A Better and Efficient DNA DATA COMPRESSOR by Fusion of Symbolical and ARLE Technique

[1]P. Surendra Varma, [2]J. N. V Harish
[1,2]Dept. of CSE, NRI Institute of Technology, Vijayawada, AP, India

Abstract
Data compression is concerned with how information is organized in data. The size and importance of these databases will be bigger and bigger in the future; therefore this information must be stored or communicated efficiently though there are many text compression algorithms, they are not well suited for the characteristics of DNA sequences. There are algorithms for DNA compression which takes advantage of repetitive nature of DNA fragments within the sequence whereas few of the other algorithms are written for the non-repeated patterns within DNA sequences.
We present a compression algorithm, "DNA DATA Compressor" for DNA sequences based on amalgamation of unparalleled symbolic representation with combination of ARLE (Altered Run length Encoding) technique. The proposed technique is very simpleton and efficient for the DNA compression. The proposed algorithm performs equally well for both repeated and non-repeated patterns within the DNA sequence. We have also defined the Worst case, Average case and Best case for DNA compression using our proposed Algorithm. Assigning symbolic representation along with ARLE technique for fragments of DNA sequence is also a unique concept introduced in this algorithm for the first time in DNA compression.

Keywords
DNACompressor, Palindromes, Approximate Repeats, ARLE

I. Introduction

DNA Sequences making up any organism represent the basic blueprint of that organism so that understanding and analyzing different genes within the DNA sequences has become an extremely important task. The Deoxyribonucleic acid (DNA) constitutes the physical medium in which all properties of living organisms are encoded. Molecular sequence databases (e.g.,EMBL,Gen bank, DDJB, Entrez, Swiss Prot, etc) currently collect hundreds or thousands of sequences of nucleotides and amino acids reaching to thousands of gigabytes and are under continuous expansion. Need for Compression arises because approximately 44,575,745,176 bases in 40,604,319 sequence records are there in the Gen Bank database. Today, increasing genome sequence data of organism's lead DNA database size two or three times bigger annually. Thus, it becomes very hard to download and maintain such data in a personal local system. Algorithms for compressing DNA sequences, such as Gen Compress [1], Bio compress [2] and Fact [5], are available as tools to manage such works. Although these algorisms use characteristics of DNA like reverse complement or point mutation, their compression rate is about 1.74 bits per base (78% in compression ratio) [3]. Therefore, compression of DNA sequences is recognized as a tough task and needs much improvement.
Although in recent years mass storage media and devices became easily affordable and ubiquitous, data compression now has an even more important role in reducing the costs of data transmission, and since the DNA files are typically shared and distributed over the Internet. Space-efficient representation of the data reduces the load on FTP service providers, as the transmissions are done faster, and it also saves costs for clients who access and download the sequences. Since the price of transmission is proportional to the sizes of accessed files, even savings of 10–20% are useful. In this article we demonstrate that this size reduction can be easily accomplished for typical eukaryotic data.
The compression of DNA has an intrinsic value in itself, but additionally it provides many useful clues to the nature of regularities that are statistically significant in the DNA data, indicating how different parts of the sequence are in relation with each other, how sensitive the genome is to random changes such as crossover and mutation, what the average composition of a sequence is, and where the important composition changes occur. Due to its many potential uses as a statistical analysis tool for DNA sequences, any DNA compression method is faced with all of the challenges that modeling tools are facing: maximizing the statistical relevance of the model used, and choosing between various models based on the overall performance-complexity merit.
In this paper, it has been tried to cope the above said problem. In this work it has been tried to achieve a better compression ratio and runs significantly faster than any existing compression program for Biological sequences. A lot of research work has Already been carried out for developing progsrammes for Biological sequence compression. It is seen that all Biological sequence compression algorithms find repetition within the sequence. Longer repetitive length implies higher compression gain. The compression ratio gain is high if highly similar subsequences are found. It is well known that there are similarities among different chromosome sequences. The objective of this paper is to subdue the above problem and propose a simple and efficient compression technique with better compression ratio.

