A Vital Approach to compress the Size of DNA Sequence using LZW (Lempel-Ziv-Welch) with Fixed Length Binary Code and Tree structure

Nishad PM
Ph.D Scholar,
Department Of Computer Science
NGM College, Pollachi, India

R. Manicka Chezian
Associate professor,
Department of computer science NGM College
Pollachi, Coimbatore, India

ABSTRACT
The genome of an organism contains all hereditary information encoded in Deoxyribonucleic Acid (DNA). Molecular sequence databases (e.g.,EMBL, Genbank, DDJB, Entrez, SwissProt, etc) represent millions of DNA sequences filling many thousands of gigabytes and the databases are doubled in size every 6-8 months, which may go to beyond the limit of storage capacity. There are several text compression algorithm used for DNA compression. This paper proposes a new hybrid algorithm is used to compress DNA sequence, the algorithm is designed by combining the fixed length binary code with the LZW (Lempel-Ziv-Welch) compression algorithm. Initially the input sequence is divided in to fragments where each fragment consist of four nucleotides and fixed length binary code is assigned to each nucleotide then the pattern (STR and CHR) in LZW used the same for creating the dictionary. Assigning a new binary code for each pattern in the dictionary using a binary tree, and the sequence is replaced binary code for the longest match in the dictionary while compression. The proposed approach attains maximum compression in DNA sequences.

Keywords
EMBL, DDJB, genome, Deoxyribonucleic acid (DNA), LZW, nucleotide.

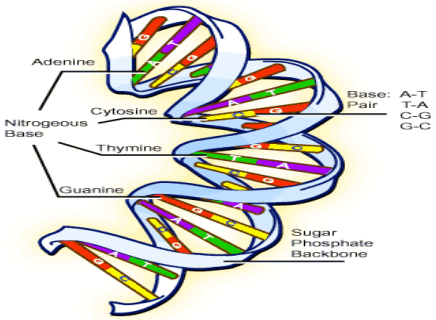
1. INTRODUCTION


Fig -1 Structure of DNA

2. EMPIRICAL REVIEW
Genbit Compression algorithm is proposed in the paper [1]. The sequence is divided in to fragments where each fragment has four characters and every fragment is replaced by eight bit binary number to differentiate the same fragment consecutive order the additional one ”1” bit is used else the bit “0” is used. So, nine bit is used to replace thirty two bit sequence. Another DNA compression algorithm is proposed in the year 2010 using hash based data structure [2]. The algorithm builds the hash table and assigns a key for unique sequences and next the replacement is made with a single character. Grumbach and Tahi [3,4] proposed two lossless compression algorithms for DNA sequences, namely *Biocompress* and *Biocompress-2*, in the spirit of the Ziv and Lempel data compression method [11]. *Biocompress-2* detects exact repeats and complementary palindromes located order-2 arithmetic coder [5] is to perform the entropy coding without knowing the modeling environment. Rivals et al. [6] give another compression algorithm, *Cfact*, which searches the longest exact matching repeat using suffix tree data structure in an entire sequence. Tabus *et al.* proposed a DNA sequence compression method based on normalized maximum likelihood discrete regression for approximate block matching [7]. One of the lossless data compression widely used is LZW data compression, it is a dictionary based algorithm. LZW compression is named after its developers, A. Lempel and J. Ziv, with later modifications by Terry A. Welch [9]. Lempel-Ziv-Welch (LZW) [9] this algorithm proposed by Welch in 1984. This algorithm is an improved implementation of the LZ78 algorithm published by Lempel and Ziv in 1978 (LZ78) [10]. The first algorithm of Lempel and Zive was published in 1977 and it is named as LZ77 [11]. The LZ77 [11] and LZ78 [10] are otherwise called LZ1 and LZ2 respectively. The LZW algorithm uses dictionary and index for encoding and decoding operation. It creates a dictionary and if a mach is found in the dictionary then corresponding string is replaced by the index. LZW compression became the first widely used universal data compression method on computers. After the invention of LZW there are lots of improvements and enhancement done in LZW for data compression that is discussed in this section. LZW compression works best for files containing lots of repetitive data.

