International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P), Volume-5, Issue-2, June 2016

Design & Implementation of Compression Algorithm for Nucleotide Sequence using Direct Coding and LZ77

Kumar Nishant , Pawan Kumar Mishra

***Abstract*— In this method we have decoded repeat sequence up to length 15 in 9 bits instead of 12 bits. If the repeat sequence length is less than 9 then the sequence will be coded in 6 bits but if the sequence length is greater than equal to 9 and less than 16 then it will be coded in 9 bits and so on. Here the method identifies repeat regions in the individual sequence and repeat regions are done by the implementation of compression algorithm for nucleotide sequence using direct coding and LZ77. To prove this method first we have to give its new algorithm and after completing the algorithm we have compress and de-compress to check the character and at last it achieves the compression ratio 1.6 which has shown in a result in the paper which is discuss below.**

***Index Terms*— DNA; Redundancy; Reference Base; Optimized Exact Repeat; Non-Repetition; LZ77 & Compression Ratio.**

I. INTRODUCTION

The genome database sizes are increasing annually with great speed. In the labs every day thousands of nucleotides are sequenced. From 1982 to the present, the number of bases in GenBank has doubled approximately every 18 months. It is found that in Dec 1982, the number of bases and the number of sequence records was 680338 and 606 respectively for GenBank and none for WGS (Whole Genome Shotgun) and with Release 129 in Apr 2002, the number of bases and the number of sequence records were 19072679701 and 16769983 respectively for GenBank and the WGS had 692266338 number of bases and 172768 number of sequences. Again in the Release 206 in Feb 2015, the number of bases and the number of sequence records were 187893826750 and 181336445 respectively for GenBank and the WGS had 873281414087 numbers of bases and 205465046 numbers of sequences [1].
The highly repetitive DNA sequences own some motivating properties [2], [3] which can be utilize to compress it. As DNA sequences it consists of four nucleotides bases A, C, G and T. In this A stands for Adenine, C stands for Cytosine,

G stands for Guanine and T stands for Thymine. A and G are the parts of Purines. C and T are the parts of Pyrimidines. The most common and is very simple form of DNA compression is just binary encode the DNA sequence bases using 2 bit for each nucleotides i.e. by replacing A/a, C/c, G/g and T/t with “00”, “01”, “10” and “11” chosen abruptly[4]. These behaviors are primary to substantial expansion in the size of DNA data sets, and are providing opportunities for novel compression techniques that take advantage of the characteristics of these new data. Our aspiration is to discover mechanisms for detecting this redundancy and use it in compression by searching optimal level of similarity within individual’s sequence.
The Study is organized that first we will discuss about the related work which tells about the related paper of the DNA sequences after that we have to discuss system overview in which it tells about LZ77 algorithm and the measure of compression and last it will discuss about the proposed method and its result.

Kumar Nishant, M.Tech CSE, Uttarakhand Technical University, Dehradun.
Pawan Kumar Mishra , Assistant Professor, in Uttarakhand Technical University, Dehradun.

II. LITERATURE SURVEY

In this chapter we will study about the specialized sequence compression algorithms. The dictionary based method like LZ77 is considered best for compressing DNA data so far. But the general text compression algorithms like Ziv-Lempel, prediction by partial matching, dynamic markov coding etc are not suitable for compressing the specialized sequences like DNA and RNA. This is because they consider the input text sequence as a plain English text. And hence increase the size of the resultant file rather than compressing it. The BioCompress and BioCompress2 are based on LZ77 and LZ78 algorithms [5-11].
Gregory Vey el at [5] has studied that differential direct coding: a compression algorithm for nucleotide sequence data. In this the Differential Direct Coding algorithm, a general-purpose nucleotide compression protocol that could be differentiated between sequence data and auxiliary data by supporting the inclusion of supplementary symbols that were not members of the set of expected nucleotide bases, thereby offering reconciliation between sequence-specific and general-purpose compression strategies. This algorithm permitted a sequence to contain a rich lexicon of auxiliary symbols that could represent wildcards, annotation data and special subsequences, such as functional domains or special repeats. In particular, the representation of special subsequences can be incorporated to provide structure-based coding that increases the overall degree of compression.

