, 20-30.
10. Hsu, W.J. and Du, M.W. (1984). BIT 24, 45-59.
11. Sobel, E. and Martinez, H. (1986). Nucl. Acids Res. 14, 1, 363-374.
12. Martinez, H. (1983). Nucl. Acids Res. 11, 4629-4634.
13. Fredman, M.L. (1984). Bull. Math. Biol. 46, 4, 553-566.
14. Murata, M., Richardson, J., and Sussman, L. (1985). Proc. Natl. Acad. Sci. USA 82, 3073-3077.
15. Johnson, M.S. and Doolittle, R. (in press). J. Mol. Evol.
16. Waterman, M.S., Arratia, R., and Galas, D. (1984). Bull. Math. Biol. 46, 515-527.
17. Galas, D.J., Eggert, M., and Waterman, M.S. (1985). J. Mol. Biol. 186, 117-128.
18. Erdmann, V.A., Wolters, J., Huysmans, E., and R. de Wachter. (1985). Nucl. Acids Res., 13, r105.