DNA Workbench

James Tisdall
University of Pennsylvania

DNA Workbench

Abstract

In this paper we describe *DNA Workbench*, a program for working with DNA, RNA, and protein sequences. It is designed to solve several problems that arise in two domains. The first domain is that of the algorithm designer and implementor who is working in the emerging field of computational biology. The second domain is that of the worker in a genetics laboratory, who needs frequently to turn to the computer to perform analysis on existing or newly acquired nucleotide or protein sequences. The problems encountered in these two domains overlap to a considerable extent. The problems, and how they are addressed by DNA WorkBench, are discussed within.

DNA WorkBench addresses both of these groups with one program. In this way, new problems that require new algorithms can be quickly brought from a theoretical solution to an implementation and to the laboratory workbench. This rapid transfer from research to development to the field is essential in a fast moving area such as biotechnology, by which term I mean to encompass such specialties as molecular biology, genetics, human gene therapy, and the current large-scale international sequencing and mapping of human DNA which has been organized as the Human Genome Project in the United States.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-93-38.

DNA WorkBench

MS-CIS-93-38
LOGIC & COMPUTATION 61

James Tisdall

LEGES SINE MORIBUS VANAE

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

March 1993

DNA WORKBENCH

James Tisdall
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
email: tisdall@cbil.humgen.upenn.edu
Advisor: Sanguthevar Rajasekaran

Technical Report MS-CIS-93-38
Logic and Computation 61

March 12, 1993

Abstract

In this paper we describe **DNA Workbench**, a program for working with DNA, RNA, and protein sequences. It is designed to solve several problems that arise in two domains. The first domain is that of the algorithm designer and implementor who is working in the emerging field of computational biology. The second domain is that of the worker in a genetics laboratory, who needs frequently to turn to the computer to perform analysis on existing or newly acquired nucleotide or protein sequences. The problems encountered in these two domains overlap to a considerable extent. The problems, and how they are addressed by **DNA WorkBench**, are discussed within.

DNA WorkBench addresses both of these groups with one program. In this way, new problems that require new algorithms can be quickly brought from a theoretical solution to an implementation and to the laboratory workbench. This rapid transfer from research to development to the field is essential in a fast moving area such as biotechnology, by which term I mean to encompass such specialties as molecular biology, genetics, human gene therapy, and the current large-scale international sequencing and mapping of human DNA which has been organized as the Human Genome Project in the United States.

Introduction

Genetics and molecular biology are in the midst of an explosive growth in the power of their techniques, and in the applicability of their results to medicine and industry. [Watson et al] Computer science is playing an increasingly important role in this growth. [Gilbert]

To put this development in perspective, one can reflect on the nature of evolution and the history of the Earth. Biotechnology reflects the emerging ability of life on Earth to begin consciously to affect the evolution of life on Earth. This is a revolution for which it is difficult to find a parallel. It is fascinating to reflect that this revolution is concurrent with the beginnings of the spread of life from the Earth to the rest of the solar system, as the Space Age and the Age of DNA overlap.

The computer tools that are coming online to address this challenge are of course strongly influenced by the nature of the genetics data. Human DNA can be represented by a string of approximately three billion characters. Each character can be one of 4 possible choices. It is quite probable that this entire sequence will be available within ten years. [Genome Project] There are important differences between the DNA of different individuals; such differences are critical in the detection and treatment of diseases, and in criminology, for example. Many other organisms are also being studied, and knowledge of their DNA is essential for the proper study of human DNA, as well as in existing and emerging industries. Thus, the data that must be maintained will soon be of very high volume and of critical importance. At present, a gigabyte disk or two is sufficient to store all of the major databases.

The emerging field of computational biology addresses the intersection of biotechnology with computer science. Aside from the data storage and retrieval aspects, there are also many significant algorithms. The most common algorithms now used by biologists are those that compare two or more sequences for similarities. Many current and anticipated data analysis and experimental design tasks require both theoretical work on algorithm design, and engineering work in systems building.

DNA WorkBench is a small, fast, distributed, portable, free program for genetics laboratory workers and algorithm designers for computational biology. In its first release, described here, it is a convenient environment for algorithm design and implementation. It provides an easy interface to multiple sequence file formats; very rapid and general database search and retrieval; a suite of standard sequence manipulation routines; and a simple and convenient data structure and operators, that provide the programmer with an interface to files, databases, users, and processes.

In addition, *DNA WorkBench* is an interactive tool for the working laboratory geneticist, in which great care has been taken to provide the simplest, fastest, and most easily learned user interface possible.

Database Retrieval

Database search and retrieval is fast and powerful in *DNA WorkBench.* Queries for locus id, accession number, organism, and keyword return instantly. (These are the most common fields on which searches are desired against the major databases.) Other searches, such as for arbitrary text or sequence, are accomplished by parallel scans of the database in its entirety.

