A Fixed-Length Coding Algorithm for DNA Sequence Compression (Draft, using Bioinformatics \LaTeX template)

Jie Liu\textsuperscript{a}, Sheng Bao\textsuperscript{b,∗}, Zhiqiang Jing\textsuperscript{c}, Shi Chen\textsuperscript{c}

\textsuperscript{a}Dept. of Computer Science and Technology, Nanjing Univ.of P \& T, Nanjing, Jiangsu 210046, CHINA, \textsuperscript{b}Dept. of Information Engineering, Nanjing Univ.of P \& T, Nanjing, Jiangsu 210046, CHINA, \textsuperscript{c}School of Life Science, Nanjing University, Nanjing, Jiangsu 210093, CHINA

1 INTRODUCTION

File compression reduces file redundancy in order to represent more information in less signs in accordance with information theory (Shannon, 1948). As specified algorithm for image, audio and video are devised, it is necessary to devise the algorithm specified for DNA compression since huge amounts of DNA sequences need to be stored and communicated to a large number of people. (Korodi et al., 2005) (Cohen, 2005) Although some universal compressors (Ziv et al., 1977) are used in bioinformatics field, new DNA sequence compressors are being devised, such as Biocompress (Grumbach et al., 1993), Biocompress-2 (Grumbach et al., 1994), GenCompress (Chen et al., 2001), Cfact (Rivals et al., 2000), DNACompress (Chen et al., 2002), CTW-LZ (Matsumoto et al., 2000) and GeNML (Korodi et al., 2005).

But they have a big problem, too slow execution. We improve our LUT (Bao et al., 2005) and use new file structure to identify different types of segment. The most advantage of this algorithm is fast execution and easy implementation. The compression and decompression speed is much faster than many newly-devised DNA-specified and well-known universal compression algorithms. Since the compression ratio is not much higher than existing ones and the compression speed is impressively fast, our algorithm is an applicable algorithm for fast DNA sequence compression, especially for database records compression.

2 METHODS

2.1 Coding non-N bases

Non-N bases have four possibilities: A, T, G or C. Each of them corresponds to a unique combination of two binary numbers. We code them as A to 00, T to 01, G to 10, C to 11. Thus, we take 1 Byte (8 bits) to store 4 bases.

2.2 File format of compressed file

We will begin discussing file structure with the definition of “section”, a DNA segment. “section” contains a series of successive Ns and ends at the last non-N base ahead the next series of successive Ns. “section” is the basic element to which we consider in compression and decompression.

Each DNA section corresponds to a “file section” which contains the information of both N and non-N bases in this section. Each file section starts with an 8 Bytes head. The first 4 Bytes records the amount of N bases whereas the following 4 Bytes records the number of non-N bases in this section. This means that each section corresponds to a real DNA segment which has at most 2\textsuperscript{32} N bases and 2\textsuperscript{32}-non-N bases respectively.

The coded values of non-N bases locate after the head. The coded information is written into destination file Byte by Byte. Considering the number of non-N bases in a section may not be a multiple of 4, the second 4 Bytes in head provides accordance for decompression program about how many bit values are effective and where the next section begins.

2.3 Compression algorithm

The compression program reads characters from source file and writes coded binary values into destination file, restricted by the file format defined above. Steps of compression algorithm is as below.

1. Preserve 8 Bytes at the beginning of file section.
2. Count the number of Ns in a successive N bases segment (To a sequence starts from non-N bases, this value is 0) until the first non-N base is encountered. Write the number of Ns into the first 4 Bytes of the section head.
3. Code all following successive non-N bases into destination file while count their number until the next N is encountered. Write the number of non-N bases into the second 4 Bytes of the section head.
4. Move the file writing pointer to the beginning of next Byte in destination file.
5. Repeat all the above until the end-of-file is encountered.

2.4 Decompression algorithm

Steps of decompression algorithm is as below.

