CABIOS

Vol. 11 no. 2 1995
Pages 219–223

A compression mechanism for sequence databases to improve the efficiency of conventional tools

R.Doelz and F.Eggenberger[1]

Abstract

This paper describes a method to compress molecular biology databases that are characterized by an increasing proportion of data derived from genome projects. The performance of our tool has been tested on various data files of the EMBL nucleotide sequence database. The best compression ratios were achieved on EST (Expressed Sequence Tags) data, typically derived from large-scale sequence projects. The compression of sequence database updates was tested in combination with the common Unix compression program 'compress'. Our tool improved the efficiency of 'compress' on average by 16%.

Introduction

Biological databases are comprehensive sources of information that allow the correlation of new experimental findings with already established results. In the field of molecular biology, the first step in the analysis of a new sequence is usually the search for homologies in a sequence database. The majority of these databases are organized as simple flat files which are distributed in separate divisions based on taxonomic aspects. The status of existing sequence databases is regularly published in *Nucleic Acids Research* (e.g. Benson *et al.*, 1993; Rice *et al.*, 1993). Currently, the collectively consumed volume of the major nucleotide and protein sequence databases needed for a fully functional search and retrieval service amounts to ~ 4 Gbyte and is growing exponentially. New releases of the major nucleotide sequence databases, the EMBL Data Library (Rice *et al.*, 1993) and GenBank (Benson *et al.*, 1993), are distributed four to six times per year. The preferred medium to distribute the databases is currently CD-ROM with magneto-optical media to be expected soon (for a review, see Mewes *et al.*, 1994). To meet the need of more rapid access to the data, both the full database releases and those entries that will be incorporated in the next release are also distributed over electronic networks. However, making available entire databases on the network causes a continuous growth of network traffic. Depending on format, the releases of the EMBL and GenBank nucleotide sequence database currently comprise 200–400 Mbytes of data and the volume of updates typically range between 20 and 100 Mbytes. To tackle the network load caused by the distribution of sequence data over the network, the development of efficient compression tools is gaining importance.

Data files with image information can efficiently be shrunk using compression algorithms that irretrievably discard part of the original data. Such methods are not acceptable for text or database files where not a single bit of information can be lost during compression or decompression. Programs that meet that requirement, so-called loss-free methods, can roughly be divided into Huffman coding, arithmetic coding and substitutional compression. The first two algorithms take into account the probability of symbol occurrence and can produce superior results than substitutional compression. Huffman coding, however, makes assumptions on the distribution of the probabilities of symbol occurrence. This is not the case for compressors based on arithmetic coding, which on the other hand consume rather large amounts of computer resources. Since the performance of a compression program depends on both speed and efficiency, generally used compressors are based on substitutional compression. Such programs work by replacing strings of the input file by references which are indexed in a dictionary. There are several variants of substitutional compressors that differ mainly in the principle how the program manages the dictionary.

Specific compressors work better on some types of files than on others and therefore should be optimized for different data types. A characterization of the data to be compressed, therefore, is essential. Each entry in a biological sequence database is composed of sequence data and accompanying descriptive information (annotation) made up of different line types that begin with their own line code. Whereas information was collected by individual research groups in the past, today's sequence databases typically contain increasing numbers of entries derived from genome projects which are automatically processed and incorporated into the databases. These data are of very different quality and can be characterized by high redundancy in the annotation sections. The

Biocomputing, Basel University, Biozentrum, Klingelbergstrasse 70, CH-4056 Basel, Switzerland