Banani Saha [6] want to say about the Genome Compress which is based on a navel algorithm for DNA compression. The genome of an organism contains all hereditary information encoded in DNA. So it is extremely important to sequence the genome which determines

how the organisms survive, develop and multiply. Since three decades, due to massive efforts on DNA sequencing, complete genome sequence of a large number of organisms including humans are now known and the genomic databases are growing exponentially with time. Also for the huge size of the genomes, an efficient algorithm is required to compress them. General text compression algorithms don't utilize the specific characteristics of a DNA sequence. DNA specific compression algorithms exploit the repetitiveness of bases in DNA sequences. A repetitive DNA sequence can be best compressed using dictionary based compression algorithm.

R.K.Bharti et al [7] has given the compression which studied about the Biological sequence Compression Based on Cross chromosomal properties using variable length LUT. In this method again a table was created, but the size of the table was variable now. It contains the table formed in fixed size LUT algorithm and in addition to this; we had some other combinations in the table, which may be a collection of more than 3 characters of DNA bases. Since the printable ASCII characters are limited and 64 out of them are already occupied. The remaining will be occupied by the one's which provide us maximum compression, out of all the other combinations of more than 3 base characters. Time complexity increases as we need to calculate, that which combinations should occupy the remaining special characters, on the basis of their frequency of repetition.

R.K.Bharti and R.K.Singh et al [8] has studied about Biological Sequences compression technique, which is based on complementary palindrome that uses variable Length Look up Table (LUT). With the progress of this technology, the size of genome database is increased by DNA sequencing. Data storage capacity had become an appreciable proportion of total capacity in the creation and analysis of DNA sequence database. The rate of increase in DNA sequencing was significantly outstripping the rate of increase in storage capacity. Efficient storage removed redundancy from the data being stored in the DNA molecule. Data compression removed redundancy used in data i.e. DNA molecule. In this paper we presented a compression algorithm based on the properties complementary palindrome of biological sequences. Our algorithm worked on both repetitive and non repetitive biological sequences. This algorithm achieved the best compression ratio for biological sequences for larger genome. It was very useful in sequence comparisons and multiple sequence alignment analysis. The simplicity and flexibility of our algorithm could make it valuable tool for biological sequence compression in clinical research.

Govind Prasad Arya et al [9] has solved a compression algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Look up Table (LUT). In this dissertation, the first chapter gave the basic description of the compression technique. Then we described that how it proved to be useful in data transfer, bandwidth utilization, etc and how was it mandatory. We also studied the two different types of compression techniques i.e. lossy and lossless techniques and also compared them on the basis on the basis of their area of application. We studied that how does compression worked and its different techniques and that there were two basic algorithms (encoding and decoding algorithms) involved, wherever the compression was concerned. We also studied the different techniques for measuring the compression rate followed by study of different Universal Data Compression Algorithms which are Ziv–Lempel algorithms, LZ77 algorithm, LZ78 algorithm, Prediction by partial matching algorithms, Dynamic Markov coding algorithm, Context tree weighting algorithm, switching method.

Deepak Harbola et al [10] has studied that the state of art: DNA compression algorithm. In this paper we discussed about the modern biological science produced the vast amounts of genomic progression data. This was igniting the needed for competent algorithms for sequence compression and investigation. Data solidity and the related techniques coming from information theory were often perceived as being of great interest for information communication and storage. In recent years, a substantial effort had been made for the application of textual data compression techniques to various computational biology tasks, ranging from storage and indexing of large datasets to comparison of genomic databases. In this paper we review the ways in which ideas and approaches fundamental to the theory and practice of data compression had been used in the area of bioinformatics. We looked at how basic theoretical ideas from data compression, such as Need of compression, lossy and lossless compression, how to measured compression ratio, how do can we compressed a biological sequence and why other universal text compression algorithms were not suitable for biological sequence. This article introduced several algorithms which were used in past twenty year for DNA sequence compression.

Subhankar Roy and Akash Bhagat et al [11] has studied about SBVRLDNACOMP: An Effective DNA Sequence Compression Algorithm. There were plenty specific types of data which were needed to compress for easy storage and to reduced overall retrieval times. Moreover, compressed sequence can be used to understand similarities between biological sequences. DNA data compression challenge had become a major task for many researchers for the last few years as a result of exponential increase of produced sequences in gene databases. In this research paper we had attempt to developed an algorithm by self-reference bases; namely Single Base Variable Repeat Length DNA Compression (SBVRLDNACOMP). There are a number of reference based compression methods but they were not satisfactory for forthcoming new species. SBVRLDNACOMP was an optimal solution of the result obtained from small to long, uniform identical and non-identical string of nucleotides checked in four different ways. Both exact repetitive and non-repetitive bases were compressed by SBVRLDNACOMP. The sound part of it was without any reference database SBVRLDNACOMP achieves 1.70 to 1.73 compression ratios _ after testing on ten benchmark DNA sequences. The compressed file can be further compressed with standard tools (such as WinZip or WinRar) but even without this SBVRLDNACOMP outperforms many standard DNA compression algorithms.