There are several genetic databases in common use. In the discussion which follows, the database used for examples is GenBank (Genetic Sequence Data Base), maintained by NCBI (National Center for Biotechnology Information) [GenBank] and considered to be a standard. *DNA WorkBench* also uses PIR (Protein Identification Resource), SwissProt (Swiss Protein Database), and several specialized databases. Other databases have been configured, such as EMBL (European Molecular Biology Laboratory) but are not currently installed, as they essentially overlap with GenBank. Adding new databases to the system has been quite straightforward.

GenBank is composed of over 100 thousand records of sequences mixed with textual data, requiring about 350 MBytes of disk storage in its uncompressed "flat file" form. It is freely available from public ftp sites. The records include several fields of text, including ID numbers, authors, known features of the DNA, translations to protein, and more. The sequence is presented as a string over a standard alphabet, formatted with whitespace and counts.

For purposes of the most common data retrieval tasks, *DNA WorkBench* can return instantly with the requested information. This is accomplished by using DBM files and precomputing the file and offset as the value for each keyword. The location data is compressed into the minimal possible number of bits (four bytes per record), and accessed via an associative array interface to the keyword-value(s) hash table. Upon retrieval, the locations are decoded, and then the files are opened via a local or NSF symbolically linked file, the offsets are applied with a seek system call, and an entire record is retrieved with one read system call. In interactive mode, instead of immediately returning the record, the compressed pointer is added to the user's array of found sequence, to be actually fetched when the user requests some operation on it.

Programming the DBM interface presented some problems. The SUN workstations we use provide the NDBM version of the software (the main advantage over the original DBM software being the ability for multiple DBM files to be opened simultaneously.) Although most of the keywords need to store only a single record pointer of four bytes, some require very much more. (For instance, doing an ORGANISM lookup on the keyword "sapiens" requires storing pointers to over 25 thousand records, a 100 thousand byte-long vector.) NDBM begins returning error messages at about the 1000 byte range, but still stores the values, overwriting the data for other keywords. This led to recompiling the Perl implementation language with the Gnu distribution GDBM package, which permits arbitrarily long values. However, once a very long value has been stored, GDBM begins preallocating very long buffers for new values, making it unusable.

Several additional alternatives were considered, such as the SDBM package, and using values that were pointers to blocks of vectors. The final solution adopted was to create two GDBM indexes when necessary, one for the common short-valued case, and one for the keywords with long value fields. In practice, the user cannot detect the difference between one or two DBM file lookups: both seem to exhibit instant response. The current indexes require about 20 Mbytes for 350 Mbytes of data; they handle several kinds of ID numbers, genus and species information, and keyword information.

Searches for arbitrary text or sequence over all of GenBank or other databases take about five minutes using a single Sun SPARCstation 2 UNIX workstation; we use a network of 10 workstations to bring this time down to about 30 seconds. (This considerably outperforms the standard commercial software, by a factor of over 100; in fact, unrestricted search is often not possible.)

The network search is done by a client program opening Berkeley sockets and sending one request to each server process to initiate the servers, and then reading the sockets and processing the hits. The servers are standard internet TCP/IP concurrent connection-oriented stateless servers. *DNA WorkBench* uses the "*inetd*" service as the master server, which simplifies the code and reduces the overall process load on the systems. A subroutine called "DBcall" handles all network requests and service. Alternate configurations are provided as plug-in's in place of the standard DBcall subroutine; these are 1) a version that is designed to be started as a daemon at boot time, as from the /etc/rc.local file; 2) a version that uses rsh instead of sockets; 3) and a non-network version for a single CPU. Since both the client and the server code are contained in one subroutine, alternate configuration is fast, and configuring *DNA WorkBench* to utilize a network is quite easy. At the top of the program the location of the database files are specified by machine and pathname; it is only necessary to put the files on various machines and to specify their locations for the program to run the searches on the various machines in parallel. Reconfiguring the program to use different servers, or to change the order in which the client controls the servers, is accomplished by just editing the list of machines and pathnames.

Database searches may include regular expressions. The regular expression package includes quite sophisticated features, which make it easy to construct a very powerful query on the text fields, especially if the user has some familiarity with the format of the database entries; a familiarity that is attainable by means of the online help system. The implementation language is optimized for regular expression operations; the operations are based on Henry Spencer's regular expression package.

Regular expressions are of particular interest to students of DNA, since the most common use of computers, aside from data storage and retrieval, is in searching for similarities in a database. Regular expressions are sometimes of more value than searches that find all similar sequences, since more specific queries are possible. For example, "point mutations" are frequently of interest, where some particular base (represented as one letter in the string) has changed to another. As another example, deletions or insertions of substrings model a common type of DNA mutation.

