An Improvised DNA Sequence Compressor Using Pattern Recognition

Panneer Arokiaraj S[1], Robert L[2]

[1]Department of Computer Science, Periyar EVRCollege, Trichy, TN, India
[2] Computer Science & Info. System Department, Community College in Al-Qwaiya, Shaqra University, KSA
[1] panksop@gmail.com
[2]robert_lourdes@yahoo.com.

***Abstract* – Genome contains the hereditary information of biological organisms. Currently, there are large number of DNA sequences are stored in DNA databases. This paper presents an improvised version of (PRDNAC) Pattern Recognition based DNA Sequence Compression algorithm which compresses the DNA sequences. The development takes place in the area of time complexity factor and compression ratio.**

Key Words - Compression Ratio, DNA Sequence Compression, PRDNAC, Time Complexity.

I. INTRODUCTION TO DNA, COMPRESSION AND PATTERN RECOGNITION

DNA (Deoxyribonucleic Acid) contain the complete set of information required for the functioning of all living organisms. The DNA of all organism have four components in common. They are the four nucleotide bases; namely Adenine, Cytosine, Guanine, and Thymine**.** They are crisply represented using the first character of their names; namely A, C, G and T respectively [1, 2]. 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, N}. The first four elements are represented as a double helix with A & T in one helix and C & G in another helix. These two helix are held together by hydrogen bonds. The element N is still remains unknown and is yet to have a pictorial representations but participates in the functionalities of a DNA. The DNA sequences are characterized as follows. i) They contain oft-repeated substrings, ii) Many of the strings are palindromes, iii) Some of them are reverse palindromes [3].

The National Center for Biotechnology Information (NCBI) maintain a repository of DNA data and is called GenBank. So also the other two repositories maintain similar data. They are European Molecular Biology Laboratory (EMBL) and DNA Database of Japan (DDJB). Since the data occupies large space it becomes necessary to compress and store the DNA data.

Pattern recognition concepts help to improve the prevailing situation by identifying patterns and to reduce memory space thereby improving compression ratio. This paper presents an improved algorithm of an earlier research paper [4] by using certain characteristics that are inherent in DNA sequences. This improvisation has a positive implications in terms of time complexity and compression ratio.

II. REVIEW OF LITERATURE

Human being possess an implicit information processing system due to the fact that they have an excellent pattern recognition capabilities. The problem of pattern recognition is about finding difference of attributes, not between individual patterns but between populations using feature search of variant or invariant attributes among the population.

Many algorithms used to compress the DNA sequences met with success, but with varying levels of compression ratio. Some of the obtained better results and some of them with lesser compression ratio**.**

Grumbach and Tahi [5], [6] proposed compression algorithms called *Biocompres*s and *Biocompress*2 with the spirit of Ziv and Lempel data compression method. They used the idea of finding replica of sequences, such as repeats, palindromes and complementary palindromes. *Biocompress*2 used the order-2 arithmetic coding technique whereas *Biocompres*s didn't use the technique.

Cfact, a two pass algorithm, was proposed by E.Rivals et al. [7] which extracted the longest repeat using Suffix Tree data structure. Substitution based compression algorithm, called *GenCompress* was proposed by Chen et al. [8], that is a widely patronized algorithm among the scientific community, which handled approximate and inexact repeats also.

Context-tree weighting [9] method was used in the algorithm '*CTW*', that was proposed by Matsumoto et al. [10] that successfully handled the short repeats and LZ77 scheme handled the long exact or approximate repeats.

Another two-phase compression tool that used the Lempel and Ziv compression scheme was designed by Chen et. al [11] called *DNA Compress*. This algorithm finds all approximate repeats in the I-phase which used a special software called *Pattern Hunter*. The next phase encodes a sequence by a pointer to the earlier

occurrences. Behshad Behzadi et al. [12] used Dynamic Programming techniques to identify the repeating sequences called *DNAPack.*

Tabus [13] proposal of an algorithm, Normalized Maximum Likelihood (NML) model, combines substitution and statistical styles. Korodi and Tabus [14] proposed an improved model of NML having capabilities of splitting the sequence into fixed size blocks and encoding them using the history of a subsequence and manipulating the same with Hamming distance called GeNML.

