COMPUTER APPLICATIONS IN THE BIOSCIENCES

Volume 10 number 1 February 1994

OXFORD UNIVERSITY PRESS

ISSN 0266-7061
Coden COABER

CABIOS

Vol.10. no.1. 1994
Pages 3–5

ADVANCE and ADAM: two algorithms for the analysis of global similarity between homologous informational sequences

Alberto Torelli and Carlo A.Robotti

Abstract

Two algorithms for the analysis of global similarity between sequences of informational polymeric molecules (nucleic acids and proteins) are proposed: one (ADVANCE) merely gives a quantification of the global similarity between two sequences, and is very fast; the other (ADAM) also provides an alignment of the sequences. Both are new algorithms, implement Sellers' theorem, do not require parameters, are able to analyze two sequences of 32 000 elements each, and are fast; nevertheless, they differ deeply in the algorithm, in the programming language type, and in their planning, and will thus have to be treated separately. In fact, the different nature of the required output in each program causes the common aim (to obtain power and speed) to be reached only through very different ways and methods.

Introduction

When we talk about global similarity, we are not searching for a particular subsequence, nor are we looking for subsequences which resemble each other: we merely require the sequences to be treated as a whole.

The Needleman–Wunsch (1970) algorithm, by now a classic of global similarity analysis, does not have a general and direct theoretical foundation, and needs some weight factors to account for the gaps inserted by the program during the sequence alignment operation. Such factors should constitute a means of managing analysis, but they introduce a systematical error (bias), because their influence on the similarity quantification cannot be evaluated *a priori* by a biologist who has no precise notions of the algorithm itself. Furthermore, this method uses a bulky matrix, and is not fast.

The analysis techniques developed on the basis of the Needleman–Wunsch algorithm are numerous (e.g. Smith, 1981), but if one wants a reliable system and a formalized environment, it is more convenient to think in terms of distances rather than similarities: this leads us to Sellers' (1974) metric space and his theorem, which represents: (i) a rigorous solution to the global analysis problem; (ii) a warranty of total absence of prejudice errors (because there are no parameters); and (iii) a possible algorithm, whose nature, unfortunately, is still matricial.

Department of Genetics, Biology, and Medical Chemistry, University of Turin, via Santena 5 bis, 10126 Torino, Italy

The two programs that we present and discuss in this paper implement Sellers' theorem in a completely new manner, avoiding both matrix usage (which occupies much memory and heavily restricts the analysis power) and alignment (which is not necessary to determine the similarity, and constitutes a big obstacle to speed), and together provide a powerful, fast and, not least, safe analysis method.

Systems and methods

ADVANCE was completely written in machine language (Assembly) for the IBM PC (all the family, from 8088/8086 to 80486), using Microsoft Macro Assembler v. 5.0. Because of the repetitive usage of interrupt requests (DOS interrupts and BIOS interrupts), the code is not portable, unless substantial modifications are introduced.

ADAM was written in a high-level language: TurboPascal v. 3.0, by Borland International. A new release of the algorithm, using more recent programming tools to get over some actual environment restrictions, is in preparation.

The hardware required for both programs is an Intel processor 8088 or higher (all members of the IBM-PC microprocessor family, or equivalents), 640 kbytes of RAM, MS-DOS 3.30 or higher, and at least a CGA video adapter. ADAM requires the ANSI.SYS driver (which is part of the operating system) to be installed and running. Neither a hard disk nor a maths coprocessor is required.

Algorithm

The logical heart: Sellers' theorem

By 'evolutionary distance' between two sequences, Sellers means 'the minimum number of mutations necessary to make the two sequences identical to one another'; these two sequences can contain neutral (dummy) elements (spacers). If the subsequence of non-neutral terms is the same in both sequences, then we can say that they are equivalent, and group them into a single equivalence class that we will refer to as 'evolutionary sequence'; its simplest member is the pure (i.e. without spacers) non-neutral monomer subsequence. Such subsequence is always defined, because non-neutral terms must be in a finite number, so that, from some position on, there will only be an infinity of neutral elements.

