DNA COMPRESSION ALGORITHM USING LENGTH LIMITED PREFIX CODE TREE

Rexline S J

Department of Computer Science, Loyola College, Chennai, India

Albert William Manuel

Department of Mathematics, Loyola College, Chennai, India

Abstract: Modern genetic science generates enormous amounts of genomic sequences. Genomic Sequence contains the genetic instructions of living things. Deoxyribonucleic acid (DNA) is a molecule that transmits the genetic instructions. Since DNA stores the genetic instructions of living things, a large amount of DNA sequences are stored in DNA databases for research purposes. In consequence, reducing the Data storage costs for DNA sequences has become an essential research in the field of lossless compression. Standard general purpose compression algorithms are not attaining good compression ratio for DNA sequences. This paper presents special Compression algorithm for DNA Sequences which uses Pattern recognition technique. The repetitive patterns recognized in the genomic sequence file is replaced with the shorter codes that are generated with the technique based on Length Limited Prefix Code Tree that compresses the DNA sequences with 80% to 82% of the reduction in its storage space.

Keywords: DNA Sequence Compression, Deoxyribonucleic Acid, Pattern Hunter, Prefix Code Tree.

Introduction: The genetic activity of every living organism is organized by billions of individual cells. The control-center of each cell is the deoxyribonucleic acid (DNA) that contains a complete set of instructions needed to direct the functioning of each and every one of the cells [4]. The substance of the DNA is the same for all living organisms. The DNA of all organisms has four components in common. DNA sequence consists of four nucleotide bases; namely Adenine, Cytosine, Guanine, and Thymine. They are symbolized using the first character of their names; namely A, C, G and T respectively [1]. There is another unknown base element symbolized by the letter N. Therefore the DNA sequence is symbolized as a set of {A, C, G, T, and N}. The first four elements are symbolized as a double helix with A & T in one helix and C & G in another helix. The element N still remains unidentified and is yet to have graphic illustration but participates in the functionalities of a DNA sequence [5, 6]. The application of DNA sequences in the field of genetic engineering, forensics, bioinformatics, and DNA nanotechnology and anthropology applications [11] has been extensive.

The DNA Database called GenBank [2] is created and maintained by the National Center for Biotechnology Information (NCBI). Similar data is maintained by the other two repositories are European Molecular Biology Laboratory (EMBL) and DNA Database of Japan (DDJB). GenBank data base contents are increasing at an exponential rate day by day [7] and require an enormous amount of storage space and so it is necessary to compress and store the DNA sequence data [21]. The heavy quantity of data generates severe storage and data communications problems [13]. Thus, enhances the necessity of the research of DNA compression algorithm to attain better compression ratio for the DNA data set.

This paper is structured as follows: Section II presents the existing DNA compression algorithms; Section III proposes the new algorithm called PHDNAC [Pattern Hunting DNA Compression Algorithm]. Section IV demonstrates the achievability and competence of the proposed method and finally, Section V contains the conclusions.

Existing DNA compression Algorithms: The standard general purpose compression algorithm such as “gzip”, “bzip2”, “WinZip” are well-known compression algorithm specially designed for text compression. But these compression algorithms are not succeeded to compress the DNA genome file with better compression ratio. Most of them attained the compression ratio for DNA sequences are greater than 2 bits per base on an average [14, 15]. The compression algorithms specifically designed for DNA sequences are also not attained average compression ratio for DNA sequences below 1.7 bits/base. Algorithms such as Biocompress, Gencompress and DNA compress compress the DNA sequences is about 1.7 bits per base on average. Grumbach and Tahi [10] proposed compression algorithms called Biocompress and Biocompress2 with the idea of Ziv and Lempel data compression method. BioCompress -1 and BioCompress -2 are compression algorithms 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 first occurrence's position. Two-bit encoding is used if the size of the code word is greater than the factor. The Substitutional method like Ziv and Lempel's encoding and statistical methods like the arithmetic encoding of order 2 enhance the Biocompress algorithm to the highest performance compression algorithm of DNA sequences.

