International Journal of Engineering & Technology, 7 (2.31) (2018) 69-73

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Compression of text files using genomic code compression algorithm

G. Murugesan[1*], Rosario Gilmary[2]

[1]*Department of CSE, St. Joseph's College of Engineering, Chennai, India.*
[2]*Department of CSE, St. Joseph's College of Engineering, Chennai, India.*
**Corresponding author E-mail:murugesang@stjosephs.ac.in*

Abstract

Text files utilize substantial amount of memory or disk space. Transmission of these files across a network depends upon a considerable amount of bandwidth. Compression procedures are explicitly advantageous in telecommunications and information technology because it facilitate devices to disseminate or reserve the equivalent amount of data in fewer bits. Text compression techniques section, the English passage by observing the patters and provide alternative symbols for larger patters of text. To diminish the depository of copious information and data storage expenditure, compression algorithms were used. Compression of significant and massive cluster of information can head to the improvement in retrieval time. Novel lossless compression algorithms have been introduced for better compression ratio. In this work, the various existing compression mechanisms that are particular for compressing the text files and Deoxyribonucleic acid (DNA) sequence files are analyzed. The performance is correlated in terms of compression ratio, time taken to compress/decompress the sequence and file size. In this proposed work, the input file is converted to DNA format and then DNA compression procedure is applied.

***Keywords:** Data compression, text compression, lossy and lossless compression, DNA, bases, bit reduction, hexa decimal format, variable length code, huffman codes.*

1. Introduction

Data compression is a minimization of the bits required to present a data. By compressing the data, storage can be effectively utilized. It also supports fast and easy file transfer. Another important advantage of data compression is inexpensive network bandwidth and hardware storage. Text file compression is beneficial for sending the information at better speed and unzipping them has low overhead. There are variety of file types. Such as .txt, Java Scripts, CSS etc. Text compression procedures functions by determining the similarities and substituting them with a transitory bit code. Text compression can be performed in a elementary way by cutting out the unwanted character or replacing the repeats by smaller bit code. It is estimated that data compression can minimize the text folders to 50%.
Compression procedures boost and develop backup storage performance. With time, storage of data grows exponentially. Compression supports data reduction. In virtual perception any type of input file can be compressed however best practice must be chosen for better performance.
Deoxyribonucleic acid DNA, which bears the genes and other nucleotides dwell in 23 pairs of chromosomes, whole of 46. The two strands are called polynucleotides. These biopolymer strands coil around each other to form a double helix structure. DNA is a combination of four bases, bound in pairs. The four bases are Adenine (A), Thymine (T), Cytosine (C), Guanine (G). The genome is made of three billion bases in a decisive order. The bases of two distinct strands are linked by hydrogen bonds. The bases are linked to each other by covalent bonds. DNA is present in every living cell apart from RBC.
In this work, the various existing compression mechanisms that are particular for compressing the text and DNA files are analyzed followed by introduction of contemporary and persuasive compression algorithm which provides better compression ratio for text files. So that the memory size of the storage is reduced and speed of transmission is improved. In the proposed work, the input text file is converted to DNA format. In the next phase, the DNA sequence is reduced to binary format and then it is converted to hexadecimal format followed by encoding using Huffman codes.
This paper is systematized as follows. Section 2 involves the survey of existing works which are brought in to compress the text and DNA files followed by motivation of the present work. Section 3 expresses the proposed work. This is followed by the conclusion.

2. Related work

MLZW [4]- Modified LZW technique is compression algorithm for Bangla characters. The dictionary contains Unicode (1-90) for Bangla characters. During the encoding technique , if the particular character exists in the dictionary then its index value is used else it is appended to the dictionary with unique index value.
PARSEC [2]- PARts- of- Speech for sEntiment Compression is compression technique that uses tags of parts of speech for textual compression. PARSEC uses sentiment analysis algorithms which adapt with minimal classification accuracy. It shows a better improvement in compression rate and accuracy.
Prediction by Partial matching technique is a lossless text compression method (PPM) [1]. By PPM , the symbols/ characters

that are not present as base alphabets are used in encoding of text. It supports extending of text in three methods. They are manual, adaptive and analyzing the passage. It is followed by encoding using arithmetical compression technique.

