BIOINFORMATICS APPLICATIONS NOTE Vol. 28 no. 19 2012, pages 2527–2529
doi:10.1093/bioinformatics/bts467

Sequence analysis

Advance Access publication July 25, 2012

DELIMINATE—a fast and efficient method for loss-less compression of genomic sequences

Monzoorul Haque Mohammed, Anirban Dutta, Tungadri Bose, Sudha Chadaram and Sharmila S. Mande*
Bio-sciences R&D Division, TCS Innovation Labs, Tata Consultancy Services Limited, 1 Software Units Layout, Madhapur, Hyderabad 500081, Andhra Pradesh, India

Associate Editor: Michael Brudno

ABSTRACT

Summary: An unprecedented quantity of genome sequence data is currently being generated using next-generation sequencing platforms. This has necessitated the development of novel bioinformatics approaches and algorithms that not only facilitate a meaningful analysis of these data but also aid in efficient compression, storage, retrieval and transmission of huge volumes of the generated data. We present a novel compression algorithm (DELIMINATE) that can rapidly compress genomic sequence data in a loss-less fashion. Validation results indicate relatively higher compression efficiency of DELIMINATE when compared with popular general purpose compression algorithms, namely, gzip, bzip2 and lzma.

Availability and implementation: Linux, Windows and Mac implementations (both 32 and 64-bit) of DELIMINATE are freely available for download at: http://metagenomics.atc.tcs.com/compression/DELIMINATE.

Contact: sharmila@atc.tcs.com

Supplementary Information: Supplementary data are available at *Bioinformatics* online.

Received on December 23, 2011; revised on July 18, 2012; accepted on July 21, 2012

1 INTRODUCTION

The volume of sequence data being deposited in major public sequence repositories is witnessing an exponential rate of growth. This presents an important challenge with respect to developing efficient compression and storage methods. Addressing this challenge directly/indirectly impacts bandwidth and cost-related issues of data transmission/dissemination.

Major public sequence repositories store sequence data either in its raw format (as fastq files) or in a processed format (as single/multi-fasta files). This study pertains to compression of files containing nucleotide sequences in single/multi-fasta format. Such files contain information of sequences along with their corresponding headers. Four nucleotide bases, i.e. A, T/U, G and C, usually constitute the majority of text characters in the sequence portion of fasta files. Using a two-bit encoding for these four characters is obviously the simplest way of reducing the file/data size by a factor of four. Several studies (see review by Giancarlo *et al.*, 2009) have proposed various approaches to further reduce the bits/base ratio to less than 2. While a majority of these approaches work by tracing (and optimally encoding) repeat patterns in genomic data, others work by capturing differences between various sequences constituting a multi-fasta file. A few ‘reference-based’ strategies (Deorowicz *et al.*, 2011; Kuruppu *et al.*, 2011) have also been reported recently. In spite of the availability of several Specialized Genome Compression Algorithms (SGCAs), which achieve reasonably high compression gains, it is observed that the majority of public sequence repositories still usey General Purpose Compression Algorithms (GPCAs) such as gzip and bzip2 for compressing and storing sequence data. The likely reasons for this observation are the following: (1) unlike GPCAs, most SGCAs are not equipped to handle non-ATGC characters. This limits their capability to perform a loss-less compression/decompression. (2) In contrast to GPCAs, the compression efficiency of most SGCAs is generally observed to be achieved at the cost of huge time and memory requirements. (3) The utility of reference-based compression approaches is also subject to availability of closely related reference genomes. In summary, the above observations indicate the need for a compression algorithm (and a practical implementation of the same) that (1) achieves loss-less compression and decompression as rapidly as GPCAs, (2) has significantly better ‘compression efficiency’ than GPCAs, (3) has low memory requirements thus enabling it to handle files of any size and (4) is compatible with popular software platforms (Unix/Linux, Windows, etc.) and system architectures (32-bit or 64-bit). In this article, we present DELIMINATE (a combination of delta encoding and progressive elimination of nucleotide characters)—a novel method that implements the above mentioned features of an ideal compression algorithm.

2 METHODS

There are two variants of DELIMINATE. Both variants (referred to as DELIM-1 and DELIM-2) separately handle the header and sequence data (in a given fasta/multi-fasta file) and perform compression of sequence data in two phases. In the first phase (common to both variants), information of all non-ATGC characters and low complexity regions (represented by lower case characters) are recorded. The file is then stripped of the non-ATGC characters, and all characters are converted to uppercase. The resulting file (now containing four distinct characters) is processed in the second phase. In this phase, the positions of two bases having the highest frequencies of occurrence are delta encoded and these bases are subsequently eliminated from the sequence. The remaining two

*To whom correspondence should be addressed.

bases (having the least frequencies) are then represented with a binary code. Various files generated in this process are further compressed using the 7-Zip archiver to generate the final compressed file. Details of various steps (in both phases) adopted by both variants of DELIMINATE are described in Supplementary Material 1.

