International Journal of Bio-Science and Bio-Technology
Vol.8, No.3 (2016), pp. 45-52
http://dx.doi.org/10.14257/ijbsbt.2016.8.3.05

Compression Techniques Applied to DNA Data of Various Species

Vilas Machhi[1] and Maulika S Patel[2]

[1] *Alumnus, G H Patel College of Engineering & Technology,* [2] *G H Patel College of Engineering & Technology*
[1,2] *Vallabh Vidyangar, Gujarat, India*
vilas_machhi@yahoo.com, maulika.sandip@gmail.com

Abstract

DNA sequences comprise of sequentially linked nucleotides, A, C, G and T. As a result of the genome projects, a significant amount of DNA sequences of various species are deposited in various databases. Human DNA contains about 3 billion base pairs. The number of genes within the DNA is 20,000 to 25,000. For storing DNA data of a single person, we require approximately 10 CD – ROMs. This amounts to huge data storage costs, subsequently making the use of these data such as analysis and retrieval quite challenging. DNA sequence analysis is useful in diverse areas such as forensics, medical research, pharmacy, agriculture etc. It is very necessary to address the storage issue of these exponentially growing data. In this paper we have implemented 4 different algorithms for DNA data compression: LZW (Lampel-ziv-Welch) algorithm, run length encoding algorithm, Arithmetic coding and Substitution method. The compression results on these algorithms are presented and compared on DNA sequence data of 10 different species.

Keywords*: DNA Data Retrieval, Lampel-ziv-Welch (LZW), Run length encoding, Arithmetic coding, Substitution Method*

1. Introduction

Biology and computers have become close cousins which are mutually respecting, helping and influencing each other. A common misconception is that bio-informatics is about creating and managing bio-data bases that is not true. Fine analytical and engineering skills are in great demand in this area. The renowned computer scientist and father of algorithms, Donald Knuth, has quoted that "Biology has 500 years of exciting problems to solve" [7]. He feels that biology is "so digital, and incredibly complicated, but incredibly useful" (Computer Literacy Interview with Donald Knuth by Dan Doernberg, December 1993) [1].

Compression of sequential data has been of interest for decades. There are many types of data which need to be compressed, for ease of storage and communication like texts, images, video, etc. In this paper we discuss compression techniques for genetic sequences. Molecular biology databases like EMBL, Genebank, and DDBJ are publicly available databases to store DNA and protein sequence. The size of the databases is increasing exponentially. DNA data bases amount to hundreds of GB making data compression in genomics a very important task [2].

Deoxyribonucleic Acid [8] is a genetic material of all cellular organisms and most viruses. DNA carries the information needed to direct protein synthesis and replication. Protein synthesis is the production of the proteins needed by the cell or virus for its activities and development. Replication is the process by which DNA copies itself for each descendant cell or virus, passing on the information needed for protein synthesis. In most cellular organisms, DNA is organized in chromosomes which are located in the nucleus of the cell.

ISSN: 2233-7849 IJBSBT
Copyright ⓒ 2016 SERSC

DNA sequence contains 4 nucleotides namely adenine (A), guanine (G), cytosine (C), and thymine (T). A human genome contains three billions characters over twenty three pair of chromosomes. Only about 10% of sequence contains genes.

DNA [9] is the blueprint of biological life from its inception to its growth and till death. Its discovery has not only revolutionized science and medicine but it has affected all walks of life; whether they are social, legal, criminal or inheritance related. DNA's discovery has become important to the extent that it has even influenced a nation's security parameters / concerns, as scientists have gone all the way to developing biological weapons. Study of DNA is used in forensic science, agricultures and genealogy.

