A Novel Algorithm for DNA Sequence Compression

K. Punitha and A. Murugan

Abstract Deoxyribonucleic Acid (DNA) sequences vary in terms of their size and fall within the range of billions of nucleotides. The value increases to twice or thrice its original value annually. Techniques of data compression and related methods that originate from the information theory are frequently understood as relevant for the field of data communication, exploration, and storage. In the present situation, it is vital to store data for biological sequences. The compressBest algorithm proposed in the paper for the compression of DNA sequences helps attain a better compression ratio and is much faster when compared to the existing compression techniques. compressBest algorithm is applicable to compression of DNA sequences with a reduction in storage space. The proposed algorithm is tested over the data from the UCI repository.

Keywords DNA sequences · Data compression · Deoxyribonucleic acid · Dynamic programming

1 Introduction

DNA databases are found and large in size [1, 2], where storage of them is a major issue. They contain considerable complexity and feature logical organization. Therefore, the data structure for storing, accessing, and processing this data in an efficient manner is a challenging and problematic task [3, 4]. So, an efficient algorithm for compression is required for storing huge data masses. The standard methods for compression [5, 6] are not relevant when it comes to biological sequences [7] owing to the subtle regularities in sequences of DNA [8, 9]. Biological sequences cannot

K. Punitha (✉)
Department of Computer Science, Agurchand Manmull Jain College, Chennai, Tamil Nadu, India
e-mail: punithsathish@gmail.com

A. Murugan
Department of Computer Science, Dr. Ambedkar Govt. Arts College (Autonomous), Chennai, Tamil Nadu, India
e-mail: amurugan1972@gmail.com

N. R. Shetty et al. (eds.), *Emerging Research in Computing, Information, Communication and Applications*, Advances in Intelligent Systems and Computing 882,
https://doi.org/10.1007/978-981-13-5953-8_13

be compressed in a competent manner by standard algorithms used for compression. The compressBest algorithm for DNA is introduced which is based on the precise matching and gives the best results for comparison of the standard benchmarks in the context of DNA sequences.

The four nucleotide bases are Cytosine, Adenine, Thymine, and Guanine, depicted using their first character, which is C, A, T, and G. The meaning of compression is the reduction of data size by way of modifying it to assume a format which necessitates less number of bits than the ones required in the original format. Considerable effort has been expanded for applying compression techniques on textual data for the accomplishment of a number of tasks in computational biology, from storing and indexing large sets of data to comparing sequences databases of DNA.

This paper proposed compressBest algorithm which can be employed for the compression of DNA sequence data, which in turn results in a reduction of storage space through the use of Dynamic programming. The mechanism which is proposed for the compression of the DNA sequence through the use of the 2-bits encoding method includes division into segments, exact matching, and the method of decompression matching. When bases are distributed in a random manner in a sequence, 2-bit encoding method serves as an efficient method. However, nonrandomness exists in an organism's life and the DNA sequences which are evident in the living organism are nonrandom in nature. They also possess certain constraints. Our algorithm resembles all other compression algorithms of the DNA sequence and is based on creating partitions for a sequence into diverse segments, which include the repeat segments which are also the copied segments, and the non-repeat segments.

2 Related Work

In [5], a two-stage algorithm suggested the combination of the features of the Huffman coding algorithm and Lempel–Ziv–Welch (LZW) algorithm. LZW is an algorithm which is based on a dictionary. It is a lossless algorithm and is unified into the standard pertaining to the consultative committee on telephony and international telegraphy [10]. In this case, the code for every character is accessible in the dictionary [11] which uses reduced bits when compared to the ASCII code (less than 5 bits).

A novel approach is proposed in [12], the compression of genomes which has the ability to modify successions in the genome to double grouping which uses the Genbit algorithm and the development of a grid takes place from the arrangement which has been encoded. In case the encoding group length is not a perfect square, a few bits are attached to the arrangement to create a flawless square with respect to its length. When both sides record the same number of times, a '0' is picked and the bit is connected again at the end to make the length a flawless square for the development of a framework.

