Volume 14 Number 1 1986 Nucleic Acids Research

Fast analysis of DNA and protein sequence on Apple IIe: restriction sites search, alignment of short sequence and dot matrix analysis

Christian Marck

Service de Biochimie, Bat–142, Département de Biologie, Centre d'Etudes Nucléaires de Saclay, 91191 Gif-sur-Yvette, Cedex, France

Received 11 July 1985

ABSTRACT

A fast restriction sites search algorithm using a quadruplet look-ahead feature has been written in 6502 assembly language code. The search time, tested on the sequence of pBR322, is 4.1 s/kilobase using a restriction site library including 112 specificities corresponding to a total site length of over 700 bases. The search for a short sequence (< 36 bases) within a longer one (up to 9999 bases) with a given number of mismatchs or gaps allowed has also been written in assembly language. Typical run time for the search of a 12 base sequence with 1,2 or 3 gaps allowed are 6.2, 9.4 or 13.6 s/kilobase, respectively. The dot matrix analysis needs 7.5 minutes per square kilobase when using a stringency of 15 matched bases out of 25. A 7/21 matrix of two 500 amino acid proteins is obtained in 3 minutes. These three routines are included in DPSA, a general package of programs allowing manipulation and analysis of DNA and protein sequences (*).

The DPSA (DNA and Protein Sequence data Analysis) program has been developed for the analysis of a yeast gene (1). It is intended to be an easy desktop tool for the manipulation and analysis of mid-length DNA sequences (up to 9999 bases). DNA sequences can be also converted into protein sequences and further analysed with the program.

DPSA includes all classical manipulation of DNA sequence: strand exchange, merging, stop codon search, translation into protein, codon usage, protein molecular weight prediction and others. These features, for the sake of shortness, will not be described here. During the development of this program, we have tried to balance between the rapidity of execution and a relative ease of programming. For this reason, the 6502 machine language is used whenever run time is a critical factor while Basic is used for dialogues and printout. In order to emphasis the benefit in run time obtained, three parts of the programs are exposed in greater detail.

* The program described in this paper is available from the author. Please send a blank 5 1/4" diskette and a self-addressed mailing label. A small charge may be requested to cover mailing.

<u>Fast quadruplet look-ahead restriction site search</u> The continuous appearence of new restriction specificities requires easy updating of the restriction sites list used. On the other hand high speed search makes the use of machine language necessary on small computers. Therefore, in order to avoid reassembly every time the restriction data file has to be modified, we use a separate basic program that builds up a binary data file containing all necessary data i.e. recognition sequences, names, point of cleavage and other data. The recognition sequences as well as the DNA sequences are stored using a binary coding of bases (2,3). We use the following code: A,C,G and T are coded as the binary equivalent of 1,2,4 and 8, respectively. This allows a fast comparison of any two degenerate bases with a byte to byte logical AND (2,3). Other numeric codes from 3 to 15 are attributed to the degenerate bases M,R,S,V,W,Y,H,K,D,B and N, respectively (4). For the sake of simplicity the above code is used throughout the program, even for the one-letter code of proteins, codes over 15 being attributed to the remaining letters.

