DNA Compression Algorithm Using Pattern Hunter

Rexline S J
Department of Computer Science
Loyola College
Chennai,India
rexlinegerard@gmail.com

Trujilla Lobo F
Department of Computer Science
Loyola College
Chennai,India
truji80@gmail.com

Abstract— Modern biological science produces vast amount of genomic progression data. Genome contains the hereditary information of creatures. Huge amount of DNA sequences are stored in DNA databases like GenBank. In consequence, decreasing the Data storage costs for DNA sequences has become an essential. Standard general purpose compression algorithms are unsuccessful to get a good compression ratio. This paper presents a Pattern Recognition based DNA Sequence Compression algorithm which compresses the DNA sequences with 80% of reduction in its storage space.

Keywords - DNA Sequence Compression, deoxyribonucleic acid, Pattern Hunter.

I. INTRODUCTION

The genetic activity of every living organism is organized by billions of individual cells [1]. The control-center of each cell is the deoxyribonucleic acid (DNA) that contains a complete set of instructions needed to direct the functioning of every cell. The substance of the DNA is the same for all living organisms. The DNA of all organisms has four components in common. They are the four nucleotide bases; namely Adenine, Cytosine, Guanine, and Thymine. They are represented using the first character of their names; namely A, C, G and T respectively [2, 3]. There is another unknown base element represented by the letter N. Therefore the DNA sequence is represented as a set of {A, C, G, T, and N}. The first four elements are represented as a double helix with A & T in one helix and C & G in another helix. The element N still remains unknown and is yet to have pictorial representations but participates in the functionalities of a DNA. The use of DNA in genetic engineering, forensics, bioinformatics, and DNA nanotechnology and anthropology applications has been extensive.

The DNA Database called GenBank is created and maintained by the National Center for Biotechnology Information (NCBI).The other two repositories maintain similar data are European Molecular Biology Laboratory (EMBL) and DNA Database of Japan (DDJB). GenBank keeps on growing at an exponential rate, it occupies large space and so it is necessary to compress and store the DNA sequence data. The volume of data creates severe storage and data communication problems. Thus, reduction of the DNA sequence storage costs has become a necessity.

The paper is organized as follows: Section II presents the existing DNA compression algorithms; Section III proposes our new approach; Section IV substantiates the achievability and competence of the proposed method and finally Section V contains the conclusions.

II. EXISTING DNA COMPRESSION METHODS

The standard general purpose compression algorithm such as “gzip”, “bzip2”, “winzip” not succeeded to compress the DNA genome file with better compression ratio. Most of them attained better compression ratio for text files only but not for DNA sequences. The compression algorithms specifically designed for DNA sequences not achieved average compression rate below 1.7 bits/base. Algorithms such as Ziv-Lempel compression algorithms [10, 11] Biocompress [12], Gencompress [13] and DNAcompress [14] compress the DNA sequences having about 1.74 bits per base on average. Grumbach and Tahi [6], [7] proposed compression algorithms called Biocompress and Biocompress2 with the idea of Ziv and Lempel data compression method. BioCompress -1 and BioCompress -2[7, 16] are compression algorithms used a window of size of the sequence to detect palindromes and factors of arbitrarily long and far from each other. They encode the factor by the pair (l, p) where l is the length of the factor and p is the first occurrence’s position. Two bit encoding is used, if the size of the code word is greater than the factor. Substitutional and statistical methods lead this algorithm to the highest compression of DNA.

CFACT was proposed by E.Rivals et al. [8] which was not only detects repeats in a text but also selected some of them according to their compressibility. Cfact is a two pass algorithm. The first pass is to find repeated segment in a sequence and the second pass is to measure their quantitative importance by the compression rate. It looks for the longest exact Matching repeat using Suffix tree structure in the sequence. The idea of Cfact is basically the same as Biocompress-2. GenCompress was proposed by Chen et al[9][10] the performance of which is based on reference sequence selection as approximate matching with edit operations uses this reference sequence for compression. It takes use of both approximate repeats and repeat complements, and encodes it with length, position and the errors. In CTW-LZ, the compression ratio is attained by the Context-Tree Weighting (CTW) algorithm which achieves an excellent compression ratio but takes time to encode the sequences.GenomeCompress [11,12,13] compresses both repetitive and non repetitive sequences by taking less execution time and memory. DNACompress [14,15] is a two phase algorithm designed by Chen et. al [11] and uses special software tool PatternHunter for finding the repeats. PatternHunter finds complementary palindromes and approximate repeats with highest score in the first phase and encodes them in second phase. Hence DNACompress involves less searching time. It checks that each repeats to see whether it saves bits to encoding, if not it will be discarded. At the end all the non-repeats are concatenated together and encoded. Behshad Behzadi et al. [12] uses Dynamic Programming techniques to identify the repeating sequences called DNAPack. Raja Rajeswari et al. [16] proposes GenBit Compress algorithm that compresses repetitive and non-repetitive sequences using the ideas of extended binary tree.