Many algorithms have been recently introduced for the purpose which frequently uses the recognition of lengthy and approximate repeats. Another algorithm is known

as the DNAPack, which use dynamic programming as its basis. When compared to the earlier programs, the DNA compression is marginally improved while its cost can be ignored. A new lossless comparison algorithm is presented which enables the vertical and horizontal compression of the data. It is based on the statistical and substitution methods [13]. The representation of hiding data through the use of compression of DNA sequences is attempted. In the initial step, data is converted to a DNA sequence and the DNA sequence is compressed later on. The techniques of four compressions are used in this case, where better compression is produced by one of the options, which depends on the DNA sequence. The decompression algorithm which is developed also enables the procurement of the original data [14].

A lossless compression algorithm which is seed-based is developed in the paper to compress the DNA sequence which utilizes the method of substitution. This method has a similarity to the LempelZiv scheme of compression. The repetition of structures is exploited in the method proposed here, which are intrinsic to DNA sequences. The process is accomplished through the creation of an offline dictionary containing repeats accompanied by mismatch details. When it is ensured that only promising mismatches are allowed [15], a compression ratio is achieved by the method that is better than or in line with the existing compression algorithms pertaining to lossless DNA compression [16].

A DNA compression algorithm is introduced [17], which originated on the basis of exact reverse matching which allows best results for compression for the benchmarks of standard DNA sequences. When the DNA sequence is enormously long, it is easy to search for exact reverses. The approximate reverses which are optimal for the compression to take place can be found by the algorithm, but the task takes considerable amount of time to complete. The time taken is a quadratic time search or greater than that [18, 19]. Compression with high speed and the best ratio can be achieved, but it is challenging to do so. The compression for proposed DNA sequences achieves a better ratio for compression and it is running faster when compared to the existing algorithms [20] for compression.

3 Proposed Work

This is a form of motivator for the advancement of compression tools with high performance whose design is targeted at genomic data. The DNA compression methods used earlier were either statistical or dictionary-based in nature. The 2-bit encoding methods in use recently have become noticeable where the bases with 4-nucleotides in the DNA sequence $\{A, G, C, T\}$ are assigned the values 00, 10, 01, and 11 respectively. The technique proposed here is used for the compression of large DNA sequence bytes which have average ratio for compression. The decompression time initially needs an input which is available at the time of sequence compression. This value is used along with the compression input strings where the

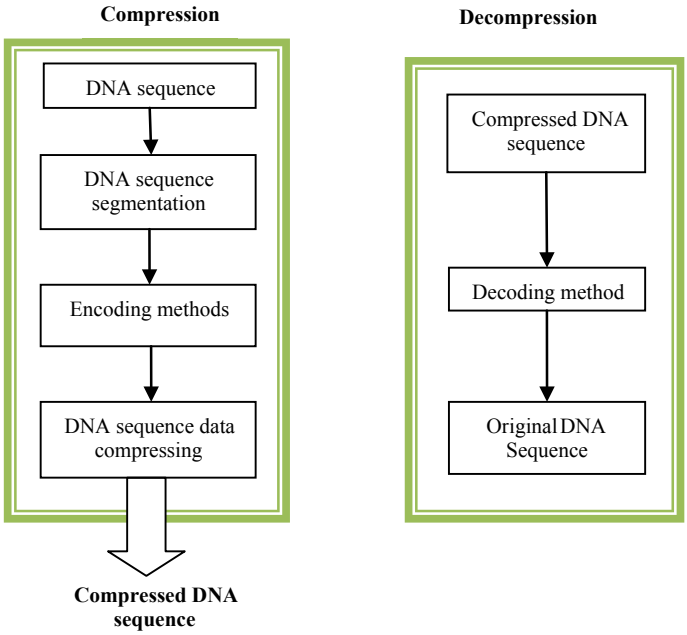

Fig. 1 Overall architecture of the proposed method

decompression takes place and the output is produced in the form of the original sequence. The decompressed input is obtained from the original sequence as per the segmented values. The DNA sequence algorithm for compression attains a compression ratio which is moderate and runs considerably faster when compared to present compression programs. The overall process of the proposed method is given in Fig. 1.

