ISSN:0975-9646

Squeezing the DNA Sequences with Pattern Recognition Techniques in Multi-Processing Environment

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

[1]*Department of Computer Science, Periyar EVRCollege, Trichy, TN, India*
[2]*CS & Info. System Department, College of Science (Al-Qwaiya), Shaqra University, KSA*

Abstract—**This paper improves an existing work of compressing DNA sequences. Previously the compression was achieved with two processors; in this paper, 4-core and 8-core capacity is used and this was done using ParPro simulator. The improvement of compression of Genome sequences which are sourced from standard DNA databanks are tabulated and compared with existing compressors. Basic technique of compression exploits concepts drawn from Pattern Recognition techniques. Usage of parallelism improved compression ratio and time appreciably.**

Keywords—**Compression Ratio, DNA Sequence Compression, ParPro, PPRDNAC, Time Complexity.**

I. INTRODUCTION

The understanding of Deoxyribonucleic acid's structure and its functions from the last century is undeniably the proficient comprehension of life and of its evolution. Since then, the use of DNA in genetic engineering, forensics and anthropology applications has been extensive. The activities of human specific biological activities are controlled and regulated with the billions of individual cells in the body [1]. Watson and Crick discovered the structure of DNA in the double helix form held together by hydrogen bonds. Primarily, it had to be understood that four nucleotide bases existed: adenine (A), guanine (G), cytosine (C), and thiamine (T). Interestingly the sequences of the nucleotides were exclusively bonded to be in pairs of A-T, T-A, C-G, G-C. And this discovery has opened the door to the belief that DNA was indeed capable of enough structural variety to serve as the molecule of heredity [2]. An addition to the family of nucleotide has also been realized with the recently identified base called 'N'.

DNA sequences contain long-term repetitions in which the subsequences are not random but are similar to each other. The properties mentioned henceforth have been identified in many sequences and have formed the basis for all DNA compression algorithms. DNA sequences contain repeated substrings that are often longer than linguistic texts. DNA sequences contain repeated palindromes. DNA sequences contain repeated reverse complements [3].

DNA data maintained in GenBank (National Center for Biotechnology Information (NCBI) Genetic Bank), EMBL (European Molecular Biology Laboratory), and DDBJ (DNA Databank of Japan) are the publicly accessible for research and education.

Compression is possible and can be achieved if there exist similarities in the DNA sequences. Data compression in the area of storage management plays a key role. Data Compression algorithms may be classified as dictionary based or statistical based. The concepts of pattern recognition methods help us improve compression ratio and the introduction of parallelism enhance the quality of compression algorithms to achieve less processing time.

To achieve high throughput along with better compression ratio is a challenging problem. In this research work, better compression ratio at high throughput with the use of parallelism in compressing the large sequences is achieved. The researching community working on genomic data, data compression algorithms and bioinformatics community would benefit at large from this research.

II. REVIEW OF LITERATURE

Researchers down the ages have used generic approaches on the natural representation of DNA sequence, as a string of characters. The compressibility of DNA sequence can take advantage of certain biological characteristics, such as, repeat content and relationship to existing sequence.

Whenever repetitive calculations on a vast data occur, Parallel Processing is resorted to increase the computation speed through concurrency [4, 5]. Researchers have developed various algorithms for compressing the sequences of DNA and brought about better compression ratios and reduced computation time appreciably. Grumbach and Tahi [6, 7] proposed DNA Compressors, to detect the exact repeats and complementary palindromes in the sequence of DNA, namely Biocompress and Biocompress2.

DNA Compressor devised using Substitution was proposed by Chen et al. [8], that is widely used by the scientific community from its inception, to compress and handle approximate repeats. Later Chen et al. [9] constructed a compressor, with the Lempel and Ziv compression scheme and a software tool Pattern Hunter, called DNA Compress to identify all approximate repeats and complementary palindromes.

Hsiao Ping et al. [10] proposed PISD (Parallel and Incremental Signature Discovery algorithm) which is an enhanced version of existing signature discovery algorithms. Kamta Nath Mishra et al. [11] proposed DNASC, an algorithm to squeeze DNA sequences online. Satyanvesh et

al. [12] proposed a compressor using Multicore and Graphic Processing Units (GPUs) with high throughput, namely GenCodex.

