International Journal of Computational Bioinformatics and In Silico Modeling
Vol. 4, No. 1 (2015): 592-596
Research Article
Open Access

ISSN: 2320-0634

A New Referential Method for Compressing Genomes

M. Mary Shanthi Rani*

Department of Computer Science and Applications, Gandhigram Rural Institute – Deemed University, Gandhigram, Tamilnadu, India.

*Corresponding author: M.M. Shanthi Rani; e-mail: drmaryshanthi@gmail.com

Received: 18 November 2014 Accepted: 15 December 2014 Online: 01 January 2015

ABSTRACT

Recent advances in genomics have brought about significant reduction in cost and time for sequencing genomes which is vital for disease prediction, drug discovery and personalized medicine. Subsequently, there is tremendous increase in genomics research data where data storage plays a crucial role. Hence, there is a great need for efficient techniques for compressing genomes. A novel reference based genome compression method has been proposed in this paper for compressing DNA sequences based on referential compression.

Keywords: Compression, DNA Sequence, Index, Reference Genome.

1.0 INTRODUCTION

The cost of sequencing the first human genome was approximately three billion US dollars and it took roughly 12 years to complete the sequencing process. The rapid development of new high-throughput DNA sequencing techniques has drastically reduced the cost of DNA sequencing and time as well. The same genome can be sequenced in a week's time at a cost of 2000 USD using the second generation sequencing techniques and furthermore reducing the time from weeks to days and cost to few hundred dollars using third generation techniques. Eventually, this sweeping drop in cost and time has triggered a large number of sequencing projects and made them feasible [1-2], resulting in generation of huge voluminous data. This exponential increase in DNA sequences poses a great challenge for their effective storage which necessitates the use of large disk arrays. Recently, compute clusters are used for storage and analysis. However, data has first to be transmitted to the cloud before being analyzed, demanding heavy bandwidth which is another bottleneck in cloud based DNA analysis [3]. One of the solutions is compression of DNA sequences which has become vital for storage and transmission of DNA sequence data [4].

Compression is a process which stores the original data in less space exploiting redundancy in data. They can be lossless or lossy. Lossless compression methods account for exact recovery of original data whereas lossy compression methods reconstruct the original signal with some loss of information. In general, lossless compression methods are preferred for biomedical applications as every single data byte is crucial for data analysis. Many novel compression algorithms based on vector quantization have become very popular due to their low computation and memory complexity as well, and have gained significance recently in the fields of image and video compression [5, 6]. Despite their efficiency, these compression methods may not be efficient for compressing DNA sequences, as they do not take account of the intrinsic biological characteristics of DNA sequences. This necessitates the development of novel techniques exclusively for DNA sequence compression.

2.0 MATERIALS AND METHODS

2.1DNA Sequence Compression Methods

DNA molecules store the digital information which represents the genetic blueprint of living organisms. A DNA sequence contains a series of four symbols representing the four nucleotide bases Adenine (A),

Thymine (T), Guanine (G) and Cytosine(C). The natural and most straightforward way of storing one base requires 1 byte. Though DNA sequences are stored in the form of text, general text compression algorithms like GZIP are not successful in compressing them efficiently due to the presence of large number of repeated fragments [7]. This inherent feature of a DNA sequence has inspired the researchers to develop DNA specific specialized compression algorithms. Some of the general purpose compression techniques that can be applied for DNA sequence compression can be categorized as bit manipulation, Dictionary based, Statistical and Referential approaches.

Bit manipulation techniques achieve compression by packing two or more bases (symbols) in a single byte [8-9]. Generally, bit manipulation algorithms achieve a compression rate of 4:1.A compression algorithm that uses seven bit encoding for three consecutive symbols is proposed in [8]. In [9], a Java based GUI for sequence compression using run-length encoding was proposed.. A novel technique based on Fibonacci code called BioCompress was developed by Grumbach and Tahi[10].They exploited the feature of the presence of palindromes in DNA sequences for effective encoding which was further improved in BioCompress2 by using markov model for compressing non-repeat regions[11]. Another encoding algorithm CTW+LZ was proposed by Matsumoto and Imai [7] which applies hash and dynamic programming strategies for optimizing the search for palindromes and repeats in DNA sequences. These methods achieve high compression ratios but at the cost of computation complexity.