3.1 Segmented in Dynamic Programming Techniques

Different symbols are in use with respect to the occurrences of different DNA sequences which are chosen for the compression of the sequence. This means that the DNA sequence data is divided into a number of parts throughout the process. A technique used for solving the optimization of a sequence of decisions is the dynamic programming technique. The underlying rationale behind this is the representation of the problem by the way of a process which is an evolution from one state to another in reaction to specific decisions.

3.2 Encoding of Numbers

The proposed algorithm needs the encoding of diverse integers. Let, $A = 00$, $C = 01$, $G = 10$, $T = 11$ is the segmentation which is accompanied by a segmented number. The process initiates with the alignment of each sequence in the dataset with respect to the reference sequence through the use of the local sequence alignment. The objective behind sequence alignment is the placement of homogenous segments in the same column through the insertion of blank space. Further, aligning similar sequences can assist in the discovery of patterns and sequence relationships, which can improve the ratio for compression.

As an example, the segments are not fixed in length, and the encoding of their length is important. For a repeated segment, the values which are segmented with respect to the reference substring of the input are taken into consideration, and the segment is copied from here, after which encoding takes place. In the case of approximate copies, rather than exact ones, it is important to encode the value which has been segmented with respect to the modifications. Any of these numbers do not feature any bounds, and the integers must be encoded in a way which is self-delimited as opposed to encoding with respect to a fixed number of bits. The reference segmented values are encoded by encoding the relative difference with respect to the reference segmentation and it is preferred to make use of its copy.

The encoding method which involves 2-bits can make use of 2 bits for the encoding of every character which means that 00 is for A, 10 is for G, 11 is for T, and 01 is for C. Therefore, "gaccgtca" can be encoded by "10 00 01 01 10 11 01 00". This requires 16 bits in all. The exact matching method can make use of repeat length and repeat position for the representation of an exact repeat. In this way, three bits are used to encode an integer and two bits are used to encode a character, and a single bit is used to indicate whether the next part contains a pair, which also signifies an exact repeat or plain character.

3.3 Compression

In experimental research work which involves the text file with a dot txt file extension, a series of four successive base pairs are present, which are a, g, t, and c. This ends in a blank space as a terminal character. The basic element, in this case, is the text file which is used to consider compression as well as decompression. The output file is also a text file containing information about four base pairs which are unmatched and an ASCII character which has a coded value.

3.4 CompressBest Algorithm

Compression Algorithm

Input: DNA Sequence
Output: Compressed data

```
Input: Compressed text file
Output: original DNA Sequence

1:  Repeat the following steps 2 to 3 until the codes are converted into bases.
2: According to the segmentation number, replace all the codes by the corresponding bases.
3:  do
{
        Read the character one by one from the DNA sequence;
        If char = $ followed by the 2-bit based sequence base then
                Assign 3 times the base
        If char = # followed by the 2-bit based sequence base then
                Assign 4 times the base
    Else
                Read two characters and
                Assign A if char = 00;
                Assign C if char = 01;
                Assign G if char = 10;
                Assign T if char = 11;

} until char = NULL;
4:  The original DNA sequence is obtained.
5:  End
```

3.5 Decompression

The input string used for compression of a particular value is decompressed to produce the original file in the form of the output. The lookup table is created from the structure of the compressed file by finding repeat patterns present in the source. The size of the lookup table is extracted from bit blocks which represent the pattern. The blocks of the compressed pattern are extracted and the pattern type is recollected in the form of pattern ID followed by patterns. Decompression takes place from the beginning of the file to the end of the file for obtaining the original sequence of DNA.