The search algorithm makes use also of the following observation. If one lists the 256 possible quadruplets and counts those found at the beginning of all restriction sites (included in a given library), it appears that 133 are never found. The 123 quadruplets found start with an A: 21, C: 33, G: 56 or T: 13 (recent addition of new specificities (5) have shown that the quadruplet usage is not drastically modified - note that nearly all 64 possible quadruplets starting with G are found at the beginning of restriction sites while T is seldom used as the first base of a restriction site). We have tested the following look-ahead algorithm. A table of quadruplet use (256 elements) is built every time the restriction site library is updated. During the restriction site search, the quadruplet value is computed for each position in the sequence and used to decide whether the search can be skipped or not. The library we use routinely includes 112 different restriction sites, the lengths of which represent a total of over 700 bases (the library includes all restriction endonucleases whose recognition sequence and cleavage are known (5)). Using pBR322 as a test sequence, the search proceeds at 4.1 s/kilobase with the look-ahead algorithm or 8 s without it. The length of the binary code is only 127 bytes while the binary restriction data file is over 2500 bytes. The use of a quadruplet code seems the best choice since: a quadruplet is just coded within a byte (the code used in this special case is 0,1,2 and 3 for A,C,G and T, respectively), the quadruplet corresponds to the shortest restriction

```
pBR322 1-200

                            Hind III
      Tth111 II        Taq I                                               Nla IV
    Nla III   Alu I   Cla I  Alu I                                         Ban I
    : :       :       ::    ::                                             :
TTCTCATGTTTGACAGCTTATCATCGATAAGCTTTAATGCGGTAGTTTATCACAGTTAAATTGCTAACGCAGTCAGGCACCGTGTATGAAATCTAACAAT   100
AAGAGTACAAACTGTCGAATAGTAGCTATTCGAAATTACGCCATCAAATAGTGTCAATTTAACGATTGCGTCAGTCCGTGGCACATACTTTAGATTGTTA
    : :       :       ::    ::                                             :
    5         15      23     30                                            76
      7                24                                                  76
                            29

                                                                          Mnl I
                                                                        Hae III
                                                                     ScrF I
                             ScrF I                                  Nci I
                             EcoR II                                 Msp I
                  Nla IV     BstN I                            Rsa I   Sau96 I
HinP I        Mnl I     Mae III SfaN I                      Msp I    Hpa II
Hha I      Fok I  Ban I  Hph I  Fok I                       Hpa II   Bcn I          EcoR V
:          :  :   :     ::   :  ::                          :  :     : :: :         :
GCGCTCATCGTCATCCTCGGCACCGTCACCCTGGATGCTGTAGGCATAGGCTTGGTTATGCCGGTACTGCCGGGCCTCTTGCGGGATATCGTCCATTCCG   200
CGCGAGTAGCAGTAGGAGCCGTGGCAGTGGGACCTACGACATCCGTATCCGAACCAATACGGCCATGACGGCCCGGAGAACGCCCTATAGCAGGTAAGGC
:          :  :   :     ::   :  ::                          :  :     : :: :         :
101        112    119    126    133                         161      170            185
101           115       125      134                        161      170
                  119        130                               164     172
                             130                                     170
                             130                                     170
                                                                     170
                                                                        173
                                                                          175

     End of restriction map
```

Fig. 1. Example of restriction map produced by the restriction search program. The complete pBR322 sequence is printed in 10 minutes.

site and finally, the look-ahead table is only 256 elements long (the use of sextuplets would require a 4096 element table). Note, however, that the look-ahead algorithm cannot be used on sequences containing degenerate codes. Several displays and printouts are possible; Fig. 1 and 2 show two examples of the restriction sites search printout: a restriction map and plasmid circular restriction map.

Fast alignment of short sequences The special type of problem we have been considering is the following one. One often searches for the occurence of a given short sequence, say up to 20 bases, within a longer one, several hundreds or thousands of bases. An example of such a problem is the search of potential regulatory (protein binding) sequences in promoters. The uncertainty about such sequences makes it advisable to allow some mismatchs. Furthermore, it would be of great help if the search could be also performed while allowing some gaps. Let us consider the following example. The sequence ABCDEFGHIJ would not be detected whenever sequences such as ABCDEGHIJ (one base is missing) or ABCDEXFGHIJ (one extra base) are

