Volume 14 Number 1 1986 Nucleic Acids Research

Computer program for the IBM personal computer which searches for approximate matches to short oligonucleotide sequences in long target DNA sequences

Eugene W.Myers[1] and David W.Mount[2]

Departments of [1]Computer Science and [2]Molecular and Cellular Biology, University of Arizona, Tucson, AZ 85721, USA

Received 16 July 1985

ABSTRACT

We describe a program[3] which may be used to find approximate matches to a short predefined DNA sequence in a larger target DNA sequence. The program predicts the usefulness of specific DNA probes and sequencing primers and finds nearly identical sequences that might represent the same regulatory signal. The program is written in the C programming language and will run on virtually any computer system with a C compiler, such as the IBM/PC and other computers running under the MS/DOS and UNIX operating systems. The program has been integrated into an existing software package for the IBM personal computer (see article by Mount and Conrad, this volume). Some examples of its use are given.

INTRODUCTION

Computers are widely used for storage and analysis of nucleic acid and protein sequences (1-11). In the course of developing computer application programs for molecular genetics, we have on several occasions required a program that would aid us in searching for near but not necessarily identical matches to a short oligonucleotide in a long DNA sequence. First, we wished to determine how many possible operator binding sites for a repressor might exist in a library of E. coli DNA sequences. This problem required the ability to search for a DNA sequence approximately 16 nucleotides long with a few mismatches, insertions and deletions compared to a consensus sequence derived from several such sites. Second, we were interested in designing a set of oligonucleotide probes for a specific protein sequence in a phage lambda library. We wished to know whether other sequences present in the phage library might be recognized by our probe mixture and might therefore give a misleading hybridization signal. Finally, we wanted to design a sequencing primer for dideoxy-type sequencing on phage lambda. We wished to

[3] A letter requesting the programs should be sent to D.M. A standard IBM microcomputer disket containing executable forms of the program and others from the laboratory will be sent. This disket should be copied and the original returned.

know whether the primer would hybridize to other phage sequences similar to the target sequence and would therefore give two or more superimposed sequences. While we were not able to determine the functional significance of near matches in the target sequence to the test sequence, it was useful to find sequences which differed from the test DNA pattern in a few positions by a small number of miamatches, deletions, additions or a combination of a small number of such changes. For the above purposes and related ones, we designed the program described below.

The program basically solves the approximate pattern matching problem which may be formally stated as follows: Given the pattern string P[1..M], subject string S[1..N], and distance threshhold D <= M, does S contain a substring whose edit distance from P is <= D? For the biological problem under consideration, M is much smaller than N (e.g. 30 versus 50,000 nucleotides) and D is a small integer in the range 0 to 10.

ALGORITHMIC METHOD

Two approaches have been used to solve the approximate pattern matching problem. The first method attributable to Sellers (12) employs a simple variation on the dynamic programming solution of Needleman and Wunsch (13) for the sequence alignment problem. Without refinement, it requires time to execute proportional to the product of M and N, designated O(NM), where N is the length of the subject string and M is the length of the pattern. A later method by Ukkonen (14) first builds a finite automaton recognizing all strings distance D or less from the pattern in a preprocessing step and then in O(N) time applies it to the subject string. Unfortunately, constructing the finite automaton takes time proportional to the product of M and the minimum of 3^M and M^D, designated as $O(Mmin(3^M, M^D))$, and slightly less space. Thus, while the scanning phase is very fast, the preprocessing step is prohibitively expensive except for very small choices of pattern length and distance threshhold. This characteristic, combined with the fact that Seller's approach can be refined to use only O(ND) time on average, motivated the authors to choose the dynamic programming approach. This decision was further necessitated by the limited memory capacity and performance of personal computers such as the IBM/PC.