Decompression Algorithm

Input: DNA Sequence
Output: Compressed data

1: Divide the sequence into segments with four nucleotides each in DNA sequences.
2: Find the matched segments in the DNA sequence and assign the numbers with symbol 'S' followed by corresponding numbers.
3: Repeat the following steps for each in the segment.
   (i). If there is three repeat bases, then assign '$' as a suffix and assign two bits based on DNA sequence values (A=00, C=01, G=10, T=11).
   (ii). If there is four repeat bases then assign '#' as a suffix and assign two bites based on DNA sequence values (A=00,C=01,G=10,T=11).
4: Repeat the steps from 2 to 3 until there are no repeat bases.
5: Assign two bits code for remaining bases.
6: End

The compression ratio can be calculated as follows:

$$\text{Compression ratio: } \frac{\text{compressed file size}}{\text{original file size}}$$

4 Results and Discussions

The program code of the compressBest algorithm is executed Java 1.7.0 JDK and tested on a dataset of DNA sequences which is from Genbank through UCI repository. Genbank is a popular public database where the human genomic data is available. The datasets include six DNA sequences such as two are HUMAN GENES [HUMDYSTROP, HUMHDABCD], two from chloroplast genomes (CHMPXX AND CHNTXX), and two are Viruses (HEHCMVCG AND VACCG).

The storage of the DNA sequence and its accuracy must be considered since even a single mutation in the base, deletion, or insertion would reflect in terms of a major change in its phenotype. For the purpose of accuracy, file compression is developed one after the other through the use of the compressBest algorithm.

The efficient result level in terms of the performance analysis of the compression ratio percentage of DNA sequence data is represented in Fig. 2 along with other techniques such as Lossless compression algorithm and BDNAS compression algorithm. The compressBest algorithm proposed here produces efficient output when compared to the other techniques in existence.

The compression of the original sequence size before compression is depicted in Table 1, and compression ratio after compression is also depicted in Table 1. The advantages of the compressBest algorithm are, it is a high-speed algorithm with minimized compression ratio, minimized execution time and it gives lossless compression data.

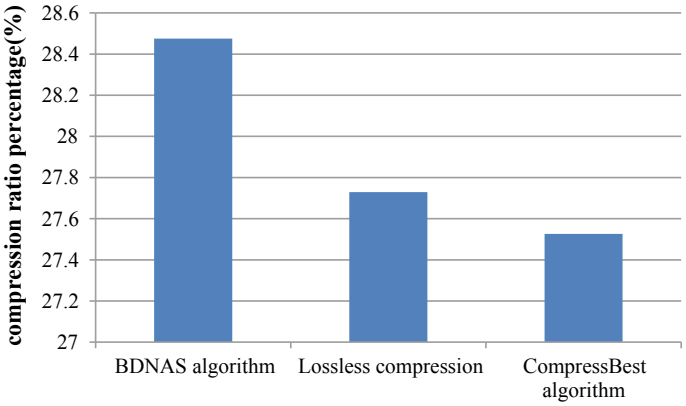

Fig. 2 Comparison of compression techniques

Table 1 Experimental analysis

S. No.	Type of DNA sequences	Original size of sequence before compression (Bytes)	Compression ratio percentage (%) after compression		
			BDNAS algorithm	Lossless compression	CompressBest algorithm
1	HUMHDABCD (Human Gene)	58,864	33.33	32.61	32.53
2	HUMDYSTROP (Human Gene)	105,265	33.33	32.62	32.59
3	CHMPXX (Chloroplast Genome)	121,024	4.18	4.02	3.66
4	CHNTXX (Chloroplast Genome)	155,844	33.33	32.42	32.35
5	HEHCMVCG (Virus Genome)	229,354	33.33	32.58	32.14
6	VACCG (Virus Genome)	47,912	33.33	32.08	31.86

5 Conclusion

The experimental result shows that CompressBest algorithm gives better compression ratio for mostly repeated sequences. It provides a reduction in file size without losses of clear data. The results from the simulator help to achieve a minimization in the time required for compression, high speed, efficiency, reduction in file size, and accuracy with respect to the original file. The UCI repository was accessed to collect the datasets and the Java environment is used for the development.