[1]*To whom correspondence should be addressed*

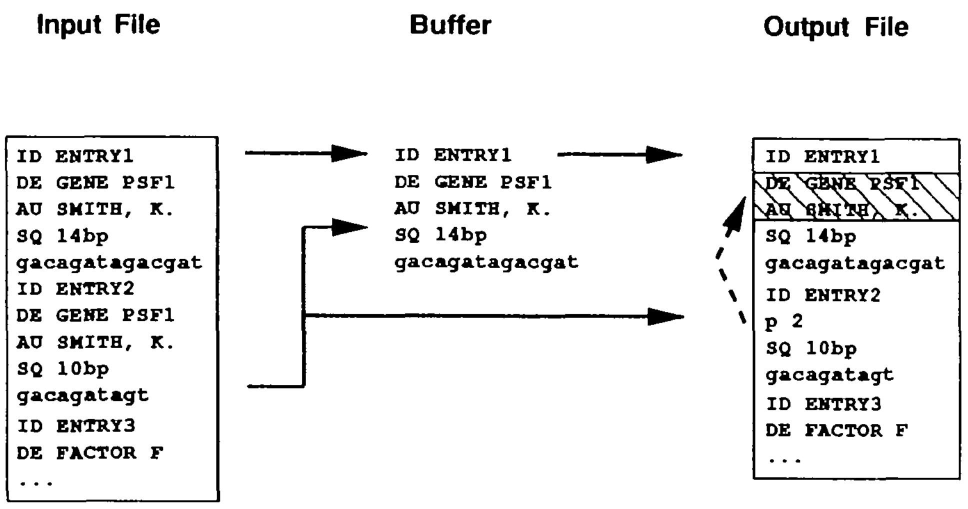

Fig. 1. Flowchart of data processing. Building the output file is achieved by comparing two consecutive database entries.

differences between two successively submitted entries derived from genome projects mostly concern only the sequence data, whereas parts of the annotation sections are often identical.

System and methods

In this paper we report on a compression tool for sequence databases that has been developed to improve the efficiency of conventional compressors by eliminating identical parts of consecutive database entries. The main benefit is the internal referencing of lines within a group of entries collected in a single file. In contrast to incremental updating of entries based on version control, the new method does not require external standards, such as defined versions of given entries. Further, as our program accepts any input of entries, it specifically works on independent entries, which implies its use for sequence database compression in general rather than for only updates to existing entries.

DBCOMP and its counterpart DBUNCOMP are part of a set of sequence database tools developed at the Biocomputing Laboratory in Basel. The code has been written in ANSI C using the NCBI Vibrant windowing system (Ostell, 1993) where applicable. DBCOMP and DBUNCOMP have been tested on Unix (SGI Irix, DEC Ultrix, DEC OSF/1, IBM AIX, Sun SunOS), VAX/VMS, AXP/VMS, DOS/Windows and MacOS. EMBL database quarterly entry statistics were based on data collections calculated with the DBTOOLS package (R. Doelz and L. Rosenthaler, unpublished: available from nic.switch.ch). Statistical analyses of heterogeneity among the observed compression ratios were performed using Kruskal–Wallis tests as implemented in the SAS program package (SAS Institute Inc., 1990).

DBCOMP starts compression by putting each line of the input file into a buffer until the beginning of the following entry is encountered. The identification code of the beginning of an entry is configurable in DBCOMP. The first entry is written unchanged to the output file. In a second step each line of the following entries is compared with the corresponding line of the previous entry that is stored in the buffer (Figure 1). Lines that are not identical with the corresponding lines of the previous entry are put into the buffer and then written to the output file. Lines found already in the buffer are referenced by a token. Having encountered the end of the input file, the program attaches a file header to the compressed output file that includes information about (i) the name of the original sequence database file (input file), (ii) the number of entries, lines, and characters of the input file, (iii) the compression ratio (percentage saved), (iv) the token used to recognize the first line of entry, and (v) the version of DBCOMP used. In order to ensure the proper reassembly of the processed input file, the program then writes information about the program version to an external file and finally prints a summary report on the screen.

DBUNCOMP is needed to decompress sequence database files that have been compressed by DBCOMP. In order to ensure the integrity of the processed output file, DBUNCOMP starts processing by reading the file header of the input file and checks the information provided by the external file. After having successfully completed this check, the program starts decompression by putting each line of the input file into the buffer until the beginning of the following entry is encountered. As in DBCOMP the first entry is written unchanged to the output file. The lines of the following entries are put into the buffer and then written to the output file, with the exception of those lines which have been replaced by a reference during

Table 1. The compression efficiency of DBCOMP in combination with 'compress' as tested using EMBL sequence database flat file sections (details see text)

