Interactive molecular biology computing

James M. Pustell

Department of Zoology, University of Vermont, Burlington, VT 05405 and International Biotechnologies Inc., 275 Winchester Avenue, PO Box 9558, New Haven, CT 06535, USA

Received August 17, 1987; Revised and Accepted November 9, 1987

ABSTRACT

It is clear that the selection of the best possible algorithm for a computer program is essential for the creation of a useful tool. After this first step is taken, however, the usefulness of such a program may be greatly enhanced or impeded by the way it is implemented. We illustrate this point by describing our implementation of the well known FASTP/FASTN algorithm in an interactive software environment.

INTRODUCTION

We introduced a molecular biology environment, now called the Cyborg environment, for microcomputers (1). The idea was to introduce a system which was independent of computer or operating system, intuitive for the molecular biologist to use and which took over as much of the tedium as possible, freeing the scientist to think and try out ideas. Access to data is simple and transparent, whether from the user's own files, in a variety of formats, or from a large database.

Here we describe how the environment can add greatly to the performance of a common task. We use the example of a global database search. As one might expect, the environment makes running such a job simpler and more flexible. Perhaps less expected at first glance, the environment can also be used to compensate for some of the inherent problems with the method and to add a new dimension of flexibility in studying the results of the search.

METHODS

The software described is the Cyborg environment, available through International Biotechnologies, Inc. All examples given are based on performance on a slow (6 mhz.) IBM AT with 640 kbytes of memory and a 30 mbyte hard disk. The system also currently runs on IBM PC, XT and Personal System 2, and compatibles under MS-DOS version 2.11 or higher. At the time this article was written the environment had been implemented on the Macintosh SE

but was not yet available for distribution.

The global database search is derived from FASTP and FASTN (2, 3). Explicit permission was obtained from the authors to use the code. It involves generating a hash table to rapidly locate uninterrupted regions of match between a query sequence and a library of sequences. The user is then prompted to optimize a certain number of matches, that is to perform a more time consuming alignment which allows insertions and deletions.

In this implementation FASTP and FASTN have been combined into a single program which uses replaceable scoring matrices. In this sense it resembles a similar combined programed called FASTA, however this is a case of convergent evolution. The program has many features FASTA does not.

It can search a compacted version of the National Biomedical Research Foundation (NBRF) protein database with a protein query sequence, the GenBank (R) floppy disk nucleic acid database with a nucleic acid, the GenBank (R) floppy disk nucleic acid database with a protein query sequence, and any individual nucleic acid or protein files in any of several additional formats.

When searching with a nucleic acid, it automatically searches BOTH strands of the library sequence, not just one as is commonly done (in FASTA/FASTN, among others). When searching the nucleic acid database with a protein, it translates the library sequences in all six frames "on the fly", and comparisons are done as proteins. The results are displayed as protein alignments, but the numbering of the nucleic acid is correctly preserved.

One of the problems of such a two pass database search is that one may locate initial unbroken matches of relatively low score which can achieve very significant scores when insertions and deletions are allowed. Since such initial low scores may be well down in a list of hundreds or thousands of matches, the user may never have the program optimize them. Thus significant matches may be missed. All database searches supported in the Cyborg environment can run in one of three modes. In the first mode, only initial scores are saved, similarly to FASTP/FASTN/FASTA. At the end of the search, instead of asking how many alignments to optimize, the program automatically optimizes them all, then presents the user with an list ranked by optimized score, instead of by initial score. However, the best mode to run the program in is the "Save Align" mode, in which EVERY alignment with a score greater than the cutoff is optimized on the fly, and a compact representation of the alignment is saved. The top 300 optimal scores from the complete search are then presented to the user. The compact representation of the alignment makes the unique interactive display described below possible. There is a third,
intermediate, mode in which all alignments are optimized on the fly, but are not saved. This method avoids dropping important alignments with low initial scores and conserves on memory on small machines.

Another difficulty involves the hashing step which requires that a certain number of residues in a row must be identical to the query sequence. In the case of proteins, a sequence may have functionally or structurally conservative substitutions which do not change the character of the protein but cause the hashing step to miss it. We have addressed this problem by allowing any set of substitutions in the hashing step as well as the scoring step. Thus one may define certain non-identical residues to be equivalent. We find this can dramatically increase the sensitivity of a protein search.

Scan times (the time for the searches themselves) are slightly faster than FASTA. 962,996 amino acids can be scanned with a 101 amino acid query sequence in 3 minutes, 46 seconds (5 minutes, 51 seconds for IFASTA). 1,602,436 base pairs of DNA can be scanned with a 303 base pair query in 7 minutes, 56 seconds (9 minutes for GFASTA - in this case GFASTA was run twice, once for each orientation of the query sequence). 1,602,436 base pairs (as a six frames translation) can be scanned with a 101 amino acid query in 12 minutes, 7 seconds (no FASTA equivalent). Processing times (the time to optimize and save the alignments on the fly) are generally one to two times the scan time. Total run time is the sum of the scan and processing times, when automatic processing is used. Thus searching the complete GenBank (R) database takes about an hour for a scan only and about two hours for both scan and automatic processing under the conditions above.