Cfact was proposed by E.Rivals et al [20] which was not only detects repeats in a text but select some of them according to their compressibility. Cfact is a two-pass algorithm. The first pass is to find the repeated segment in a sequence and the second pass is to measure their quantitative importance by the compression rate. It searches for the longest repeat pattern using Suffix tree structure in the DNA sequence. The idea of Cfact is basically the same as Biocompress-2. Gen Compress was proposed by Chen et al the performance of which is based on reference sequence selection as approximate matching with edit operations uses this reference sequence for compression. It takes both approximate repeats and repeal complements and encodes it with length, position, and the errors. Context-Tree Weighting (CTW) algorithm is used in the CTW-LZ DNA compression algorithm, which achieves an outstanding compression ratio but takes a long time to encode the sequences. Genome Compress [8] compresses both repetitive and nonrepetitive sequences. DNA Compress is a two-phase algorithm designed by Chen et al [24, 25] uses special algorithm called PatternHunter for finding the repeats of DNA sequences. PatternHunter finds complementary palindromes and approximate repeats with the highest score in the first phase and encodes them in the second phase. Hence DNA Compress involves less searching time; it checks 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 to get the compressed data set. Behshad Behzadi et al [3] used Dynamic Programming techniques to identify the repeating sequences called DNAPack. Raja Rajeswari et al proposed GenBit Compress algorithm [18] that also compresses the repetitive and nonrepetitive DNA sequences [22] by assigning shorter binary code based on the method of the extended binary tree [19]. SBVRLDNA Comp [24] is designed for the compression of DNA and for RNA sequence. Rahul Vishwakarma [17] introduced mapping table to encode the data into nucleotide sequence.

Proposed DNA Compression Method PHDNAC: DNA sequences are the composition of four nucleotides bases {A, C, G, T}, hence to compress a single DNA base, two bits are more than enough. And therefore, the storage space required for DNA sequences can be an average of 2 bits per base [23]. The standard DNA compression algorithm like Genome Compress, DNA Compressreduced the compression ratio of DNA sequences up to 1.7 bits per base. Reducing the DNA compression ratio below 1.7 bits per base is a very challenging task [9]. The special characteristics of the DNA sequences can help to the Standard DNA compression algorithms to reduce the storage space of the DNA sequences below 1.7 BPB. The special characteristics of the DNA sequences are given as follows.

- One of the special characteristics of DNA sequences is the "complements" of the DNA base. In DNA sequences, A and T are complements of each other; G and C are also complements of each other. The complement of the DNA sequence AGGGCAAACGT is TCCCGTTTGCA.
- Another important characteristic of DNA sequences is the "reverse complements" of the DNA base. In DNA sequences, A and T are complements of each other; G and C are also complements of each other. CCCTTTGCA is the complement of the DNA sequence GGGAAACGT. ACGTTTCCC is the reversed complement of CCCTTTCGA.
- Some other characteristics of the DNA sequences are i)They contain many tandems repeats [17] ii) Many of the strings are palindromes iii) Some of them are reverse palindromes and iv) nucleotide may appear only nine consecutive times.

The above discussed characteristics of the DNA sequences conclude that better compression for DNA sequences is possible.

The algorithm proposed to compress the DNA sequences is of Two-Pass algorithm. In order to achieve a good compression ratio for DNA sequences, it finds the longest repetitive patterns and also their frequency of occurrences and generates the Prefix Code Table for the Pattern of size 3 to 9 with Codeword in the first Pass. The frequencies of occurrences of the DNA patterns are used to calculate the pattern probability to generate the Codeword using the Extended Binary Tree. It is enough to generate the Prefix 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 Codeword is assigned to the patterns found in the DNA sequences based on the technique Extended Binary Tree called Huffman coding, introduced by David Huffman in 1952, that compresses texts by assigning shorter codes to more frequently used patterns and longer codes to less frequently used patterns. And in the second pass only, the DNA sequence file is encoded using the Codeword.

Algorithm for the Proposed PHDNAC Method:

- Scan the given input DNA sequences and identify the repeating patterns.
- If the pattern is not an existing one from $\prod_{i=1}^{n} P\text{i}$ then it is encoded as P_{i+1} and the Prefix Code Table is generated for the Pattern of size 3 bytes to 9 bytes.
- Once the repeating patterns are identified, and then order them according to their frequency of occurrences.
- The probabilities of patterns in an input file are computed by their frequency of occurrences of each pattern.
- Patterns are coded by generating the Prefix code tree based on Extended Binary Tree called Huffman coding with uniqueness according to the probabilities of patterns in an input file.
- The length of the longest Codeword is the height of the Prefix code tree generated by the Huffman coding. The proposed PHDNAC method limits the lengths of certain Huffman codes to the maximum of n x 2 bits to a set of $\boldsymbol{2^n}$patterns where n is the number of bases.
- If there are any individual bases (nonrepeat 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 the individual base.
- The additional bit 0 is used as Most Significant Bit in the Pattern Code to indicate the Pattern as the regular one.
- The additional bit 1 is used as Most Significant Bit in the Pattern Code to indicate the Pattern as the reverse pattern.
- To indicate the uncompressed DNA base, the pattern code 111 is used that instruct the file header to read the next short integer to obtain the data about the uncompressed base.