	1988	1989	1990	1991	1992	1993	Heterogeneity[a]
ESTs				5.93 (4.52)	7.28 (5.36)	8.40 (5.57)	NS[b]
Viruses	2.48 (2.47)	2.71 (2.65)	3.46 (3.14)	3.60 (3.21)	4.15 (3.49)	3.99 (3.38)	**
Primates	3.11 (2.85)	3.24 (2.93)	3.47 (3.16)	3.63 (3.24)	3.94 (3.40)	3.99 (3.38)	*
Unclassified		2.81 (2.60)	3.64 (3.10)	3.08 (2.61)	3.07 (2.77)	3.39 (2.92)	NS
Rodents	3.12 (2.84)	3.12 (2.85)	3.41 (3.16)	3.51 (3.18)	3.82 (3.31)	3.75 (3.30)	NS
Other mammals	2.77 (2.55)	3.05 (2.81)	3.31 (3.04)	3.39 (3.10)	3.54 (3.18)	3.72 (3.25)	NS
Synthetics	1.86 (1.96)	2.42 (2.23)	3.30 (2.81)	3.17 (2.87)	3.30 (3.04)	3.38 (3.00)	NS
Other vertebrates	2.83 (2.62)	3.04 (2.72)	3.36 (3.08)	3.39 (3.11)	3.54 (3.18)	3.50 (3.17)	NS
Plants	2.98 (2.65)	2.99 (2.74)	3.33 (3.10)	3.35 (3.10)	3.43 (3.15)	3.42 (3.18)	NS
Invertebrates	2.90 (2.62)	2.84 (2.65)	3.51 (3.13)	3.41 (3.11)	3.53 (3.19)	3.45 (3.21)	NS
Organelles	2.80 (2.45)	2.88 (2.68)	3.18 (2.89)	3.58 (3.15)	3.16 (3.27)	3.52 (2.38)	NS
Prokaryotes	2.86 (2.76)	2.97 (2.87)	3.19 (3.10)	3.30 (3.13)	3.31 (3.15)	3.29 (3.14)	NS
Fungi	2.67 (2.63)	2.92 (2.77)	3.21 (3.08)	3.28 (3.16)	3.38 (3.19)	3.26 (3.15)	NS
Bacteriophages	2.23 (2.24)	2.62 (2.59)	2.89 (2.79)	2.92 (2.85)	2.97 (2.92)	2.89 (2.86)	NS

Compression ratios (input characters divided by output characters) of 'compress' alone are given in parentheses. The change in the performance of DBCOMP on database divisions from 1988 onwards is given as heterogeneity among differences between compression ratios achieved with DBCOMP in combination with 'compress' and 'compress' alone.
[a]Kruskal–Wallis test (chi-square approximation).
[b]Not significant.
*$P < 0.005$.
$P < 0.01$.

compression. Having encountered a reference, the program puts the corresponding lines of the previous entry from the buffer into the output file. When reading the end of the input file, the integrity of the decompressed file is checked by comparing the number of entries, lines and characters of the decompressed file with those of the original database file. The programs terminates after reporting a summary.

The sections of the EMBL database were calculated by collecting the new entries added in each quarter and subdivision of the EMBL database in the years 1988 to 1993. Using the Date information of the entry creation, the programs as available in the DBTOOLS package (R. Doelz and L. Rosenthaler, unpublished) were used with slight modifications to create quarterly incremental EMBL database files in the original EMBL flat file format. Weekly updates were collected from the EMBnet update stream as provided by EMBL Heidelberg, after separation for new entries showing up the first time in the current release (named XEMBL set), and those entries which have experienced changes in their contents but were available already earlier (XXEMBL).

Implementation and discussion

The idea behind all substitutional compressors, including DBCOMP, is to replace a particular piece of data with a reference to a previous occurrence of that piece. Typical substitutional compression schemes such as the LZW method used in 'compress' (Sperry Corporation, 1983; IBM, 1983), generate a dictionary of substrings that is built up character by character. In contrast, DBCOMP is line-oriented. By using a flexible wordsize of one line, the program replaces identical lines with pointers to the previous occurrence of those lines. Conventional substitutional compressors, however, reduce redundant phrases of rather small wordsize, but do not eliminate identical lines of consecutive entries, which is a typical source of redundancy of automatically acquired biological sequence data.