GLZA [3] is a open source grammar based compression that supports low entropy grammar. It uses hill climbing algorithm considering the encoded string length estimations. Further compression is done by Markov modeling and selective recency modeling.

Universal Text Preprocessing technique [18] is method which does not use any information from the external repositories like dictionaries etc. It explains that compression ratio can be significantly improved by preprocessing the files followed by compression. It is a reversible technique that is applied before encoding.

Symbol Mapping Transformation Method for Text Compression (ETAO) [9] is a procedure for transforming the textual passage based on mapping procedures. Here, each letters of canonical alphabetical order is mapped into single letters rearranged in accordance with their relative frequencies. The procedure uses complementary algorithm. ETAO uses Arithmetic or Huffman encoding as backend and Average Code Length (ACL) can be reduced to 5%.

DNA sequences are capable to incorporate repeated substrings between them. The immeasurable genomic data are stored in nucleotide databases. In order to maintain such databases, number of compression algorithms has been designed. Nevertheless, nearly all the existing algorithms are ill-suited. Data compression algorithms can be classified as loss algorithms and lossless algorithms. In loss compression algorithms, the original input is not recovered fully during decompression. A part of data is lost for all time. Whereas, in the lossless compression algorithm, the actual data is retrieved without loss when the file is being decompressed. Most of the DNA compression algorithms are lossless compression algorithms because losing a single base will misdirect the entire sequence.

DNA sequences can be compressed by two modes, namely horizontal and vertical mode. Vertical modes include compression using file formats. Biological sequences can be compressed by considering only the substrings of the entire genome. This procedure falls under horizontal mode wherein the substrings are made as reference.

GenCompress[21] is a substitution based lossless compression technique by searching the approximate repeats from the DNA sequence. This procedure was introduced specially for genomic sequences. GenCompress seeks the average repeats present in the sequence. Here, an ideal prefix is determined followed by encoding. It also explains the amplitude of similarity or relevance between two DNA sequences.

Biocompress-2[24] is a combination of statistical and substitutional procedure. It was specially designed to compress biological sequences without any loss in original data. Here, the regularities present in the sequence is discovered. One of the regularity considered is existence of palindromes. It determines the repeats and non repeats present in the DNA sequences and encodes them.

DNA Compress program[20] is persuasive, faster and has better running time when compared with the previous compression algorithms. It applies a software called Pattern Hunter [19] to determine the typical average repeats followed by encoding.

Statistical compression algorithm[15] is designed specifically to compress the genomic sequences. It considers the repeats present in the sequence as well as the statistical properties of the sequence. The algorithm anticipates the next symbol to be encoded. Arithmetic coding is used to encode the symbols.

CDNA algorithm[23] is a DNA compression technique which is pure statistical and considers the entropy estimates. Each of the symbol to occur is anticipated by considering the average partial matches. Each match is done between subsequences of the genome that have low hamming distance.

NML[16] is called Normalized Maximum Likelihood. It was introduced to compress the DNA sequences by selecting the models. NML uses the Minimum Description Length (MDL) principle. In this procedure, the input data is recognized as codes. These codes are then compressed by the model selected. The model that provides data with least description length is chosen from other candidate models.

Biological sequence compression algorithm[22] uses the distinctive structure of the genomic sequences. The two major features considered here are the average repeats and palindromes and it is done by dynamic and hash programming. This approach provides higher compression ratio when compared with canonical compression procedures.

