International Journal of Advanced Trends in Computer Applications
www.ijatca.com

Multi step DNA Sequence Compression & Encryption

***Syed Mahamud Hossein[1,2], Pradeep Kumar Das Mohapatra[1], Debashis De[2]**
[1,2]Research Scholar, Vidyasagar University ,Midnapur-721102
[1]Department of Microbiology, Vidyasagar University,Midnapur-721102, West Bengal
[2]Department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, BF-142, Sector-I, Kolkata-700064,India.
[1,2*]mahamud123@gmail.com, [1]pkdmvu@gmail.com, [2]dr.debashis.de@gmail.com

Abstract: *The preponderance of short repeating patterns is an important phenomenon in biological sequences. Here present Compression algorithm, which data compresses by searching exact repeat, genetic palindrome and palindrome (RGP^2) substring substitution and create a Library file. The output of RGP^2 again compressed by Huffman's algorithm. It can provide the data security, by using ASCII code, on line Library file acting as a signature and Huffman's tree on at a particular level on the basic of a key tree node. Over all compression rate is 2.263592 bit per base. The algorithm can approach a moderate compression rate, provide strong data security, the running time is very few second and the complexity is O(n^2).*
.

Keywords: *DNA Sequence, Huffman Code, Compression Rate, Node, Encode, Decode, Lossless Compression, Repeat, Genetic Palindrome, Palindrome, Substitution and Encryption.*

Abbreviation: *RGP^2 -Repeat, Genetic Palindrome and Palindrome.*

I. INTRODUCTION

The DNA databases are too large [1-8], complex, must contain some logical organization [9-10], hence data structure to store, access and process this data efficiently is a difficult & very challenging task [11-12]. So it needs an efficient compression algorithm to store these huge mass of DNA data. The available compression algorithm [13-14] cannot compress the genome sequences well because the regularities in DNA sequences are evasive [15]. The two bit encoding is efficient if the bases are randomly distributed in the sequence, but the life of living organism is non-random have some limitations [15]. Huffman's code both in the static and adaptive model are not applied on DNA sequence well because they contain only four different occurrence [11,15]. The phenomenal characteristics of genomic data have contain so many repeats (e.g. ATGC) within a given DNA sequence [11]. The DNA sequence have some special structures [11,16-17], researchers are kept in mind and develope several DNA compression algorithms. This DNA sequence Compression algorithm achieves a moderate compression ratio and runs significantly faster than any existing compression program on benchmark DNA sequences. This algorithm consists of two phases: i) find all exact RGP^2 and ii) encode RGP^2 and non-match regions. This algorithm developed on the basis of fast and sensitive homology search [18], as our exact RGP^2 search engine. This RGP^2 substring creates a dynamic library file and place ASCII character in appropriate places on source file.

The RGP^2 technique convert the DNA sequence into 256 ASCII characters with unmatch a,t,g and c, in that situation the Huffman's algorithm is easily applied. Also develop related other supporting algorithm are string matching, string orientation changing and file size calculating etc.

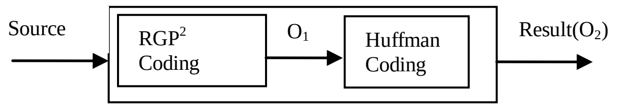
This technique can provide two phase information securities. In phase one, the output contain 256 characters including a, t, g & c, so, the output file is secure than input file. In phase two data encryption purpose swapping of the branches in the Huffman's tree on at a particular level on the basic of a key and decode the encoded symbols using the modified Huffman's tree which are specified in scheme I and II. In scheme-I apply swapping method on two nodes at specified level on Huffman's tree, and in scheme-II perform swapping method between two specified nodes at different level on Huffman's tree [19-20].

Now discuss details of the algorithm, provide experimental results and performance comparison of our result with other exiting compression results [21-22].