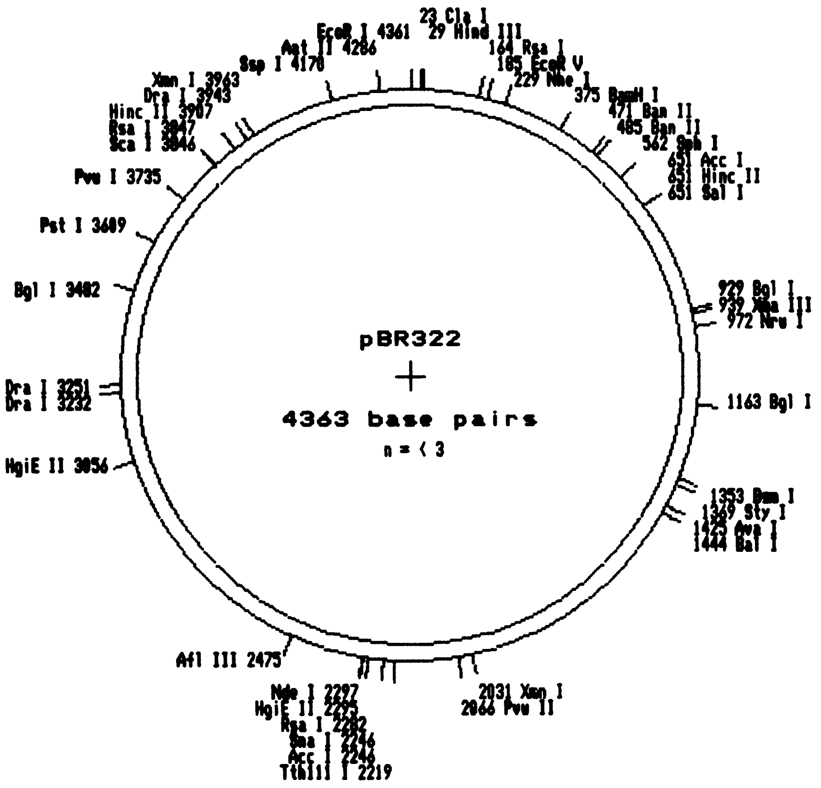

Fig. 2. Circular restriction map of pBR322 sequence. In this example are included all restriction sites appearing three times or less (unique sites are printed in bold face). Computation and printing of such maps are long due to the large number of trigonometric functions necessary (5 to 10 minutes). This printout (as fig. 1) is obtained on an Apple Imagewriter printer.

encountered unless allowing 5 mismatchs. Such a degree of freedom is unpractical since non significant homologies would be too numerous. The solution is therefore to slide sequence s along S and, for every position of s, search for a possible alignment with a maximum number k of gaps and/or mismatchs.

The search for an alignment of two sequences is usually done using the algorithm introduced by Needleman and Wunsh (6) (hereafter referred to as the NW algorithm). Successive variations adapted to specific cases have been proposed (7-10) that lead to the following choice: 1- the original NW algorithm always gives the best alignment but is time consuming and requires a large storage capacity 2- less time consuming algorithms may miss the

```
                                   S            s

                              a  b  c    A B C D E F G H

       ABCDEFGH                          0 1 2 3 4 5 6 7 8
 a      :::: ::               A  A  A    1 x . . . . . . .
     ...ABCD--GHIJKL...       B  B  B    2 . x . . . . . .
                              C  C  C    3 . . x . . . . .
       ABCDEFGH               D  D  O    4 . . x x x x . .
 b      ::::::::              G  E  P    5 . . x . x . x .
     ...ABCDEFGHMN...         H  F  D    6 . . . x . x . x   a
                              I  G  E    7 . . . . x . x .
       ABC--DEFGH             J  H  F    8 . . . . . x . x   b
 c      ::: :::::             K  M  G    9 . . . . . . x .
     ...ABCOPDEFGH...         L  N  H   10 . . . . . . . x   c
```

Fig. 3. A sequence s of n=8 elements ABCDEFGH is looked for along S with a maximum gap number of K=2. A window of n+K=10 elements is slidded along S and the alignment with s tested for every position. The two sequences are arranged to define a matrix as shown. Three particular cases are illustrated with the corresponding paths indicated in the matrix: a/ 2 gaps in S, b/ perfect match - no gaps, c/ 2 gaps in s.