Sellers (12) gives an O(NM) dynamic programming algorithm where N is the length of a subject string S[1..N] and M is the length of the pattern P[1..M]. It computes an (M+1) by (N+1) matrix d[i,j], i in [0,N] and j in

[0,M], where d[i,j] is the smallest distance between the prefix P[1..j] and a substring of S ending at position i. The dynamic programming recurrence defining d[i,j] can be deduced to be:

$$d[i,j] = \begin{cases} 0 & \text{if } j = 0 \\ j & \text{if } i = 0 \\ d[i-1,j-1] & \text{if } i,j > 0 \text{ and } S[i] = P[j] \\ \min(d[i,j-1], d[i-1,j], d[i-1,j-1]) + 1 & \text{otherwise} \end{cases}$$

This matrix can be computed column by column starting with the column given by the i=0 clause of the recurrence. The existence of an approximate match can thus be determined in O(NM) time and O(M) space by computing each column d[i,*] in sequence and checking if d[i,M] is less than or equal to D.

Seller's algorithm does more work than is actually needed to solve the approximate string matching problem since it computes for each position i in S, the smallest edit distance between the pattern P and a substring of S ending at position i. It is only necessary to determine those entries of the matrix d that are less than or equal to D and to report an affirmative outcome when such an entry lies in row M of a given column. Thus, it suffices to retain for each column i, only those entries up to row k where k is such that d[i,m] > D for all m > k. The recurrence for d guarantees that d[i+1,m] > D for all m > k+1. Thus, to compute the relevant portion of column i+1 it suffices to compute those entries through row k+1 and then discard those in rows k+1, k, k-1, etc. until an entry less than or equal to D is encountered. Figure 1 below illustrates the matrix d and the portions of d that need to retained and/or computed when D = 1.

Suppose that the symbols of the pattern and subject strings are randomly and independently chosen and each symbol is chosen with uniform probability 1/A

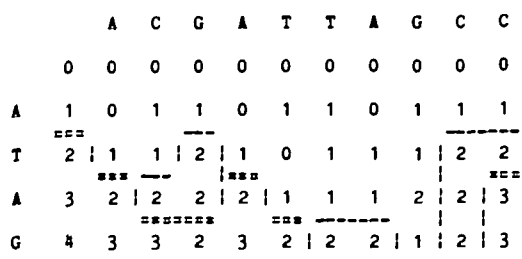

FIGURE 1: Matrix d, retained portions, and computed portions when D=1.

where A is the size of the domain alphabet (e.g. A = 4 for DNA and A = 20 for proteins). The probability that the row m entry of a given column is less than or equal to D, Pr(d[*,m]<=D), is less than or equal to

$$\text{Sum } k=0 \text{ to } D \left[\binom{m+D-k}{D-k} \binom{M}{k} A^{(k-M)} \right]$$

But since these terms quickly vanish as m grows larger than D, it follows that the expected number of entries computed in each column is O(D). Thus, the expected performance of the refined algorithm is O(ND).

Simulations were run to determine the exact number of computed items per column for the cases where the alphabet size is four and twenty. For each alphabet size, 50 experiments were run with D set at 0, 1, 2, ... 15, with M = 50, and N = 75000. The subject and pattern sequences were randomly generated as stipulated above. Applying linear regressions to the results determined that on average 1.17+1.55D and x.xx+y.yyD entries per column are computed when A is four and twenty, respectively. Both regressions had a correlation coefficient greater than .9997.

To construct an alignment between the pattern and subject when an approximate match is found requires that the computed portions of previous columns be available. However, since the substring of the subject that approximately matches the pattern can have at most M+D symbols (in the case where all changes are deletions of subject symbols), it follows that only the most recently computed M+D columns need to be kept. Since D<=M, it follows that only $O(M^2)$ space is needed to construct the matches.

The software permits a user to individually specify limits on the number of inserts, deletes, and mismatches that can occur in an "acceptable" approximate match. This feature is realized by finding approximate matches within a distance threshold equal to the sum of the individual limits. Whenever a match position is found, the software engages in an exhaustive backtracking search for a match that satisfies the specified limits. A particular search path is aborted when any particular limit is exceeded but in general the search can be exponential in M. Fortunately, this rarely occurs in practice and the number of such searches is few as the number of sites of significant homology is small.

T. Smith and W. Ralph (personal communication) have written a similar program based on the dynamic program method of Waterman and Smith (15).