3. PROPOSED ALGORITHM
These paper discuses the problem of DNA compression both for repetitive and non repetitive DNA sequences. To improve the compression rate, a new technique is proposed been devised, which is much effective with respect to compression rate. DNA sequence contains only four nucleotides namely A, C, G, T. So the input sequence is divided in to fragments where each fragment equal to four nucleotides, then construct a unique fixed length code (8 bit) for fragment by assigning binary code by replacing A =’00’, C=’01’, G=’10’ and T=’11’ in a fragment so this leads to 8 bit unique code for example ‘CATG’ is the first fragment in the sequence shown in the fig -1 so after assigning the code the algorithm generates ‘01001110’. Then Binary value is placed in STR so STR =‘01001110’. Table-1 Shows this action shown in line1. This is followed by the algorithm looping for each additional

fragment is fetched form the Sequence and assigns a fixed length binary code for it as shown earlier. Each time the code is generated from the sequence it is stored in the variable, CHR. The data table (Dictionary) is then searched to determine if the concatenation of the two variables, *STR+CHR*, has already been in the table. If a match in the code table is not found, the following actions are taken place. The concatenation of *STRG+CHR* is stored in the variable, *STR.* Else the CHR is assigned to the STR and the STR+CHR is added to the table. In table -1 line number-14 STR ='01001101' and CHR= '01001101' so the Concatenated (STR+CHR) is already in the table then STR is reassigned into "0100110101001101" then next fragment Code is fetched for CHR is "00000101" so the STR+CHR= "01001101 01001101 00000101" (Table-1 Line number -15), the presence is failed for STR+CHR hence the STR+CHR is added to the Table.

CATGCCATCAACCCACCATCCATCAACCCATCCATCA
ACCCATCCATCCATCAACCCATCCATCCATCCATCCG
AG

Fig -2 DNA Sequence

The next phase of the algorithm is binary tree construction for the dictionary. The tree is built with the following steps. The two free nodes from the right side are located. A parent node of these two nodes is created and the parent node is added to the list of free nodes of next level. The process is continued up to the level is completed and any free node left after the parent creation then that node also treated as the free node of next level. One child node is designed path taken from parent node when decoding a 0 bit. The other is arbitrarily set to 1 bit. The previous steps are repeated until only free node left. This free node is designated the root of the tree. The tree creation is illustrated in Fig -3. In Level-0 '010011010100110101100010' and '0100110101001101010011101' located first then parent node is created for the Left child '0' and for the right child '1' is assigned. The same is repeated until the free node is available in the level-0. In level-2 fig-1 after creating the parent for the right child one node is free so that node is set to the free node of next level, So new binary codes is generated for the members of dictionary, that is shown in the Table-2. For '0100111001010011' code is '000', '0101001101000001' is '001' and so on.

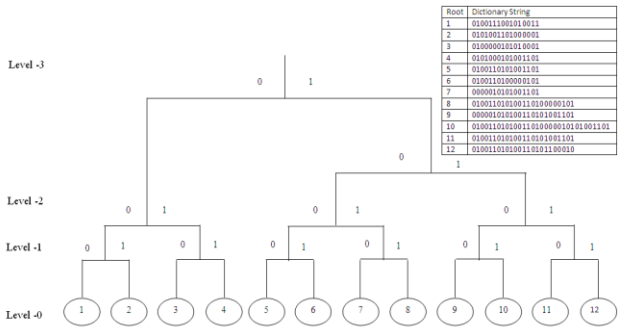


Fig-3 Binary Tree for the LZW Dictionary

Table 2. Dictionary after Tree Substitution

S no	Dictionary String	New Code after Tree Substitution
1	0100111001010011	000
2	0101001101000001	001
3	0100000101010001	010
4	0101000101001101	011
5	0100111010011101	1000
6	0100110100000101	1001
7	0000010101001101	1010
8	0100110101001101010000101	1011
9	0000010101001101010011101	1100
10	0100110101001101010000101010011101	1101
11	0100110101001101010011101	1110
12	0100110101001101011100010	1111

The next phase of the algorithm is replacement of fragment code with the new binary code. The first fragment code is accepted from the DNA sequence then assign to the STR and set CNT (count) equal to zero initially, This is followed by the algorithm looping for each additional fragment is fetched form the sequence and assigns a fixed length binary code for it as shown in the earlier. Each time the code is generated from the sequence, it is stored in the variable CHR. The data table (Dictionary) is then searched to determine if the concatenation of the two variables, STR+CHR, has already been in the table. If a match in the code table is not found, the following actions are taken place. STR is reassigned to STR+CHR else check for CNT equal to Zero the dictionary is dynamically updated by new entry ((i.e.), the code entry for '0000010101001101' not available in the dictionary so the new code is assigned and updated the dictionary.) and code is assigned to STR+CHR the New binary code of STR is assigned, that is coded as the output for STR. STR equal to next fragment code (i.e.), initially the STR='01001110' and CHR= '01010011' and Search For STR+CHR ('0100111001010011') is available in the Dictionary So the STR is set to '0100111001010011' then CHR is assigned as the next fragment code is '01000001' so the STR+CHR is Failed in the table so the output is the new binary code of STR that is '000' and CHR is set to STR. The output sequence of the proposed algorithm is shown in Fig -4. Decoding phase is done by fetch the compressed code and matches with the dictionary and if a match is found then the corresponding original value is taken from the table and converts the value to the original nucleotides (the decompression phase is related to the Huffman coding [12]) and the fixed length code is replaced with the appropriate nucleotides.