Other, more powerful automata are also extremely useful, and the author hopes to add a subroutine which runs on a networked parallel machine for context-free language searching in the future. Of course, the programmer has a Turing machine available for arbitrary computations via the implementation language.

Data Storage

As mentioned in the previous section, the genetic data is stored as ASCII "flat files", requiring over 350Mbytes for GenBank. Simply reading these files from disk on a single CPU requires about 3 minutes in our environment. Clearly, some form of data compression could significantly decrease the I/O time.

The implementation language provides a full set of operators for working with various forms of data compression. Significantly, most DNA sequence data is over a four letter alphabet, permitting a fourfold compression from the ASCII representation.

Certain compression formats are de facto standards in the genetics community. One is that provided by the BLAST software, which is further discussed below. This achieves very small space requirements by discarding most of the text, and using an efficient compression on the sequence.

An emerging standard is ASN1, the ISO presentation layer protocol format, which has been adopted by NCBI.

Also in common use is the GCG format that comes with the Genetics Computer Group software package.

In addition, other data compression schemes suggest themselves. However, it is probably most practical to attempt to integrate the system with compressed data that already exists on disk, rather than introducing yet another new compression scheme, unless significant benefits can be shown to accrue.

Our plans are to explore the use of data compression formats. There will clearly be a tradeoff between compression and processing time; plus there is the issue of discarded data with BLAST. It is not inconceivable, however, that we may achieve comparable time performance with significantly reduced disk storage requirements. This is the next major task we expect to undertake on the system.

User Interface

Once sequences have been loaded by reading a file or performing a database search or retrieval, the user prompt shows how many sequences have been loaded, and which of them is the "current sequence". One moves to a different sequence by giving a number or displacement from the current sequence, after the style of the UNIX "ed" or "vi" editors. One can also specify ranges of sequences to operate on.

Most commands act upon the current sequence by default. Thus once a desired sequence has been loaded and verified, typing a short command will run the desired algorithm on that sequence, saving the user the trouble of continually remembering and specifying filenames. The simple user interface requires few keystrokes, due to a shortest distinguishable command abbreviation facility.

There is only one menu, available by a single keystroke. Most commands simply return to the main loop if a useful response is not given to a prompt, so that aborting a procedure only requires one to hit RETURN one or two times.

No additional user documentation beyond the on-line menus and help is necessary or provided. Approximately five minutes self-training time has been shown to be sufficient for the laboratory worker to use the program effectively. Users are advised of the most common system commands by examples, and the on-line system manual pages are referred to when necessary.

No special graphics terminal is required. In future, XWindows, Mac, and Windows NT graphics are planned, but in a simple, modular fashion that will not unduly complicate using and programming in the system.

Sequence File Formats

The genetics programming community has developed an unfortunate number of alternate file formats for essentially the same purpose, that of presenting sequence data with relevant textual information. This causes significant impediments to the effective use of computer tools in the laboratory. *DNA WorkBench* addresses this problem with a comprehensive set of subroutines for sequence recognition, extraction, and formatting in all of the most common sequence file formats. These are integrated into the system so that the user rarely need be aware of them. Most files are read with no user intervention required.

Files that are to be read are by default analyzed for their format, so that user intervention only becomes necessary when the file is in an uncommon format, or has become garbled. Reformatting for editing or output is easy. The user can reformat any entry into any format by specifying the desired output format; *DNA WorkBench* determines the existing format, reformats, and either loads the newly formatted sequence into the work space, or writes it out. Files with entries in mixed formats are also handled. The user can specify a default output format for bulk reformats and writes to files. All functions included in the package take similar actions to hide the formatting problem from the user.

The low-level formatting routines are packaged into alternate higher-level subroutines for reading, writing, display, and sequence and text extraction. The programmer can build an application calling these subroutines for all common nucleotide and protein sequence file format I/O.

It is an interesting observation about our implementation language, that our format-handling subroutines are a fraction of the size of functionally equivalent C code, and are much easier to amend or extend. This highlights the programming efficiency of using a string-oriented interpreted language to deal with string data.

Commands

Besides database retrieval, *DNA WorkBench* can also perform all standard sequence manipulation tasks.

- It can take the reverse complement, adding the opposing strand of DNA as a new entry in the work space.
- It can perform translations from nucleotides to proteins, and report on reading frames, translations of "codons", groups of three nucleotides that specify an amino acid, that do not contain a "stop codon". (This is the "genetic code".)
- It can edit sequence in your choice of editor, loading the changes into the workspace automatically when the editor is exited.
- It can x-out undesired bases specified as ranges.
- It can apply regular expressions globally and present the result as a report and a choice of actions to take.
- It can submit commands to the local database management system (DBMS). At our location this takes the form of a general SQL command interface to our Sybase data system, and as a single command that extracts all locally entered sequence and performs similarity comparisons on it.
- It can enter the running program in an interpreter mode to perform arbitrary commands not available through the menus. (See below.)
- It can easily submit sequence for BLAST or FASTA similarity searches against several general and specialized sequence databases.
- It can call the PRIMER program to study a sequence preparatory to laboratory cloning.
- New algorithms and commands are easy to add, and are being added as practice dictates.