RESULTS

One of the strengths of an integrated environment is that the same tools are available to the user in any program, so very little additional learning is required to perform a new task. If we wished to search a DNA database with a protein, we would select it from the main menu, and, just as for any simpler task, we would be presented with an initial screen (similar to Figure 1B) on which to name sequences and variables to use. In this case, a number of empirical variables must be used because the global searching problem is not fully understood. These variables are hidden, so as to not intimidate the casual user. That is, the user could enter a sequence name and hit RUN and get a result. However, the less naive user has instant access to all such variables through a function key called "Tweak" which reveals the hidden
variables and permits them to be changed (Figure 1A). Pushing RUN starts the search.

In Figure 1B we see the screen displayed during the search. On the left the user has entered a request to compare the human gastrin gene (HUMGAST) in its entirety (Region 1-0, where 0 means the "end of the sequence"). The GenBank (R) database is in the directory "\genbank\" and a number of the user's own files (in a variety of formats) in the directory "\mydir\" will be searched as well. A scoring matrix, a hash level, automatic processing level, and how much to report the progress of the search can also be selected or changed. On the right are the available GenBank (R) directories. The user has elected to search all vertebrate sequences by setting the Yes/No toggles on the directory names. The lines next to the active directories show an "x" for each file which has been searched. This search is about half completed and the program is currently working with the fourth rodent file.

Upon completion of the search, an interactive menu (not shown) is reached from which the user may call a number of different types of display. One may look at a list of entries ranked by optimized score (Figure 1C) or a graphic map of alignments (Figure 1D) or aligned, automatically annotated sequences (eg. Figure 3) by pushing the appropriate function keys. These display modes function exactly the same in other programs as well, so once a user has learned how to use one program, using any other program requires no further learning.

One may interact with the results on the screen. So, we may notice from the list (Figure 1C) that the Xenopus caerulein genes all seem to have the same score and (Figure 1D) cover the same region. In the graphic map (Figure 1D) the arrows indicate the orientation of the DNA from which the protein was translated, "+" indicates an insertion, and "." indicates a deletion. From either the graphic map or the list we may select pig gastrin, human gastrin, human calcitonin, only one example of a Xenopus caerulein, and two examples of cholecystokinin. We then set our display to show only selected entries, and return to the graphic map (Figure 2A). If we decide to keep this particular subset as useful, we can get a hard copy. We may opt for only a graphic map and aligned sequences, then push PRINT. Figure 3 results. Note that the program is also capable of automatically printing the references for all the database entries displayed in the print out. The first one in our example appears at the bottom of Figure 3. This feature is obviously valuable since one so commonly must consult the relevant literature regarding the sequences found by global searches. Naturally it requires that the references be
The default values from PAM_250 are:

- P1: Length of Query Sequence
- P2: Maximum Hash Level, CH: Current Hash Level
- QL: Length of Query Sequence
- NH: Maximum Hash Level, CH: Current Hash Level

Cutoff = (QL / P2) + (P3 * (HH - CH))

If Cutoff > P4, then Cutoff = P4

- Gap: Penalty for one residue indel
- Del: Penalty per residue for continuing indel

You may tweak below (0 means use default value):

- Pl: Length of Query Sequence
- P2: Maximum Hash Level
- P3: Length of Hash Table
- P4: Cutoff
- Gap: Penalty for one residue indel
- Del: Penalty per residue for continuing indel

For Scoring Matrix:

- PAM_250

For Hash Level:

- 2

For Automatic Processing:

- Save Align

For Source:

- G: Genbank
- N: Nbrf

Figure 1. Screen dumps from four phases of a global search
Figure 2. Screen dumps from four phases of a global search
available in the database, as they are in the GenBank floppy disk database and the IBI compacted NBRF database.

Now we may wish to examine a specific region of our query sequence.
Since this is a global alignment, the optimized scores reflect the value of the alignment to the WHOLE query sequence. A sequence may match a small region very well (a common functional site, for example) but the whole sequence only poorly (and thus be very low on the ranked list, mixed with hundreds of other alignments). We can define a subregion by picking from the features table of the query sequence (Figure 2B). In this case we have selected the signal peptide. We can then ask the program to rescore the optimized score ONLY over the defined subinterval. In this example, we are asking to rerank all entries based only on their similarity to the signal peptide of human gastrin. In less than a minute, we have quite a different map (Figure 2C) or aligned sequence display (Figure 2D). If these look useful, we can print a hard copy of this result as well.

Interactively, we can then work through many different aspects of our data to obtain maximal insight.

DISCUSSION

We have seen how two common molecular biology computing tasks can be greatly enhanced through the use of an interactive user environment. In addition to ease of use and attractiveness of display, important scientific results can be gained, which would be difficult or impossible to achieve otherwise. In any new or creative science, it is essential that scientists be able to interact with their analyses, to try ideas and evaluate options quickly and easily. The Cyborg environment is a first step in that direction.

REFERENCES