CABIOS

Vol.1. no.2. 1985
Pages 83 – 87

Optimal alignments of biological sequences on a microcomputer

Kazutada Watanabe, Yoshio Urano and Taiki Tamaoki

Abstract

An algorithm and a program have been developed which enable optimal alignments of biological sequences on an 8-bit microcomputer. The compiled program can process sequences up to 1000 residues on a Commodore 64. Since this program was written originally in the BASIC language, it may readily be adapted to other microcomputers with small changes.

Introduction

Recent advances in the isolation and characterization of DNA sequences from a variety of sources have increased the necessity for investigating the degree of similarities among these sequences. Needleman and Wunsch (1970) have devised an elegant algorithm that determines the maximum alignments between two sequences. This algorithm has been modified by Sellers (1974) to measure evolutionary distances and further modified by Waterman *et al.* (1976) to deal with any length of gaps. Although this algorithm is highly useful for comparing a pair of DNA sequences, its use has been limited to main frame computers or minicomputers.

There has been an increasing use of microcomputers in research as they are much less expensive and easier to use than main frame computers. It is, therefore, highly desirable to develop a program which will allow the optimal alignments of DNA sequences on a microcomputer. Gotoh (1982) has reported an algorithm which saves a huge space of memory when the gap weight has a form of $w_k = uk + v$ (k is the gap length and u,v ($u \geq 0, v \geq 0$) are constants). This algorithm yields an evolutionary distance in a very small memory space. However, the D matrix which produces an evolutionary distance must be backtracked in order to obtain optimal alignments (Gotoh, 1982; Goad and Kanehisa, 1982; Smith *et al.*, 1981; Taylor, 1984) and the amount of elements necessary for the alignments is more than a microcomputer can accommodate. Therefore, it is still extremely difficult to obtain the optimally matched alignments on a microcomputer through direct application of this algorithm. For this reason, little application of a microcomputer to obtain optimal alignments of DNA sequences has been accomplished so far (for example see Söll and Roberts, 1984).

Department of Medical Biochemistry, University of Calgary, Calgary, Alberta T2N 4N1, Canada

In this paper, we describe an improved algorithm and a program which can investigate the similarity between a pair of DNA sequences up to 1000 residues on an 8-bit microcomputer and print out the optimal alignments. This program is also applicable to align amino acid sequences.

Systems and Methods

Hardware

A Commodore 64 equipped with a 5-inch, single-sided, single-density disk drive (Commodore 1541) was used as a microcomputer. The printer and CRT used were Commodore 1526 and Zenith data systems, respectively.

Software

The DNA sequence data were prepared by a word processing program 'Paper Clip' to produce sequential files. The alignment program was written in the BASIC language for Commodore 64 and compiled by the BASIC compiler 'PETSPEED' (Oxford Systems, UK).

Algorithm for a microcomputer

In order to apply the optimal alignment algorithm of Waterman *et al.* (1976) to a small microcomputer, it is essential to make the memory size used in the program small. By the application of Gotoh's algorithm (1982), a huge amount of memory can be saved. However, the route for the optimal alignments must be backtracked from the right bottom elements of the D matrix guided by the path matrix (Gotoh, 1982; Goad and Kanehisa, 1982; Smith *et al.*, 1981; Taylor, 1984). There is no space in a small microcomputer to store either the whole D matrix or the path matrix.