1. Read the head (the first 8 Bytes) to obtain information about how many Ns are in this section and how many non-N bases are effective.
2. Write Ns into destination file, the decompressed file, according to the number written in the first 4 Bytes of the head.
3. Read the following 4 Bytes to determine how many bits should be decoded then and where the next section begins. The next section begins from the most nearby next Byte of compressed file.
4. Decode effective bits whose amount is recorded in last 4 Bytes of this section’s head. Move reading pointer to the next section.
5. Repeat all the above from the beginning of the next section until the end-of-file is encountered.
2.5 Algorithm implementation
The C++ and C source codes of algorithm implementation are available at the website provided in Abstract of this paper.

3 EXPERIMENTS
Experiments are operated to test our algorithm. Codes for testing the algorithm are continually revising.(Liu et al., 2005)These tests are performed on a computer whose CPU is AMD Duron 750MHz and operating system is MagicLinux 1.2 (Linux Kernel 2.6.9) without swap partition. Testing programs are executed at multiuser text mode and compiled by gcc 3.3.2 with optimization level O3. The file system is ext3.Files are stored on a 4.3 GB Quantum Fireball hard disk with 5400 RPM. Table 1 compares compression ratio while table 2 compares running time.

Table 1. Comparison on compression ratio

| sequence     | size   | ours | DNA | Gzip    | bzip2 |
|--------------|--------|------|-----|---------|-------|
| atatsgs      | 9647   | 2.0068 | –   | 2.1702  | 2.15  |
| atefla23     | 6022   | 2.0113 | –   | 2.0379  | 2.15  |
| atrdna       | 10014  | 2.0068 | –   | 2.2784  | 2.15  |
| atrdna       | 5287   | 2.0125 | –   | 1.8846  | 1.96  |
| chmpnx       | 121024 | 2.0005 | 1.6716 | 2.2821  | 2.12  |
| chntx        | 155939 | 2.0004 | 1.6127 | 2.3349  | 2.18  |
| hechmcvg     | 229354 | 2.0003 | 1.8492 | 2.3278  | 2.17  |
| hsg6dpdgen   | 52173  | 2.0013 | –   | 2.2444  | 2.07  |
| humdystrop   | 38770  | 2.0018 | 1.9116 | 2.3633  | 2.18  |
| humghcsa     | 66495  | 2.0010 | 1.0272 | 2.0655  | 1.31  |
| humhdbcd     | 58864  | 2.0011 | 1.7951 | 2.2399  | 2.07  |
| humhnrpb     | 56737  | 2.0012 | 1.8165 | 2.2670  | 2.09  |
| mmzp3g       | 10833  | 2.0065 | –   | 2.3225  | 2.13  |
| mptmctg      | 186609 | 2.0004 | 1.8920 | 2.3291  | 2.17  |
| mtpacg       | 100314 | 2.0007 | –   | 2.2922  | 2.12  |
| vaccg        | 191737 | 2.0004 | 1.7580 | 2.2520  | 2.09  |
| xlfxf512     | 19338  | 2.0035 | –   | 1.8310  | 1.80  |
| chr10(rice)  | 22432531 | 2.0000 | –   | 2.4498  | 2.3033 |
| Average      | –      | 2.0031 | 1.7037 | 2.3224  | 2.0674 |

The compression ratio of other algorithms are cited from their original papers. As the compression ratio of newly-devised algorithms are similar,we take DNACompress as an example,"ours" refers to our algorithm.DNA stands for DNACompress. The unit of file size is bit rather than Byte.

4 DISCUSSION
The performance of a compression algorithm has two sides, the compression ratio and the running time. Many newly-devised DNA compression algorithms focus on compression ratio while ignore the running time. But the time occupation of obtaining a little lower compression ratio is very high. Many of them run 100 times slower than universal compression algorithm, according to Table 2 of Chen’s paper (Chen et al., 2002). Our algorithm runs many times faster than Gzip which is 100 times faster than newly-devised algorithms. Considering the compression ratio and running time both advance traditional compressors (Gzip and bzip2) considerably, our algorithm is a wise choice of replacing them. It is more useful in those fields which need fast running, such as database.