II. DNA Structure

DNA consists of two long polymers of simple units called nucleotides, with backbones made of sugars and phosphate groups joined by ester bonds. These two strands run in opposite directions to each other and are therefore anti-parallel. Attached to each sugar is one of four types of molecules called nucleo bases (informally, bases). It is the sequence of these four nucleo bases along the backbone that encodes information. This information is read using the genetic code, which specifies the sequence of the amino acids within proteins. The code is read by copying stretches of DNA into the related nucleic acid RNA in a process called transcription.
The deoxyribonucleic acid (DNA) constitutes the physical medium in which all the properties of a living organism are coded [TG93]. A DNA sequence consists of four alphabets namely Adenine (A), Cytosine(C), Guanine (G) and Thymine (T). The ribonucleic acid (RNA) sequence is also consisting of four alphabets Adenine (A), Cytosine(C), Guanine (G) and Uraceil (U). RNA's nucleotide is similar to DNA's nucleotide but Thymine (T) is replaced by Uraceil (U) in RNA nucleotides

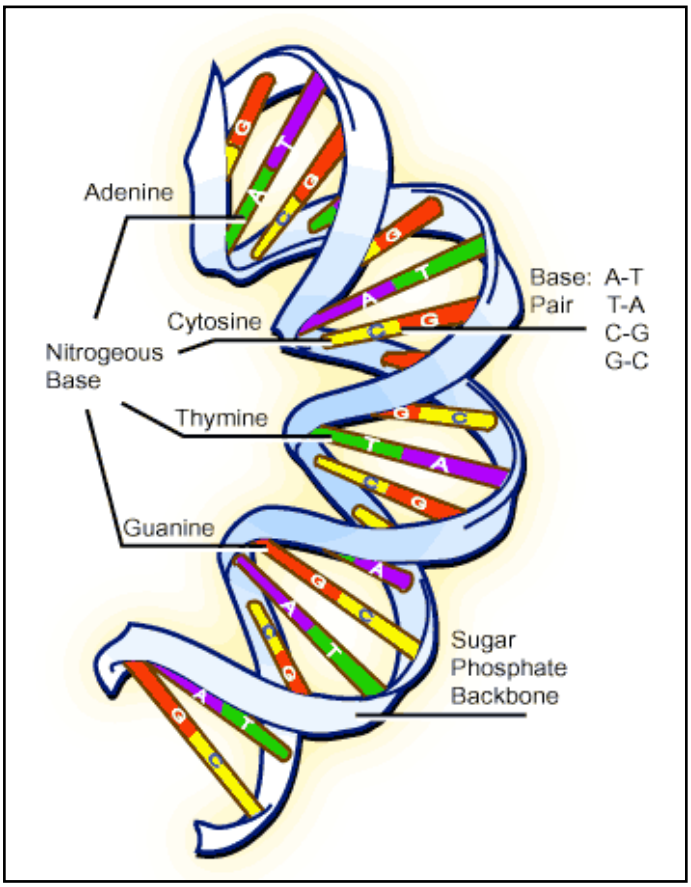

Fig. 1: DNA Structure

Within cell's DNA is organized into long structures called chromosomes. During cell division these chromosomes are duplicated in the process of DNA replication, providing each cell its own complete set of chromosomes. Eukaryotic organisms (animals, plants, fungi, and protists) store most of their DNA inside the cell nucleus and some of their DNA in organelles, such as mitochondria or chloroplasts. In contrast, prokaryotes (bacteria and archaea) store their DNA only in the cytoplasm. Within the chromosomes, chromatin proteins such as histones compact and organize DNA. These compact structures guide the interactions between DNA and other proteins, helping control which parts of the DNA are transcribed.

III. Related work

Compression or source coding is the process of encoding information using fewer bits (or other information-bearing units) than an unencoded representation would use, through use of specific encoding schemes. Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. Compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications. Therefore, the design of data compression schemes therefore involves trade-offs among various factors, including the degree of compression and the computational resources required to compress and un compress the data. Further some compression algorithms can introduce distortion in data which are known as lossy compression. Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter 'e' is much more common than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very small. Lossless compression exploits the repeats, palindromes and patterns present in the digital data to reduce the overall size.