best possible fit. An elegant solution has been proposed by Fickett (11), the main idea being to reorder the computation of the distance matrix so that only distance below a fixed value are to be computed. In the present case, we search recursively for the possible alignment of a sliding sequence s along a longer one S. The usual result of such a search is that s will be found only a few times, once or not at all. Therefore, in order to speed up the complete search over S, it is advisable to detect as quickly as possible that the alignment is NOT possible. Performing the complete NW algorithm when the alignment is possible will not be too time consuming since such cases are rare and s is a short sequence. Unfortunately the algorithm proposed by Fickett (11) requires many manipulations of the i and j indices of the distance and path matrices which are not easy to program in 6502 machine language. We have therefore devised a simpler procedure that stops the computation of the matrices as soon as the allowed number of gaps or mismatchs is surpassed.

A brief description of the algorithm is given below. Alignment is searched between s (s(1), ... , s(n)) and S (S(1), ... , S(N)), n << N, first with s(1) aligned with S(1), then with S(2) up to the end of S. For every position of s the distance and path matrices are computed (see ref. 11 for the definition and computations of these two matrices) and k gaps or mismatchs are allowed. Fig 3. shows that the size of the matrices has to be (1+n+K)*(1+n) in order to support the case where all gaps are found in s.

Since we wish the whole s sequence included in the alignment, all paths should end in the rightmost column. The distance between two alignments is computed using a penalty of 1 for every gap or mismatch, however a mismatch is always preferred to a gap. The classical NW algorithm is used to compute successive complete rows of the distance matrix while recording two characteristic values dr and dc defined as follows:

- dr is the least distance found in the current l row: dr= min(d(l,j) j=1 to n)
- dc is the least distance found up to i=l in the rightmost column: dc= min(d(i,n) i=1 to l)

After each row as been computed, it is decided whether the computation should be

1/ resumed if dr is still =< k (the computation then proceeds to the next row unless the end of the matrix has been reached)

2/ given up if dr > k and dc > k (no path ending in the rightmost column with distance less than k exists)

3/ stopped if dr > k but dc =< k (a path ending in the rightmost column exists).

One advantage of this procedure is that there is no need of reinitializing the distance or path matrix between two successive positions of s. The corresponding 6502 machine code is 442 bytes long. Runtime depends upon the number of gaps allowed. For example the sequence AAACCCGGGTTT was added at the end of the pBR322 sequence and the search for this sequence with allowance of 1,2 or 3 gaps or mismatchs allowed proceeds at a speed of 6.2, 9.4 or 13.6 s/Kilobase, respectively. If the accelerating algorithm described above is not used, the run times become 28, 30 and 33 s/Kilobase. The maximum length n of s has been set to 36 and the value of k (number of gaps or mismatchs) is limited to n/2. The S file may be up to 9999 bases. If gaps are not allowed, the program switches to a classical search algorithm and the search for the above sequence needs only 0.5 s/Kilobase with 3 mismatchs.

<u>Fast dot matrix analysis</u> The criterium chosen to plot the dot matrix is the simplest one: two windows of length l are slidded along each of the two sequences to be compared and one point is plotted whenever k bases over l are found to match (12). Protein sequences in one letter code can also be compared. The dot matrix is computed row by row and printed out every eight rows; this insures a minimal core memory space. The binary code is 192 bytes long. The graphic printout uses a 72*72 dot/inch accuracy and examples can

be found in ref. 1. A 500*500 protein matrix with a 7/21 stringency is computed and printed in 3 minutes. Run time increases as the product of the lengths of the sequences compared: a 2000*2000 matrix is obtained in 20 minutes for a stringency of 10 bases matched out of 15 or 30 minutes for 15 bases out of 25.