Minh Due Cao et al. [15] produced another sequence compression algorithm called *Expert Model* (XM) using statistical methods. Raja Rajeswari et al. [16] developed *GenBit Compress,* an algorithm that compresses repetitive and non-repetitive sequences using the ideas of extended binary tree. "*LCA*", a Lossless Compression Algorithm was given by Taysir Soliman et al. [17] for handling several approximate repeats and complimentary palindromes. An algorithm, "*DNASC",* that combines statistical and substitution methods, was proposed by Kamta Nath Mishra et al [18].

III. METHODOLOGY

The five bases of DNA sequences {A, C, G, T, N} cannot be represented using 2 binary bits. A regular compression tool may not be able to achieve appreciable compression due to the fact that such regular compressors are designed for normal texts. The patterns of English text are different from the patterns of DNA sequences. The regularity of patterns available in DNA sequences can be taken advantage for better compression. To demonstrate the above claim, consider a sequence of given below having 128 bases.

AGTCAGTCCTGAAAGCACCTAAGCCGAATCCANTACNTACCCGTCCGTANTTTTTAATTTTTNA
CCGTTGCCTCCACTGACTGACGAACGTNCGAACTGATNGTATNGCTAAATNGCTAAAATCGNTN.

Repeatedly, scan the given input sequence and identify the repeating patterns. Once the repeating patterns are identified, then they are coded based on their uniqueness. For the above given sequence, the following basic patterns are identified: AG, AA, AC, AT, AN, CA, CC, CT, CG, GT, GA, GC, TC, TG, TA, TT, TN, NT, AGT, AAA, AAG, AGC, ACC, AAT, ATC, ANT, CCT, CTG, CAC, CTA, CCG, CGA, CCA, CGT, GTC, GAA, GCC, GTA, TCC, TGA, TAA, TAC, TTT, NTA, AGTC, AAGC, AATC, CTGA, CTAA, CGAA, CCGT, GTCC, TCCA, TTTT and TTTTT. In order to achieve a compression ratio, this compressor finds the longest repeating patterns and their reverse are identified. For example, TTTTT is the longest pattern and palindrome; AGTC is reverse of CTGA, ACCT is the reverse of TCCA and CGAA is reverse of AAGC and etc. are identified and coded in the following way: if the pattern is not an existing one from $\prod_{i=1}^{n} p_i$ then it is coded as P_{i+1} and the symbol table is generated. Table I represents the symbol table with 3 bytes required for coding the above mentioned sequence after eliminating the redundant patterns. The number of bits required to represent a pattern is determined by satisfying the condition that for any 'n', number of pattern formation $\leq 2^n$, possible cases. An additional bit is used for indicating the reverse or regular pattern apart from the required bits. The following are some sample symbol tables for the better understanding of this paper.

TABLE I
Symbol Table for Storing the Identified Patterns of Size 3 Bytes (Sequence Code 000)

Pattern Id	Pattern	Bit Patterns used (MSB=0)	Reverse	Bit Patterns used (MSB=1)
P_0	CCT	0x00	TCC	0x10
P_1	GCC	0x01	CCG	0x11
P_2	ANT	0x02	TNA	0x12
P_3	CGA	0x03	AGC	0x13
P_4	AAA	0x04	---	---
P_5	CCC	0x05	---	---
P_6	TTT	0x06	---	---
P_7	GGG	0x07	---	---
P_8	GTA	0x08	ATG	0x18

TABLE II
Symbol Table for Storing the Identified Patterns of Size 4 Bytes (Sequence Code 001)

Pattern Id	Pattern	Bit Patterns used (MSB=0)	Reverse	Bit Patterns used (MSB=1)
P_0	AGTC	0x00	CTGA	0x10
P_1	ACCT	0x01	TCCA	0x11
P_2	ATGA	0x02	AGTA	0x12
P_3	AAAA	0x03	---	---
P_4	CCCC	0x04	---	---
P_5	TTTT	0x05	---	---
P_6	GGGG	0x06	---	---
P_7	ACTG	0x07	GTCA	0x17

TABLE III
Symbol Table for Storing the Identified Patterns of Size 5 Bytes (Sequence Code 010)

Pattern Id	Pattern	Bit Patterns used (MSB=0)	Reverse	Bit Patterns used (MSB=1)
P_0	AAAAA	0x00	---	---
P_1	CCCCC	0x01	---	---
P_2	GGGGG	0x02	---	---
P_3	TTTTT	0x03	---	---

The symbol tables generated with a unique sequence code for the identified repeating patterns of varying sizes are shown in Table II and Table III. The sequence code 111 is used for representing the uncompressed data and to instruct the file header to read the next short integer to obtain the information about the uncompressed bytes.