In this paper, if not otherwise mentioned, consider lower case letters u, v, to denote finite strings over the alphabet {a, c, g, t}, |u| denotes the length of u, the number of characters in u. u_i is the i-th character of u. $u_{i:j}$ is the substring of u from position i to position j. The first character of u is u_1. Thus u = $u_{1:|u|-1}$, where $u_{i:j}$ represent the original substring and |v| denotes the length of v, the number of characters in v. v_i is the i-th character of v. $v_{i:j}$ is the another substring of v from position i to position j. The first character of v is v_1. Thus v = $v_{1:|v|-1}$. $u_{i:j}$ match with $v_{i:j.}$ The minimum different between u-v is of substring length. The $v_{i:j}$ represent the repeat, genetic palindrome and palindrome substring .The match found if $u_{i:j=}$ $v_{i:j}$ and count exact maximum repeat, genetic palindrome and genetic palindrome of $u_{i:j..}$ We use ε to denote empty string and ε=0.

The multi step algorithm as a two step procedure in which first step consists of RGP^2 Coding and result in ASCII form. The second step consists of Huffman's Algorithm which gives the final result.

The procedure for Multi step DNA Sequences Compression & Encryption is



Define as - Let, s be source file to be coded.
Step 1: $O_1 \leftarrow RGP^2$ Coding(s)
Step 2: $O_2 \leftarrow$ Huffman Coding(O_1)

II. METHODS

2.1: File format: File type is text file and blank space ahead the end of file. The output file also text file, contains the information of both unmatch four base pair and coded values.

2.2: Generating the substring from input sequence

a t g g t a g t a a t gtacatg …… ...n_n

It is clear that for i^{th} substring W_i .
i, is the starting position of the substring and.
j= (i-1) + l, is end position of the substring; where l is the substring length.

The substring length is less than 3(three) has no importance in matching context therefore we consider the substring size in the range: 3 <=1 <= n.

Therefore range for I and j are as 1<=i<=n-1+1 and 1<= j<=n respectively.

***2.3: Searching for exact* RGP^2**

Let a string S over the DNA alphabet {a, c, g, t}. An exact RGP^2 is a substring, convert it into another string *s by* edit operations (repeat, genetic palindrome and palindrome, insertion). Now encode those substring match apropos maximum that provide profits on overall compression.

This method of compression is as below

1. Run the program and output all exact repeat, genetic palindrome and palindrome into a list s in the order of descending scores;
2. Extract match of repeat, genetic palindrome and palindrome *(r)* with highest score from list s, then replace all r by corresponding ASCII code into another intermediate list o and place r in library file.
   Where r is repeat, genetic palindrome & palindrome.
3. Process each repeat, genetic palindrome and palindrome in s so that there's no overlap with the extracted repeat, genetic palindrome and palindrome *r*;
4. Goto step 2 if the highest score of repeat, genetic palindrome and palindrome in s is still higher than a pre-defined threshold; otherwise exit.

2.4 Encoding RGP^2

An exact RGP^2 can be presented as two kinds of triples. first is (l, p), where l means the repeat, genetic palindrome and palindrome substring length and p show the starting positions of the substrings of repeat, genetic palindrome and palindrome, respectively. Second replace this operation is expressed as (r, p, char) which means replacing the exact repeat, genetic palindrome & palindrome substring at position p by ASCII character char.

2.5: Decoding

Decoding time, first require on line Library file, which was created at the time of encoding the input file.

On this particular value, the encoded input string is decoded and produces the output original file.

2.6: Algorithms

2.6:1: Encoding algorithm for RGP^2

1. Check for replaced character, if found just in shift in right direct ran.
2. Replace the first three consecutive replaceable symbol by the available special symbol in sequential order.
3. Check for the repeat, genetic palindrome and palindrome for the rest of the part of the string it repeat found replace it by the symbol used for the replacement of the first three symbol for

genetic palindrome and palindrome respectively use the equivalent character of additive ASCII value 72 and 144 respectively.

4. During each pass place only one entry in the library file against the original replaceable characters with the replaced one Rest, means genetic palindrome and palindrome can be calculated during replacement by adding 72 and 144 respectively.
5. Continue step 1 to 4 until no three consecutive replaceable symbol exit.
6. stop.

2.6:2: Decoding algorithm RGP^2

1. Extract the character
2. Check if it is within 'a', 't', 'g', 'c' just directly put if not among those character replace by equivalent umbination reading from 'a', 't', 'g', 'c' by checking it with all replace character entry from library file .
3. If direct match replace exactly with the entrees available in the library else replace by genetic palindrome or palindrome of that if match found with the 72 and 144 adetive value ASCII character of the give in library.
4. Continue until full string lossy either of 'a', 't', 'g' and 'c'.