In order to backtrack the route in a microcomputer, the D matrix was divided into small submatrices and parts of the submatrices are calculated twice. This procedure made it possible to obtain optimally aligned sequences in a small memory size. Actually, D matrix of $(M+1)\times(N+1)$ order was divided into small submatrices D_{ij} of $(L+1)^2$ order (Figure 1). M and N are the lengths of sequences under comparison and L is determined from the maximum matrix size which the microcomputer can accommodate in the core memory with the program and other variables. Here, $0 \leq i < M/L$, $0 \leq j < N/L$ (i and j are integers). The *L* value used for Commodore 64

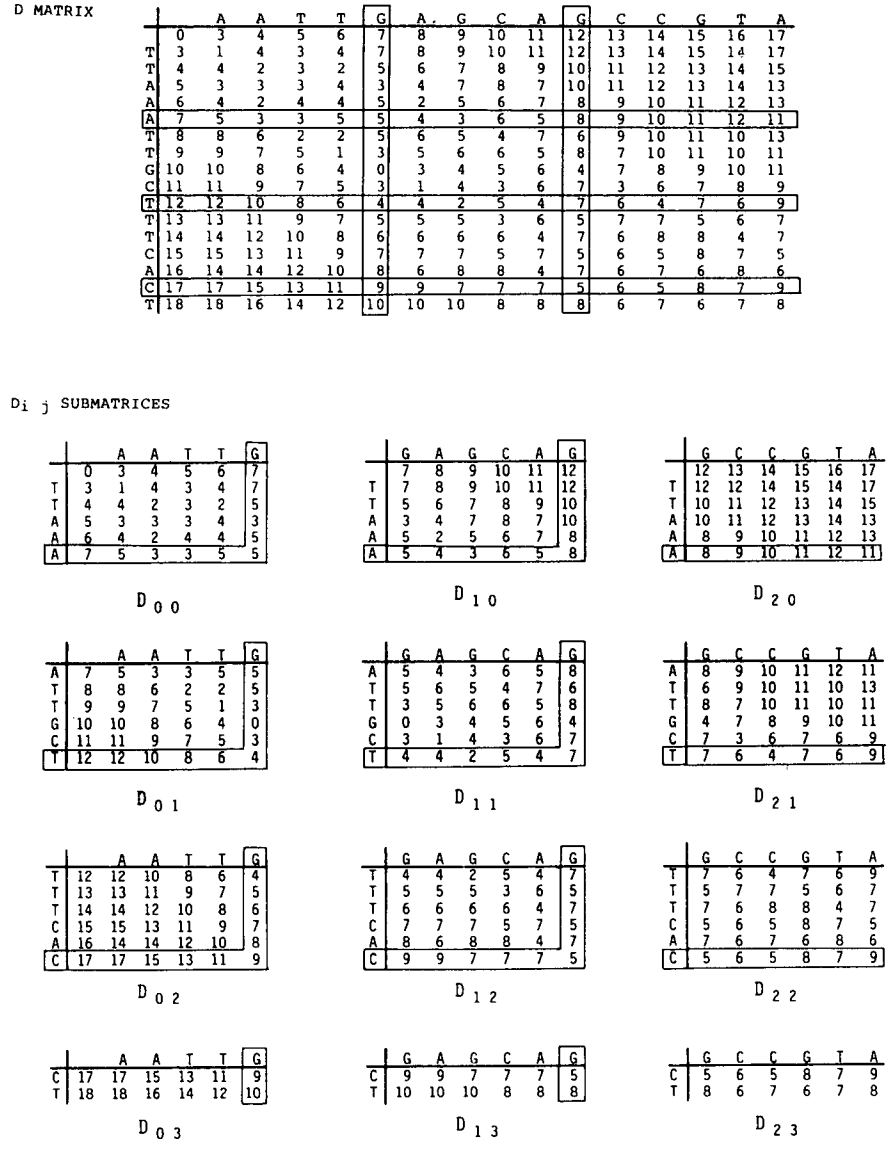

Fig. 1. Submatrices in the D matrix. The D matrix was divided into small submatrices of D_{ij}. The D_{ij} submatrices of (6×6) order (i.e., L=5) are shown here for explanation. In the program, the L value of each D_{ij} submatrix is 90. The arrays in the boxes were shared with the neighboring submatrices and stored in the diskette in the first run. The weight values used here were d=−1 for matched residues, d=+1 for mismatched residues, and k+2 for gap.

was 90. These submatrices shared the same element at the border arrays with the neighboring submatrices (arrays in the boxes in Figure 1).