In [3], compression techniques such as Gzip, Bzip, DNACompress, GeneCompress, Binary and Huffman encoding are used for DNA data compression. Table I shows the sizes of H3N2 sequences and compression ratios achieved with different compression methods. The first two rows contain the results of general compression tools. The third row contains the results of encoding text-based sequences to binary codes with position/base-pair tables. The last row contains the results using Huffman codes for compression. Since GenCompress cannot deal with large data sets and DNACompress compresses only single sequence files, due to which only 20 randomly selected H3N2 sequences are compressed for comparison. So from their results Huffman algorithm gives better compression ratio.

Table 1. Comparison of Compression Results on h3n2 Virus (Primitive Size 1,995,960 bytes) [3]

Algorithm	Size	Compression ratio
Gzip	101,494	94.92%
Bzip	54,314	97.28%
DNACompress	474,139	76.26%
GeneCompress	25,837	98.70%
Binary	73,901	96.30%
Huffman	52,927	97.35%

2. Method

DNA sequence contains a large number of approximate repeats. In literature, researcher used Gzip, Bzip, DNACompress, GeneCompress, Binary and Huffman for compression. We used LZW, Run length encoding, Arithmetic coding & substitution method for DNA data compression and these methods & results are discussed below.

A. LZW Algorithm [4][10]

LZW is a "dictionary"-based compression algorithm. It encodes data based on dictionary instead of tabulating character counts and building tree like Huffman encoding. For encoding a substring a single code number corresponding to that substring's index in the dictionary needs to be written to the output file. It gives good result on files with repeated substrings like text files. It was widely used in Unix file compression utility "compress", and in the GIF image format. In LZW, dictionary has 256 characters (in case of 8 bits) and that is used as "standard" character set. It then reads 8 bits data at a time (e.g.'t', 'r', etc.) and encodes that data as the number which represents its index in the dictionary. It adds new substring in the dictionary when it comes across new substring (say 'tr'). When it comes across a substring which is already seen, it just reads new character and concatenates it with the current string to get a new substring. When the next time LZW revisits a substring, it will be encoded using a single number. Maximum numbers of entries (say, 4096) are defined for the dictionary, so that the process doesn't run away with memory. So, the codes which are taking place of the substrings are 12 bits

 Copyright ⓒ 2016 SERSC

long (2^{12} = 4096). It is necessary for the codes to be longer in bits than the characters (12 vs. 8 bits).

Codes 0-255 in the dictionary are always used to represent single bytes from the input file. When encoding begins the dictionary contains only the first 256 entries and remaining table is blanks. To represent sequence of bytes codes 256 through 4096 is used by which compression is achieved. During encoding, LZW identifies repeated sequences in the data, and adds them to the dictionary. Time Complexity:-O(n) expected time, where n is the length of the text that is being compressed.

LZW Encoding Algorithm [10, 6]

```
1   Initialize table with single character strings
2   P = first input character
3   WHILE not end of input stream
4       C = next input character
5       IF P + C is in the string table
6         P = P + C
7       ELSE
8         output the code for P
9       add P + C to the string table
10        P = C
11      END WHILE
12  output code for P
```

B. Run length Encoding Algorithm [11]

Run length encoding works by reducing the physical size of a repeating string which is called a run. Run is typically encoded into two bytes in which the first byte represents the number of characters in the run, which is called the *run count* and the second byte is the value of the character in the run which is called run value and the range of it is in the 0 to 255. Encoded run may contain 1 to 128 or 256 characters. The run count usually contains the number of characters minus one. Uncompressed string of length 10 would normally require 10 bytes of storage. For example we have sequence of H3N2 virus DNA data:

TTCCCCCCCT

The same string would require only six bytes using RLE encoding:

2T7C1T

RLE is used in Windows 3.x, used to compress the Windows 3.x startup screen. It is well suited to palette-based bitmapped images such as computer icons. Run-length encoding is also used in fax machines (combined with other techniques into Modified Huffman coding).