III. METHODOLOGY

The process of compaction is a dire need as huge volume of space is required for the nucleotides of DNA {A, C, G, T, N} using binary digits. Various efficient compaction techniques, designed for compression of normal texts, would turn out to be ineffective and impractical while employed in the compaction of increasing DNA sequence length. Nevertheless, the presence of regularity of patterns exhibits the compression likelihood in the DNA sequences. An Improvised Pattern Recognition based DNA Sequence Compressor (IPRDNAC) [13] algorithm is supplemented with parallelism to reduce the computation time. The parallelization of the DNA sequence compression as shown in Figure 1 involves number of processors waiting for the task. The core of this paper is to complement the existing Parallelised Pattern Recognition based DNA sequence Compressor (PPRDNAC)[14] algorithm.

The power of computer can be propelled to farther extent by replacing single central processing unit with multiple processors as the compression of the sequences requires more computational power and time for the analysis and encoding of the data. Plying the sequence for long and repetitive patterns of the given data set is carried to compact the data. The repeating patterns are coded according to their uniqueness. With the given dataset, 8 processors from C_0 to C_7 recognize the patterns simultaneously. The job scheduler or central coordinator (in this case it is C_0) schedules and organizes the tasks among the processors.

The large and subsequent patterns are identified with the parallelized PRDNAC algorithm and are stored into the persistent storage. To illustrate, consider the following sequence having 128 bases.

AGTCAGTCCTGAAAGCACCTAAGCCGAATCCAN
TACNTACCCGTCCGTANTTTTTAATTTTTNACCGTT
GCCTCCACTGACTGACGAACGTNCGAACTGATNGT
ATNGCTAAATNGCTAAAATCGNTN.

The patterns that are deduced with the processors C_0 through C_7 at time slice t_0 are TTTTT, AGTC, AAGC, TCCA, NTAC, CCGT, AATC, CCT, GCC, ANT, CGA, AAA, CCC, TTT, GGG, GTA, AN, AA, CG, NG, TN, GT, AT. The discovered patterns are yielded to the central coordinator C_0 which in turn is handed over to the Improvised PRDNAC algorithm for compression.

Steps involved in Parallelizing the Pattern Recognition based DNA Sequence Compressor with 8 processors:

1. The sequence with 'n' bases is supplied to all processors and the central coordinator is rendered the scheduled tasks S_i to be performed at each time interval t_j by all processors.
2. Extract the patterns of varying sizes simultaneously.
3. The process is iterated till the completion of tasks.
4. Subsequently, employ the Improvised PRDNAC algorithm for the formation of compressed file.

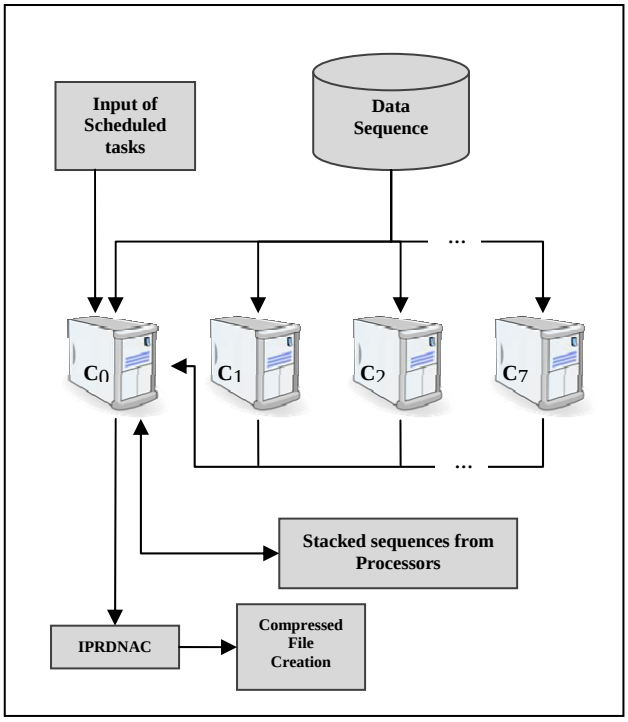

Fig 1. Model of Improvised PPRDNAC with 8 Processors

The following is the pseudo code to compress the DNA sequence using Improvised PRDNAC algorithm.

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.
   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.

The components of the compressed file are organized as follows: a set of all blocks of bit patterns in a contiguous form representing the compressed and uncompressed patterns, variable length file header in coded form to represent the symbol table, size of the file header and the end-of-file marker.

IV. EXPERIMENTAL RESULTS

The compression ratio and time complexity are the two factors that decide the efficiency of any algorithm and serve as a Bench mark. The efficiency (measured in terms of BPB - Bits Per Base) of the proposed PPRDNAC algorithm is compared with the existing DNA Sequence Compressors such as BioCompress2, Genome Compress, DNAC, GeNML, DNASC and the general compression software WinRAR and is tabulated in Table 1.