2.6.3 Algorithm for Scheme-I of Huffman's

This algorithm recursively find a weighted binary tree with n given weights $w_1, w_2, \ldots w_n$. (Here weights mean frequency of n characters in text). LEVEL is the input where the tree is altered.

1. Arrange the weights in increasing weights.
2. Construct two leaf vertices with minimum weights, say w_i and w_j in the given weight sequence and parent vertex of weight $w_i + w_j$.
3. Rearrange remaining weights (excluding w_i and w_j but including parent vertex of weight $w_i + w_j$) in increasing order.
4. Repeat step 2 until no weight remains.
5. Find out left most node and right most node at specified LEVEL and interchange their position with respect to their parent node.
6. To find out code for each given weights (i.e. frequency of characters) traversing tree from root assign 0 when traverse left of each node & 1 when traverse right of each node.

2.6.4 Algorithm for scheme-II of Huffman's

This algorithm recursively find a weighted binary tree with n given weights $w_1, w_2, \ldots w_n$. (Here weights mean frequency of n characters in text). LEVEL is the input where the tree is altered.

1. Arrange the weights in increasing weights.
2. Construct two leaf vertices with minimum weights, say w_i and w_j in the given weight sequence and parent vertex of weight $w_i + w_j$.
3. Rearrange remaining weights (excluding w_i and w_j but including parent vertex of weight $w_i + w_j$) in increasing order.
4. Repeat step 2 until no weight remains.
5. Find out two nodes at specified LEVEL by binary digits and interchange their position with respect to their parent node.
6. To find out code for each given weights (i.e. frequency of characters) traversing tree from root assign 0 when traverse left of each node & 1 when traverse right of each node.

III. ALGORITHM EVALUATION

3.1: Accuracy

The DNA sequence storage, accuracy must be taken firstly in that even a single base mutation, insertion, deletion would result in huge change of phenotype. It is not tolerable that any mistake exists either in compression or in decompression. For accuracy purpose develop string matching algorithm to check one by one character.

3.2: Efficiency

This algorithm can compress original file from substring length (l) into 1 character for any DNA segment, and destination file uses less ASCII character to represent successive DNA bases than source file.

3.3: Space Occupation

This algorithm reads characters from source file and writes them immediately into destination file. It costs very small memory space to store only a few characters. The space occupation is in constant level.

.

IV. EXPERIMENTAL RESULTS

This technique test on standard benchmark data used in[12,23]. The definition of the compression ratio [29]; $1-(|O|/2|I|)$,

where $|I|$ is number of bases in the input DNA sequence and $|O|$ is the length (number of bits) of the output sequence,

the compression rate[29], which is defined as $(|O|/|I|)$,

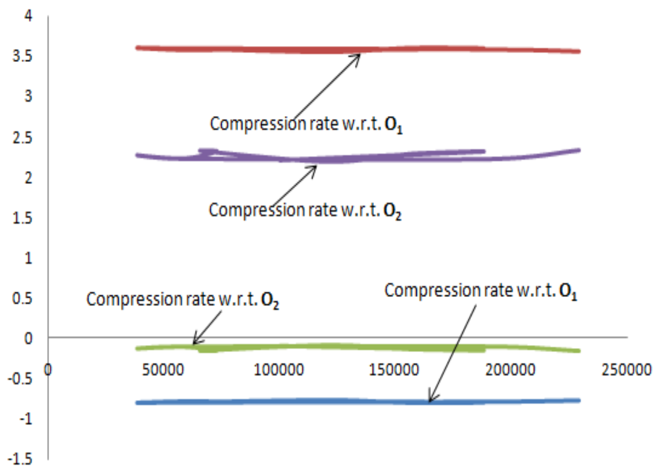
where $|I|$ is number of bases in the input DNA sequence and $|O|$ is the length (number of bits) of the output sequence and improvement[29] over *RGP*^{2};

$(\text{Ratio\_of\_O}_1 - \text{Ratio\_of\_O}_2)/\text{Ratio\_of\_O}_1)*100$.