III. PROPOSED METHOD

DNA sequences consists of only four nucleotides bases {A, C, G, T}, and therefore two bits are enough to store each base. To reduce the compression ratio of DNA sequences below 1.7 bits per base is a very challenging task [5]. Standard DNA compression algorithms to reduce the storage space of the DNA sequences can be developed using the special characteristics of the DNA sequences. The special characteristics of the DNA sequences are given as follows.

- One of these characteristics is the appearance of “complements”. In DNA, A and T are complements of each other; G and C are also complements of each other. The complement of the DNA sequence GGGAAACGT is CCCTTTGCA.
- Another characteristic is the appearance of “reverse complements”. In DNA, A and T are complements of each other; G and C are also complements of each other. The complement of the DNA sequence GGGAAACGT is CCCTTTGCA. If we then reverse CCCTTTCGA, the reversed complement obtained is ACGTTTCCC. ACGTTTCCC is defined as being the reverse complement of GGGAAACGT.
- Some other characteristics of the DNA sequences are i) They contain many tandem repeats ii) Many of the strings are palindromes iii) Some of them are reverse palindromes and iv) Nucleotide may appear only nine consecutive times.

All these facts conclude that better compression for DNA sequences is possible.

The proposed algorithm is based on the binary representation of nucleotides. The algorithm that compresses the DNA sequences is of Two Pass. In the first Pass, it finds the repeats, palindromes, complements and reverse complements and generates the PatternCode Table for the Pattern size of 3 to 9 with PatternCode. It is enough to generate the Pattern Code Table up to 9 bytes because of the characteristics of the nucleotide that they may appear only nine consecutive times in the DNA sequences. The source file is encoded using the PatternCode during the second pass.

- In order to achieve a better compression ratio, the compressor finds the longest repeating patterns and their reverse, complements and reverse complements of those patterns. The PatternCode table is generated by taking the advantage of the characteristic of DNA sequences. For example, the PatternCode table would store the pattern of the DNA sequence AAACGT. The reverse pattern of the DNA sequence AAACGT would be TGCAAA. The complement of the DNA sequence AAACGT would be TTTGCA. The reverse complement of AAACGT is ACGTTT. These reverse, complement and reverse complement patterns are not necessary to store in the PatternCode table.
- The PatternCode table would store only the repeated pattern and palindromes not reverse pattern, complement and reverse complement of the pattern. Presumably, there is a need of two bits indicating that it is a regular pattern [00], reverse pattern [01], complement [10] and reverse complement [11]. It reduces the space required for the PatternCode table. The Pattern Code Tables are generated with a unique Pattern code for the identified repeating patterns of varying sizes are shown in Table I.

TABLE I. PATTERNCODE TABLE

Pattern ID	Pattern	Pattern Code & Type =00	Reverse Pattern	Pattern Code & Type =01	Complement Pattern	Pattern Code & Type =10	Reverse Complement Pattern	Pattern Code & Type =11
Patterns of Size 3 Bytes – 9 Bytes (Pattern Code 000-110)								
P0	AAA --- AAAAAAAAA	000X0	Nil	Nil	TTT --- TTTTTTTTT	Nil	Nil	Nil
P1	CCC --- CCCCCCCCC	000X1	Nil	Nil	GGG --- GGGGGGGGG	Nil	Nil	Nil
P2	3 Bytes – 9 Bytes Sample Code AAACGT ACCATAG ATCGAATG	000X2	Reverse of 3 Bytes – 9 Bytes Sample Code TGCAAA GATACCA GTAAGCTA	010X2	Complement of 3 Bytes – 9 Bytes Sample Code TTTGCA TGGTATC TAGCTTAC	100X2	Reverse Complement of 3 Bytes – 9 Bytes Sample Code ACGTTT CTATGGT CATTCGAT	110X2
P3		000X3		010X3		100X3		110X3
P4		000X4		010X4		100X4		110X4
P5		000X5		010X5		100X5		110X5
P7		000X6		010X6		100X6		110X6
P7		000X7		010X7		100X7		110X7
P8		000X8		010X8		100X8		110X8
P9		000X9		010X9		100X9		110X9
P10		000XA		010XA		100XA		110XA
P11		000XB		010XB		100XB		110XB
P12		000XC		010XC		100XC		110XC
P13		000XD		010XD		100XD		110XD
P14		000XE		010XE		100XE		110XE
P15		000XF		010XF		100XF		110XF