The D matrix by Waterman *et al.* (1976) is described by the following relations.

$$D(R,S)=\mathrm{Min}[D(R-1,S-1)+d,\ P(R,S),\ Q(R,S)]$$

$$P(R,S)=\mathrm{Min}_{1\le k\le R}[D(R-k,S)+w_k]$$

$$Q(R,S)=\mathrm{Min}_{1\le k\le S}[D(R,S-k)+w_k]$$

(d is a matching weight between two aligned residues.)

P(R,S) (or Q(R,S)) can be written as follows.

$$P(R,S)=\mathrm{Min}\{\mathrm{Min}_{1\le k\le A}[D(R-k,S)+w_k],\mathrm{Min}_{A+1\le k\le R}[D(R-k,S)+w_k]\}$$

$$=\mathrm{Min}\{\mathrm{Min}_{1\le k\le A}[D(R-k,S)+w_k],\mathrm{Min}_{1\le k\le R-A}[D(R-A-k,S)+w_{k+A}]\}$$

According to Gotoh's algorithm (1982), D matrix calculation can be done in MN steps in the case that the gap weight has the form of $w_k=uk+v$. When this form of gap weight was applied,

$$P(R,S)=\mathrm{Min}\{\mathrm{Min}_{1\le k\le A}[D(R-k,S)+w_k],P(R-A,S)+uA]\}$$

If R=(L × i)+A and S=(L × j)+B ($1\le A\le L$, $1\le B\le L$), then $D(R,S)=D_{ij}(A,B)$, $P(R,S)=P_{ij}(A,B)$ and

$Q(R,S)=Q_{ij}(A,B)$ in the D_{ij} submatrix. Therefore,

$$P(R,S)=P_{i\,j}(A,B)=\underset{1\le k\le A}{\mathrm{Min}}\{\mathrm{Min}[D_{i\,j}(A-k,B)+w_k],P_{i\,j}(0,B)+uA\}$$

and

$$Q(R,S)=Q_{i\,j}(A,B)=\underset{1\le k\le B}{\mathrm{Min}}\{\mathrm{Min}[D_{i\,j}(A,B-k)+w_k],Q_{i\,j}(A,0)+uB\}$$

From these relations, it is possible to generate the $D_{i\,j}$ submatrix independent of other submatrices once the values of $D_{i\,j}(0,0)$, $D_{i\,j}(A,0)$, $D_{i\,j}(0,B)$, $P_{i\,j}(0,B)$, $Q_{i\,j}(A,0)$ $(A,B=1, 2, \ldots, L)$ are given. Since the submatrices were overlapped at the borders in the case of $1\le i<M/L$ and $1\le j<N/L$,

$D_{i\,j}(0,B)=D_{i-1\,j}(L,B)$, $P_{i\,j}(0,B)=P_{i-1\,j}(L,B)$

$D_{i\,j}(A,0)=D_{i\,j-1}(A,L)$, $Q_{i\,j}(A,0)=Q_{i\,j-1}(A,L)$

$D_{i\,j}(0,0)=D_{i\,j-1}(0,L)$

Then, the elements in D submatrix can be determined from the relations below.

$$D_{i\,j}(A,B)=\mathrm{Min}[D_{i\,j}(A-1,B-1)+d,\ P_{i\,j}(A,B),\ Q_{i\,j}(A,B)]$$

$$P_{i\,j}(A,B)=\underset{1\le k\le A}{\mathrm{Min}}\{\mathrm{Min}[D_{i\,j}(A-k,B)+w_k],\ P_{i-1\,j}(L,B)+uA\}$$

$$Q_{i\,j}(A,B)=\underset{1\le k\le B}{\mathrm{Min}}\{\mathrm{Min}[D_{i\,j}(A,B-k)+w_k],\ Q_{i\,j-1}(A,L)+uB\}$$

Therefore, a part of any submatrix is reproduced without calculation of the D matrix from the beginning if the values of $D_{i\,j-1}(0,L)$, $D_{i-1\,j}(L,B)$, $P_{i-1\,j}(L,B)$, $D_{i\,j-1}(A,L)$ and $Q_{i\,j-1}(A,L)$ $(A,B=1, 2, \ldots, L)$ have been memorized beforehand. In the case of $i=0$ and/or $j=0$, the following values were used as initial conditions.