The compression of DNA sequences is considered as one of the most challenging tasks in the field of data Compression. Standard compression algorithms are notable to compress DNA sequences. Rapid advancements in research in the field of DNA sequence discovery has led to a vast range of compression algorithms. The number of bits required for storing four bases of any DNA sequence is two. With the constant decrease in prices of memory and communication channel bandwidth, one often doubts the need of such compression algorithms. There are many text compression algorithms implemented in various tools like WinZip, winrar, gzip, bzip2 but they cannot compress DNA sequences. It is very well known that one of the main features of DNA sequences is that they contain substrings which are duplicated frequently except for a few random mutations. For this reason, most DNA Compressors are built to searching and encoding these repeats. However, searching for repeated patterns takes a long time and takes more memory. The algorithms designed specifically to compress DNA sequences assuming the frequent occurrences of repeated patterns in DNA sequence may not efficiently compress with non repeated patterns in the DNA sequences where as some of the algorithms work well but built for short sequences. Therefore, the results for these algorithms under the best and worst case vary in a high degree. There are several approaches for encoding of text s which are Huffman Encoding, Adaptive Huffman Encoding, Arithmetic coding, Arithmetic adaptive coding, Context Tree weighted method etc. Algorithms designed for DNA Compression like Gen Compress, Bio Compress, DNA Compress, CTW+LZ, C fact have achieved an approximate compression ratio of 22% use the above mentioned approaches. With Bio Compress, at each step, the longest factor beginning at the current position which matches with a factor starting before is chosen. BioCompress-2 uses arithmetic coding of order 2. C fact performs in two pass in that it looks for longest exact matching repeat and uses a suffix-tree for finding the longest repeat. Gen Compress works on approximate repeats in which at each step, it looks for the optimal prefix of the not yet encoded part of the DNA sequence. It uses Hamming distance (v1) and edit distance (v2) for approximate repeats. CTW+LZ is a Combination of Gen Compress and CTW uses Context Tree Weighting method. DNA Compress uses Pattern Hunter as preprocessing. Found repeats are sorted in decreasing order of size. Greedy approach of Gen Compress and DNA Compress has not been proved well. DNA Pack was made using dynamic programming instead of greedy approach and proved better than above mentioned algorithms. Common components of most of DNA compression algorithms are:

- Finding the candidate repeat segments.
- Considering approximate repeats.
- Encoding of the repeat segments.
- Encoding of the non-repeat segments.

Increasing genome sequence data of organisms lead DNA database size two or three times bigger annually. Thus it becomes very hard to download and maintain data in a local system. For a four letter alphabet in DNA (A, C, G, T), an average description length of 8 bits can be assigned per Fragment. Algorithms for Compressing DNA sequences, such as Gen Compress, Bio compress and C fact are available to compress DNA sequences. Their compression rate is about 1.74 bits per base i.e., 78% in compression rate hence we present a new compression algorithm named" Gen Bit Compress" whose compression rate is below 1.2 bits per byte (for Best case)

even for larger Genome (nearly 2, 00,000 characters).

IV. Methodology

In this paper we consider the problem of DNA compression both for repetitive and non-repetitive DNA sequences. DNA is a long polymer of more echoing sequences. As explained earlier, DNA sequence can consist of four letters {A, C, T, G}. So these four letter can maxim generate possible 256 combinations as per the probabilistic theory. These combinations can be exemplified by a unique 256 symbolic representation where it can yield each one of only 8bits or one byte for each sequence of 4 four letter. This type of representation or encoding will drastically result in the decline of memory requirement of up to nearly 75% of the representation. .The major part of second phase is to still compress the resulting sequence which can be achieved with the help of ARLE (Altered run length encoding) technique. By the end of all the phases of compression we are capable of achieving a very good compression ratio in tallying to various best compression algorithms for DNA on an average 86.23 % and ranges in between the minimal of 75% to maximal of nearly 95.23% These rates of compression are generally been categorized to best, average and worst case considerations in performance metrics calculations. This type of representation or encoding will drastically result in the decline of memory requirement of up to nearly 86.23% of the representation.

A. Encoding

1. Pre Requisite

Before starting with initial phase of our encoding it is needed that the DNA file is to be referenced whether it contains any junk or invalid characters other than (A-Adenine, G-Guanine-Cytosine-Thymine) this can be done by manually for very small DNA sequences or else it can be programmed to check whether it has valid or any junk or invalid characters. The Proposed algorithm assumes that the sequence only consists of only valid (A, G, C, T) characters and no more junk data and proceeds further.