Algorithm* to *generate the Prefix Code Table for the Pattern of size 3 to 9 bases with Codeword:

1. To create Codeword for each Pattern of size 3 to 9 bases, the Patterns present in the DNA sequence file are sorted in increasing order by frequency. Totally 7 Prefix code Table is required for the Pattern of size 3 to 9 bases. For each Prefix code Table,Extended Binary Tree called Huffman coding is generated and unique Codeword is assigned. Procedure for building a Huffman Prefix code tree is given as follows.
   a. Create a list of free nodes, where each node corresponds to each Pattern P0, P1, P2, and P3 and so on.
   b. Select two free nodes with the lowest frequency Pattern from the list.
   c. Create a parent node for two nodes found in b) with a frequency equal to the sum of the two child nodes.
   d. Remove the two nodes found in b) from the list of free nodes, and add the parent node created in c) to the list.
   e. Repeat the process starting from b) until only a single tree remains.

f. After building the Huffman Pattern Code tree, Assign 0 for a left branch and 1 for a right branch.
g. Create a Prefix Codeword for each Pattern by traversing the binary tree from the root to the node, which corresponds to the Pattern.The Prefix Codeword generated by the Huffman Pattern Code tree is uniquely decodable because Prefix Codeword is a Codeword in which prefix part of a Codeword is not a Codeword by itself.

2. Repeat the steps a to g forPattern of size 3 to 9 bases and generate 7 Pattern Code Table.
3. The number of patterns in each of the Prefix code Table is limited by checking the number of bits required to represent a pattern is less than or equal to n x 2, where n is the number of bases in the pattern.Length limited Prefix code tree is used to attain a minimum weighted path length.
4. The remaining Patterns from the Prefix code Table can be removed for better compression.
5. The compressed file consists of three different regions: Header, Compressed file and the length of the file header. The Header contains all the information like Prefix code Table content that must be known to decode the compressed code regions.

The compressed file is structured as:

- Blocks of bits representing the content of the Prefix code Table [000-110] of exact 3 bytes to 9 bytes pattern that provide the maximum amount of compression in the encoding pass generated by the Extended Binary Tree based on Huffman Coding required at the time of encoding as well as decoding process.
- Blocks of bits representing the Pattern Code [000-111].
- Blocks of bit representing the Codeword of identified patterns with the pattern type [0/1] in a contiguous form to indicate the repeated patterns and also the reversed pattern.
- Blocks of bit patterns representing compressed data in a contiguous form.

Performance Analysis of the Proposed Method: The proposed method using the Extended Binary Tree called the Huffman Coding is illustrated in the following DNA sequences with 24 Patterns of size 4 bases. The Single base needs 8 bits of storage space. The efficiency of the proposed method is measured in terms of bits per base (BPB).
P1 P2 P1 P0 P1 P0 P3 P2 P0 P1 P0 P2 P0 P3 P0 P3 P0 P1 P0 P1 P0 P0 P0 P1
Where P0 P1 P2 P3 are the 4 base DNA Pattern
P0 = AGTC
P1 = ACCT
P2 = ACTG
P3 = ATGA
Space required for the above DNA sequences: 4 x 8 x 24 = 768 bits = 96 bytes.

Fig.1 shows the Prefix Code tree generated using the Extended Binary Tree called Huffman coding for the above DNA sequences. Table 1 gives the Codeword derived from the Prefix Code tree for the above DNA sequences.