If a, b, c are non-neutral monomers in the set of all possible elements, and if we assume that that set is a metric space [i.e.

there is a distance $d(a,b)$ between any a and b, and for every a, b, c it is true that: (i) $d(a,b)$ is zero when $a = b$; (ii) $d(a,b) = d(b,a)$; (iii) $d(a,b) + d(b,c) \geq d(a,c)$], then the distance between sequences can be defined as

$$d(a_1,a_2, \ldots; b_1,b_2, \ldots) = \sum_{i=1}^{\infty} d(a_i,b_i)$$

and the distance between evolutionary sequences will be

$$[d] = \min d(a_1,a_2, \ldots; b_1,b_2, \ldots)$$

where $a_1,a_2, \ldots$ and $b_1,b_2, \ldots$ are those members (of the respective equivalence classes) that return, for the sequence distance, a minimum value, which will be taken as the evolutionary distance between the corresponding evolutionary sequences (i.e. equivalence classes, according to the above definition).

By proving that d is a metric space on sequences and that $[d]$ is the same on evolutionary sequences, Sellers demonstrates that such minimum value $[d]$ $(a_1, a_2, \ldots, a_m; b_1,b_2, \ldots, b_n)$ is determined by mathematical induction as follows.

Let $i \in \{0 \ldots m\}$ and $j \in \{0 \ldots n\}$, and interpret the sequences as spacers (//) when i = zero or j = zero. Induction is initiated by the formulas

$$[d]\ (a_1,a_2, \ldots, a_i; //) = \sum_{h=1}^{i} d(a_h,//)$$

$$[d]\ (//; b_1,b_2, \ldots, b_j) = \sum_{h=1}^{j} d(//, b_h)$$

and the inductive step is made by giving $[d]$ $(a, a, \ldots,a ; b, b, \ldots,b)$ the minimum of the following three values:

$$[d]\ (a_1,a_2, \ldots,a_{i-1}; b_1,b_2, \ldots,b_j) + d(a_i,//)$$
$$[d]\ (a_1,a_2, \ldots,a_{i-1}; b_1,b_2, \ldots,b_{j-1}) + d(a_i,b_j)$$
$$[d]\ (a_1,a_2, \ldots,a_i; b_1,b_2, \ldots,b_{j-1}) + d(//,b_j)$$

For the demonstrations, see Sellers (1974).

If we assign a geometrical meaning to the indexes, clearly the induction is done when a $(m+1) \times (n+1)$ matrix (which makes room for the neutral element, too) has been filled with distances. Going backward along the matrix again, it is also possible to obtain a sequence alignment, as Sellers suggests.

Two new algorithms

The exploitation of the theorem's inductive nature makes it possible to enormously increase the power of analysis: in this case, in fact (m and n being the sequence lengths), on a 16-bit machine the required amount of memory is proportional no longer to $4mn$, but to $4(m+n)$, and in a 32 000 × 32 000 analysis this means that 256 kbytes will suffice (instead of 4097 Mbytes).

Moreover, both algorithms read from disk two files containing the sequences, and recognize only the letters corresponding to legal monomers, regardless of their case (uppercase or lowercase); the special characters LineFeed, CarriageReturn and EOF are allowed but ignored. If any other character is present, an error is signalled. Along the reading, only legal monomers are allocated, and particular care is dedicated to the memory management optimization. The output is in text mode. Because the calculation process for $[d]$ implies a minimalization of three values, in case of parity there are $3 \times 2 \times 1 = 6$ possible choice priorities. The two programs implement the priority that makes the gap number minimal.

The other characteristics are program-exclusive, which is why we must proceed by distinguishing the two algorithms.