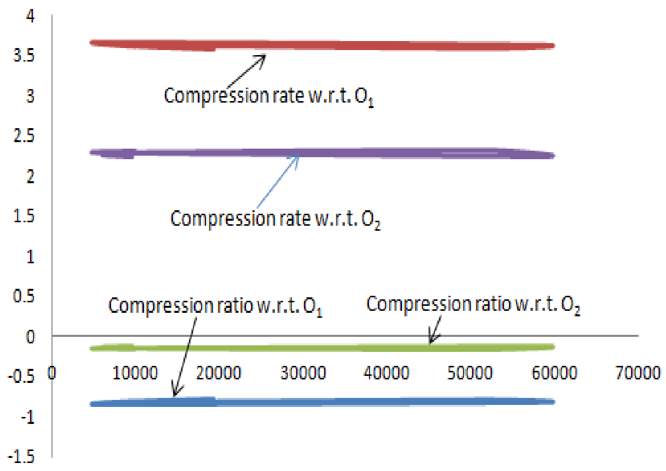
The compression ratio and rate shown in the table-I, improvement result also show in the same table. Our result compare with Table-II result, also last Colum showing the improvement over 'gzip'.

Data set	Sequence Name	Base pair/File size	Result of RGP^2(O_1)		Our result(O_2)		Improvement
			Compression ratio	Compression rate(bits /base)	Compression ratio	Compression rate(bits /base)	
Data set-I	MTPACGA	100314	-0.784636	3.569272	-0.104253	2.208505	36.82 %
	MPOMTCG	186608	-0.797222	3.594444	-0.154677	2.309354	
	CHNTXX	155844	-0.800621	3.601242	-0.14109	2.28218	
	CHMPXX	121024	-0.77697	3.55394	-0.096477	2.192953	
	HUMGHCSA	66495	-0.795443	3.590887	-0.159365	2.318731	
	HUMHBB	73308	-0.791564	3.583129	-0.157309	2.314618	
	HUMHDABCD	58864	-0.788801	3.577603	-0.113822	2.227643	
	HUMDYSTROP	38770	-0.802218	3.604436	-0.134382	2.268765	
	HUMHPRTB	56737	-0.798403	3.596806	-0.1137	2.2274	
	VACCG	191737	-0.791642	3.583283	-0.111418	2.222836	
	HEHCMVCG	229354	-0.780061	3.560121	-0.163267	2.326535	
	Average	----	---	3.583196	---	2.263592	
Data set-II	atatsgs	9647	-0.820669	3.641339	-0.117445	2.234891	37.19%
	atef1a23	6022	-0.831949	3.663899	-0.131849	2.263699	
	atrdnaf	10014	-0.808667	3.617335	-0.153984	2.307968	
	atrdnai	5287	-0.836202	3.672404	-0.146964	2.293928	
	celk07e12	58949	-0.816010	3.632020	-0.120290	2.240580	
	hsg6pdgen	52173	-0.795181	3.590362	-0.160216	2.320433	
	mmzp3g	10833	-0.815563	3.631127	-0.146866	2.293732	
	xlxfg512	19338	-0.787568	3.575137	-0.136001	2.272003	
	Average			3.627953		2.278404	

Table I: Each row displays the compression ratio and rate for each DNA sequence



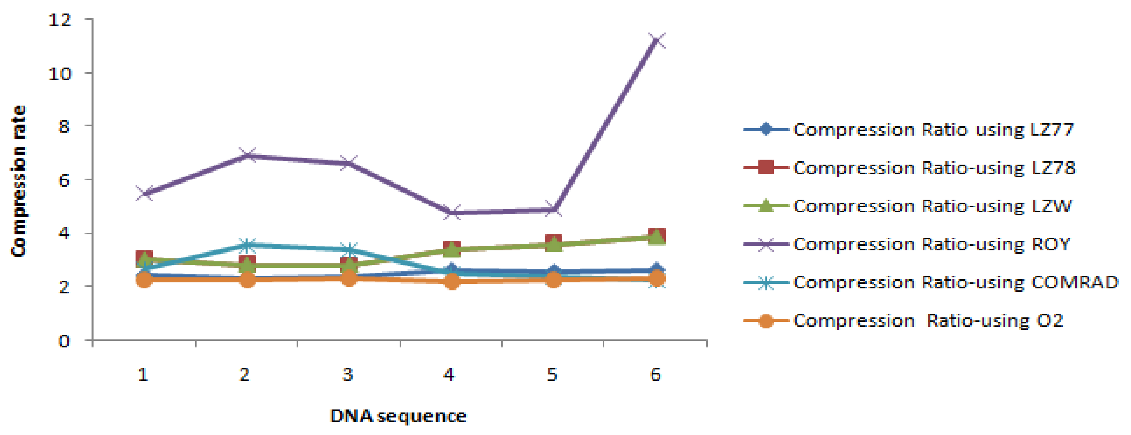
Graph I: Compression ratio & rate w.r.t.O_1 and O_2 (data set-I)