DNAPack[17] is a compression algorithm for DNA sequences that uses dynamic programming rather than greedy approach. The procedure is less expensive and yields better compression ratio. First, the repeats, complementary palindromes and non repeats of the substrings of the genome is determined. The repeats and complementary palindromes uses hamming distance where as the non repeat parts uses arithmetic 2 compression or Context Tree Weighting.

GenBit Compress[13] is a compression Tool for compressing the genomic sequences. A new principle applied here is allocating binary bits for parts of DNA sequences. This approach differs from other approach by considering only the exact repeats and encoding them rather than considering the approximate repeats.

Differential compression algorithm[14] is done by considering the likeliness of the genetic sequence repository. Every sequence is not stored separately but a storage is created only for particular data. The data encloses cited sequences, differences and their locations.

DNABIT compress[12] involves removal of redundancy from the genomic sequences so that storage is made competent. Here, both the repetitive and non repetitive regions are compressed by allocating binary bits for smaller fragments.

GenCodex[10] was introduced to compress the genomic sequences present in multi cores and GPUs. The prime target of this approach is produce prominent throughput. GenCodex yields a speed up of 11 , 23 on multi cores and GPUs , respectively. It is better than GenBit and DNABit. It produce a compression ratio of 0.017bpb and 2.25 bpb for best and worst case, respectively.

DNACRAMP tool[11] is a technique proposed for compressing DNA sequences with or without duplicates. Here, the DNA sequences are encoded in bits. The sequence is partitioned into n/4 sections. The quadrupled sections are partitioned into sub partitions followed by assignment of header and trailer. The terminals are grouped to form a cluster. DNACRAMP does not use dynamic programming and encodes each base by 1.19 bits.

Biocompress[25] is a lossless compression algorithms for biological sequences. It is based on regularities which determines and analyze the duplicates of substring that occur in the prior . It is followed by encoding with repeat length and position of prior occurrence.

Seed based compression technique[5] was designed to compress DNA sequences that utilize the substitution procedure. Initially, the repeat structure present in the DNA sequences are determined by forming a offline dictionary. The dictionary posses the knowledge of duplicates and mismatches present in the sequence. This technique considers only the promising mismatches.

High throughput compression[6] classifies and provides an idea of existing compression mechanisms designed particularly for biological sequences. This paper will also provide the achievements of those techniques.

Referential compression algorithm[7] introduces an innovative procedure to compress the genomic sequences by references considered. Here, set of input sequences are chosen for which reference is determined. A reference is combination of value and key.

DNA sequence compression algorithm[8] uses Extended -ASCII depiction. Here, the DNA sequences considered are represented by extended ASCII codes . The processed sequences are encoded using Run length procedure.

3. Proposed work

Proposed work represents the Genomic Code based compression for the given text file. Here, the text file is reconstructed into a DNA sequence format. Later, the sequence of data is compressed. The whole process of the system can be defined in two major phases as shown in FIGURE I. In phase I, the text file converted to ASCII file and then to binary representation. The binary file is then represented in DNA sequence format. In phase II, the DNA sequence is bit reduced and expressed in binary format. The binary form of representation is converted to hexadecimal format. It is followed by encoding using Huffman codes.

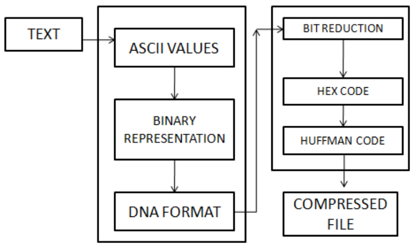

Figure I: Block Diagram of text compression

Algorithm

```
ALGORITHM TEXT Compression ( TEXT )
//ALGORITHM  to ASCII ( TEXT )
// ALGORITHM  to Simplify ASCII ( ASCII file )
// ALGORITHM  to Bin Rep ( cASCII file)
// ALGORITHM  to DNA ( Bin file )
// ALGORITHM Bit  reduction  ( DNA file )
// ALGORITHM Hex code ( BR file )
// ALGORITHM  Huff code ( Hex file )
ALGORITHM  to ASCII ( TEXT )
```