IV. SYSTEM OVERVIEW

A. ***LZ77 Algorithm:-*** We are going to describe the LZ77 algorithm[9] using the sample sequence abracadabra

at the beginning we choose two numbers: ls—the maximum length of the identical subsequences, and lb—the length of the buffer storing the past characters. Let us set ls = 4 and lb = 8. The LZ77 algorithm works on the buffer sequence b that is composed of lb previous and ls future symbols.There are no previous symbols at the beginning of the compression because the current position i in the x sequence equals 1. Therefore we initialize the buffer b with lb repetitions of the first symbol from the alphabet (a in our example), and the ls first symbols from the sequence. Now we find the longest prefix of the current part of component b(lb+1)..(lb+ls) (Underlined characters in column "Buffer"), in the buffer starting not further than at the lbth position. We find the prefix a of length ll = 1 starting at the 8th position in our example. Then we output a triple h8, 1, bi describing the repeated part of the sequence. The first element of the triple, 8, is a position where the identical subsequence starts, the second, 1, is the length of the subsequence and the third, b, is the character following the repeated sequence, xi+ll . Then the buffer is shifted ll +1 characters to the left, filled from the right with the component x(i+ls+1).(i+ls+1+ll), and the current position, i, is changed to i + ll + 1. The output is a sequence of triples; in our example: h8, 1, bi, h1, 0, ri, h6, 1, ci, h4, 1, di, h2, 4,_i. (The special character _denotes no character).The parameters lb and ls are much larger in real implementations, so we can find significantly longer identical components. Choosing lb and ls we have to remember that using large values entails the need to use much more space to store the triples. To achieve better compression, in modern versions of the LZ77 algorithm the sequence of triples is modeled by simple algorithms, and then encoded with the Huffman or arithmetic coder.

B. ***Measuring the Compression Ratio:-***
Compression rate[9] is the measurement of the reduction in size of the original file. There are four main methods of measuring the compression rate. The first one known as Bit per Byte or bpb refers to the replacement of one byte (particularly the collection of 8 bits) by less than 8 bits. It can be formulated as follows: (compressed length / original length) * 8. If a file of 800 bytes has been compressed to a file of 200 bytes, the compression will be - (200/800)*8=2 bpbThe second method is measuring of compression in terms of percentage, which can be formulated as (compressed length / original length) * 100. If a file of 800 bytes has been compressed to a file of 200 bytes, the compression will be - (200/800)*100= 25% of the original file.Third method can be representation in ratio form, which is (original size: compressed size). This is a general representation technique and is widely used. But it has low precision. i.e (4: 1) or (3:1). Bit per Char is another technique. It is same like bpb in some cases only and it cannot be used to compress binary files.

V. PREVIOUS WORK

In this related work we are going to discuss about work which is done in the previous thesis. All Genome compression algorithms utilize redundancy within the sequence, but vary greatly in the way they do so. In general, compression algorithms can be classified into Naive Bit Manipulation, Dictionary-based, Statistical and Referential Algorithms. They have formed nucleic acid sequences fragments of {A, C, G, T} and {K, M, R, S, W, Y, B, D, H, V, N} into 152=255 combinations and then converting them into 255 ASCII [12] characters out of 256. For repeat count in the later method they have used 9 characters from digit '1' to '9'. If repeat is greater than 9; then it recounts the repeats.
Here,

- A DNA sequences S
- Three bits coding rules { S_1,S_2,S_3,S_4,S_5,S_6,S_7, S_8 and S_9}
- Flag variable v for repeat B_0 and non-repeat B_1 segments
- Two bits coding rule (A=00, C=01, G=10, and T=11)
- Count C
- From sequence we know that 2=000, 3=001, 4=010, 5=011, 6=100, 7=101, 8=110 , 9=111

Step	Input Sequence AGTTTTTTTTTTTTC	Output Sequence
1	AGTTTTTTTTTTTTC	000
2	AGTTTTTTTTTTTTC	000010
3	AGTTTTTTTTTTTTC	000010111111
4	AGTTTTTTTTTTTTC	000010111111111001
5	AGTTTTTTTTTTTTC	0000101111111111001001

Table-Encoding Process

Due to this table we can see that here the value of input sequence T has repeated 12 times and the sequence of T has written in 12 bits.