2. Initial Phase

After the immediate prerequisite satisfaction of sequences as per the principle .The first phase of encoding methodology begins. Let the set S represent {A, C, T, G}. If m is the total elements in the set and n is the length of the word that should be constructed using the set s, possible words that can be generated would be mPn where element cannot be repeated. In case of DNA sequences, the bases can be repeated. So the possible words of the length n with m characters in the set S would be m^n where in any character in the set can be repeated any number of times within the word length n. Considering this theory , we build the concept of symbolical representation keys of word length=4 with no collisions within the representation. We scan the whole DNA sequence and fragment it in small DNA fragments of length=4 and represent this fragment in one character. The result would be compressed DNA. Therefore, the algorithm initially builds and assigns unique character to each of the fragment and then scan entire DNA sequence into a single character finally resulted into semi compressed DNA and proceed to the next phase. The detailed description is as follows

The input sequence is divided in to fragments, where each fragment = 4 characters. Thus in this coding scheme, 256(2 power 8 = 256) combinations can be represented because the maximum number of possible combinations are 256 with four letters (A, G, C, T).The process of entry of each fragment in the input sequence is pre defined with a special symbol. So the input file is to be given as input in the text field of the tool then the algorithm scans the entire sequence with four letters at a time and immediately verifies with the suitable symbol and replaces it. So it traverses from the initial position to final Null string and replaces each of the four letter combination irrespective of the order. The special symbols can be any extended ASCII representation or any possible 256 unique representation with totally 8 bits of representation. The problem with extended ASCII representation is that is inaccessibility of certain character. This job can be overwhelm with an special representation like palindrome reverse sequences can be represented with an special like its complement with the reversal of the special symbol used along with an negation symbol followed indicating its reverse compliment.

In order to have clear understanding of above mentioned step let us consider an example and remember that for the sequences given in the input the following symbols are already been pre-defined.

For example

let us consider a sequence of DNA with 80 letters (A, G, C, and T) which is known as HEHCMVCG.These are divided in to fragments of each four bytes each given as follows:-

CTCTCTCTCTCTCTCTCTCTCTCTCTCTCT
GCCAGCCTGAACATCCATCCATCCATCCATCCATCCATC

CTCT CTCT CTCT CTCT
CTCT CTCT CTCT CTCT
AGCT AGCC AGCC AGCC
TGAA CATC CATC CATC
CATC CATC CATC CATC

Here the specified sequence is to be converted as per the specified specifications. Therefore, logically it needs 96 bytes to store this sequence in a text file. Normal text compression algorithm like WinZip supplied with Windows or zip, tar supplied with UNIX cannot actually compresses this sequence. With the proposed algorithm, sequence will be compressed by 75% irrespective of the number of repeated or non-repeated patterns within the sequence.

So by the end of the first phase we have concluded that we are capable of achieving the compression rate of about 75%.Because a size of 80 bytes is been converted to size of 20 bytes. But further there is capable of decreasing its size to much more which is done in next phase.

Repeating this process for the entire compressed sequence and adding these fragments result in an original DNA sequence without any loss of information.

3. Second Phase

Now the resulting Sequence can still be further compressed. By the end of the first phase we are capable of compressing the above DNA sequence to about 75%(Initially it requires about 80 bytes for 80 characters but by the end of first phase we minimized it to 20 bytes there by achieving it to 75%) .The major part of second phase is to still compress the resulting sequence which can be achieved with the help of ARLE technique(Altered Run length encoding). This can be achieved in the following manner .Initially the compressed sequence from the first phase is taken as output and given as input to the second phase the compressed file is scanned in the following manner:-Each byte of data is been scanned to find out adjacent repetitious occurring. If the occurrence of same symbol is been seen consequently more than of two times the total repetitive sequences are supplanted with its length indication .If there is no

adjacent repetitious of the same symbol then its representation is continued without any change. The algorithm does not depend on the size of the sequence and capable to compress even larger sequences having sizing in hundreds of MBs. So if the fragment within DNA sequences appears to be AAAA…200 times, it would be compressed to special symbol.length ie..@50 times (98% compression). Algorithm can be extended to check if it finds such fragments in which same base is being repeated for multiple times, it can be compressed as base{n} where n is the frequency of repetition.

Each byte of data is scanned and if there is more than one repetition of the same sequence consequently in any position then immediately it counts the number of occurrences of same letter repeatedly and uses altered Run length encoding to represent the maximum number of repeats consequently.

So at this stage it generally consists of two important notations they are the sequence letter and its length following it immediately. The specialty with this representation is that if there is more than two consequent repetitions then only its length is indicated or else cannot use. So the length of each sequence which is greater than 2 is found out and only those whose consecutive repetition is more than two time are indicated with length special bit. So by the end of the second step of our compression algorithm the sequence memory can be drastically be compressed.