C. Arithmetic Coding [5]

Arithmetic coding is a method for lossless data compression. A string of characters such as the words ‘hello there’ is represented using a fixed number of bits per character, as in the ASCII code. Arithmetic coding is variable-length entropy encoding which converts a string into another form that represents frequently used characters in the string. In other entropy encoding techniques, the input message is separated into its component symbols and each symbol is replaced with a code word. While in arithmetic coding, it encodes the entire message into a single number n, where $0.0 \leq n < 1.0$. It is used in JPEG2000.

Algorithm:

1. Take the number of independent characters in the words to be encoded. Arrange these characters into a table with their corresponding frequency.

Copyright ⓒ 2016 SERSC

2. Each of the characters will be assigned a range between 0 and 1 based on its frequency/ probability of occurrence.
3. The algorithm for encoding is as below:
   set low to 0.0
   set high to 1.0
   while there are still input symbols do
   get an input symbol
   code_range = high - low.
   high = low + code_range * high_range of the symbol being coded
   low = low + code_range * low_range of the symbol being coded
   end of while
   output low

D. Substitution Method

This is the simple statistical encoding method. In this method, we substitute a frequently repeating pattern with a code. The code is shorter than that pattern given for compression. It is used in compression of source program files.

Algorithm:

1. Decide all the possible combinations of A, C, G, T for the length of 3 alphabets (triplet).
2. Assign 1 alphabet or symbol to the triplet
3. Replace the triplet with that symbol or alphabet

This will result in new compressed file.

3. Results

DNA data of different species likes Tomato, Bacteria, Rabbit, Ecoli, H3N2 virus, Tobacco, Oryza Sativa (Asian rice) , Aspergillus niger (fungus) ,Saccharomyces cerevisiae (yeast) and Nemastoma Gelatinosum (algae) are collected from NCBI [10] (National Center for Biotechnology research) on which we have applied various algorithms which are implemented in C#. The results of these algorithms on different data sets are shown in the figures 1 to 5. In figures 1 to 5, the first bar shows actual size of DNA data and the rest of the bars show the compressed size of different algorithms.

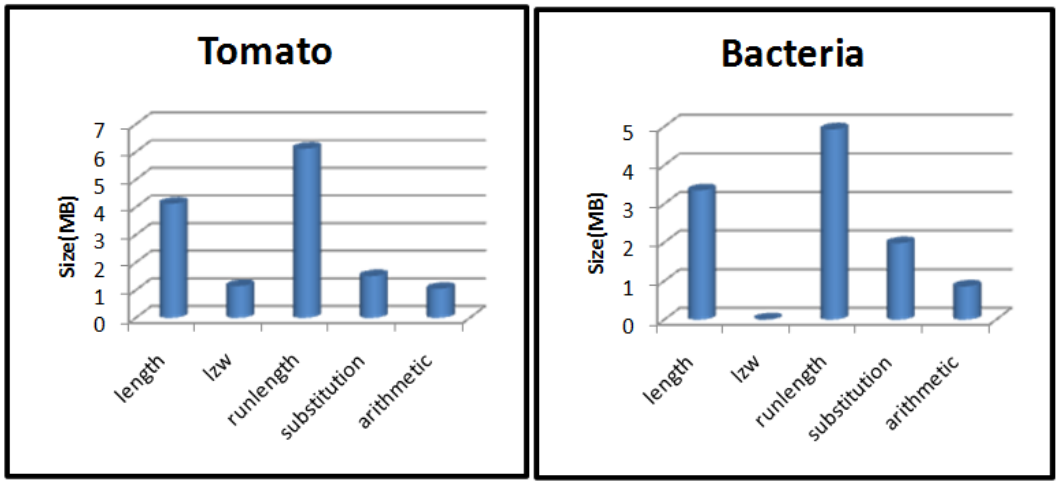

Figure 1. Comparison of Compression Ratio of Different Algorithms on Tomato & Bacteria