$D_{i\,0}(A,0)=P_{i\,0}(A,0)=uA+v+(L\times i)$ $(1\le A\le L)$

$D_{0\,j}(0,B)=Q_{0\,j}(0,B)=uB+v+(L\times j)$ $(1\le B\le L)$

$D_{0\,0}(0,0)=0$

In practice, the complete set of elements in the D matrix were first obtained through the calculation of all submatrices. In this calculation, Gotoh's algorithm (1982) was applied to each submatrix. After calculation of each submatrix, the values of $D_{i\,j}(0,L)$, $D_{i\,j}(L,B)$, $P_{i\,j}(L,B)$, $D_{i\,j}(A,L)$, $Q_{i\,j}(A,L)$ obtained were stored on a diskette. It was not necessary to input these values from the diskette in order to obtain the other submatrices in the first calculation, since these submatrices were calculated successively. These values were used to backtrack the path matrix later.

After calculation of all submatrices, parts of the submatrices were recalculated using the boundary values stored in the diskette. The recalculation was started from the submatrix at the rightmost bottom of the D matrix (Figure 2) in order to obtain the routes for optimal alignments. Actually, the submatrix of $D_{i\,j}$ was first recalculated with the path matrix using the values $D_{i-1\,j}(L,B)$, $P_{i-1\,j}(L,B)$, $D_{i\,j-1}(A,L)$, $Q_{i\,j-1}(A,L)$ and $D_{i\,j-1}(0,L)$ $(A,B=1, 2, \ldots, L)$ which were stored in the diskette. Here, i and j are integers of $(M-1)/L-1<i\le(M-1)/L$ and $(N-1)/L-1<j\le(N-1)/L$, respectively.

When the second calculation of this submatrix was finished, a route for the optimal alignment was traced up to the edge

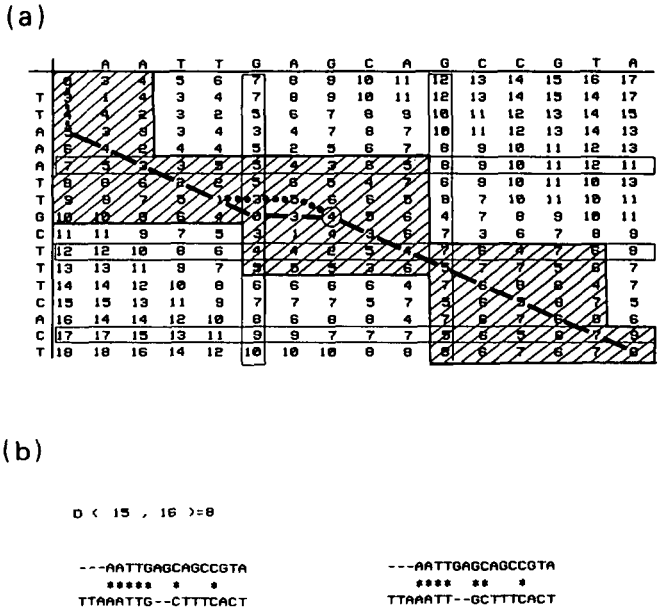

Fig. 2. **(a)** Submatrix calculation in the second run and the route for the optimal alignment. In order to obtain optimal alignments, parts of the submatrices were recalculated (hatched area) starting from the rightmost bottom of the D matrix ($D_{2\,3}$ submatrix in this case). A route for the first alignment is indicated by solid line. The positions and elements in path matrix at the branching points (in the circles) were memorized. Another possible route is indicated by broken line. **(b)** Two possible alignments obtained by this program. Evolutionary distance (D(M,N)) is also printed out.

of the submatrix directed by the path matrix. Then, the submatrix and path matrix were erased for the next submatrix which was linked to the alignment route obtained. Through successive input of the boundary values from the diskette, calculation of the submatrices terminated at the origin of the D matrix. Only parts of the submatrices which are related to the alignments were recalculated (Figure 2).

