Biogrep: a multi–threaded pattern matcher for large pattern sets

Kyle Jensen[1], Greg Stephanopoulos[1*], Isidore Rigoutsos[2]

[1] Department of Chemical Engineering, Massachusetts Institute of Technology,
Cambridge, MA 02139, USA

[2] IBM Research Division, Thomas J. Watson Research Center
Yorktown Height, NY 10598, USA

[*] To whom correspondence should be addressed; E-mail: gregstep@mit.edu.

Abstract

Summary: This paper introduces a new pattern matching tool called Biogrep. This tool is designed to quickly match large sets of patterns against biosequence databases and is optimized for multi–processor computers. Biogrep uses standard POSIX extended regular expressions and can divide the pattern–matching task between a user–specified number of processors.
Availability: Source code for Biogrep and packages for common GNU/Linux distributions are available under a GPL license at
`http://web.mit.edu/cheme/gnswebpage/`.
Contact: `biogrep@mit.edu`

Introduction

As more genomes are sequenced, increasing numbers of functional DNA and protein sequence motifs (or *patterns*) are being discovered. Searching for these motifs in biosequences can be an important part of the annotation process. Many databases such as Prosite (Hofmann *et al.*, 1999), PRINTS (Attwood *et al.*, 2003), and BLOCKS (Henikoff & Henikoff, 1991) contain collections of biologically significant patterns that correlate with the function of protein families. For example, the Prosite the motif `[AG]....GK[ST]` is indicative of ATP/GTP binding proteins.

There are a variety of tools available for pattern–matching. Most common are the "grep" family of Unix tools, including a number of very fast and sophisticated variants such as agrep (Wu & Manber, 1992) and NR–grep (Navarro, 2001). Also, there are many excellent bioinformatics–specific pattern–matching tools including Patscan (Dsouza *et al.*, 1997), tagc (Mangalam, 2002), and fuzzpro (Rice *et al.*, 2000). However, all of these tools are optimized for searching for single patterns, that is, one–at–a–time. In contrast, Biogrep is designed to match large pattern sets (100+ patterns) against large biosequence databases (100+ sequences) in a parallel fashion.

Method and Implementation

Biogrep is written in the C programming language using the GNU regular expression (Hargreaves & Berry, 1992) and POSIX threads (pthreads) (Mueller, 1993) libraries. The program reads query patterns from either a plain text file, one–per–line, or from a Teiresias–formated pattern file (Rigoutsos & Floratos, 1998). These patterns are treated as POSIX extended regular expressions and are searched against a user supplied file, which can be either a FastA (Pearson & Lipman, 1988) formatted biosequence database or any text file.

Table 1 shows a comparison of Biogrep with a few common programs. The grep family of pattern matching tools are absent from the table because their run times are extremely long. This is because many of these tools cannot take sets of patterns and have to be used on a per pattern basis. The next best alternative to Biogrep is a simple PERL script split between multiple processors.

Biogrep has a number of user options, which are described in the documentation that comes with the software. Most importantly, Biogrep can divide the pattern–matching task between a user–specified number of processors using threads. This drastically reduces the user–time required to match large sets of patterns (see Table 1). In addition, Biogrep is distributed with detailed documentation, numerous examples, and various helper–scripts for interfacing with other pattern matching/discovery programs.

Table 1: Performance of Biogrep matching all the 1333 patterns in Prosite (release 17.01) against the 782370 protein sequences in Swiss–Prot/TrEMBL (Bairoch & Apweiler, 2000) (release as of 8 July 2002). Runs were carried out on an IBM p670 eserver running AIX 5L with 8 Power4 processors.

program	# processors	execution time (s)
biogrep	1	8683
biogrep	2	4477
biogrep	4	2266
biogrep	6	1620
perl	1	11780
perl	6	1916
patscan	1	28466

Acknowledgments

This work was supported by the DuPont–MIT Alliance (DMA) and the Singapore–MIT Alliance (SMA).

References

Attwood, T. K., Bradley, P., Flower, D. R., Gaulton, A., Maudling, N., Mitchell, A. L., Moulton, G., Nordle, A., Paine, K., Taylor, P., Uddin, A. & Zygouri, C. (2003) PRINTS and its automatic supplement, prePRINTS. *Nucleic Acids Research,* **31** (1), 400–402.

Bairoch, A. & Apweiler, R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBl. *Nucleic Acids Research,* **28** (1), 45–48.

Dsouza, M., Larsen, N. & Overbeek, R. (1997) Searching for patterns in genomic data. *Trends in Genetics,* **13** (12), 497–498.

Hargreaves, K. A. & Berry, K. Regex. Free Sotfware Foundation, 675 Mass Ave, Cambridge, MA 02139.

Henikoff, S. & Henikoff, J. G. (1991) Automated assembly of protein blocks for database searching. *Nucleic Acids Research,* **19** (23), 6565–6572.

Hofmann, K., Bucher, P., Falquet, L. & Bairoch, A. (1999) The prosite database, its status in 1999. *Nucleic Acids Research,* **27** (17), 215–219.

Mangalam, H. J. (2002) tagc – a grep for DNA. *BMC Bioinformatics,* **3** (8).

Mueller, F. (1993) A library implementation of POSIX threads under unix. In *Proceedings of the Winter 1993 USENIX Technical Conference and Exhibition* pp. 29–41, San Diego, CA, USA.

Navarro, G. (2001) NR–grep: a fast and flexible pattern-matching tool. *Software Practice and Experience,* **31** (13), 1265–1312.

Pearson, W. R. & Lipman, D. J. (1988) Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences,* **85**, 2444–2448.

Rice, P., Longden, I. & Bleasby, A. (2000) EMBOSS: the European molecular biology open software suite. *Trends in Genetics,* **16** (6), 276–277.

Rigoutsos, I. & Floratos, A. (1998) Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics,* **14** (1), 55–67.

Wu, S. & Manber, U. (1992) Agrep — a fast approximate pattern-matching tool. In *Usenix Winter 1992 Technical Conference* pp. 153–162, San Francisco.