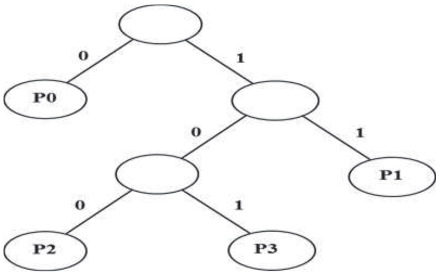

Fig 1: Prefix Code Tree

Table 1: Prefix Code Table

Pattern Id	Frequency	Probability	Pattern Codeword
P0	11	0.46	0
P1	7	0.29	11
P2	3	0.125	100
P3	3	0.125	101

Codeword structure for the proposed method for the above DNA sequences is given as: 3 bits are used to indicate the Patterns of Size 3 Bytes – 9 Bytes (Pattern Code 000-110, 111 to indicate the unaltered Pattern) + single bit is used to indicate the Pattern Type + Pattern Codeword

Space required for the above DNA sequences as per the proposed method is: Bits required to store the Patterns with its Pattern Codeword (3+16 x 2 + 9+1) + Bits required to indicate the Pattern Type(1) + Bits required to store the Pattern Code Size [000 -110] with its Pattern Codeword(129). Bits required to store the Pattern Code Size with its Pattern Codeword (129) are calculated as follows as 129 bits.
P0 = 11 x 5 = 55 bits P1 = 7 x 6 = 42 bits
P2 = 3 x 7= 21 bits P3 = 3 x 7 =21 bits

Total Space required for the aboveDNA sequences as per the proposed method is 139 bits +16 x 2 +9 + 1 +3 bits= 184 bits = 23 bytes approximately 80 % reduction of storage area. According to the proposed DNA compression method, the compression ratio for above DNA sequence is 1.91 BPB.

In the decoding process, the first three bits indicate the pattern code [3 bytes to 9 bytes 000-110/unaltered byte 111] and the next single bit indicates the pattern type[0-repeated pattern,1 – reversed pattern] and finally, the pattern Codeword generated by the extended binary prefix code tree. During the decoding process, the pattern Codeword is extracted bit by bit and finds the match in the pattern code table and replaced with the pattern.

Table 2: Sample Prefix Code Table for Patterns of Size 3 Bytes To 9 Bytes

P. ID	Pattern	Probability	P.Code & Type =0	Reverse Pattern	P.Code&Type =1
Patterns of Size 3 Bytes – 9 Bytes (Pattern Code 000-110)					
P1	AAA ---	0.1300	000	Nil	000
P2	CCC --- CCCCCCCCC	0.0900	0010	Nil	0010
P3	3 Bytes – 9 Bytes **Sample Code** ACT ACTG ATGAC AAACGT ACCATAG ATCGAATG AATGCGTCG	0.0800	0011	Reverse of 3 Bytes – 9 Bytes **Sample Code** TCA GTCA CAGTA TGCAAA GATACCA GTAAGCTA GCTGCGTAA	0011
P4		0.0800	0100		0100
P5		0.0700	0101		0101
P7		0.0650	0110		0110
P7		0.0650	0111		0111
P8		0.0600	10000		10000
P9		0.0600	10001		10001
P10		0.0400	10010		10010
P11		0.0350	10011		10011
P12		0.0300	10100		10100
P13		0.0300	10101		10101
P14		0.0300	10110		10110
P15		0.0200	10111		10111
P16		0.0200	11000		11000
P17		0.0200	11001		11001
P18		0.0150	11010		11010
P19		0.0150	11011		11011
P20		0.0150	11100		11100
P21		0.0100	11101		11101
P22		0.0050	111100		111100
P23		0.0050	111101		111101
P24		0.0050	111110		111110
P25		0.0025	1111110		1111110
P26		0.0025	1111111		1111111

Table 2 shows the general Prefix Code Table for Patterns of Size 3 Bytes – 9 Bytes with 26 sample Codeword generated by the Extended Binary Tree. In the case of equal probability of the occurrence of the pattern, the prefix code tree can generate the unique Codeword as shown in the Table II. According to Table II, it is understood that generation of Codeword for more patterns are possible with less number of bits. The minimum number of bits used to generate the Codeword will increase the performance of the DNA compression algorithms [16]. The decoding of the DNA sequence is also possible for the extended binary prefix code tree with high speed.

To analyze the competence of the proposed PHDNAC algorithm, standard data set of DNA sequences are compressed and the results are compared with the compression ratio of other efficient DNA compressors. The standard specially designed DNA compression algorithms used for comparison are BioCompress2, Genome Compress, CTW+LZ, DNA Compress, DNAPACK and the general purpose compression software are WinRAR, WinZipand Bzip2.