Copyright ⓒ 2016 SERSC

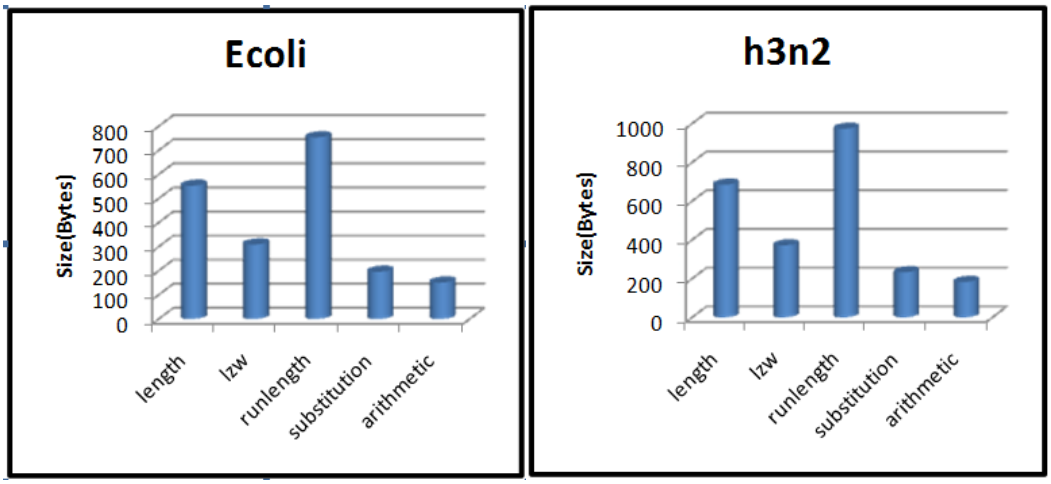

Figure 2. Comparison of Compression Ratio of Different Algorithms on Ecoli & H3N2 Virus

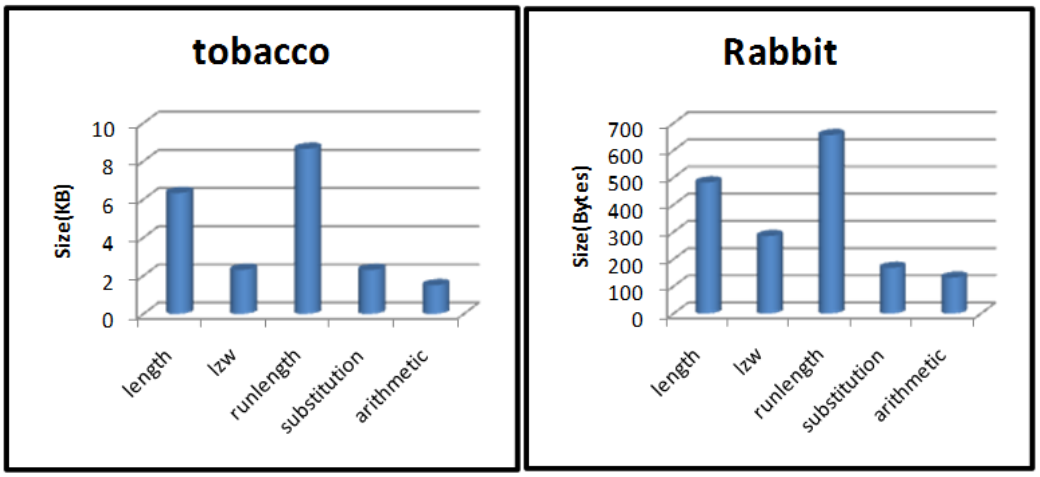

Figure 3: Comparison of Compression Ratio of Different Algorithms on Tobacco & Rabbit