Statistical methods are based on deriving a probabilistic model of data for predicting the symbols and use arithmetic coding for encoding [12-14]. XM [14] is a popular statistical based compression algorithm which uses a panel of experts like Markov and Context Markov models to predict the next symbol. Dictionary based methods are substitution methods which compress sequences by replacing repeated substrings with short pointers to a dictionary. The most popular dictionary based algorithms are Lempel-Ziv algorithms LZ77 and LZ78 [15].

The recently emerged referential compression is gaining very popular due to the growing number of projects on resequencing of genomes[16].These projects analyze genomes belonging to the same species which means they handle highly similar genomes. This similarity is fully utilized by reference based algorithms by encoding only the differences between the input and a reference genome from the same species achieving even 500:1 compression ratio specifically human genomes [17-18]. Reference based algorithms generate a mapping between input data and reference by replacing substrings with reference to reference genome.

Brandon et. al [19] proposed a reference based algorithm in which only the differences between the input genome and the reference genome are stored. These differences are categorized as three types of single base differences due to inserts, deletes and replacements respectively. The authors have also analyzed the entropy coding methods for encoding the referential positions within the reference genome. Another referential compression algorithm has been proposed in [20] which identifies only SNPs (Single Nucleotide Polymorphisms) and multi-base INDELS between input and reference genome sequences. Wang and Zang [21] has presented a novel referential compression tool GRS which finds longest common subsequences between two strings using Unix diff command. A self-indexing approach RLZ based on LZ77 is proposed in [22]. A novel LZ77 based compression method was proposed in [23] which consider more than one reference sequence for encoding.

Motivated by the non-referential compression scheme XM, an expert based reference compression method GReEn has been recently proposed which uses a copy expert to find matching k-mers between input and reference genomes [24].A number of reference based compression methods have been proposed with different methodologies to achieve good compression rate.

2.2 Proposed Method

The proposed method is a lossless compression method based on referential compression which reconstructs the input genome without any loss of information. The problem in hand is compression of the input genome with respect to a reference genome that is available at both the encoder and the decoder. For convenience, let us denote the input genome as I and the reference genome as R. The encoding process can be defined as a mapping f (I , R) between the input genome sequence and the reference genome generating an index codebook. Each index vector of the index codebook contains two parameters: Reference block index and pos that represents the starting position of input block within the reference block. If the input block cannot find a perfect match in all the reference genome blocks, its index vector will be the raw string of input block.

Both the input and the reference genome sequences are given as a set of compressed FASTA-files, one for each chromosome. The general idea of the mapping algorithm is to divide the input genome and the reference genome sequences into fixed size blocks. The reason for using fixed-size blocks is twofold. First, it helps in effective and convenient handling of data. Second, it enhances the compression speed.

Considering the fact that there would be high similarity between two genomes of the same species, we assume the block size of the reference genome BS_R to be much higher than the block size of the input genome BS_I intending to get more number of matches by local search within a few reference blocks. In addition, we

use a sliding window of size BS_I over the reference block which slides just one character forward for each comparison of input block, thus trying to optimize and expedite the matching process within a reference block.

Let N denote the number of chromosomes in the input genome and $IS^1, IS^2 \ldots IS^i \ldots IS^N$ represent the string representation of N chromosomes respectively. Each IS^i is further divided into n substrings (blocks) of length BS_I. Let us denote the these n blocks as $IS_1^i, IS_2^i \ldots IS_n^i$ where i represents the i^{th} chromosome of the input genome. Similarly, let M be the number of chromosomes in the reference genome and $RS^1, RS^2, \ldots RS^j \ldots RS^M$ be their string representations respectively. Each RS^j is divided into m substrings (blocks) of length BS_R. Let us denote these m blocks as $RS_1^j, RS_2^j \ldots RS_m^j$. For convenience and for reducing the computation time, we compare input and reference blocks of the same chromosome. Let IS_k^i represents the k^{th} block(substring) of input chromosome IS^i, k $\in$ {1,2,…n}.