Existing Algorithm:-

Step1: Store the current character of remaining S into R
Step2: Check the next character relative to R, store it to T
Step3: c _ 0
Step4: **while** *R* != null **do**
Step5: **if** *R* = *T* **then**
Step6: *v* = *B*0
Step7: **else**
Step8: *v* = *B*1
Step9: **end if**
Step10: The compressed codes are (*B*0*RS*1…*r*) or (*B*1*R*); (*RB*1…*rB*0) or (*RB*0); (*RB*1) or (*RB*0); (*B*0*R*) or (*B*1*R*1…*r*) or (*R*1… *n''*)
Step11: Update *c* according to coding Splice
Step12: **end while**
Step13: The total number of bits *c* and optimal method for a specific sequence is obtained Step14: Compressed by optimal method
Step15: Use LZ77 for final stage compression

VI. PROPOSED METHODOLOGY

In this method we have decoded repeat sequence up to length 15 in 9 bits instead of 12 bits. If the repeat sequence length is less than 9 then the sequence will be coded in 6 bits but if the sequence length is greater than equal to 9 and less

than 16 then it will be coded in 9 bits and so on which is shown in the table.
Here,

- A DNA sequences S
- Three bits coding rules { S_1,S_2,S_3,S_4,S_5,S_6,S_7, S_8 and S_9}
- Flag variable v for repeat B_1 and non-repeat B_0 segments
- Two bits coding rule (A=00, C=01, G=10, and T=11)
- Count C
- From sequence we know that 2=000, 3=001, 4=010, 5=011, 6=100, 7=101, 8=110 , 9=111

Proposed Algorithm:-

While (EOF)
Read a character R from remaining sequence S
Read one more character T to check repetition
IF R≠T then
Decode R in 3 bits as 0R, where R is Nucleotide base (A/a=00, C/c=01, G/g=10, T/t=11)
End IF
Else IF R=T then
Count number of repeats (C) of R in remaining sequence S
IF (2>C<=8) then
Decode R in 6 bits as 1RC, where C (2=000, 3=001, 4=010, 5=011, 6=100, 7=101, 8=110)
End IF
Else IF (16>C>8) then
C_1=C-8
Decode R in 9 bits as 1R111C_1, where C_1 (2=000, 3=001, 4=010, 5=011, 6=100, 7=101, 8=110)
End Else IF
End Else IF
End While
At last use LZ77 for final stage

Step	Input Sequence AGTTTTTTTTTTTC	Output Sequence
1	AGTTTTTTTTTTTC	000
2	AGTTTTTTTTTTTC	000010
3	AGTTTTTTTTTTTC	000010111111001
4	AGTTTTTTTTTTTC	000010111111111001001

Table- Proposed Process

VII. RESULTS

The following table shows the results of compression applied on different types of standard DNA sequences. As the results proved that on an average simple SBVRLDNACOMP: An Effective DNA Sequence Compression Algorithm (Existing algorithm) compresses the DNA sequences of 121024 Bytes into 25749 Bytes while our proposed algorithm Design & implementation of a compression algorithm for nucleotide sequence using direct coding and LZ77 (Proposed algorithm) compresses 121024 Bytes into 24228 Bytes.

S.N	Type of Sequence	Original Size in bytes	Existing Method	Proposed Method	Compression Ratio
1	Chmpxx	121024	25749	24228	1.6015
2	Chntxx	155844	33158	31199	1.6016
3	Humdystrop	38770	8252	7763	1.6019
4	Humghcsa	66495	14150	13316	1.602
5	Humhdabcd	58864	12527	11786	1.6018
6	Humhprtb	56737	12704	11360	1.6018
7	Mpomtcg	186608	39701	37352	1.6013
8	Mtpacg	100314	21344	20081	1.6015
9	Hemcmvcg	229354	48801	45910	1.6014
10	Vaccg	191737	407945	38381	1.6014
	Average				1.6016