The following standard DNA sequences are used for the purpose of analysis of the efficiency of the various DNA compression algorithms.

- A chloroplast genomes (CHMPXX)
- Three human genes (HUMHDABCD, HUMHBB, HUMHPRTB)
- Two virus genomes (HEHCMVCG, VACCG)

Table 3: Space Required For the General Purpose Compression Algorithm for DNA Sequences

Sequence Name	File Size(Bytes)	WinRAR(Bytes)	WinZip(Bytes)	Bzip2(Bytes)
Chmpxx	121024	34022	34775	32099
Humhbb	73308	20306	20780	19684
Humhdabcd	58864	16110	16624	15215
Humhprtb	56737	15804	16228	14854
Hehcmvcg	229354	66233	67757	62169
Vaccg	191737	53188	54561	50209

These standard DNA sequence data sets are used for comparison purposes by many researchers who work in the field of DNA sequence compression algorithm. The DNA sequence files can be downloaded from the GENBANK database. The DNA sequence files are made available in FASTA file format in DNA databases which can also retrieve by any text processor like notepad [12]. Table 3 gives the storage space required for the general purpose compression algorithm for DNA sequences and Table 4 shows the compression ratio of the general purpose compression algorithm for DNA sequences.

Table 4: Compression Ratio of The General Purpose Compression Algorithm for DNA Sequences

Sequence Name	File Size(Bytes)	WinRAR (BPB)	WinZip(BPB)	Bzip2(BPB)
Chmpxx	121024	2.248	2.298	2.122
Humhbb	73308	2.215	2.267	2.148
Humhdabcd	58864	2.189	2.259	2.067
Humhprtb	56737	2.228	2.288	2.094
Hehcmvcg	229354	2.310	2.362	2.168
Vaccg	191737	2.219	2.276	2.095
Average(BPB)		2.235	2.292	2.116

Table 5: Space Saved For the DNA Sequences Using Various the General Purpose Compression Algorithm

Sequence Name	File Size (Bytes)	WinRAR (BPB)	WinZip(BPB)	Bzip2(BPB)
	% of Space Saved			
Chmpxx	121024	71.89	71.27	73.48
Humhbb	73308	72.30	71.65	73.15
Humhdabcd	58864	72.63	71.76	74.15
Humhprtb	56737	72.15	71.40	73.82
Hehcmvcg	229354	71.12	70.46	72.89
Vaccg	191737	72.26	71.54	73.81
Average		72.06	71.35	73.55

Table 6: Space Saved for the DNA Sequences using various DNA Compression Algorithms

Sequence Name	BioCompress	Genome	CTW+LZ	DNAcompress	DNAPACK	PHDNAC
	% of Space Saved					
Chmpxx	78.94	79.12	79.14	79.10	79.25	82.00
Humhbb	76.50	77.25	77.40	77.63	77.79	80.10
Humhdabcd	76.54	77.25	77.23	77.63	78.26	80.10
Humhprtb	76.17	76.88	76.96	77.29	77.64	80.00
Hehcmvcg	76.90	76.88	76.98	76.89	77.07	79.61
Vaccg	77.98	78.00	77.98	78.03	78.02	79.90
Average	77.17	77.56	77.61	77.76	78.01	80.23

The following conclusion is drawn from Table 5 and Table 6.

- The proposed method compresses the DNA Sequences much better than the other specially designed DNA compression algorithms.
- This algorithm works extremely well in compression over the general purpose compression algorithms like WinRAR, WinZip and Bzip2 as evident from Table 5.
- According to Bio-Compress DNA compression algorithm taken from Behshad Behzadi [3], average BPB for the above discussed DNA data set is 1.83. And for Genome Compress is 1.80 BPB.CTW+LZ give the compression ratio of 1.79 BPB and DNA Compress gives 1.78 BPB compression ratio. According to DNAPACK, compression ratio is of 1.76 BPB. But the proposed method gives the compression ratio of 1.59 BPB.
- It is also observed that, in all category of standard data set, a better compression gain in terms of the percentage of storage space saved by the proposed method is approximately 80 % and above as an average.
- Speed has been part of DNA compression algorithms. The proposed algorithm achieves a good compression speed too.