References

1. International nucleotide sequence database collaboration, http://www.insdc.org. (2013).
2. Karsch-Mizrachi, I., Nakamura, Y., & Cochrane, G. (2012). The international nucleotide sequence database collaboration. *Nucleic Acids Research, 40*(1), 33–37.
3. Deorowicz, S., & Grabowski, S. (2011). Robust relative compression of genomes with random access. *Bioinformatics, 27*(21), 2979–2986.
4. Brooksbank, C., Cameron, G., & Thornton, J. (2010). The European Bioinformatics Institute's data resources. *Nucleic Acids Research, 38,* 17–25.
5. Shumway, M., Cochrane, G., & Sugawara, H. (2010). Archiving next generation sequencing data. *Nucleic Acids Research, 38,* 870–871.
6. Kapushesky, M., Emam, I., & Holloway, E. (2010). Gene expression atlas at the European bioinformatics institute. *Nucleic Acids Research, 38*(1), 690–698.
7. Ahmed, A., Hisham, G., & Moustafa, G. (2010). EGEPT: Monitoring Middle East genomic data. In: *Proceedings of 5th Cairo International Biomedical Engineering Conference* (pp. 133–137). Egypt.
8. Korodi, G., Tabus, I., & Rissanen, J. (2007). DNA sequence compression based on the normalized maximum likelihood model. *Signal Processing Magazine, IEEE, 24*(1), 47–53.
9. Deepak, H. (2013). State of the art: DNA compression algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering*, 397–400.
10. Kaur, S., & Verma, V. S. (2012). Design and Implementation of Lzw data compression algorithm. *International Journal of Information Sciences and Techniques (IJIST), 2*(4). https://doi.org/10.5121/ijist.2012.240771.
11. Shravan Kulkarni, S., & Kini, Y. (2017). Pre equal architecture for lossless data compression and decompression using hybrid algorithm. *International Journal of Advance Electrical and Electronics Engineering (IJAEEE), 6*(1_2). ISSN (Print): 2278-8948
12. Bhukya, R., Viswanath, B. V., Mahendra Kumar, D., Swathi Kiran, D. S., & Bagdia, P. (2017). DNA sequence decompression using bitmap matrix & wavelet transformation in image processing. Received: 20th April 2017, Accepted: 28th April 2017, Published: 1st May 2017, Copyright © 2016 Helix ISSN 2319– 5592 (Online)- Helix Vol. 8: 1491–1497.
13. Mishra, K. N., & Aaggarwal, A. (2010). An efficient horizontal and vertical method for online DNA sequence compression. *International Journal of Computer Applications, 3*(1), 0975–8887.
14. Bandyopadhya, S. K., & Chakraborty, S. (2011). Data hiding using DNA sequence compression. *Journal of Global Research in Computer Science, 2* (1), Vol. 27–33. © JGRCS 2010, All Rights Reserved.
15. Cover, T. M., & Thomas, J. A. (2012). *Elements of information theory*. New York: Wiley.
16. Eric, P. V., Gopalakrishnan, G., & Karunakaran, M. (2015). An optimal seed based compression algorithm for DNA sequences. Correspondence should be addressed to Pamela Vinitha Eric; pamela.vinitha@gmail.com Received 28 November 2015; Revised 9 May 2016; Accepted 19 June 2016.
17. Mukherjee, R., Mandal, S., & Mandal, B. (2016). Reverse sequencing based genome sequence using lossless compression algorithm. *International Research Journal of Engineering and Technology (IRJET), 3(5)*. e-ISSN: 2395-0056. www.irjet.net p-ISSN: 2395-0072.
18. Jacob, G., & Murugan, A. (2013). DNA based cryptography an overview & analysis. *International Journal of emerging science, 3*(1), 36–42.
19. Chouhan, D. S., & Mahajan, R. P. An architectural framework for encryption and generation of digital signature using DNA cryptography. In *International Conference on Computing for Sustainable Global Development (INDIACom)*.
20. Jahaan, A., Ravi, T. N., & Panneer Arokiaraj, S. (2017). Bit DNA squeezer (BDNAS): A unique techniques for data compression. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology ©,* 2 (4). ISSN: 2456-3307.