Table.5.2 Final Result Analysis

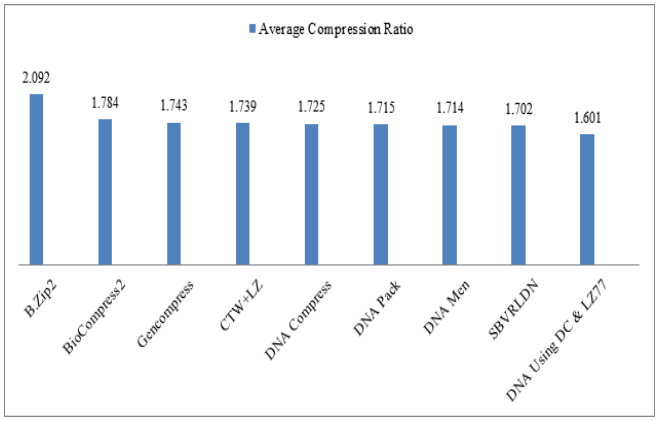

Fig 1: Average bits per base for DNA compression algorithms

VIII. CONCLUSION

In this chapter we decode a sequence greater than 8 lengths in 9 bits instead of 12 bits. Here the method identifies repeat regions in the individual sequence and repeat regions are done by the implementation of compression algorithm for nucleotide sequence using direct coding and LZ77. To improve this we have to give its new algorithm to make the bits more less than 9 bits which I had done in this paper. It should make to improve the compression ratio and to give a better result for the next any compression of DNA compression algorithm.
Although the proposed algorithm is providing better results than the existing one, but still it can be improved for the better compression & execution time.
In this we told about how to measure the compression, Now a days there are so many nucleotide or we can say that there are large number of nucleotide and we have to send from one place to another from mail or from cloud computing for this we have to compress more less bits. In this we have

learn about LZ77 algorithm and how to measure the compression to take out the result.

ACKNOWLEDGMENT

It gives me great sense of pleasure to present the thesis of M.Tech undertaken during M.Tech, 2nd year. I owe special date of gratitude to **Mr. Govind Prasad Arya (Assistant Professor),** Department of Computer Science & Engineering, Faculty of Technology at Shivalik College of Engineering, Dehradun for his special constant support and guidance throughout the course of work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me.

REFERENCES

[1] Kircher M and Kelso J, High-throughput DNA sequencing – concepts and limitations Bioessays, Wiley Online Library, 32, 6, 524–536, 2010.
[2] Paula WCP, An Approach to Multiple DNA Sequences Compression-A thesis submitted in partial fulfillment of the requirements for the Degree of Master of Philosophy, The Hong Kong Polytechnic University, Hong Kong, 2009.
[3] Shiu HJ, Ng KL, Fang JF, Lee RCT and Huang CH, 2010, Data hiding methods based upon DNA sequences, Information Sciences, Elsevier, 180, 2196–2208.
[4] Mridula TV and Samuel P, Lossless segment based DNA compression, Proceedings of the 3rd International Conference on Electronics Computer Technology, IEEE Xplore Press, 298- 302, 2011.
[5] Gregory Vey, "Differential direct coding: a compression algorithm for nucleotide sequence data", Database, doi: 10.1093/database/bap013, 2009.
[6] Banani Saha and Umesh Ghoshdastider, “Genome Compress: A Novel Algorithm for DNA Compression.” 2010.
[7] R.K.Bharti et al., "Biological sequence Compression Based on Cross chromosomal properties Using variable length LUT", CSC Journal, Volume 4, Issue 6, 2011.
[8] Rajendra Kumar Bharti and R.K.Singh, “Biological Sequences compression based on complementary palindrome using variable Length Look Up Table(LUT)”, IJRRAS Journal, Volume 9, Issues 3, December 2011.
[9] Govind Prasad Arya et al, “A Compression algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Look up Table (LUT)”, IJCSIT Journal, Volume 3, Issue 3, 2012.
[10] Deepak Harbola et al, “State of Art: DNA Compression Algorithms.” IJARCSE Journal, Volume 3, Issues 10, October 2013.
[11] Subhankar Roy & Akash Bhagat et al, “SBVRLDNACOMP: An Effective DNA Sequence Compression Algorithm.” , IJCSA Journal, Volume 5, August 2015.
[12] ASCII code. [Online]. Available: http://www.LookupTables.com/

Kumar Nishant has received his Bachelor of Engineering degree in Computer Science Engineering from M.G.R. University, Chennai in year 2012. At present he is pursuing M.E from Uttarakhand Technical University(UTU), Dehradun.

Pawan Kumar Mishra At present he is an Assistant Professor at Uttarakhand Technical University(UTU), Dehradun.