Table 2. Comparison on running time

| sequence     | Gzip(s) | encode(CLK) | decode(CLK) | encode(s) | decode(s) |
|--------------|---------|-------------|-------------|-----------|-----------|
| atatsgs      | 0.013   | <10000      | <10000      | <0.01     | <0.01     |
| atefla23     | 0.011   | <10000      | <10000      | <0.01     | <0.01     |
| atrdna       | 0.014   | <10000      | <10000      | <0.01     | <0.01     |
| atrdna       | 0.010   | <10000      | <10000      | <0.01     | <0.01     |
| chmpnx       | 0.105   | 10000       | 10000       | 0.01     | 0.01     |
| chntx        | 0.135   | 20000       | 20000       | 0.02     | 0.02     |
| hechmcvg     | 0.198   | 30000       | 30000       | 0.03     | 0.03     |
| hsg6dpdgen   | 0.044   | <10000      | <10000      | <0.01     | <0.01     |
| humdystrop   | 0.037   | <10000      | <10000      | <0.01     | <0.01     |
| humhdbcd     | 0.050   | <10000      | <10000      | <0.01     | <0.01     |
| humghcsa     | 0.055   | 10000       | 10000       | 0.01     | 0.01     |
| humhnrpb     | 0.049   | <10000      | <10000      | <0.01     | <0.01     |
| mmzp3g       | 0.014   | <10000      | <10000      | <0.01     | <0.01     |
| mptmctg      | 0.100   | 20000       | 30000       | 0.02     | 0.03     |
| mtpacg       | 0.088   | 10000       | 10000       | 0.01     | 0.01     |
| vaccg        | 0.164   | 30000       | 20000       | 0.03     | 0.02     |
| xlfxf512     | 0.018   | <10000      | <10000      | <0.01     | <0.01     |
| chr10(rice)  | 9.5     | 3460000     | 3510000     | 3.46     | 3.51     |

"Gzip" includes the total of time elapsed in both compression and decompression by Gzip. More experiments indicate that bzip2 takes more time to perform same operation. Following four fields list the time elapsed in compression and decompression respectively, "encode" means compression while "decode" means decompression. Each operation is evaluated in two units, CPU clock and second.

REFERENCES
Bao,S., Chen,S., Jing,Z.-Q., Ren,R. A DNA Sequence Compression Algorithm Based on LUT and LZ77(2005) [Online] http://arxiv.org/cs/0504100
Chen,X., Kwong,S. and Li,M. (2001) A compression algorithm for DNA sequences IEEE Engineering in Medicine and Biology 20, 61-66
Chen,X., Li,M., Ma,B. and Tromp,J.(2002) DNAcompress: fast and effective dna sequence compression, Bioinformatics, 18, 1696-1698
Cohen,J.(2005) Computer science and bioinformatics, COMMUNICATIONS OF THE ACM, 48, 72-78.
Grumbach,S. and Tahi,F. (1994) A new challenge for compression algorithms: Genetic sequences. In Proceedings of the Data Compression Conference, 340-350
Grumbach,S. and Tahi,F. (1994) A new challenge for compression algorithms: Genetic sequences. J. Inform.Process.Manage 30, 875-886
Korodi,G. and TABUS,I,(2005) An efficient normalized maximum likelihood algorithm for DNA sequence compression, ACM Transactions on Information Systems, 23, 3-34
Liu,J. and Bao,S. (2005) C and C++ implementation of algorithm and binary executable files for Linux [Online] http://grandlab.cer.net/topic.php?TopicID=50
Matsumoto,T.,Sadakane,K. and Imai,H. (2000) Biological sequence compression algorithms Genome Informatics Workshop Universal Academy Press, 43-52
Rivals, E., Delahaye, J.,Dauchet, M., AND Delgrange,O. (1995) A guaranteed compression scheme for repetitive DNA sequences. Tech. Rep. IT285, LIFL Lille I Univ.
Shannon, C.E. A mathematical theory of communication (1948) *The Bell System Technical Journal, 27*, 379-423, 623-656
Ziv, J. and Lempel, A. A Universal Algorithm for Sequential Data Compression (1977) *IEEE Transactions on Information Theory*, 23, 337-343