The work file is updated with the following bits "000000000010000" to represent the 4 byte pattern called "AGTCAGTCCTGA" and the entries of variable length file header for the above mentioned sequence is " 001 001 001". From this example, the 12 bytes of data is represented as "0000000000100000 01001001"in 3 bytes form. Similarly, the construction of the file is formed in the same way till to the end of the file is encountered.

With an intention to achieve a higher compression ratio, the compressed file is organized as: a) set of all blocks representing the bit patterns of compressed and uncompressed data in a contiguous form, b) variable length file header representing the sequence code of identified patterns c) 4 bytes long unsigned integer to represent the length of the file header and d) end of the file marker – which contains the offset and the EOF. The Fig. 1 shows the components of the compressed file.

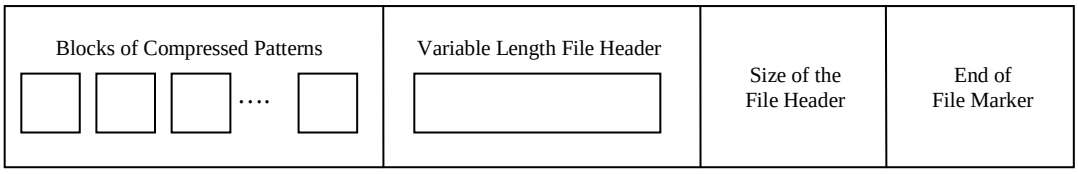

Fig. 1. Components of the Compressed File.

The following is the pseudo code incorporating all the above ideas. Steps to compress the DNA sequence.

1. Read the Sequence repeatedly and form the symbol tables with sequence codes.
2. Determine the number of bits required for representing the patterns identified.
3. Using the symbol tables generated in step 1, Construct the work file to represent the compressed and uncompressed sequences.
4. Repeat the step3 till the end of the file is encountered.
5. Update the work file with the indices required to retrieve the patterns without any loss.

Steps to decompress the compressed DNA sequence file.

1. Read the compressed file from the end to obtain the size of the file header.
2. Locate the starting point of the file header through the offset found at the end of the file.

3. Recollect the size of the blocks to be read, bit patterns and the sequence code to recollect the patterns.
4. Read the blocks of compressed patterns and explode it from the beginning to the end of the file.

IV. EXPERIMENTAL RESULTS

Performance comparison of this algorithm against many other standard algorithms presented in Table IV. The standard algorithms used for comparison are BioCompress2, Genome Compress, CTW, DNACompress, GeNML and DNASC and the widely used general compression software WinRAR. The following 11 benchmark standard data set of DNA sequences are used in this paper for the purpose of analysis:

- Two chloroplast genomes (CHMPXX and CHNTXX)
- Five human genes (HUMDYSTROP, HUMGHCSA, HUMHDABCD, HUMHBB and HUMHPRTB),
- Two mitochondria genomes (MPOMTCG and MTPACG) and
- Two virus genomes (HEHCMVCG and VACCG).

The above data sets are used by many authors who work in DNA sequence compression. The DNA sequences are made available in FASTA file format in DNA databases which can also retrieved by any text processor such as Notepad of Microsoft. A typical DNA sequence is in the form of a single word having no white spaces, soft return or an end of line marker, with a constraint that a nucleotide may appear only nine consecutive times. Efficiency of PRDNAC is measured in terms of bits per base (BPB), the bits required to store a nucleotide. The time complexity measures [19] such as the time required to compress and decompress are dealt in Table V. The following inferences are drawn from Table IV and Table V.