Commands not special to *DNA WorkBench* are passed directly to the computer's command interpreter system ("shell"). Thus the user can interact with the computer in the normal fashion, while maintaining an additional environment of sequence and commands. The advantage of this approach is that it maintains the accessibility of the normal command environment, while adding an array of sequence and commands that can be easily manipulated without a lot of file reading, writing, and naming overhead. The functional result is similar to having a special-purpose "genetics shell" command interpreter.

Providing access to FASTA and BLAST proved to be slightly non-trivial. BLAST, the popular favorite among genetics researchers due to its very fast calculation of sequence similarities, is an I/O bound program. It was necessary to execute the program on the remote system that houses the compressed databases in order to achieve the speed for which the program is noted. This is exactly the type of task at which the implementation language excels.

FASTA similarly runs much faster on the CPU that controls the storage disk. In addition, there is a peculiarity about FASTA that had to be overcome. At present, FASTA is optimized to run from its own FASTA-formatted version of the GenBank database, which we do not store. Therefore, we configure FASTA to use the GenBank flat files. However, although this is an option provided by FASTA, the program then does not allow for command-line, non-interactive execution. Once again, the implementation language provided convenient mechanisms by which FASTA could be executed and controlled on a remote machine by a local process, appearing to the user as a function to be called by a few keystrokes.

Programming Environment

DNA WorkBench offers an attractive programming environment. With a simply designed and convenient interface to sequence, files, databases, and users, it supports rapid program development and use. Existing and new functions can be called and are callable from such other languages as C, FORTRAN, and Prolog.

The entire code is only, as of this writing, 15 printed pages, under 2000 lines, in one file. Almost half of the code comprises the file handling and help subroutines. This brevity eases the reading and modifying of the code. All functions, including client, server, and index making, are packaged together, which helps integrate the package and promotes porting to new systems. *DNA WorkBench* has been written with portability in mind, although it has not yet been ported.

DNA WorkBench is written entirely in the Perl language. [Wall] Thus the code bears much resemblance to C, Shell, and AWK program syntax. Perl is an excellent language for string manipulation and for the other tasks for which *DNA WorkBench* has been designed, such as internet programming, user and program interfaces, string manipulation, file format handling, and ease of modification. In addition, Perl is free, well documented, and has an active and rapidly growing international community of users. Perl has been ported from its UNIX birthplace to MacIntosh, DOS, DOS/Windows, Windows NT, and VMS systems. Although it may at some point be desirable to recode some of the "most inner" loops in C, so far this has not proved to be necessary.

One command option in *DNA WorkBench* is to enter the running program as a command interpreter. Thus the entire program state can be examined, and all functions are accessible interactively to the programmer. By using the Perl "require" program loader, it is easy to add specialized functions interactively, and to perform customized computations without altering the existing program code.

Conclusion

DNA WorkBench began as a question: How far can we take Perl to solve problems in computational biology program development and in meeting the computer needs of genetics laboratory workers?

The answer has far exceeded our expectations, producing a system that matches or outperforms the standard commercial software packages in several key areas, such as database retrieval and search, user interface, sequence file format handling, programming environment, and the use of multiple processors.

As a rough measure of performance, we note that doing a regular expression search for sequence over all of GenBank takes about 30 seconds using a network of 10 CPU's; the popular BLAST program, running on a single CPU (as it is written to do) takes over a minute to perform a similarity search in our environment. Of course, these are different kinds of searches, and one is multiprocessing while the other is not; yet this comparison does provide some measure of performance.

DNA WorkBench has a small size, is easily reconfigurable, and took one person three months of part-time work to write. It is in daily use in genetics laboratories, and has begun service as a programming environment for algorithm development for computational biology.

REFERENCES

- Watson,J.D., Gilman,M., Witkowski,J., and Zoller,M. **Recombinant DNA 2nd Edition** New York, Scientific American Books, 1992
- Walter Gilbert *Towards a paradigm shift in biology* **Nature** Vol 349 10 Jan 1991 p.99
- U.S. Department of Health and Human Services and U.S. Department of Energy **Understanding Our Genetic Inheritance-The U.S. Human Genome Project: The First Five Years FY 1991-1995** National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia, 1990
- **GenBank Release 75** National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, Maryland, March, 1993
- Wall,L. and Schwartz,R.L. **Programming perl** O'Reilly and Associates, Sebastopol, California, 1990