Graph II: Compression ratio & rate w.r.t.O_1 and O_2 (data set-II)

Sequence Name	Base pair/ File size	gzip	Our result(O_2)	Improvement	Sequence Name	Base pair/ File size	LZ77	LZ78	LZW	ROY	OMRAD	O_2
		Compression Ratio	Compression Ratio				Compression Ratio	Compression Ratio	Compression Ratio	Compression Ratio	Compression Ratio	Compression Ratio
MTPACGA	100314	2.2922	2.2085		atatsgs	9647	2.428981	3.023654	3.023654	5.503137	2.685691	2.234891
MPOMTCG	186608	2.3291	2.3093		atef1a23	6022	2.334222	2.830354	2.830354	6.929804	3.552802	2.263699
CHNTXX	155844	2.3349	2.2821		atrdnai	5287	2.392984	2.817479	2.817479	6.617621	3.382597	2.293928
CHMPXX	121024	2.2821	2.1929		chmpxx	15180	2.620404	.393696	3.393696	4.781102	2.484452	2.192953
HUMGHCSA	66495	2.0655	2.3187		humdystrop	38770	2.567358	3.604835	3.604835	4.924425	2.387021	2.268765
HUMHBB	73308	-----	2.3146		humghcsa	66495	2.628261	3.868575	3.868575	11.22847	2.265125	2.318731
HUMHDABCD	58864	2.2399	2.2276	0.51%	Average		2.495368	3.256431	3.256431	6.664093	2.792948	2.26216
HUMDYSTROP	38770	2.3633	2.2687		Improvement							
HUMHPRTB	56737	2.2670	2.2274									
VACCG	191737	2.2520	2.2228									
HEHCMVCG	229354	2.3278	2.3265									
Average	----	2.2753	2.2635									
atatsgs	9647	2.1702	2.234891									
atef1a23	6022	2.0379	2.263699									
atrdnaf	10014	2.2784	2.307968									
atrdnai	5287	1.8846	2.293928									
celk07e12	58949	--	2.240580									
hsg6pdgen	52173	2.2444	2.320433									
mmzp3g	10833	2.3225	2.293732									
xlxfg512	19338	1.8310	2.272003									
			2.2784									

Table II: Comparison of Compression rate



Graph III: Line chart shows the comparison of compression ratio of above algorithm in table1I

V. RESULT DISCUSSION

In Table 1 showing the result of this two algorithms for data set –I and II, drawn the corresponding graphical representation in graph I & II. Also in table 2 showing the earlier result of gzip, LZ77, LZ78, LZW, ROY etc and compared this result with others and improved showing the same table in last Colum. The comparative result also graphically shown in graph III. From this experiment, conclude that internal repeat, genetic palindrome and palindrome matching patter are same in all type of sources and Look up Table plays a key role in finding similarities or regularities in DNA sequences. Output file are encrypted by the Human's key, so the compress data is very important for data protection over transmission point of view. These techniques provide the high security to protect nucleotide sequence in a particular source.

VI. CONCLUSION

This DNA compression algorithm overcomes the binary coding and Huffman's coding problem. This method is fails to achieve higher compression rate than others standard method, but it has provide very high information security.

Important observations are:

a) Repeat, genetic palindrome and palindrome substring length vary from 2 to 5 and no match found in case the substring length becoming six or more. The substring length, three is highly compressible over substring length of four or above.

b) Also library file of subsequence size 3 is the key role of codon table.

FUTURE WORK

Try to reduce the time complexity, improve compression rate and ratio.

ACKNOWLEDGEMENT

Above all, authors are grateful to all our colleagues for their valuable suggestion, moral support, interest and constructive criticism of this study. The author offer special thanks to Ph.D guides for helping in carrying out the research work.