The previously simulated set of separate divisions of a total of 21 partial data sets of the EMBL nucleotide sequence database was compressed using DBCOMP and then 'compress' and using 'compress' alone. The resulting compression ratios are summarized in Table I. As expected, DBCOMP performs better on database files of divisions which contain rather large proportions of automatically acquired data. The highest compression ratios were observed with database files of the EST (Expressed Sequence Tags) division. EST data are partial DNA sequences derived from randomly selected cDNA clones (Benson *et al.*, 1993). ESTs are submitted in bulk and annotated automatically, which results in high redundant annotation section and explains the observed compression ratios ranging from 5.9 to 8.4 (Table I). We observed a similar performance with the recently introduced STS division in the EMBL and GenBank nucleotide sequence databases.

The change in the observed compression efficiency on EMBL database divisions from 1988 onwards, as calculated using Kruskal–Wallis tests on differences between ratios achieved with DBCOMP in combination with 'compress' and 'compress' alone, is given in Table I. As a result of the constant proportion of the automatically

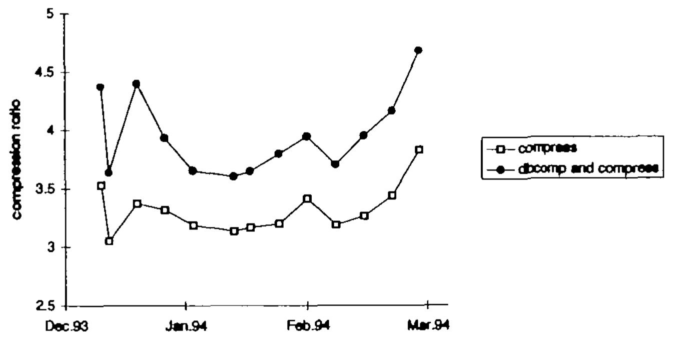

Fig. 2. The performance of DBCOMP in combination with 'compress'. Compression ratios achieved with new entries of weekly updates of the EMBL nucleotide sequence database (XEMBL).

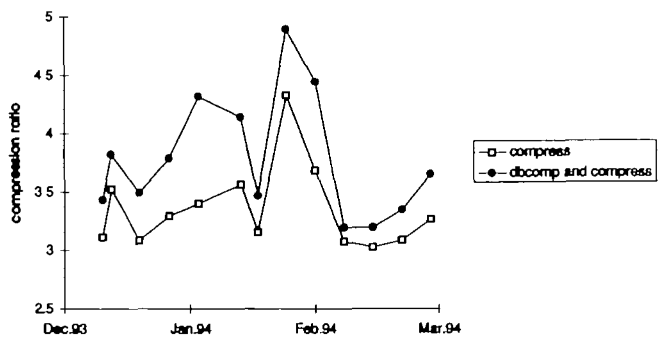

Fig. 3. The performance of DBCOMP in combination with 'compress'. Compression ratios achieved with updated entries of weekly updates of the EMBL nucleotide sequence database (XXEMBL).

processed EST entries, there was no significant increase in the compression efficiency on EST data collected since 1991. In contrast, the compression efficiency on database files of the 'primates' and 'viruses' divisions increased significantly from 1988 onwards (Table I). This might reflect the growing importance of the automatically acquired data in these divisions. Indeed, the incorporation into the database of sequence data derived from large-scale genome projects is increasing and even expected to become the major source of data for sequence databases (Higgins *et al.*, 1992). The EMBL Data Library, for example, started the incorporation of data from genome projects in 1991. Only two years later, release 34 of the nucleotide sequence database contained 20% of automatically processed sequence data (Rice *et al.*, 1993).