(i). Algorithm to Compress Procedure Encode

Begin

1. Divide the given DNA sequence in to fragments, where each fragment consists of 4 characters.
2. Generate all possible combinations of DNA sequence (A, C, G, T). (Since the sequence contains 4 different bases, there will be 4^4 = 256 combinations).
3. Assign unique character to each of the fragment and then scan entire DNA sequence into a single character sequence finally resulting into semi compressed DNA format.
4. Each character of data is scanned and if there is more than one repetition of the same unique letter sequence consequently in any position then immediately it uses altered Run length encoding to represent the maximum number of repeats consequently.
5. If the consecutive fragments are different, no changes are been made and then continued to next step.
6. Repeat the steps 3 and 4 until the length of sequence is "n- τ". (Where n = length of the given sequence and τ = n mod 4)
7. 7: This result in conversion of entire given DNA sequence to some compressed format.
8. 8: Transfer the compressed DNA data to the output String.

OUTSTRING

Previously by the end of the first phase we have concluded that we are capable of achieving the compression rate of about 75%. Because a size of 80 bytes is been converted to size of 20 bytes. But by the end of second phase we are further capable of decreasing its size to 8 bytes because the special symbols each take 1 byte and the b digits each require only one byte so totally there are 5 special symbols and 3 bytes of data totally constituting of 8 bytes of data. So finally we can prove by our algorithm that we can compress the data to about 90 % which cannot be done existing DNA compression algorithms.

B. Decoding

The decoding can be followed in the same manner as Encoding but completely in the reverse manner .First the encoded data should be given as a input which initially converts the special combined representation to symbols representation and these data is send to the next phase where each symbol is converted to fragment of four bytes which results in the original data.

1. Procedure Decode

Begin

1. Reverse the process of altered Run length Encoding and then replace the length field to their respective symbolic representation.
2. Repeat the step 1 for entire compressed sequence given and the resulting output is given as input to next phase of decoding.
3. The resulted sequence is then converted in to a segment of 4 bytes each based on the special key value generated and based on the probability of occurrence.
4. The step 3 is repeated for the whole sequence until all the special symbols are been converted in to fragment bits of each 4 bytes.
5. The resulting sequence from the final phase is the real input string.
6. Transfer the outer string which is the actual output.

End

V. Performance Measurement

The most benchmark for our algorithms is a comparison with the other algorithms designed to compress DNA sequence. We had compared the performance of the method mentioned versus many compression algorithms and to even very recently proposed algorithms. We have include the results of various algorithms .Our proposed methodology suggests that this method suits one of the best compression methodology in comparison with various existing methods. If a sequence is compressed using proposed algorithm it will be easier to make sequence analysis between compressed sequences. It will be easier to make multi sequence alignment. High compression rate suggests that a highly repetitive sequence.

The various chromosomal structure compression percentages based on various existing techniques and are compared to our proposed method are given as follows:-

Table 1: Performance Measurement Analysis

Type of Sequence	Original Size	Hash Based	Variable LUT	Bio Compress	Proposed Technique
Gallus Beta Globin	752	75%	86%	74.4%	92%
Goat alanine beta globin	732	75%	83.25%	73.7%	88%
Human Beta globin	440	75%	82.32%	76.5%	86.2%
Lemur Beta Globin	312	75%	87%	76.5%	89%
Mouse Beta globin	776	75%	94.2%	94.7%	92%
Opossum Beta Hemoglobin	760	75%	92%	92.4%	91.6%
Rabbit Beta Globin	736	75%	87%	73.12%	82%
Rat Beta Globin	752	75%	91.2%	68.4%	93%
Avg	752.5	75%	86.32%	78.75%	89.8%

Basis on the above thesis specifies the percentage of compression that each can achieve. The DNA compress, LZ77, Gen compress are of are of good model for compressing the DNA sequences but truly these are not the best compression algorithms as of compared to the next recent compression techniques of Hash based and Variable LUT compression methodologies. The basic criterion that generally befits any compression criteria is the basic two significant compression rate and compression time. The two compression techniques DNA compress and LZ77 are capable of attaining best pre-processing time when compared to the other techniques. But they defray the compression percentage.

The following are the family of sequences of Mendilian Genome Sequence which is sun constituent of Chromosome-(PBD-rPBN). In the present Performance metrics comparison we are testifying the sequences of only type Nucleiotide.The best compression algorithms are capable of achieving an compression ratio of about 86.2% and approx in range of (80-91%).The variable LUT compression technique are reckoned to be the beat compression technique for Nucleotide sequence where there is a greater scope of repeated Palindromes and repeated echo's of the sequence. The variable LUT is able of achieving on an average of 86.2% compression rate but however its features doesn't suit where there is lesser scope for percentage when there is non-repeated fragment. The Hash based compression is capable of achieving better compression rate when there are both for repeated as well as non repeated fragments it's compression ratio generally will be in around of least 75% in the worst ratio which is of best patch considering to other compression algorithms .While its best case is also around the same ratio so it is not suitable for best optimized compression.