Precondition :Text file only with upper/ lower case alphabets and punctuation marks

Input :A simple text file

Output :A text file with ASCII values

```
   While not end of TEXT file do
       Read every character from TEXT file and assign to ' ch'
       ch = ASCII of ch
       Write ' ch' to ASCII file
   [End of While]
ALGORITHM  to Simplify ASCII ( ASCII file )
   Input          :An input ASCII file
   Output         :A modified ASCII file as cASCII file
   While not end of file do
       Read every character from ASCII file and assign to 'ch'
       If ch=13 then
              Write '0' to cASCII file
       else if ( 31 < ch < 127 )
              x = ch-31
              Write 'x' to cASCII file
       else
       Write  '96' to cASCII file
       [End of If]
   [End of While]
ALGORITHM  to Bin Rep ( cASCII file)
   Input          :Input  cASCII  file
   Output         :A binary Binfile
   While not end of file do
       Read every value from cASCII file and assign to 'ch'
       Convert 'ch' into Binary value of 7 bit
       Write 'ch' to Binfile
   [End of While]
ALGORITHM  to DNA ( Binfile )
   Input          :An input Binfile
   Output         :A DNAfile
Assign
       A     =     '00'
       T     =     '01'
       G     =     '10'
       C     =     '11'
If  ch = '00' then
       Write 'A' to the DNA file
else if ch='01'   then
       Write 'T' to DNA file
else if       ch='10'   then
       Write 'G' to DNA file
else
       Write 'C' to DNA file
[End of If]
[End of While]
ALGORITHM Bit  reduction   ( DNAfile )
Input          :A DNA file with A, T, G, C as bases.
Output         :A Bit reduced BR file.
While not end of DNA file do
       Read first character from DNA file
       Write corresponding assigned Bit code to BR file
       if ( first character == second character )
              Write '0' to BR file
       else
              Write corresponding Bit code to BR file
       [End of If]
[End of While]
ALGORITHM Hex code ( BR file )
Input          :A Bit Reduced binary BR file
Output         :A reduced Hex file
While not end of BR file do
       Read 4 bit values and assign to 'ch'
       If ch = 0000 then
              Write '0' to Hex file
       else if ch=0001     then
              Write    '1' to Hex file
       else if      ch=0010   then
              Write    '2' to Hex file
       else if      ch=0011 then
              Write    '3' to Hex file
       else if      ch=0100 then
              Write    '4' to Hex file
       else if      ch=0101 then
              Write    '5' to Hex file
       else if      ch=0110 then
              Write    '6' to Hex file
       else if      ch=0111 then
              Write    '7' to Hex file
       else if      ch=1000   then
              Write    '8' to Hex file
       else if      ch=1001 then
              Write    '9' to Hex file
       else if      ch=1010 then
              Write    'A' to Hex file
       else if      ch=1011   then
              Write    'B' to Hex file
       else if      ch=1100  then
              Write    'C' to Hex file
       else if      ch=1101 then
              Write    'D' to Hex file
       else if      ch=1110   then
              Write    'E' to Hex file
       else
              Write    'F' to Hex file
       [End of If]
[End of While]
ALGORITHM  Huff code ( W, n )
Input          : A list 'W' of 'n' positive weights from Hex
                 file
```

```
Output      :An extended binary tree T with weights
             taken from 'W' that gives minimum
             weighted path length
Create list 'F' from singleton trees formed from element of
'W'
While 'F' has more than one element do
    Find T1, T2 in that have minimum values associated
    with their roots
    Construct new tree T by creating a new node and setting
    T1 and T2 as its children
    Let the sum of the values associated with the roots of T1
    and T2 be associated with root of T
    Add T to F
    Huffman code : = Tree stored in F
[End of While]
```

Modules of the proposed algorithm

The whole process is divided as 2 phases. The modules of the proposed system are Conversion of text file to DNA sequence and Compression of DNA sequence by a genomic code compression procedure.