The DNA and protein sequence analysis system that we have written has two aims. 1- As a desktop instrument, the Apple IIe allows a quick interactive search of DNA sequences. The three examples given above show that the Apple IIe, with its 8 bit 6502 microprocessor may run very fast when programmed directly in assembly language (3,10). For pure search tasks, run times obtained are up to 10 times better than that obtained on more recent 16 bit machines programmed in compiled Basic (13-15). 2- The second orientation is the production of documents for publications, posters or simply daily lab use.

One of the most tedious tasks in DNA sequencing is certainly sequence entering from gel reading. For this reason, the design of the editor has been given great care. A short description of it is given below. The editor can handle three types of sequences: 1- DNA sequences, 2- Degenerate DNA sequences and 3- protein sequences in one-letter code. Upon further use of a sequence, its type is recognized and the keyboard is automatically restricted to the allowed letters. A file is presented on the screen one page at a time, a page is 5*80 bases long, plus the first line of the next page. One can move from page to page or inside a page using the four arrow keys; immediate insertion or deletion at the cursor position is possible as well as immediate strand exchange. DNA sequences can be entered with the A,C,G,T keys or using the "special keyboard": in this mode the keyboard is reprogrammed so that the A,S,D,F keys (for the left hand) and the J,K,L,;, keys (if the right hand is used) act as A,C,G,T. This allows a single user to type in the sequence while reading it without the need of looking at the keyboard.

Hardware requirements Apple IIe with 80 columns card, Apple Imagewriter printer (driven with an Apple Superserial card) and two 5" 1/4 floppy disk units. The program resides in disk 1 and the data in disk 2. All runtime examples given above are obtained with the serial 1 Mhz 6502 processor.

ACKNOWLEDGMENTS I wish to thank P. Cottrelle for producing the loads of DNA sequence whose analysis inspired me to create this program. I thank F. Dardel for helpful discussions about alignment problems, C. Mann for correcting the manuscript and A. Sentenac for enthousiastic support.

REFERENCES

1 - Cottrelle,P., Thiele,D., Price,V.L., Memet,S., Micouin,J-Y., Marck,Ch., Buhler,J-M., Sentenac,A. and Fromageot,P. (1985) J. Biol. Chem. 260 , 3090-3097.

2 - White,C.T., Hardies.S.C., Hutchinson,C.A. and Edgell,M.H. (1984) Nucleic Acids Res. 12 , 751-766.

3 - Dardel,F. (1985) CABIOS 1 , 19-22.

4 - Cornish-Bowden,A. (1985) Nucleic Acids Research 13 , 3021-3030.

5 - Roberts,R.J. (1985) Nucleic Acids Research 13 , r165-r200.

6 - Needleman,S.B. and Wunsch,C.D. (1970) J. Mol. Biol. 48 , 443-453.

7 - Kruskal,J.B. (1983) SIAM review 25 , 201-237.

8 - Theory and practice of Sequence Comparison: Time-warps, String Edits, and Macromolecules, David Sankoff and Joseph Kruskal, Eds., Addison-Wesley, Reading, Massachussets, 1983.

9 - Wilbur,W.J. and Lipmann,D.J. (1983) Proc. Natl. Acad. Sci. 80 , 726-730.

10 - Dardel,F. (1985) CABIOS, submitted.

11 - Fickett,J.W. (1984) Nucleic Acids Res. 12 , 175-179.

12 - Heiter,P.A., Max,E.E., Seidman,J.G., Maizel,J.V. and Leder,P. (1980) Cell 22 , 197-207.

13 - Queen,C. and Korn,L.J. (1984) Nucleic Acids Res. 12 , 581-599.

14 - Schwindinger,W.F. and Warner,J.R. (1984) Nucleic Acids Res. 12 , 601-604.

15 - Lagrimini,L.M., Brentano,S.T. and Donelson,J.E. (1984) Nucleic Acids Res. 12 , 605-614.