Our proposed compression is capable of achieving of nearly 95.2% compression ratio when compared to the sequence of family CHNMKXXXX which is of best till date not achieved by any compression technique. The worst case compression rate if 75% as similar to of Hash based compression technique. The following data shows that the current leader s in the market of DNA compression is SNA compressor (even though the above results are of large variation but it is best of all compression techniques because of its convey ability of any type of sequence and also its running time) and variable LUT.So we can say that our proposed technique is capable of achieving the best compression rate in comparable to these algorithms.. Note there is further capability to reduce the space complexity of our technique which can yield much better results. The various compression ratios of sequences are as follows for various sequences versus various compression techniques

The following table illustrates the detailed compression ratio of various sequences for the family of sequences of Human Chromosome by various compression techniques.

Table 2: Sequence Size Comparison

Sequences	Size	BIO	Gen compress	RAR	ZIP	Our Technique
MTPACGA	164	60	52	132	164	41
MPOMTCG	496	72	63	154	176	124
CHNTXX	1488	378	393	279	268	271
CHMPXX	484	132	120	193	185	121
HEHHPTRV	732	272	376	398	485	274
CHROMOBETA	760	172	360	390	480	260
OPOSSUM	752	480	312	392	-	142
CHBGVHXXX	732	272	306	394	-	160

The above table shows the performance between various compression techniques between Bzip, RAR, DNA, ZIP and our algorithm for DNA sequences after including unknown nucleotides as a separate entity. The practical evaluation of the performance of this approach is done in such a way that it is easily comparable with the published results. From the above data it is clear that our algorithm gives better compression result when compared to other techniques as well compression techniques like ZIP, as well Archive which are the beat compression techniques for data other than DNA sequences.

VI. Performance Cases

The following is the separate analysis for the proposed algorithm is given for all the three cases (worst case, best case and average case).

Input: Input String (INSTRING) Containing A, T, G and C and for RNA is A, U.G, C.

Output: Encoded String (OUTSTRING) Compression Rate = Number of Bits/Total number of Bytes.

Worst case: In the worst case there are no repetitive fragments and individual bases are maximum and

In this algorithm, the worst-case compression rate at about 75% which is the best compression ratio of some algorithms.

Best Case: The best case is generally observed when there is more number of consecutive repetitions.

The best compression rate of our proposed technique is generally at an average of 91.5% which is best compared to compression techniques

Average Case: The Average case efficiency of this algorithm defines the compression rate of a typical input or a random input which is given by neither the worst case nor the best-case efficiency

The average case compression is around in range of 82.5-87.67% which is undone by various existing compression techniques

VII .Conclusion & Future work

A simple DNA compression Tool which is completely new in its design is proposed to compress DNA sequences which are repetitive as well as no repetitive in nature. If the sequence is compressed using our Compress algorithm Tool, it will be easier to compress large bytes of DNA sequences with the Compression ratio of about 92.5 %, when the repetitive fragments are maximum. When the repetitive fragments are less, or nil, the compression Ratio is 2.238 Bits/bytes. The compressed Genomes will be very useful in sequence comparisons and multiple sequences Alignment Analysis. Thus our compress algorithm can act as a better trade-off between Time complexity and Space complexity

References

[1] Ateet Mehta, 2010, et al.,"DNA Compression using Hash Based Data Structure", IJIT&KM, Vol. 2, No. 2, pp. 386.

[2] Choi Ping Paula Wu, 2008, et al.,"Cross chromosomal similarity for DNA sequence. Compression", Bioinformatics 2(9), pp. 412-416.

P. Surendra Varma received his M.Tech Computer science Engineering from Acharya Nagarjuna university campus. He is working as an Assistant Professor in NRI institute of technology, Vijayawada. His research interests includes Bioinformatics, compression techniques, operating systems, theory of computation, compiler design, programming languages, data mining and warehousing, software engineering.

J.N.V.Harish has received B.Tech in Computer science Engineering from NRI Institute of technology, Vijayawada. He is currently working as a software engineer in Exilant Technologies, Banglore. His research interests includes Bioinformatics, compression techniques, Android programming, cloud computing, data mining and warehousing.