Conversion of text file to DNA sequence

This module explains the reconstruction of text files into DNA sequence. The input text file is converted to genomic code in three steps. Initially, each of the character in text file is replaced with its corresponding ASCII value. In order to depict the ASCII value with minimal (7 bit) code, the ASCII values are simplified and then presented using binary format. Then, each two bit of the binary file is replaced with its corresponding DNA base.

Example 1

Input text	: day
ASCII value	: 100 97 121
Simplification of ASCII	: 69 66 90
Binary representation	:1010101 0111010 10010110
Assign	: A=00;T=01;G=10; C=11.
DNA sequence	:GGGGCGGGTTG

Compression of genomic data

This module explains the bit reduction of DNA sequence and representing them in binary format. Binary code uses the digits of 0 and 1 (binary numbers) to represent the DNA bases. Each base or the symbol gets a bit value assignment. The bit string can corresponds to the DNA bases.
The four DNA bases are : {A,T,G,C}
Assign: A=00; T=01; G=10; C=11.
Initially, for the first base of the DNA sequence the assigned bit code is given. Then, the next base is compared with its prior base. If they are same, it is represented by '0' else it is represented by its corresponding binary code. Thus, we get a stream of binary output.

- *Conversion of Binary Format to Hexadecimal Format*

This sector describes the conversion of binary values to hexadecimal values. It is useful and effective approach for compressing long binary strings. Here, both bases are powers of 2. Thus, it is a simple procedure than other general conversions. For converting long binary strings, partition the entire string of binary numbers into groups of four bits each. Hexadecimal converts 4 bits into one hexadecimal unit. So, in order to convert the number, first divide the entire bit sequence. For each four digit group, convert the 4 bit binary number to its equivalent hexadecimal value.
In binary to hexadecimal Conversion, the binary values for 0 to 9 will take the same values as their hex values. The binary values from 10 to 15 are represented as characters from A to F.
Example 1:
Conversion of binary number 10110101 to a hexadecimal number

Divide into groups of 4digits	:	1011	0101
Convert each group to hex digit	:	B	5

- *Encoding By Huffman Codes*

This module explains the encoding of hexadecimal string by Huffman codes. It is a lossless data compression algorithm. The procedure is to allocate variable-length codes to input hexadecimal characters. The lengths of the assigned codes are based on the frequencies of the corresponding characters. The character that appears the most gets the smallest code and the character that appears the least gets the largest code. In a variable-length code words may have different length as shown in TABLE I.

Table I: Variable Code Length

HEX-BASES	A	1	9	C
FREQUENCY	7	60	85	20
A VARIABLE CODE	111	10	0	110

Given a hexadecimal string and its corresponding variable code as shown in TABLE I, it is simple to encode the hex string just by replacing the hex characters by the code words.

Input	:	1A99CA1
Output	:	10 111 0 0 110 111 10

Applications

- It reduces the storage space required by the text files in the database.
- Processing costs of text files can be economized.
- Transmission costs of stored data can be diminished.
- Provision of quick access to any record and superior functionality.

Result analysis of sample text

Here, a sample text "Good day" is considered. The step wise procedure of proposed compression algorithm is explained keeping the sample text as base. In phase I, the sample text is converted to a model DNA fragment with A,T,G and C as bases. It is done using ASCII conversion and symbolize them in binary format. In phase II, each of base in DNA sequence is given a corresponding bit code. TABLE II explains Bit reduction of sample DNA sequence. It is followed by restructuring the sequence in hexadecimal format and Huffman encoding.