The encoding algorithm for mapping process is given as follows:

1. k=1
2. Repeat
3. Call compare(IS_k^i)
4. If compare (IS_k^i) returns true
5. Set index (IS_k^i)= (ref_index,pos)
6. Else
7. Set index(IS_k^i)= IS_k^i // Encode RAW
8. k=k+1
9. until k>n

Compare() finds a perfect match of a input block(IS) in reference block(RS) as follows:

Algorithm Compare(IS_k^i)

```
repeat
For each substring RS_r^i of reference chromosome RS^i , r ∈{1,2,…m}
   r=1;
   repeat
        p=1;
        repeat
             Compare the strings IS_k^i and RS_r^i (p,p+7) ;
             If perfect match found,
             Set pos=p;
             Set ref_index=r;
             Return pos and ref_index;
             else
             p=p+1;
        until p>64;
   r=r+1;
   until r>m;
   Return false;
```

Steps 1 – 9 are repeated for each chromosome i, i $\in$ {1,2,…N} in a given genome. The process of comparison of input block with the reference block substring has been visualized in Table 1.

Table 1. Comparison of Input block and Reference Block Substring

Chromosome	First Comparison	Second Comparison
Reference (*RS*) :	AATCGAGGTAAGT……..	AATCGAGGTAAGT……..
Input (*IS*)	*AATGTATT*	*AATGTATT*

3.0 RESULTS AND DISCUSSION

The performance of the proposed method is demonstrated using real genomic data from UCSC genome browser. In particular, HG18 release genome, Watson JW genome, Korean genomes KOREF20090131 and KOREF20090224 and the Han Chinese genome YH were used. The size of the compressed genome is used as the parameter for evaluating the performance of the proposed method.

The block size of the reference chromosome BS_R is set to 4 MB for human genomes and 2 MB for TAIR genomes and rice genomes so as to keep the size of the index book small. To expedite the matching process, the block size BS_I of the input block is set to 8 bytes. The size of each index vector in bits is equal to the sum of the number of bits required for storing the reference block index and the position within the reference block. Therefore,

Size of each index vector (I_s) = $\log_2 m + \log_2 BS_R$ where m is number of reference blocks.

The size (C_s) of each compressed chromosome = n× I_s Where n is the number of input blocks in each chromosome.

Therefore, the total size (T_s) of the compressed genome is given by

$T_s = \sum_{i=1}^{N} C_s$

where N is the total number of chromosomes in a genome.

The size (C_s) of the index book is further compressed by using delta encoding which efficiently compresses the differences in reference block indices of successive input blocks [25].

The increase in compression ratio is greatly influenced by three parameters: m, BS_R and BS_I respectively. A huge block size of the referential block will have greater probability of matching more number of input blocks within the same block. This will speed up the matching process and accounts for greater compression ratio if the index entries are encoded efficiently with reference to the previous index.

Furthermore, compression time decreases with the increase in block size but at the cost of increase in block index size. Thus, the proposed referential compression is an optimization problem where we have to choose an optimum block size for reference and input block as well to achieve a good balance between compression time and compression ratio.

Table 2 compares the size of the compressed files of the proposed algorithm with the popular Gzip algorithm for different choices of reference and input genomes.

Table 2: Comparison of Compressed File Sizes (in Megabytes) of Gzip and Proposed method

S. No.	Reference Genome	Input Genome	Size of Input Genome	Compressed Size using Gzip	Proposed Algorithm (T_s)
1	HG18	YH	2987	832	298
2.	HG18	JW	2991	834.8	299
3.	KO224	YH	2987	832	298.7
4.	KO131	HG18	2996	836.2	299.6
5.	TAIR8	TAIR9	113.63	34.1	11.36
6.	TIGR5.0	TIGR6.0	355.07	108.67	35.50

It has also been observed that proper choice and size of the reference genome plays a vital role in enhancing the performance of the compression algorithm. The proposed method outperforms Gzip algorithm by achieving a higher compression ratio of 10:1. Similar referential based algorithms like GReEn are able to achieve very high compression ratios like 100:1 but with increased computation complexity. The proposed method is relatively simple and the decompression process is simply replacing the substrings from the same reference genome available at the decoder using the index book.

4.0 CONCLUSION

In this paper, a novel referential compression algorithm has been proposed and the results are competitive with traditional Gzip algorithm as shown in Table 2. The compression speed depends on the block size of the reference genome which can be tuned depending on available main memory capacity. The compression ratio can be further improved if the block size of the input sequence is increased so as to have long matches which will be investigated in our future work thereby leading to compact storage and transmission of genome data in genomics research.