- PRDNAC algorithm compresses the data set much better than the other compression algorithms.
- PRDNAC algorithm excels in compression over the list of six techniques as evident from Table IV. It is observed that, in all cases, a better compression gain in terms of the percentage of storage space saved by PRDNAC is 88.73 at the maximum and 80.81 as an average.
- The proposed PRDNAC algorithm achieves a compression of below 1 bit to represent a base of DNA sequence for HUMGHCSA.
- As PRDNAC performs both compression and decompression at a very short span, whereas the GZIP executed only for compression process.
- PRDNAC algorithm does the compression process, 4 times faster than GZIP for the sequence CHNTXX, shown in Fig. 2.

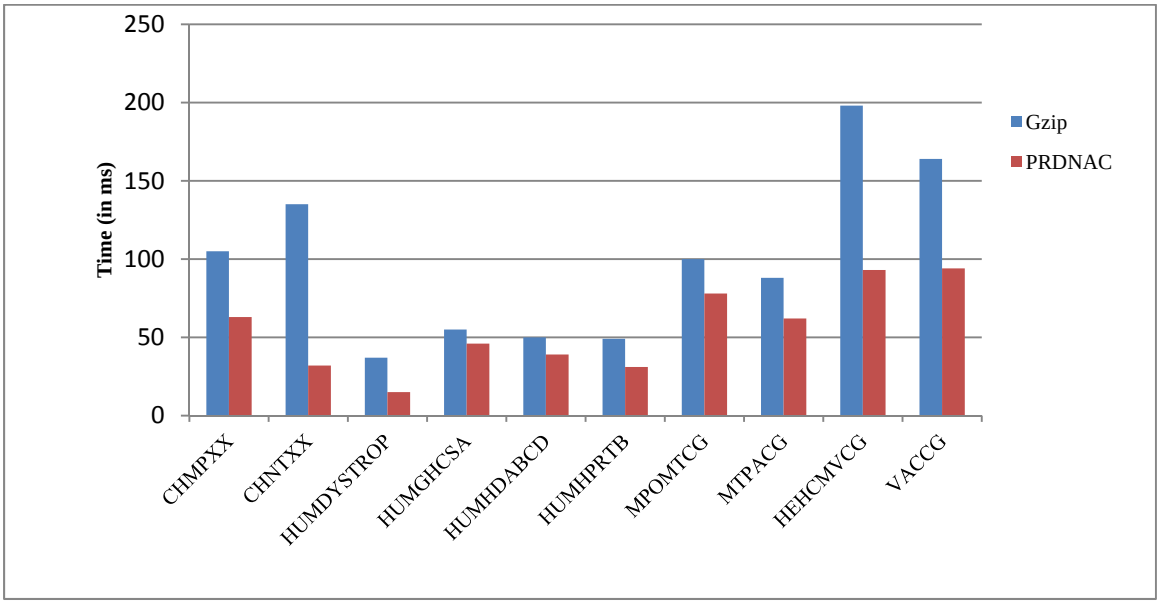

Fig. 2. Chart Showing Time Efficiency of PRDNAC with GZip.

TABLE IV
Efficiency Comparison of PRDNAC with Other DNA Compressors (Bits Per Base).

SEQUENCE	Size in Bytes	Compressed File size by PRDNAC in bytes	WinRAR	Bio-Compress2	Genome Compress	CTW	DNA Compress	Ge-NML	DNASC	PRDNAC	Percentage of Space Saved by PRDNAC
			BPB (Bits Per Base)								
CHMPXX	121024	22240	**2.25**	1.68	1.67	1.67	1.67	1.66	1.50	**1.47**	**81.62**
CHNTXX	155844	29050	**2.24**	1.62	1.61	1.61	1.61	1.61	1.51	**1.49**	**81.36**
HUMHBB	73323	14673	**2.22**	1.88	1.82	1.84	1.79	---	---	**1.60**	**79.99**
HUMDYSTROP	38770	8170	**2.37**	1.93	1.92	1.92	1.91	1.91	1.89	**1.69**	**78.93**
HUMGHCSA	66495	7495	**1.38**	1.31	1.10	1.10	1.03	1.01	0.91	**0.90**	**88.73**
HUMHDABCD	58864	11712	**2.19**	1.88	1.82	1.82	1.80	1.71	1.61	**1.59**	**80.10**
HUMHPRTB	56737	11501	**2.23**	1.91	1.85	1.84	1.82	1.76	1.71	**1.62**	**79.73**
MPOMTCG	186608	38497	**2.30**	1.94	1.91	1.91	1.89	1.88	1.88	**1.65**	**79.37**
MTPACG	100324	20506	**2.23**	1.88	1.86	1.86	1.86	1.84	1.80	**1.64**	**79.56**
HEHCMVCG	229354	46758	**2.32**	1.85	1.85	1.84	1.85	1.84	1.80	**1.63**	**79.61**
VACCG	191737	38621	**2.23**	1.76	1.76	1.76	1.76	1.76	1.70	**1.61**	**79.86**
Average bits per base (BPB)			**2.19**	1.78	1.74	1.74	1.72	1.70	1.63	**1.54**	**80.81**