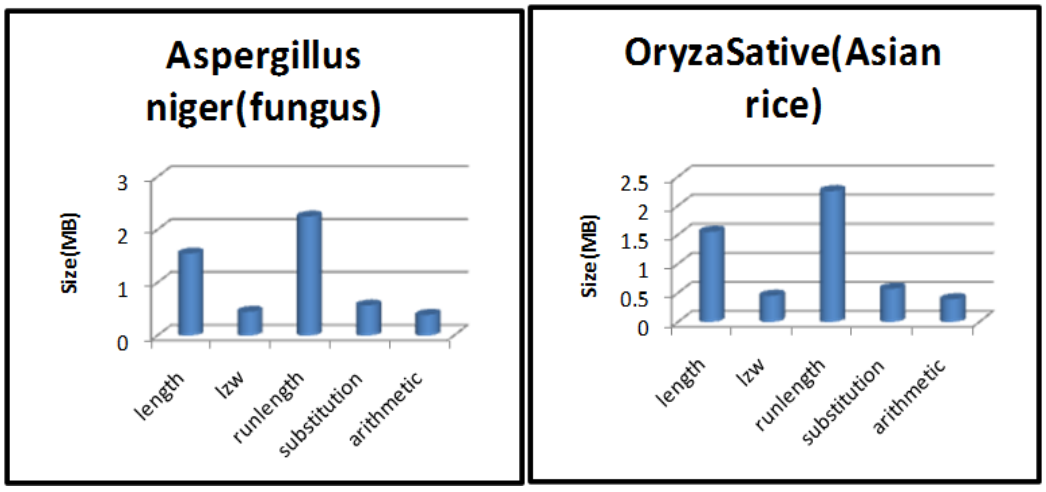

Figure 4. Comparison of Compression Ratio of Different Algorithms on Aspergillus Niger (Fungus) & Oryza Sativa (Asian rice)

Copyright ⓒ 2016 SERSC

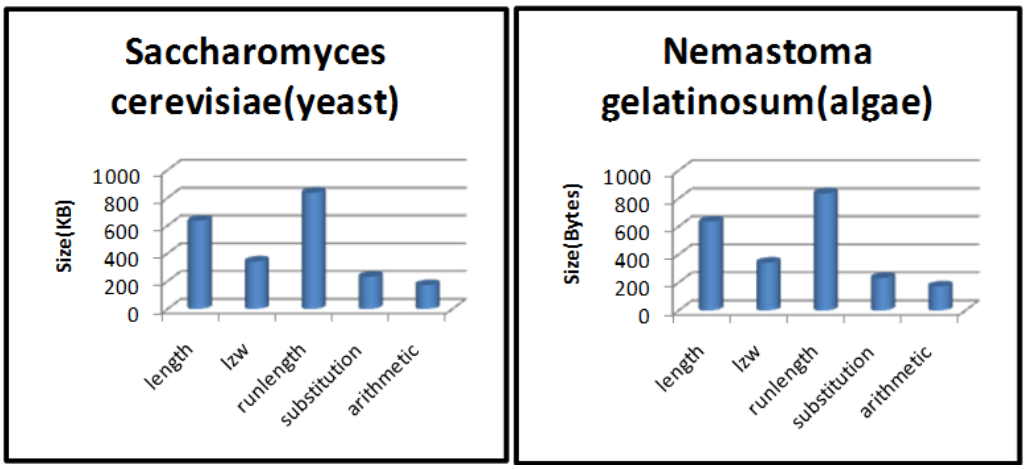

Figure 5. Comparison of Compression Ratio of Different Algorithms on Saccharomyces Cerevisiae (Yeast) & Nemastoma Gelatinosum (Algae)

From the graphs it is concluded that for Bacteria, LZW algorithm provides better compression ratio and for other organisms, Arithmetic coding provides better compression ratio. Run length Encoding is not suitable for DNA data compression. Table II & III show compression ratio of different algorithms for different species.