5.0 REFERENCES

1. Consortium IHGS: Initial sequencing and analysis of the human genome. Nature 2001, 409(6822),860-921.
2. Schadt EE, Turner S, Kasarskis A (2010), A window into third-generation sequencing. Human Mol Genet, 19(R2):R227-R240.
3. Trelles O, Prins P, Snir M, Jansen RC (2011),Big data, but are we ready? Nat Rev Genet, 12(3):224.
4. Nalbantoglu ÖU, Russell DJ, Sayood K(2010),Data compression concepts and algorithms and their applications to bioinformatics. Entropy, 12:34-52.
5. T. Lookbaugh, E. A. Riskin, P. A. Chou, and R. M. Gray(1993), "Variable rate vector quantization for speech, image and video compression," IEEE Trans. Commun., 4:186–199
6. K. Somasundaram and M. Mary Shanthi Rani (2009), "Adaptive Classified Pattern Matching Vector Quantization for compressing images", The 2009 International Conference on Image Processing, Computer Vision & Pattern Recognition Proceedings, Las Vegas, USA :pp.532-538.
7. Matsumoto T., Sadakane K., and Imai H. (2000), "Biological sequence compression algorithms," in Proc. GIW :pp. 43–52.
8. Gregory vey(2009), Differential direct coding: a compression algorithm for nucleotide sequence data, The journal of biological databases and curation,
9. Raja rajeswari P., Allam apparo and Kumar.Genit V.K. compress tool(gc)(2010):A Javabased tool to compress DNA sequences and compute compression ratio(its/ase) of genomes.CoRR,abs/1006:1193.
10. Grumbach S.and Tahi F.(1993), "Compression of DNA sequences," in Proc.DCC: pp. 340–350.
11. Grumbach S.and Tahi F.(1994), A new challenge for compression algorithms:Genetic sequences, Inform. Process. Manag., 30(6): 875–886.
12. Loewenstern D. and Yianilos P.N. (1999). Significantly lower entropy estimates for natural DNA sequences, J. Comput. Biol., 6(1): 125–142.
13. Allison L., Edgoose T., and Dix T.I.(1998), Compression of strings with approximate repeats, in Proc. 6th Int. Conf. ISMB : pp. 8–16.
14. Cao M.D., Dix T.I., Allison L., and Mears C.(2007), "A simple statistical algorithm for biological sequence compression," in Proc. DCC: pp. 43–52.
15. Ziv J. and Lempel A.(1977),A universal algorithm for sequential data compression, IEEE Trans. Information Theory, IT-24:337-343.
16. Thomas J. Hudson, Warwick Anderson, Axel Artez et al(2010), International network of cancer genome projects, Nature,464(7291):993-998.
17. Sebastian Wandelt and Ulf Leser(2012), Adaptive efficient compression of genomes, Algorithms for Molecular Biology:7(30).
18. Sebastian Deorowicz and Szymon Grabowski(2011), Compression of DNA sequence reads in fastq format, Bioinformatics, 27(6):860-862.
19. Marty C.randon, Douglas C.Wallace and Pierre Baldi(2009), Data Structures and compression algorithms for genomic sequence data, Bioinformatics, 25(14):1731-1738.
20. Christley S, Lu Y, Li C, Xie X (2009), Human genomes as email attachments, Bioinformatics, 25(2),274-275.
21. Congamo Wang and Dabing Zhang(2011), A novel compression tool for efficient storage of genome sequencing data, Nucleic Acids Research, 39(7):e45.
22. Kuruppu S, Puglisi SJ, Zobel J(2010): Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval. In Proceedings of the 17th international conference on String processing and information retrieval, SPIRE'10 ,Heidelberg: Springer-Verlag:pp.201-206.

23. Szymon Grabowski and Sebastian deorawicz(2011), Engineering relative compression of genomes, CoRR,abs/1103:2351.
24. Armando J.pinho, Diogo pratas and Sara P.Garcia(2011), GreEn: a tool for efficient compression of genome resequencing data, Nucleic Acids Research,2011.
25. Suel, Torsten, and Nasir Memon. "Algorithms for delta compression and remote file synchronization." (2002):1-24.

*****