REFERENCES

[1]. International nucleotide sequence database collaboration, (2013),[Online]. Available: http://www.insdc.org.

[2]. Karsch-Mizrachi, I., Nakamura, Y., and Cochrane, G., 2012, The International Nucleotide Sequence Database Collaboration, Nucleic Acids Research, 40(1), 33–37.

[3]. Deorowicz, S., and Grabowski, S., 2011, Robust relative compression of genomes with random access, Bioinformatics, 27(21), 2979–2986.

[4]. Brooksbank, C., Cameron, G., and Thornton, J., 2010, The European Bioinformatics Institute's data resources, Nucleic Acids Research, vol. 38, 17-25.

[5]. Shumway, M., Cochrane, G., and Sugawara, H., 2010, Archiving next generation sequencing data, Nucleic Acids Research, vol. 38, 870-871.

[6]. Kapushesky, M., Emam, I., Holloway, E., et al. , 2010, Gene expression atlas at the European bioinformatics institute, Nucleic Acids Research, 38(1), 690-698.

[7]. Ahmed A., Hisham G., Moustafa G., et al., 2010, EGEPT: Monitoring Middle East Genomic Data, Proc., 5th Cairo International Biomedical Engineering Conf., Egypt, 133-137.

[8]. Korodi, G., Tabus, I., Rissanen, J., et al., 2007, DNA Sequence Compression Based on the normalized maximum likelihood model, Signal Processing Magazine, IEEE, 24(1), 47-53.

[9]. Mr Deepak Harbola1 et al. State of the art: DNA Compression Algorithms, International Journal of Advanced Research in Computer Science and Software Engineering, 2013, pp 397-400.

[10]. A. Postolico, et al., Eds., DNA Compression Challenge Revisited: A Dynamic Programming Approach, Lecture Notes in Computer Science, Island, Korea: Springer, 2005, vol. 3537, 190–200.

[11]. Nour S. Bakr1, Amr A. Sharawi, 'DNA Lossless Compression Algorithms: Review ', American Journal of Bioinformatics Research, 2013 pp 72-81

[12]. 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.

[13]. X. Chen, S. Kwong and M. Li, "A Compression Algorithm for DNA Sequences and its Applications in Genome Comparison,*Genome Informatics*, 10:52–61, 1999.

[14]. Bell, T.C., Cleary, J.G., and Witten, I.H., *Text Compression*, Prentice Hall, 1990.

[15]. Matsumoto, T., Sadakane, K., and Imai, H., 2000, Biological Sequence Compression Algorithms, Genome Informatics, 2000,pp 43–52.

[16]. Giancarlo, R., Scaturro, D., and Utro, F., 2009, Textual data compression in computational biology: a synopsis, Bioinformatics, 25(13), 1575–1586.

[17]. Nalbantoğlu, Ö. U., Russell, D.J., and Sayood, K., 2010, Data Compression Concepts and Algorithms and their Applications to Bioinformatics, Entropy, 12(1), 34-52.

[18]. Ma,B., Tromp,J. and Li,M. (2002) PatternHunter—faster and more sensitive homology search. Bioinformatics, 18, 440–445.1698

[19]. Syed Mahamud Hossein et al.A Compression & Encryption Algorithm on DNA Sequences Using Dynamic Look up Table and Modified Huffman Techniques, I.J. Information Technology and Computer Science, 2013, pp 39-61

[20]. Md. Syed Mahamud Hossein,A Compression and Encryption Algorithms on DNA Sequences using R^2CP and Modified Huffman Technique, International Journal of Computer Applications , 2012 ,pp 1-10

[21]. Dhajvir Singh Rai et al., Survey of Compression of DNA Sequence, *International Journal of Computer Applications, 2013, pp*- 52-58

[22]. Jie Liu et al., A Fixed-Length Coding Algorithm for DNA Sequence Compression(Draft,using Bioinformatics LATEX template), Bioinformatics,*2005,pp 1–3*

[23]. Xin Chen, San Kwong and Mine Li, "A Compression Algorithm for DNA Sequences Using Approximate Matching for Better Compression Ratio to Reveal the True Characteristics of DNA", IEEE Engineering in Medicine and Biology, 2001, pp 61-66