Table 2. Compression ratio results of Bacteria, Tomato, E-coli, H3N2 virus, Tobacco and Rabbit

Algorithm	Bacteria	Tomato	E-coli	H3N2	Tobacco	Rabbit
LZW	73.1	72.15	44.12	45.69	63.51	40.87
Runlength	-47.16	-48.18	-36.35	-42.19	-36.49	-36.1
Subsitution	64.78	63.44	64.56	65.99	63.67	65.15
Arithmetic	74.33	74.58	72.69	73.28	76.3	72.61

Table 3. Compression Ratio Results of Aspergillusniger (Fungus), OryzaSative (Asian Rice), Saccharomyces Cerevisiae (Yeast), Nemastomagelatinosum (Algae)

Algorithm	Aspergillusniger (fungus)	OryzaSative (Asian rice)	Saccharomyces cerevisiae (yeast)	Nemastomagelatinosum (algae)
LZW	70.77	71.03	60.77	46.14
Runlength	-45.16	-40	-39.06	-31.65
Subsitution	63.23	62.58	71.72	63.46
Arithmetic	74.58	97.45	78.62	88.66

4. Conclusion

The DNA data is used in forensics and research of medicine and for genealogy. In future there is a great demand of storing the DNA sequences. DNA database size will continue to increase with the increased sequencing efforts all over the world. This demands the need for efficient data storage methods. We have reviewed and implemented five methods namely Run-Length encoding, LZW, Huffman, Arithmetic Encoding and Substitution Method for data compression. The results on 10 different DNA datasets are presented. From the compression ratio, we can say that Runlength algorithm is not suitable for DNA data compression. Arithmetic coding is better for DNA data compression because it has good compression ratio.

Copyright ⓒ 2016 SERSC

5. Future Work

With few of the governments having started giving incentives for depositing one's DNA, it is very much possible to have a larger repository of individual human DNA data. In future, compression of these types of target genomes could be carried out using a reference genome for better compression results.

References

[1] A. S. Nair, "Computational Biology & Bioinformatics: A Gentle Overview", **(2007)**.
[2] S. Grumbach and F. Tahi, "A new Challenge for Compression Algorithms: Genetic Sequences", Information Processing & Management, vol. 30, no. 6, **(1994)**, pp. 875-886.
[3] L. S. Heath, A. P. Hou, H. Xia and L. Zhan, "A Genome Compression Algorithm Supporting Manipulation", http://www.lifesciencessociety.org/CSB2010/toc/PDF/38.2010.pdf
[4] C. Sumija, A. Thillipan, M. M. Thomas, S. Rajesh, "Privacy Protection of Medical Data using Histogram Shifting based Reversible Data Hiding", International Journal of Advance Research in Computer Science and Management Studies, vol. 2 , iss. 4, **(2014)**.
[5] R. N. M. S. Sindhu and G. R. Krishna, "An Efficient Adaptive Binary Arithmetic Coder and Its Application in Video Coding", International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, iss. 8, **(2014)**.
[6] K. Sayood, "Introduction to Data Compression", Fourth Edition.
[7] C. Setubal and J. Meidanis, "Introduction to Computational Molecular Biology", Cengage Learning, **(1997)**.
[8] http://mediways.com
[9] www.whatisdna.net
[10] http://kramli.staff.ui.ac.id/files/2012/03/Lecture-3-Data-Compression.pdf.
[11] http://www.fileformat.info/mirror/egff/ch09_03.htm
[12] http://www.ncbi.nlm.nih.gov/

Authors

Vilas Machhi, she received the Bachelor's degree in Computer Engineering from G. H. Patel College of Engineering & Technology, Vallabh Vidyanagar, Gujarat, India in 2013. She is currently pursuing her Master's degree in Computer Engineering from B.V.M. Engineering College Vallabh Vidyanagar, Gujarat, India.

Maulika S. Patel, she received her Bachelor and Master degrees in Computer Engineering in 1997 and 2004 respectively. She acquired her doctorate degree from Dharmsinh Desai University, Nadiad, India in the year 2015. Her current research interests include Computational Biology, Artificial Intelligence and Big Data. She is working as Associate Professor & Head of the Department in Computer Engineering at G H Patel College of Engineering & Technology, Vallabh Vidyanagar, India.

Copyright ⓒ 2016 SERSC

Copyright ⓒ 2016 SERSC