3 RESULTS

Four different datasets [FNA, FFN, eukaryotic and next-generation sequencing (NGS) dataset] were used for evaluating the compression efficiency of DELIMINATE. Details of these datasets are provided in Supplementary Material 2. Files constituting these four datasets were compressed using DELIMINATE (both variants), bzip2, gzip and lzma. All experiments were performed on a Linux workstation (32-bit) having a 2.33 Ghz dual core processor and 2GB RAM. The results obtained were compared in terms of percentage compression ratio (PCR) and the time taken for compression and decompression. PCR was calculated using the following formula.

$$\text{PCR} = (\text{size of compressed dataset}/\text{size of original dataset}) \times 100.$$

The results of both variants of DELIMINATE (in terms of PCR) when compared with various GPCAs are summarized in Table 1. Detailed results for individual files in all validation datasets (for all methods) are provided in Supplementary Tables 1–4. The results indicate that both variants of DELIMINATE achieve better compression ratios when compared with other algorithms. Except for the NGS dataset, all GPCAs fail to achieve a compression ratio lower than 2 bits/base (<25%) in most cases. In contrast, for all datasets, both variants of DELIMINATE generate compression ratios <25%, thus indicating significant compression gains. Files constituting the NGS dataset are observed to be highly compressible using GPCAs. This is expected since NGS datasets are typically characterized by high sequencing coverage, and consequently end up having extensively repeated sequence strings, making them highly amenable to compression. Interestingly, even for the NGS dataset, both variants of DELIMINATE are observed to significantly outperform GPCAs. The percentage improvement in compression ratio (PICR) of DELIM-2 (i.e. the better performing DELIMINATE variant) when compared with GPCAs was quantified using the following formula.

$$\begin{aligned}&\text{PICR}\\&=[1-(\text{PCR of DELIMINATE}/\text{PCR of compared algorithm})]\\&\quad\times 100.\end{aligned}$$

Values of PICR are summarized in Supplementary Table 5. The results in this table indicate that the compression ratios obtained using DELIM-2 are on an average 7–27% better when compared with that obtained using GPCAs. In some cases, the compression gains (in terms of PICR) obtained using DELIM-2

Table 1. Summary of results obtained using DELIMINATE and other compression methods indicating (1) size of compressed dataset (SCD) in MB, (2) PCR[a], (3) compression time (CT) in seconds and (4) decompression time (DT) in seconds

Validation dataset with size	Evaluation Parameter	DELIM-1	DELIM-2	DELIM-S[b]	bzip2	gzip (default)	gzip (-9 option)	lzma (default)	lzma with 2 GB memory allocated
FNA dataset (2679 Prokaryotic Genome files) 5067 MB	SCD	1221	1219	1232	1415	1521	1463	1335	1335
	PCR	24.10	24.07	24.31	27.92	30.01	28.87	26.36	26.36
	CT	1238	1134	1208	1097	1303	7639	5120	6043
	DT	1109	1085	1142	485	101	117	215	221
FFN dataset (2679 files containing gene sequences) 4835 MB	SCD	1111	1111	1120	1278	1455	1402	1224	1224
	PCR	22.98	22.97	23.16	26.43	30.09	29.00	25.33	25.33
	CT	1175	1063	1119	1038	1165	6030	4461	4946
	DT	1108	1035	1078	421	103	199	254	248
Eukaryotic dataset (25 Human Chromosome files) 2996 MB	SCD	649	629	631	770	836	802	686	681
	PCR	21.65	21.00	21.06	25.71	27.91	26.77	22.91	22.74
	CT	799	705	829	651	691	3781	5470	6406
	DT	477	451	487	253	94	91	86	97
NGS dataset (14 files containing genomic/metagenomic reads) 8836 MB	SCD	1124	1093	1124	1471	1838	1762	1201	1129
	PCR	12.72	12.37	12.72	16.65	20.80	19.94	13.59	12.77
	CT	2897	2566	2259	2823	2044	5945	9636	11518
	DT	1844	1783	1771	652	120	119	220	238
TOTAL 21735 MB	SCD	4105	4053	4107	4934	5650	5429	4447	4370
	PCR	20.36	20.10	20.31	24.18	27.20	26.78	22.05	21.80
	CT	6109	5468	5415	5609	5203	23395	24687	28913
	DT	4538	4354	4478	1811	418	526	775	804

[a]PCR = (size of compressed dataset/size of original dataset) × 100. [b]Discussed at the end of Section 3.

are observed to be as high as 40%. The performance efficiency of DELIM-2 was also compared with two specialized genome compression algorithms, namely, GenCompress (Chen *et al.*, 1999) and XM-Compress (Cao *et al.*, 2007). Details of the datasets used in this evaluation and a discussion of the results obtained are provided in Supplementary Material 3.