EXECUTION OF PROGRAM

To use the program, two disk files, the first containing the subject DNA sequence to be searched and the second containing the source DNA pattern are prepared using a text editor.

DNA Sequence Format.

The program accepts sequences in the same free format that is used by other programs from this group (see article by Mount and Conrad, this volume). Any line with ; or > in the first column is ignored by the program so that such lines may be used for comments. Only standard base symbols in the test pattern sequence and the large subject sequence (A,G,C,T,U) are recognized. Future modifications will include the capability to analyze all possible ambiguous base symbols. Text entered by editors that set the parity bit is acceptable. By convention, the sequence should read 5' to 3'. Spaces and tabs may be placed anywhere in the sequence and lines may be any length.

DNA sequence file handling

Prompts by the program request the names of disk files containing the subject and pattern-containing sequences. The sequences in these two disk files are loaded from the disk files into the computer memory.

Sequence length restrictions. A combined total of 420,000 nucleotides in the two sequences can be analyzed on the IBM personal computer with 512 kbytes of memory so that entire viral genomes such as that of phage lambda can be searched. The two most important limitations on the size of the sequences that can actually be analyzed by the program are: (1) the length of the subject sequence can be any value up to the memory capacity of the machine and cannot be less than the length of the test pattern; and, (2) the program is presently set for a pattern-containing sequence not longer than 50 nucleotides, but with a recompilation, this value can readily be set to much larger values without a significant degradation of performance. Also, this pattern sequence may be extracted by the program from a much longer sequence that has been loaded into the computer memory.

Analysis of Parts of Sequences. Any portion of the subject and pattern-containing sequences can be analyzed. If only a portion of a sequence is required, a 'no' response is entered. A prompt for the extent of the sequence to be analyzed then appears. Two numbers representing the first base in the sequence to be analyzed and the second are entered. These

numbers should be separated by one or more spaces and the second is followed by a carriage return.

Data Output Options

Several types of data output are possible. The most useful options is to send the data to a disk file for later editing or printing. Terminal output can also be turned off for faster output of data to a disk file. Bases can be colored on an IBM color monitor.

TYPES OF ANALYSIS POSSIBLE

Once the subject and test-pattern sequences have been selected, the maximum number of allowed mismatches, insertions and deletions must be decided. In the presently compiled version, the total number of such variations must not be greater than 30. The program can be recompiled to accomodate larger values. Two choices are available for selecting these numbers. First, the total number of mismatches, insertions and deletions allowed in the subject sequence can be set. In this case, the program searches for any match with a combination of mismatches, insertions and deletions which is less than or equal to this number. Alternatively, the maximum number of allowed mismatches, insertions and deletions can each be set individually, including the possibility that any two of these parameters can be set to zero. The program will then search for matches that satisfy any combination of these values. In order to limit the voluminous data output that is possible, the program lists only one of the possible matches at each position where a match is found. However, other possible alignments may be found by varying the mismatch, insertion and deletion parameters and re-examining the region.

EXAMPLES OF PROGRAM USE

In Tables 1 and 2, program data output for the two options described in the above paragraph are shown. In this example, the subject sequence was the complete sequence of *E.coli* phage lambda of length 48,502 nucleotides and the pattern-containing sequence was an operator binding site of length 16 nucleotides for LexA protein in front of the *E. coli recA* gene. Program output was sent to a disk file which was then edited and printed using a standard text editor. Table 1 displays the output allowing a total of 3 mismatches, insertions and deletions, and Table 2 allowing 5 mismatches, 0 insertions and 0 deletions. The base positions at the beginning and end of the subject sequence are shown in the first line of output. These numbers are followed by the number of insertions (Add), deletions (Del) and