ADVANCE

Entirely written in Microsoft Macro Assembler v. 5.0, ADVANCE.EXE gets the two filenames from the command line, and calculates the time elapsed from its launch to the DOS return. On a 80486 DX (33 MHz) a 1000 × 1000 analysis is done in only two seconds. Register-to-register type instructions, which require fewer clock cycles, were used intensively. In such a way, any CPU register serves different purposes in different moments. All the function or procedure calls that can significantly slow down the program have been substituted by right macros, and all memory-to-memory instructions are 16-bit instructions (the 8088, even though having an 8-bit bus, uses 16-bit instructions). For total portability on the entire IBM PC family, only 8086-recognized instructions have been used, and any memory addressing operation is done into the 640 kbytes of base memory. The program can be interrupted at any time by pressing Scroll Lock, but if the interrupt is not confirmed, ADVANCE begins computations again after a few seconds. The critical error handler control lets the program manage every type of hardware error; in the case of memory shortage, it is a DOS responsibility to give an error message. In the induction procedure six arrays are involved, which continuously exchange data with one another. The output consists of displaying the two values of the evolutionary distance and the elapsed time. No sequence alignment is performed. The stack is used for parameter passing only at load time (when data files are loaded and cleared); otherwise the CPU register usage has been preferred because it is a faster method. All the messages are shown in a window with a direct video memory access, so that the display is instantaneous. All is planned for maximum speed.

ADAM

ADAM.COM is written in Borland TurboPascal v. 3.0. Since this version generates only .COM files and no more than 64 kbytes of data can be allocated as static (whereas managing memory with pointers allows breaking the 64 kbyte wall, albeit at the expense of speed), it was chosen to address the memory in direct mode, i.e. with pointers managed by the program,

without any intermediation from TurboPascal. Thus, if there is not sufficient memory, the error message comes from the program, and not from DOS or from TurboPascal. This is a temporary solution, which will be overcome in the near future by rewriting the algorithm in TurboPascal 6.0. The critical error handler control is taken, and ANSI escape sequences are used for message display. In the alignment procedure, which includes the induction one, six normal arrays are involved, plus a 120 × 120 matrix buffer completely managed as if it were two distinct arrays: in other words, the same data structure is referenced in two different ways.

The alignment algorithm is organized in essentially five steps (these steps are inserted in nested loops, so that the number of times of their occurrence and the relative order of their execution depends each time from the change, at run time, of the logical conditions that control the program flow): (i) dynamic buffer adjustment to the actual analysis size; (ii) induction of eight distinct arrays management; (iii) logical decisions and alignment; (iv) repeated induction usage when the support buffer cannot be utilized any longer; (v) automatic completion of the induction when the repeated induction cannot be used any more. The two moments, induction and alignment, are intimately correlated.

At any time, the current size of Sellers' matrix necessary to perform the analysis is shown; this value is decreasing up to zero. The output (aligned sequences and various statistics) can be redirected. In case of insufficient disk space in writing the output file, the program goes on anyway, and even though an error message appears at the end, the analysis comes to completion in any case (but this feature slows down the program).

Implementation

ADVANCE opens a window in the middle of the screen, and loads the two files specified in the command line (in case of error, a help window appears which tells the user what to do). After load and filter operations, the analysis begins. Pressing Scroll Lock interrupts the program at any time; otherwise the message 'COMPUTING DISTANCE' is kept on the screen (no life signal is implemented, to avoid slow-down effects on calculations). After interruption, if the user does not confirm or deny the quit action, ADVANCE waits a few seconds (in this time the chronometer is disabled) and then goes back to its job. Finally, it gives the Sellers' evolutionary distance and the elapsed time. On a 80486 DX/33 MHz, a 1000 × 1000 analysis takes place in 2 s (3000 × 3000 in 26 s and 32 000 × 32 000 in 53 min). Hitting Scroll Lock now causes the window to disappear and the original screen to be restored; a return to DOS is then performed.

ADAM uses the whole screen: the upper part contains the program header; the subsequent lines are used for the filename input (input-data-file names and out-file name) and Carriage-Return is the valid output-file name which redirects the output alignment to the screen only (no disk files are generated). The analysis is brought to an end even when a disk-full error occurs; the current calculated piece of alignment is shown (in scroll mode) in real time, and the same is done for size and dimensions (in bytes) of the Sellers' matrix that is in use. The program is fully interruptable and makes extensive use of the ANSI escape sequences. A 1000 × 1000 analysis requires 20 s (on a 80486 DX/33 MHz machine) and a 3000 × 3000 one (corresponding to two proteins of 1000 amino acids) is done in 7 min. In the end, a table of the alignment characteristics appears. This alignment is completely consistent with the calculated distance: if a manual count is made of the differences between the aligned sequences, the total that is obtained is equal to the evolutionary distance. This is a consequence of the particular choice priority implemented (as we have seen, six priorities are possible) and allows a direct control of the right operation of the algorithm.