The compression performance of DBCOMP and DBUNCOMP could be increased by the use of several buffers of a larger size. Similarly, a further increase of compression efficiency can be achieved by a reduction of the minimal wordsize of less than one line. We tried several changes of parameters (data not shown), and measured efficiency after compression with LZW-type compressors. The resulting improvements proved to be minimal. This can be attributed to the fact that smaller segments of lines are well captured by the substitutional methods used in the conventional type of compressor. As a drawback, increased sophistication resulted in larger code and lower processing speed. Data compression, however, should be a reasonable compromise between efficiency and speed. To meet this requirement, the resources used by DBCOMP and DBUNCOMP were kept as minimal as possible. The program allocate a single buffer with a size that has been set to 49 Kbyts, the current DOS version even compiles using the small memory model that produces the fastest run-times. Moreover, DBCOMP and DBUNCOMP are not intended to tackle the redundancy that can already be eliminated by using established compression schemes. Our progams are rather intended to work together with conventional compressors such as the Unix program 'compress', as mentioned above.

The performance of DBCOMP has been tested on incremental updates of the EMBL nucleotide sequence database. The EMBL database is updated every 3 months. In Basel, the daily updates of the EMBL database that contain new and updated entries are accessible as XEMBL and XXEMBL database respectively. Since these data are delivered to the end-user exclusively via a network, an efficient compression scheme is indispensable. Thus, we analyzed the efficiency of DBCOMP on weekly updates of both the XEMBL and XXEMBL database. A set of 13 weekly updates collected since December 1993 were compressed using 'compress' alone and using DBCOMP and then 'compress'. The observed compression performance achieved with the two procedures is illustrated in Figures 2 and 3. The average compression ratio (input characters divided by putput characters) was 4.0 ± 0.1 for new entries (XEMBL) and 3.8 ± 0.1 for updated entries (XXEMBL). The difference between the compression ratio of DBCOMP in combination with 'compress' and 'compress' alone is attributed to the compression performance of DBCOMP. The average of this difference was 0.65 ± 0.05 for XEMBL and 0.43 ± 0.06 for XXEMBL updates (Figures 2 and 3), reflecting a higher proportion of automatically generated sorted entries in XEMBL than in XXEMBL, which is to be expected. The large variations seen in both graphs reflect the heterogenous composition of weekly updates.

To summarize, the tools described in this paper provide both a simple and efficient method to improve the compression of nucleotide sequence databases distributed over the network. Due to its reasonable compromise between efficiency and speed, this new method could also be adapted for real-time compression that could easily be incorporated into molecular biological applications relying on large database input.

Availability

The source code of DBCOMP and DBUNCOMP suited to run on most of the operating systems in use today can be downloaded by anonymous FTP from nic.switch.ch in the mirror/embnet-ch directory. Both text-driven versions and code to link to the NCBI toolbox to obtain a graphical user interface are provided.

Acknowledgements

Thanks to J.Epstein and J.Kans for providing help with the NCBI toolbox and C.Wadley for improving the manuscript. This code has been written at the Biocomputing Laboratory in Basel. Financial supported has been provided by the University of Basel and the Swiss National Science Foundation.

References

Benson,D., Lipman,D.J. and Ostell,J. (1993) GenBank. *Nucleic Acids Res.*, **21**, 2963–2965.

Higgins,D.G., Fuchs,R., Stoehr,P.J. and Cameron,G.N. (1992) The EMBL Data Library. *Nucleic Acids Res.*, **20**, 2071–2074.

IBM (1983) Software patent 4814746. Inventors Miller,V.S. and Wegman,M.N.

Mewes,H.W., Doelz,R. and George,D.G. (1994) Sequence databases: an indispensable source for biotechnological research. *J. Biotechnol*, **35**, 239–256.

Ostell,J. (1993) NCBI Software Development ToolKit Available by anonymous FTP from the National Center for Biotechnology Information at ncbi.nm.nih.gov.

Rice,C.M., Fuchs,R., Higgins,D.G., Stoehr,P.J. and Cameron,G.N. (1993) The EMBL Data Library. *Nucleic Acids Res.*, **21**, 2967–1971.

SAS Institute Inc. (1990) *SAS/STAT User's Guide*, version 6. SAS Institute Inc., Cary, NC.

Sperry Corporation (now Unisys) (1983) Software patent 4558302. Inventor Welch,T.

Received on November 1, 1994, revised on November 11, 1994; accepted on January 13, 1995