At the end of the recalculation, only one route for an optimal alignment and the branching points which lead to the other alignments were memorized in the core memory of the computer. The first alignment was obtained by this route. After a print-out of this alignment, parts of the submatrices were recalculated from the branching points to obtain the other alignments. Possible alignments were thus obtained by repeating this procedure.

Figure 3 shows a typical output of the optimal alignments of nucleotide sequences of human beta-globin and mouse beta(minor)-globin genes (Lawn *et al.*, 1980; Konkel *et al.*, 1979) obtained by this program. The weight of alignments (d value) was fixed to be -1 for matched residues, $+1$ for mismatched residues, and the gap weight (w_k) was $k+2$ in the program. Execution times are presented in Table I.

The total size of the BASIC program including the matrices and variables was 31380 bytes and compiled program was 37154 bytes. These sizes can be changed according to the size of submatrices.

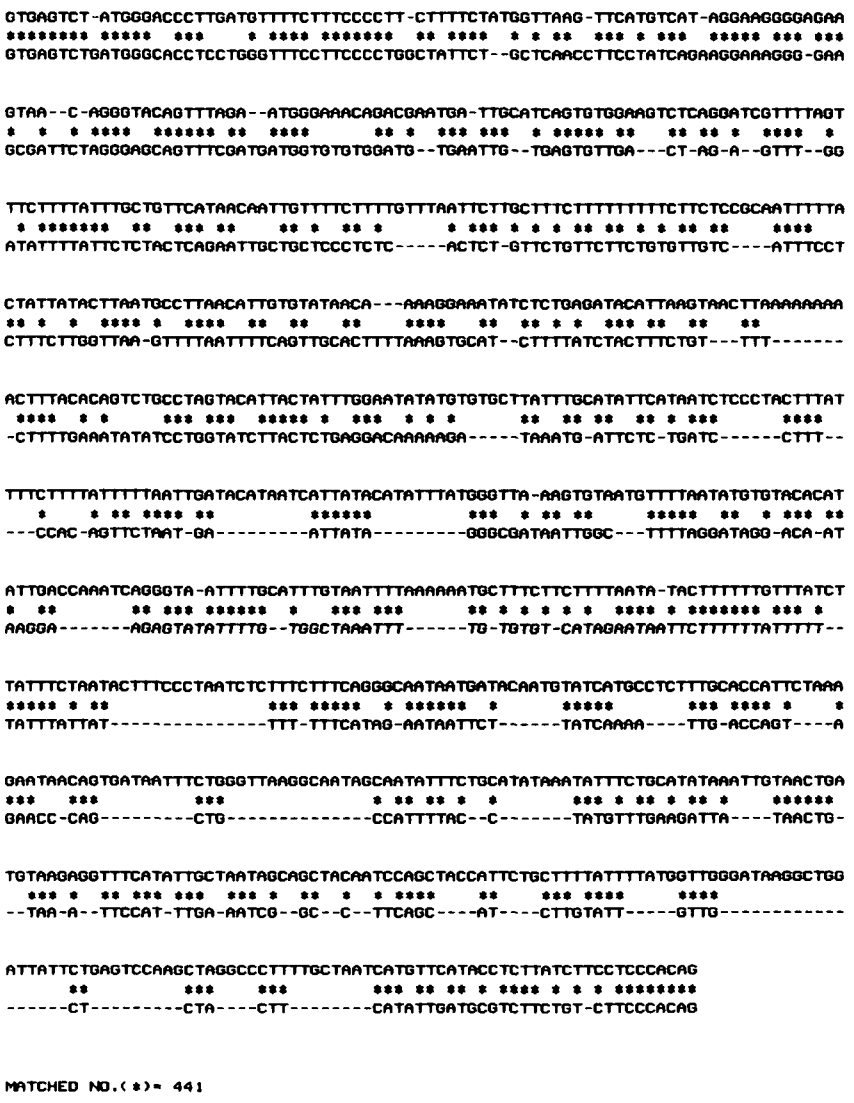

Fig. 3. An example of the optimal alignment printed out by this program. These sequences are 850 bp from the second intron of the human beta-globin gene (Lawn *et al.*, 1980) and 628 bp from the second intron of the mouse beta(minor)-globin gene (Konkel *et al.*, 1979).

Table I. Execution times of the program. The execution time was measured starting from the moment when the input of DNA sequences into the microcomputer from the diskette was completed.

DNAs (bp)	D matrix calc.	Backtracking	Total
45	–	–	33 s
90	–	–	2 min 04 s
180	5 min 18 s	4 min 36 s	9 min 54 s
360	26 min 42 s	9 min 15 s	35 min 57 s
720	112 min 37 s	18 min 23 s	131 min 00 s
1000	222 min 00 s	25 min 29 s	247 min 29 s

Operation of the program

At first, sequence data must be prepared using the word processing program 'Paper Clip' or any other program in the form of sequential files. The alignment program only requires sequential files of sequence data.

In the alignment program, sequence names (file names) and sequence positions to be compared must be typed-in from the keyboard. After this has been done, the program will start calculation.

Applicability of the program