- Let there be a finite sequence made up of A, G, T, C which can have many repeats. Included only those repeats into the Pattern Code table that provide maximum amount of compression in the encoding pass. In such a way that the number of bits required representing the Pattern ID is determined by satisfying the condition that for any 'n' number of bases maximum of 2^n possible combinations of pattern are used. Otherwise, two bits per base encoding will be used.
- The compressed file consists of three different regions: Header, Compressed file and the length of the file header. The Header contains all the information that must be known to decode the compressed code regions.

  The compressed file is structured as:
- Blocks of bits representing the content of the PatternCode Table [00-06] of exact 3 bytes to 9 bytes pattern that provide maximum amount of compression in the encoding pass.
- Blocks of bits representing the Pattern Code [000-111].
- Blocks of bit representing the pattern code of identified patterns with the pattern type [00-11] in a contiguous form to indicate the repeated patterns, reversed pattern, complement pattern and reversed complement pattern.
- Blocks of bit patterns representing compressed data in a contiguous form.

The following is the pseudo code incorporating all the above ideas:

- Procedure to compress the DNA sequence.
- Read the Sequence repeatedly and create the Pattern CodeTables for the Pattern of size 3 to 9 with pattern IDs and Pattern Codes.
- Identify the number of bits required to represent pattern IDs and Pattern Codes.

- Hexadecimal numbers [0x1-0xF] are enough to represent pattern ID based on the number of bases [A, C, G, and T]. The number of bits required to represent a Pattern ID is determined by satisfying the condition that for any 'n' number of bases possible cases of pattern formations are maximum of 2^n.
- Octal numbers [00-06] are enough to represent Pattern Code for 7 Pattern Tables of pattern size Exact 3 Repeat Bases to 9 Repeat Bases, since the DNA sequences are in the form of only four nucleotides bases {A,C,G,T} with a constraint that a nucleotide may appear only nine consecutive times.
- i) Using the Pattern Code Tables generated in step 1, encode the compressed file using the following procedure.
- Nine bits are used to represent a pattern size of Exact 3 Repeat Bases to 9 Repeat Bases; 3 bits to represent the pattern code and 4 bits to represent pattern ID and 2 bits to represent pattern type.
- Octal numbers [00-06] are enough to represent pattern code for 7 pattern Code tables.
- Hexadecimal numbers [0x1-0xF] are enough to represent pattern ID based on the number of bases [A, C, G, and T].
- Two bits are used to represent the Pattern type: 00-Regular Pattern, 01- Reverse Pattern, 10-Complement Pattern and 11- Reverse Complement Pattern.
- ii) If there are individual bases (non repeat regions) and for Repeat Bases but not in the Pattern Code Table, the corresponding code gets transformed. Assigned code for bases is: A="00", G="01", C="10", T="11" and indicated by 3 bit (07) Pattern Code. Totally 5 bits are enough to represent individual base.
- Repeat the step3 till the end of the file is encountered.
- Modify the encoded file with the required field like the presented patterns to retrieve the DNA sequences.

Procedure to decompress the DNA sequence:

- Create the Patterns Code Table by locating the repeated patterns present in the source file from the compressed file structure.
- Extract the size of the Pattern Code from the blocks of bit representing the pattern Code of identified patterns in a contiguous form.
- Calculate the size of the blocks to be read based on the value of the Patterns Code.
- Extract the blocks of compressed Patterns and recollect the Pattern Type, pattern ID and then the Patterns.
- Decode it from the beginning to the end of the file to get the original DNA sequence.