Conclusion: DNA sequences which characteristics are repetitive with nine consecutive times as well as nonrepetitive in nature. Because of these characteristics, the better compression ratio is possible by making use of the pattern recognition and extended prefix code tree to develop special DNA compression algorithm. The prefix code tree is well known for its unique code generation. In this specially designed DNA compression algorithm PHDNAC, both the pattern recognition and Length limited extended binary prefix code tree are used to get higher compression as well as to get high speed. The ease and flexibility of this proposed DNA Compress algorithm could make it as an invaluable tool for DNA compression in the field of bioinformatics research.

References:

1. Alam Jahaan, T.N. Ravi, S. Panneer Arokiaraj, “A Comparative Study and Survey on Existing DNA Compression Techniques”, International Journal of Advanced Research in Computer Science, Volume 8, No. 3, March – April 2017.
2. Bacem Saada, Jing Zhang,“Vertical DNA Sequences Compression Algorithm Based on Hexadecimal Representation” , Proceedings of the World Congress on Engineering and Computer Science 2015 Vol II WCECS 2015, October 21-23, San Francisco, USA,2015.
3. Behzadi, B. and Le Fessant F,“DNA Compression Challenge Revisited”, Symposium on Combinatorial Pattern Matching (CPM2005), pp.190-200,June 2005.
4. Calladine, C. R. and Horace A. Drew. “Understanding DNA: The Molecule and How It Works”, San Diego: Academic Press, 1997.
5. 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.

6. 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.
7. Don Adjeroh, Yong Zhang, Amar Mukherjee, Matt Powell and Tim Bell,"DNA Sequence Compression using the Burrows-Wheeler Transform", Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB'02), 2002.
8. U. Ghoshdastider et al., "GenomeCompress: A Novel Algorithm for DNA Compression", ISSN 0973-6824,2005.
9. Grumbach, S. and Tahi F., "Compression of DNA Sequences", In Proc. IEEE Symp. On Data Compression, pp. 340-350, 1993.
10. Grumbach, S. and Tahi F, "A new challenge for compression algorithms: Genetic Sequences", Journal of Information Processing &Management, Vol. 30, pp. 875-886, 1994.
11. Heba Mahmoud Mohmmed Affify, "Lossless Differential Compression Algorithm For Genomic Sequence Databases", Thesis in Systems And Biomedical Engineering ,Cairo University, Giza, Egypt, 2012.
12. Jan Mra´ zek and Shaohua Xie, " Pattern locator: A new tool for finding local sequence patterns in genomic DNA sequences",University of Georgia, Athens, GA 30602, USA,October 19, 2006.
13. Kamtanath Mishra, Dr.Anupam Agarwal, Dr.Edries Abdelhadi 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.
14. 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.
15. Matsumoto,T. et al. "Biological sequence compression algorithms. Genome", Inform. 11, 43–52,2000 .
16. Raffaele Giancarlo, Davide Scaturro and Filippo Utro, "Textual data compression in computational biology", synopsis, Vol. 25 no. 13 , pages 1575–1586 ,2009.
17. Rahul Vishwakarma and Newsha Amiri, "High Density Data Storage in DNA using an Efficient Message Encoding Scheme", International Journal of Information Technology Convergence and Services (IJITCS) Vol.2, No.2, April 2012.
18. 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.
19. Raja Rajeswari and Dr.AlamApparao, "DNABIT Compress – Genome compression algorithm", Bioinformation Volume 5 ,Issue 8, January 22, 2011.
20. 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.
21. Sebastian Wandelt, Marc Bux, and Ulf Leser, "Trends in Genome Compression", Institute for Computer Science,Humboldt-University at Zu Berlin, Germany,June 4, 2013.
22. Shanika Sewwandini Kuruppu,"Compression of Large DNA Databases", Melbourne School of Engineering,The University of Melbourne,January, 2012.
23. Soliman, T., "A Lossless Compression Algorithm for DNA sequences", International Journal of Bioinformatics and Applications, Vol. 5(6), pp. 593, 2009.
24. Subhankar Roy,Akash Bhagot,Kumari Annapurna Sharma and SunirmalKhatua, "SBVRLDNACOMP: An Effective DnaSequence Compression Algorithm",International Journal on Computational Science & Applications (IJCSA) Vol.5, No.4, August 2015.
25. Xin Chen, et al., "DNA Compress: fast and effective DNA sequence Compression" Bioinformatics Applications Note, Vol. 18 no. 12, Pages 1696–1698, 2002.
26. Xin 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.

***