Discussion

As it is concerned with global similarity, the analysis of two sequences of comparable length can be used successfully for recognizing similar patterns (as clearly appeared from some crossed tests between the numerous exons of the human salivary α-amylases). Here is a specimen result from ADAM, obtained using two short hypothetical sequences planned to demonstrate clearly the operation of the algorithm:

(1)
```
w w w x x x x x y y z z z
      ■ ■ ■ ■ ■       ■ ■ ■
· · · x x x x x · z z z z
```

The final table supplied is:

sequence 1 length :	13	
sequence 2 length :	9	
matching pairs :	8	
mismatching pairs :	1	
gaps in file 1 :	0	
gaps in file 2:	4	
least distance :	5	
mutation :	38.462 %	

Although the algorithm aims at the calculation of a minimum distance, the alignment, which is not necessary, should be interpreted as a visual representation of such calculated distance, and not as a similarity alignment. In this sense, the finding of local similar patterns is always a consequence of the fact that the sequences have comparable lengths; in case of very different lengths, the algorithm appears exactly as it is, and here the minimum distance value is not generally paired with a meaningful alignment in order to discover similar subsequences (except particularly lucky cases):

(2) w w w x x x x x y y z z z

■ ■ y y ■

w · · x · · · · · · z · ·

(typical result: probe disintegrated)

(3) a b c d e f g h i j a b c d e f g h i j a b c d e f g h i j

■■■■■

· · · · · f g h i j · k l m n

(a lucky case)

The sequences tested are short and suitable for manual control on the calculated alignments. Sellers' theorem assures in any case that the distance (in the sense specified by Sellers) found is the minimum one theoretically possible, independently of the sequence lengths or compositions.

The algorithm is also totally successful in the local similarity analysis (i.e. it returns an alignment which certainly detects the similar regions) only on condition that the two sequences are of the same nature: two cytochromic sequences, two mitochondrial genomes, and so on. In this case the most efficient (operatively speaking) alignment is generated (the theorem itself guarantees a constant quality—see example 1). However, it is necessary to underline that the algorithm is not aimed at this, even if the program can obtain it, and therefore the algorithm must be used with wisdom.

Analyzing two sequences that are very different in composition and/or length generates safe results only in the sense of Sellers' theorem, i.e. a minimum number of mutations necessary to make the two sequences identical. Other interpretations of the results may be non-meaningful: 84 tests done using unlike sequences proved that the alignment obtained is simply the best possible alignment (in the Sellers' sense), but it is difficult to judge the meaning of the similarity patterns obtained; an example from these tests is reported here:

(4)

A T T G A G G C T T A C G C G T C C A T A T C C · G · G G A A T ·

■ ■■ ■■ ■ ■■ ■■ ■■■ ■■ ■ ■

A · · G A T C C T · · · G · · T C T A T · T C C C G C G C A G T G G

sequence 1 length :	30
sequence 2 length :	26
matching pairs :	17
mismatching pairs :	5
gaps in file 1 :	4
gaps in file 2 :	8
least distance :	17
mutation pressure :	50.000 %

The strength of these two programs is essentially based on the size of the analysis, which is performed without any method simplification or compromise, irrespective of sequence length.

In any case, work is in progress to extend the program in order to compare (and to establish relations among) whole viral, bacterial or eukaryotic genomes, and to detect regions with higher or lower variability, or to study other features linked to the sequences' nature.

Certainly, using a program is profitable only when its operation is well known; as for the applicability field, this is definable in an approximate manner only because, generally, an automated elaboration process implies an appropriate usage and a critical evaluation of the results.

Acknowledgements

We are particularly grateful to Gabriella Sella for her precious assistance in polishing this paper; Lorenzo Silengo for fruitful discussions; and Piero Cervella for his advice. This research was partially funded by a grant from Ministry of University and Scientific Research (40%, Genetica Evoluzionistica).

References

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequences 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. (1981) Comparison of biosequences. *Adv. Appl. Math.*, **2**, 482–489.

Received on December 12, 1992; accepted on March 16, 1993