IV. EXPERIMENTAL RESULTS

In order to illustrate our algorithm's approach, the following sequence is used as an example:

AAACGT ACCATAG ATCGAATG TGCAAA GATACCA GTAAGCTA TTTGCA TGGTATC TAGCTTAC ACGTTT CTATGGT CATTCGAT AAACGT ACCATAG ATCGAATG TGCAAA GATACCA GTAAGCTA TTTGCA TGGTATC TAGCTTAC ACGTTT CTATGGT CATTCGAT

TABLE II. PATTERN CODE TABLE FOR THE GIVEN DNA SEQUENCE

PID	Pattern	Pattern Code & Type =00	Reverse Pattern	Pattern Code & Type =01	Complement Pattern	Pattern Code & Type =10	Reverse Complement Pattern	Pattern Code & Type =11
	Patterns of Size 6 Bytes (Pattern Code 011)							
P0	AAACGT	000000	TGCAAA	010000	TTTGCA	100000	ACGTTT	110000
	Patterns of Size 7 Bytes (Pattern Code 100)							
P0	ACCATAG	000000	GATACCA	010000	TGGTATC	100000	CTATGGT	110000
	Patterns of Size 8 Bytes (Pattern Code 101)							
P0	ATCGAATG	000000	GTAAGCTA	010000	TAGCTTAC	100000	CATTCGAT	110000

Encoded string:

011000000 100000000 101000000 011010000 100010000 101010000 011100000 100100000
101100000 011110000 100110000 101110000 011000000 100000000 101000000 011010000
100010000 101010000 011100000 100100000 101100000 011110000 100110000 101110000

Total number of encoded binary bits = 24 x 9 = 216 bits

Total number of bits required to store the repeated Patterns = 6 x 2 +7 x 2+ 8 x2 = 42 bits

Total bits required to store the above said bases = Total number of encoded binary bits + Total number of bits required to store the repeated Patterns = 216 +42 = 258 bits= 32.25 bytes.

According to 2 bits per symbol technique, the total bits required to store the above said bases = 168 x 2 = 336 bits = 42 bytes.

In general, the total bits required to store the above said bases = 168 x 8 = 1344 bits.

From the above said example, the 168 bytes of data is compressed as 32.25 bytes; which gives approximately 80.8% reduction in storage space of DNA files.

To experiment the efficiency of the proposed algorithm, standard set of DNA sequences are compressed and the results are compared with the compression ratio of other efficient DNA compressors. The standard algorithms used for comparison are BioCompress2, Genome Compress, CTW, DNA Compress, DNAPACK, DNABIT and the general purpose compression software WinRAR. The sequence files can be downloaded from the GENBANK database: http://www.ncbi.nlm.nih.gov:80/entrez/query.fcgi. The DNA sequence files are made available in FASTA file format in DNA databases which can also retrieved by any text processor. Efficiency of the proposed method is measured in terms of bits per base (BPB). The results are shown in Figure I have proved that the proposed algorithm performs better than other DNA Compression algorithm.

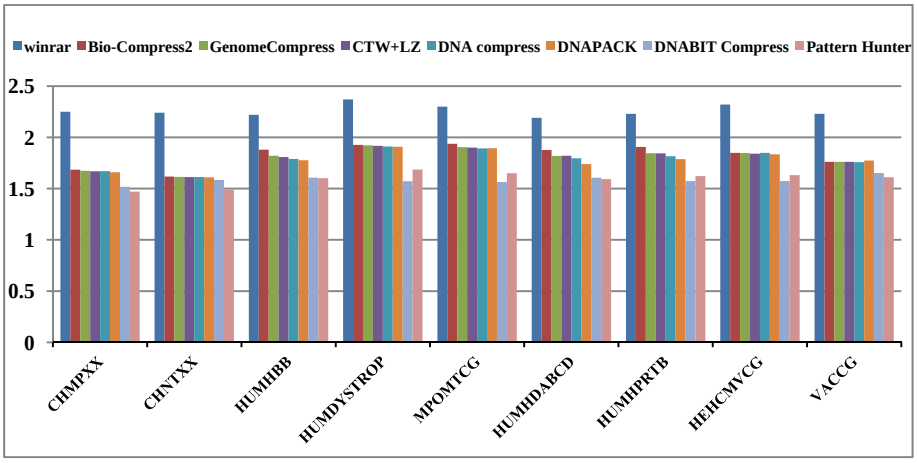

FIGURE 1: EFFICIENCY COMPARISON OF PATTERN HUNTER WITH OTHER DNA COMPRESSORS

V. CONCLUSION

A simple DNA compression algorithm which gives better compression is proposed to compress DNA sequences which are repetitive as well as non repetitive in nature. The simplicity and flexibility of DNA Compress algorithm could make it an invaluable tool for DNA compression in clinical research.