The following eleven DNA sequences, namely, CHMPXX, CHNTXX, HUMDYSTROP, HUMGHCSA, HUMHDABCD, HUMHBB, HUMHPRTB, MPOMTCG, MTPACG, HEHCMVCG and VACCG, are used as standard datasets in this work. From Table 1, it is witnessed that PPRDNAC excels in achieving good compression ratio than the listed compressors. The amount of storage space saved by PPRDNAC algorithm ranges from 78.93% to 88.73% with a mean of 80.81%.

The introduction of parallelism with ParPro simulator[15] having processors viz. 2/4/8 in identifying the patterns through the pattern recognition process in PPRDNAC has an appreciable improvement with respect to time consumption in compressing the DNA sequences as shown in Table 2. The decompression process of PPRDNAC algorithm follows the procedure of PRDNAC algorithm, due to which the time span remains the same, even though the number of processors is increased.

TABLE 1-EFFICIENCY COMPARISON OF PPRDNAC 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	PPRDNAC	Percentage of Space Saved by PPRDNAC
			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 2 THROUGHPUT COMPARISON OF PPRDNAC OVER PRDNAC (TIME IN SECONDS)

SEQUENCE	Size in Bytes	Improvised PRDNAC		PPRDNAC (Using 2 Processors)		PPRDNAC (Using 4 Processors)		PPRDNAC (Using 8 Processors)
		Compress (S)	Decompress (S)	Compress (S)	% age of Time improvement over PRDNAC	Compress (S)	% age of Time improvement over PRDNAC	Compress (S)
CHMPXX	121024	0.063	0.031	0.035	44.44	0.027	57.14	0.055
CHNTXX	155844	0.032	0.047	0.027	15.63	0.019	40.63	0.037
HUMHBB	73323	0.047	0.031	0.025	46.81	0.021	55.32	0.045
HUMDYSTROP	38770	0.015	0.016	0.01	33.33	0.009	40.00	0.02
HUMGHCSA	66495	0.046	0.047	0.031	32.61	0.029	36.96	0.042
HUMHDABCD	58864	0.039	0.07	0.021	46.15	0.017	56.41	0.037
HUMHPRTB	56737	0.031	0.016	0.018	41.94	0.015	51.61	0.028
MPOMTCG	186608	0.078	0.094	0.047	39.74	0.033	57.69	0.081
MTPACG	100324	0.062	0.078	0.035	43.55	0.029	53.23	0.067
HEHCMVCG	229354	0.093	0.062	0.052	44.09	0.043	53.76	0.088
VACCG	191737	0.094	0.108	0.061	35.11	0.055	41.49	0.091