TABLE V
Comparison of Compression Time of PRDNAC with Gzip

SEQUENCE	Size in Bytes	GZIP (S)	PRDNAC		
			Compress (S)	Decompress (S)	Total time taken (S)
CHMPXX	121024	0.105	0.063	0.031	0.094
CHNTXX	155844	0.135	0.032	0.047	0.079
HUMHBB	73323	---	0.047	0.031	0.078
HUMDYSTROP	38770	0.037	0.015	0.016	0.031
HUMGHCSA	66495	0.055	0.046	0.047	0.093
HUMHDABCD	58864	0.050	0.039	0.070	0.109
HUMHPRTB	56737	0.049	0.031	0.016	0.047
MPOMTCG	186608	0.100	0.078	0.094	0.172
MTPACG	100324	0.088	0.062	0.078	0.140
HEHCMVCG	229354	0.198	0.093	0.062	0.155
VACCG	191737	0.164	0.094	0.188	0.282

V. CONCLUSION

An improved compression ratio and lesser compression time have been achieved thru this improvised version of PRDNAC algorithm. Further development may be made by introducing any newer techniques drawn from various paradigms that have not been used. The authors hope that this algorithm will be appreciated by the scientific community of genomic data.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] PanneerArokiaraj, S. and Robert, L., "Pattern based DNA sequence Compressor", *Proc. of IEEE International conference on Computational Intelligence and Computing Research (ICCIC'12)*, Coimbatore, India. pp. 1-5, 2012.

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

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

[7] 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.
[8] Chen, X., Kwong, S. and Li, M., "A compression algorithm for DNA sequences and its applications in genome comparison", *The 10th workshop on Genome Informatics (GIW-99)*, pp.51–61, Tokyo, Japan, 1999.
[9] Williems, Shtarkov and Tjalkens, "The context tree-weighting method: Basic properties", *IEEE Trans. Info. Theory*, pp.653-664, 1995.
[10] Matsumoto, T., Sadakane, K., Okazaki, T. and Imai, H., "Implementing the context tree weighting method by using conditional probabilities", *Proc. of 22nd Symposium on Information Theory and its Applications*, pp. 673–676, SITA, December 1999.
[11] Chen, X., Li, M., Ma, B. and Tromp, J., "DNACompress: Fast and effective DNA sequence compression", *Bioinformatics*, Vol. 18(12), pp. 1696–1698, 2002.
[12] Behzadi, B. and Le Fessant, F., "DNA Compression Challenge Revisited", *Symposium on Combinatorial Pattern Matching (CPM2005)*, pp.190-200, June 2005.
[13] Tabus, Korodi and Rissanen, "DNA sequence compression using the normalized maximum likelihood model for discrete regression", *DCC*, p253, 2003.
[14] Koradi and Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression", *ACM Trans. Info. Systems*, Vol. 23(1), pp.3-34, 2005.
[15] Minh Due Cao, Dix and Lloyed Allison, "A simple statistical algorithm for biological sequence compression", *In Proc. Data Compression Conf. (DCC-07)*, Snowbird, UT. pp. 43-52, March, 2007.
[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] Soliman, T., "A Lossless Compression Algorithm for DNA sequences", *International Journal of Bioinformatics and Applications*, Vol. 5(6), pp. 593, 2009.
[18] 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.
[19] Jie Liu, Sheng Bao, Zhiqiang Jing and Shi Chen," A fixed length coding algorithm for DNA sequence compression", *Journal of Bioinformatics, Vol (0)*, pp.1-3,2005.