000 010 1011 1011 1110 10000 1110 10001

Actual size of DNA Sequence is 76 Bytes.

After Compression: 4 Bytes.

Compression ratio 94.59 %

Fig -4 Compressed DNA sequence

4. CONCLUSION

A simple DNA compression algorithm which is completely new in its design is lossless compression is proposed to compress DNA sequences which are repetitive as well as non repetitive in nature. DNA sequence analysis i.e. single and multiple alignments are areas of active research in

bioinformatics. The algorithm used both statistical and Dictionary based concepts. The proposed algorithm gives the high compression ratio on DNA sequences.
The proposed work can be further enhanced and expanded for the authentication of compression and decompression techniques to obtain optimum compression.

5. ACKNOWLEDGMENT

Late Dr. N. Nalayini. Associate Professor, Department of computer science, NGM College, Pollachi.

6. REFERENCES

[1] Raja Rajeswari and Dr. Allam Apparao "Genbit compress – algorithm for repetitive and non-repetitive DNA sequences" Journal of Theoretical and Applied Information Technology 2005

[2] Ateet Mehta and Bankim Patel "DNA compression Using Hash Based Data Structure" International Journal of Information Technology and Knowledge Management July-December 2010, Volume 2, No. 2, pp. 383-386

[3] S. Grumbach and F. Tahi, "A new challenge for compression algorithms: Genetic sequences," J. Inform. Process. Manage., vol. 30, no. 6, pp. 875-866, 1994.

[4] S. Grumbach and F. Tahi, "Compression ofDNA sequences," in Proc. IEEE Symp. Data Compression, Snowbird, UT, 1993, pp. 340-350.

[5] I.H. Witten, R. Neal, and J.G. Cleary, "Arithmetic coding for data compression," Commun. ACM, vol.30, pp.52-541, Jun. 1987.

[6] É. Rivals, J.P. Delahaye, M. Dauchet, and O. Delgrange, "A Guaranteed Compression Scheme for Repetitive DNA Sequences," LIFL Lille I Univ., Tech. Rep. IT-285, 1995.

[7] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," ACM Trans. on Information Systems, vol. 23, no. 1, pp. 3–34, Jan. 2005.

[8] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Reading, Massachusetts: Addison-Welsley Publishing Company, 1992.

[9] WELCH, T. A. 1984. A technique for high-performance data compression. IEEE Comput. 17, 6, 8–19. 9

[10] ZIV, J. AND LEMPEL, A. 1978. "Compression of individual sequences via variable-rate coding". IEEE Trans. Inform. Theory 24, 5, 530–536.

[11] ZIV, J. AND LEMPEL, A. 1977. "A universal algorithm for sequential data compression". IEEE Trans. Inform. Theory 23, 3, 337–343.

[12] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102.

7. AUTHORS PROFILE

Nishad PM M.Sc., M.Phil. Seven months Worked as a project trainee in Wipro in 2005, five years experience in teaching, one and half year in JNASC and Three and half year in MES Mampad College. He has published eight papers national level/international conference and journals. He has presented three seminars at national Level. Now he is pursuing Ph.D Computer Science in Dr. Mahalingam center for research and development at NGM College Pollachi.

Dr. R.Manicka chezian received his M.Sc., degree in Applied Science from P.S.G College of Technology, Coimbatore, India in 1987. He completed his M.S. degree in Software Systems from Birla Institute of Technology and Science, Pilani, Rajasthan, India and Ph D degree in Computer Science from School of Computer Science and Engineering, Bharathiar University, Coimbatore, India. He served as a Faculty of Maths and Computer Applications at P.S.G College of Technology, Coimbatore from 1987 to 1989. Presently, he has been working as an Associate Professor of Computer Science in N G M College (Autonomous), Pollachi under Bharathiar University, Coimbatore, India since 1989. He has published thirty papers in international/national journal and conferences: He is a recipient of many awards like Desha Mithra Award and Best Paper Award. His research focuses on Network Databases, Data Mining, Distributed Computing, Data Compression, Mobile Computing, Real Time Systems and Bio-Informatics.