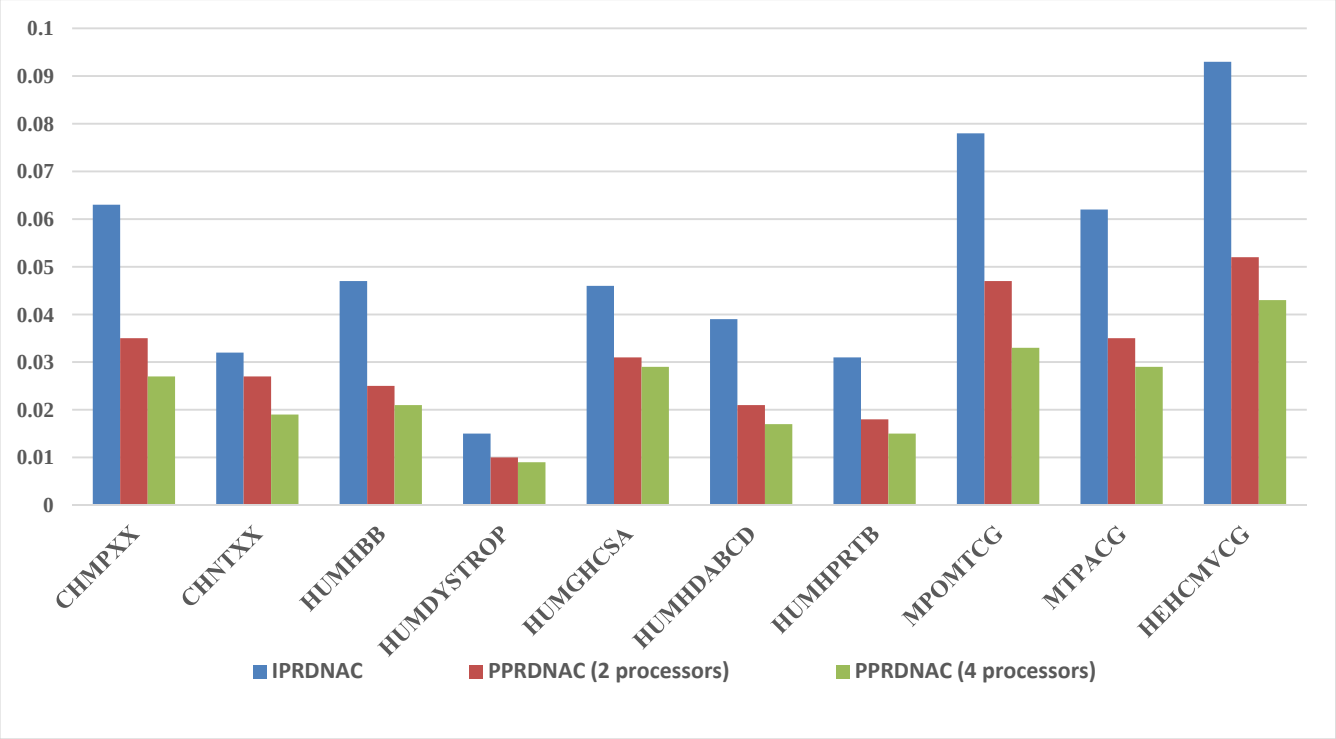

Fig. 2. Chart Showing the Efficiency of PPRDNAC over PRDNAC.

On the other hand, the percentage of improvement in time consumption achieved by PPRDNAC over PRDNAC with 2 processors has a minimum of 32.61 and a maximum of 46.81 except for the CHNTXX sequence. Upon using 4 processors in PPRDNAC, the improvement ranges between 40% and 57.69%. The results in Table 2 are represented graphically in Figure 2. While engaging 8 processors, the percentage of time consumed by PPRDNAC has no significant improvement over the improvised PRDNAC. Since, much of the time is wasted in scheduling and coordinating the processors.

V. Conclusion

Compression of DNA sequences exploiting multiprocessors was discussed in this paper. There is substantial gain in compression ratio and time while employing multiprocessors. Results are tabulated and compared with existing, popular compressors. This work may be further done by employing any improved techniques drawn from PR paradigms or from any other paradigms and also by employing Multi-core programming techniques.

References

[1]. Calladine, C. R. and Horace A. Drew. Understanding DNA: The Molecule and How It Works. San Diego: Academic Press, 1997.
[2]. Didier G. Arques and Christian J. Michel, “Analytical Expression of the Purine/Pyrimidine Autocorrelation Function after and before Random Mutations”, Mathematical Bio-Sciences 123:103-125, Elsevier Science Inc., 1994.
[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]. Michael J. Quinn, “Parallel Computing, Theory and Practice”, McGraw Hill International Edn. Singapore, 1994.
[5]. Ananth Grama, George Karypis, Vipin Kumar, Anshul Gupta, “Introduction to Parallel Computing”, 2nd Edition, Addison Wesley, Pearson Education, 2003.
[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]. 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]. Chen, X., Li, M., Ma, B. and Tromp, J., “DNA Compress: Fast and effective DNA sequence compression”, Bioinformatics, Vol. 18(12), pp. 1696–1698, 2002.
[10]. Hsiao Ping Lee, Tzu-Fang Sheu and Chuan Yi Tang, “A parallel and incremental algorithm for efficient unique signature discovery on DNA databases”, Biomed Central, 2010, http://www.biomedcentral.com/1471-2105/11/132.
[11]. Kamnath 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.
[12]. Satyanvesh D, Kaliuday Balleda, Ajith Padyana and Baruah P K, “GenCodex – A novel algorithm for compressing DNA sequences on Multicores and GPUs”, http://www.hipc.org/hipc2012/documents/SRSPapers/Paper%2037.pdf.
[13]. Panneer Arokiaraj, S. and Robert, L., “Improvised Pattern Recognition based DNA sequence Compressor”, International Journal of Engineering and Technology, Vol. 5(6), pp. 5017-5022, Dec 2013-Jan 2014.
[14]. Panneer Arokiaraj, S. and Robert, L., “Parallelised Pattern Recognition based DNA sequence Compressor”, Proc. of WCCCT 2014, IEEE International Conference,Trichy, India, pp. 13-19, March, 2014.
[15]. Ravi, T.N., “ParPro Scheduler - A Simulator for Parallel Job Scheduler”, Thesis on “A Study on Job Scheduling Strategies for Parallel Systems”, pp. 105, 2010.