The results with respect to time indicate that the compression time of DELIM-2 is slightly higher when compared with default gzip and bzip2 algorithms (Table 1). However, it is observed that the compression speed of DELIM-2 is around two to seven times faster when compared with gzip with −9 option (i.e. the best compression mode) and lzma algorithms. Although the decompression speed of DELIM-2 is faster than its compression speed, it is not as fast as the decompression speed of GPCAs. It is to be noted that the values (of compression time) provided for both variants of DELIMINATE (Table 1) refer to the total time taken for compressing a fasta file, which includes the final 7-Zip archiving step. Furthermore, all values (of compression and decompression time) indicated in Table 1 correspond to the 'real' or wall-clock elapsed time (not 'user + system' time). Although measuring 'real' time confers an advantage to programs such as DELIMINATE (both variants) and 7-Zip (which can simultaneously utilize 2 or more available processing cores, unlike single-CPU compression tools such as gzip and bzip2), this practice was adopted given that most of the present day workstations possess two or more processing cores. In the present comparison, both variants of DELIMINATE (including the piped calls to 7-Zip) were run with two processing cores. On another note, lzma—the best performing GPCA, in terms of compression ratio—uses a low amount of memory during compression (default mode) compared with DELIMINATE. It may appear that providing higher amount of memory would enable lzma to achieve better compression ratio. However, results in Table 1 indicate that no significant compression gains (when compared with the default mode) could be achieved by lzma even when 2 GB of memory (maximum possible allocation on the benchmarking system) was allocated to it for compressing the validation datasets.

Overall, validation results suggest that both variants of DELIMINATE (especially DELIM-2) are able to achieve significant gains in compression ratio as well as in processing time by providing genome sequences represented in a unique (partially compressed) format as input to a general purpose compression algorithm. In order to verify as to what extent the splitting and transformation steps adopted by DELIM-2 contribute to these compression gains, two experiments were performed. In the first experiment, the F1 file obtained at the end of the first phase (see Supplementary Material 1) containing a single un-split stream of A, T, G and C was converted to binary format (2 bits for each nucleotide base) and was provided as input to 7-Zip. The results of this experiment, shown in Supplementary Table 6 (under the head DELIM-B), demonstrate that the compression ratio of DELIM-2 and DELIM-B are more or less comparable. In the second experiment, the bits corresponding to nucleotides at odd and even positions in the F1 file were split into two binary data streams which were then compressed in parallel using 7-Zip. The results obtained in this experiment (provided in Table 1 under the head DELIM-S) also indicate that DELIM-2 and DELIM-S obtain similar compression levels. However, a comparison of compression time of DELIM-2, DELIM-S and DELIM-B (Supplementary Table 6) suggests that splitting the data streams and processing them using parallel threads (as in DELIM-2 and DELIM-S) positively impacts the overall time required for compression. These results therefore imply that though the transformation steps adopted in DELIM-2 do not help it attain any significant gain in compression ratio (compared with what could be attained by providing a homogeneous stream of four standard nucleotides to 7-Zip/lzma), they contribute to a faster compression process.

4 CONCLUSION

All versions of DELIMINATE discussed in this study are observed to perform significantly better than GPCAs. The promising results obtained for typical genome sequence files imply the possibility of significant reduction of storage requirements for not only individual users/research laboratories but also major repositories harboring massive amounts of sequence data. Besides simple storage-related costs, these results also have direct implications with respect to the cost of sequence data transmission and dissemination. Interestingly, the compression gains of DELIM-2 with NGS datasets are drastically high (40% on an average when compared with gzip). Given that NGS data constitute a major fraction of sequence data in public sequence repositories, replacing gzip (the routinely used compression tool) with DELIM-2 would naturally result in massive savings of storage- and transmission-related costs.

Conflict of interest: None declared.

REFERENCES

Cao,M.D. *et al.* (2007) A simple statistical algorithm for biological sequence compression. In: *Proceedings of the IEEE Data Compression Conference (DCC)*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 43–52.

Chen *et al.* (1999) A compression algorithm for DNA sequences and its applications in genome comparison. *Genome Informatics Workshop on Genome Informatics*, **10**, 51–61.

Deorowicz,S. and Grabowski,S (2011) Robust relative compression of genomes with random access. *Bioinformatics*, **27**, 2979–2986.

Giancarlo,R. *et al.* (2009) Textual data compression in computational biology: a synopsis. *Bioinformatics*, **25**, 1575–1586.

Kuruppu,S. *et al.* (2011) Reference sequence construction for relative compression of genomes. In Grossi,R. *et al.* (ed.) *String Processing and Information Retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 420–425.