It is not difficult to apply this program to other microcomputers through minor changes in the program, mainly in the file control. However, a compiler program is necessary to deal with DNAs of large sizes. For example, the compiled program can execute 1000 bp DNAs in 4 h (Table I). On the other hand, BASIC program takes about 66 hours to process the same length of DNAs.

It is also possible to deal with sequences of longer than 1000

residues with small modifications in the program at the expense of execution time. A program for Commodore 64 or a print-out of the BASIC program will be available on request. For the program, please send a blank 5-inch, single-sided diskette to us.

Acknowledgements

We thank Mr. David Pot and Mr. Marc Nixon for helpful discussion and technical assistance. This work was supported by the National Cancer Institute of Canada and the Medical Research Council of Canada. T.T. is a Research Associate of the National Cancer Institute of Canada. Y.U. is a recipient of Alberta Heritage Medical Research Fellowship.

References

Goad,W.B. and Kanehisa,M.I. (1982), Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries, *Nucleic Acids Res.*, **10**, 247-263.

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

Konkel,D.A., Maizel,J.V.,Jr. and Leder,P. (1979), The evolution and sequence comparison of two recently diverged mouse chromosomal β-globin genes, *Cell*, **18**, 865-873.

Lawn,R.M., Efstratiadis,A., O'Connell,C. and Maniatis,T. (1980), The nucleotide sequence of the human β-globin gene, *Cell*, **21**, 647-651.

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

Sellers,P.H. (1974), On the theory and computation of evolutionary distances, *SIAM J. Appl. Math.*, **26**, 787-793.

Smith,T.F., Waterman,M.S. and Fitch,W.M. (1981), Comparative biosequence metrics, *J. Mol. Evol.*, **18**, 38-46.

Söll,D. and Roberts,R.J. (eds.) (1984), The applications of computers to research on nucleic acids II, part 1 and 2, IRL Press, Oxford, UK, pp. 569-854.

Taylor,P. (1984), A fast homology program for aligning biological sequences, *Nucleic Acids Res.*, **12**, 447-455.

Waterman,M.S., Smith,T.F. and Beyer,W.A. (1976), Some biological sequence metrics, *Adv. Math.*, **20**, 367-387.

Received on 11 March 1985; accepted on 14 March 1985