Table 1. Example of program use to search the sequence of phage lambda for close matches to a 16 nucleotide pattern representing a repressor binding site. In this case, a total number of 3 mismatches, insertions and deletions was specified as the match criterion. All the matches found are shown. See text for explanation of data format.

```
Match at Base Pairs 986-1001 (Add=1,Del=1,Chg=1)
Subject: CTTTATAGAGCATA-AG
           *
Pattern: CTGTAT-GAGCATACAG

Match at Base Pairs 28733-28746 (Add=2,Del=0,Chg=1)
Subject: CTGTAT-AGCTT-CAG
                 *
Pattern: CTGTATGAGCATACAG

Match at Base Pairs 31187-31200 (Add=2,Del=0,Chg=1)
Subject: CT-T-TGTGCATACAG
             *
Pattern: CTGTATGAGCATACAG
```

mismatches (Chg) which must be made to the subject sequence to match the pattern. A '-' in the subject sequence on the next line indicates that the addition of a single nucleotide was necessary to achieve a match. A '-' in the test pattern sequence, shown below in the output, shows that a deletion

Table 2. Example of program use with the number of mismatches, insertions and deletions each specified separately by the user. These numbers were 5 mismatches, 0 insertions and 0 deletions. Only 4 of a total of 21 matches are shown for illustration. The sequences were the same as those described in Table 1.

```
Match at Base Pairs 1374-1389 (Add=0,Del=0,Chg=5)
Subject: CAGATTGAGCGTGCAG
          * **     * *
Pattern: CTGTATGAGCATACAG

Match at Base Pairs 19878-19893 (Add=0,Del=0,Chg=5)
Subject: GTGTATGAAGATTCAC
         *       **  *  *
Pattern: CTGTATGAGCATACAG

Match at Base Pairs 36142-36157 (Add=0,Del=0,Chg=5)
Subject: CTCTATGAGCTGAAAA
           *       ** * *
Pattern: CTGTATGAGCATACAG

Match at Base Pairs 45962-45977 (Add=0,Del=0,Chg=5)
Subject: ATGTATGAGCAGAGTC
         *           * ***
Pattern: CTGTATGAGCATACAG
```

of a base in the subject sequence was necessary. Finally, a '*' between the two sequences indicates the presence of mismatch in the two sequences at that position. Each of the two searches shown took approximately 4 min to execute on an IBMXT personal computer.

PORTABILITY OF PROGRAMS

The program has been compiled and linked with the Unix C compiler to run under the UNIX operating system and with the Lattice C compiler to run under the MS/DOS operating system on a microcomputer. Any microcomputer running under MS/DOS can run this program.

ACKNOWLEDGEMENTS

This research was supported by grants from NSF and NIH.

REFERENCES

1. Conrad, B. and Mount, D. W. (1982) Nucleic Acids Res. 10, 31-38.
2. Mount, D. W. and Conrad, B. (1984) Nucleic Acids Res. 12, part 2, 811-818.
3. Mount, D.W. and Conrad, B. (1984) Nucleic Acids Res. 12, part 2, 819-824.
4. Little, J. and Mount, D.W. (1984) Gene 32, 67-73.
5. Mount, D. (1985) BioTechniques 3, 102-112.
6. Martinez, H. (1984) Mathematical and computational problems in the analysis of molecular sequences - a special commemorative issue honoring Margaret Oakley Dayhoff. Bull. of Math. Biol. 46, No. 4.
7. Sankoff, D. and J.B. Kruskal. eds. 1984. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. Addison-Wesley Pub@. Co@., Inc@., Reading, Mass.
8. Soll, D. and R.J. Roberts, eds. 1982. The applications of computers to research on nucleic acids. I. Nucl. Acids Res. 10, no. 1.
9. Soll, D. and R.J. Roberts, eds. 1984. The applications of computers to research on nucleic acids. II. Nucl. Acids Res. 12, no. 1.
10. Jungck, J.R. and R.M. Friedman. (1984) Bull. of Math. Biol. 46, 699-744.
11. Korn, L.J. and C.L. Queen. (1984) DNA 3, 421-436.
12. Sellers, P.H. (1980) J. Algorithms 1, 359-373.
13. Needleman, S.B. and C.D. Wunsch. (1970) J. Molecular Biology 48, 443-453.
14. Ukkonen, E. (1985) J. Algorithms 6, 132-137.
15. M. Waterman and T. Smith. (1981) J. Mol. Biol. 147, 195-197.