REFERENCES

[1] Calladine, C. R. and Horace A. Drew. Understanding DNA: The Molecule and How It Works. San Diego: Academic Press, 1997.

[2] Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, "Cross chromosomal similarity for DNA sequence compression",Bioinformation 2(9), pp. 412-416, 2008.

[3] Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, "Analysis of cross sequence similarities for DNA multiple sequence compression", International journal of Computer Aided Engineering and Technology, 2009.

[4] Manzini, G. and Rastero, M., "A simple and fast DNA Compressor, Software: Practice and Experience", MIUR support projects(ALINWEB), Vol. 34(14), pp.1397-1411, 2004.

[5] X Chen et al. A compression algorithm for DNA sequences and itsapplications in Genome comparison. In Proceedings of the FourthAnnual International Conference on Computational Molecular Biology, Tokyo, Japan, April 8-11, 2000.

[6] Grumbach, S. and Tahi, F., "Compression of DNA Sequences", In Proc. IEEE Symp. On Data Compression, pp. 340-350, 1993.

[7] Grumbach, S. and Tahi, F., "A new challenge for compression algorithms: Genetic Sequences", Journal of Information Processing &Management, Vol. 30, pp. 875-886, 1994.

[8] Rivals,E., Jean Paul Delahaye, M., Dauchet and Delgrange,O.,"A Guaranteed Compression Scheme for Repetitive DNA Sequences", In Proc. Data Compression Conf. (DCC-96), Snowbird, UT. p453, 1996.

[9] Chen, X., Kwong, S. and Li, M., "A compression algorithm for DNA sequences and its applications in genome comparison", The 10thworkshop on Genome Informatics (GIW-99), pp.51–61, Tokyo, Japan, 1999.

[10] Textual data compression in computational biology: a synopsis Raffaele Giancarlo∗, Davide Scaturro and Filippo Utro, Vol. 25 no. 13 , pages 1575–1586 ,2009.

[11] U. Ghoshdastider et al., "GenomeCompress: A Novel Algorithm for DNA Compression", ISSN 0973-6824,2005.

[12] Xin Chen, Sam Kwong and Ming Li, "A Compression Algorithm for DNA Sequences and Its Applications in Genome Comparison," Genome Informatics, Vol. 10, pp. 51-61, 1999

[13] Matsumoto,T. et al. Biological sequence compression algorithms. Genome Inform. 11, 43–52,2000

[14] Xin Chen, et al.," DNA Compress: fast and effective DNA sequence Compression" Bioinformatics Applications Note, Vol. 18 no. 12, Pages 1696–1698,2002.

[15] Chen, X., Li, M., Ma, B. and Tromp, J., "DNACompress: Fast and effective DNA sequence compression", Bioinformatics, Vol. 18(12), pp. 1696–1698, 2002.

[16] Raja Rajeswari and Dr.AlamApparao, "GenBit Compress-Algorithm for repetitive and non repetitive DNA sequences", Journal of theoretical and applied information technology, pp. 25-29, 2010.

[17] Raja Rajeswari and Dr.AlamApparao," DNABIT Compress – Genome compression algorithm", Bioinformation Volume 5 ,Issue 8, January 22, 2011.

[18] Soliman, T., "A Lossless Compression Algorithm for DNA sequences", International Journal of Bioinformatics and Applications, Vol. 5(6), pp. 593, 2009.

[19] Kamnath Mishra, Dr.Anupam Agarwal, Dr.EdriesAbdelhadi and Dr. Prakash C. Srivasatava, "An Efficient Horizontal and Vertical Method for Online DNA Sequence Compression", IJCA, Vol. 3(1), pp.39-46, June, 2010.

[20] Behzadi, B. and Le Fessant, F., "DNA Compression Challenge Revisited", Symposium on Combinatorial Pattern Matching (CPM2005), pp.190-200, June 2005.