Example

Sample text	:Good day
ASCII conversion	: 71 111 111 100 32 100 97 121
Simplification of ASCII	: 40 80 80 69 1 69 66 90
Binary representation	: 0000010 1100111 0011010 0101110 1010111 1111101 1010101 01110110010110
The four DNA bases are	: { A,T,G,C}
Assign	: A=00 ; T=01 ; G=10 ; C=11
DNA Sequence	: AATTGTCACTAGCGG GCCCCTGGGGCGGGT TG
Bit Reduction	:

Table II: Bit Reduction Of Bases

A	A	T	T	G	T	C	A	C	T	A	G	...
00	0	01	0	10	01	11	00	11	01	00	10	...

Reduced Binary format of text in DNA format:
000010100111001101001011100101100001100
00011100001010

Bits in partition :
0000 1010 0111 0011 0100 1011 1001 0110 0001 1000 0111 0000 1010
Hex format : 0A734B961870A

Table III: Huffman Code

HEX-BASES	0	A	7	3	4	B	9	1	8
FREQUENCY	2	2	2	1	1	1	1	1	1
A VARIABLE CODE	000	010	101	0010	1101	0110	1000	1111	0011

From TABLE III the corresponding Huffman Codes for the hexadecimal values were obtained.
Huffman Codes for the sample text:

000 010 101 0010 1101 0110 1000 1111 0011 101 000 010

Input sample text	= 8 Characters
	= 8* 8 Bits
	= 64 Bits
Proposed Work	= 42 Bits
8 Characters	= 42 Bits
1 Character	= 42/8
	= 5.25 bits per character

$$Compression\ Ratio = \frac{Uncompressed\ size}{Compressed\ size}$$

$$= \frac{64}{42}$$

The above ratio shows that around 35 % of memory is saved by using our new algorithm technique and which is the considerable amount of compression compared with the existing approach given in the literatures.

4. Conclusion

In this paper, the existing work related to the compression of text files and biological sequences are discussed. It is followed by designing of a new algorithm to compress the substantial text file by genomic code compression procedure. The proposed algorithm yields better results with significant improvement in the compression ratio.

References

[1] Radescu R & Pasca S, "String Matching in Text Compression", *ECAI 2017-International Conference, Targoviste, Romania, 9th edition*, (2017).
[2] Dufourq E & Bassett BA, "Text Compression for Sentiment Analysis via Evolutionary Algorithms", *PRASA-RobMech International Conference, Bloemfontein, South Africa*, (2017).
[3] Conrad KJ & Wilson PR, "Grammatical Ziv-Lempel Compression: Achieving PPM-Class Text Compression Ratios with LZ-Class Decompression Speed", *Data Compression Conference (DCC)*, (2016).
[4] Barua L, Dhar PK, Alam L & Echizen I, "Bangla text compression based on modified Lempel-Ziv-Welch algorithm", *International Conference on Electrical, Computer and Communication Engineering (ECCE)*, (2017), pp.855-859.
[5] Eric PV, Gopalakrishnan G & Karunakaran M, "An Optimal Seed Based Compression Algorithm for DNA Sequences", *Advances in Bioinformatics*, (2016).
[6] Zhu Z, Zhang Y, Ji Z, He S & Yang X, "High - throughput DNA sequence data compression", *Briefings in bioinformatics*, (2015).
[7] Mehta K & Ghrera SP, "DNA compression using referential compression algorithm", *Eighth International Conference Contemporary Computing (IC3)*, (2015).
[8] Saada B & Zhang J, "DNA Sequences Compression Algorithm Based on Extended-ASCII Representation", *Proceedings of the world congress on engineering and computer science*, (2015).
[9] Baloul FM, Abdullah MH & Babikir EA, "ETAO: Symbol Mapping Tranformation Method for Text Compression", *International Conference on Computer Electrical and Electronics Engineering (ICCEEE)*, (2013), pp.384-389.
[10] Satyanvesh D, Balleda K & Padyana A, "GenCodex- A Novel Algorithm for Compressing DNA seuences on Multi-cores and GPUs", *Proc. IEEE, 19th International Conf. on High Performance Computing (HiPC)*, (2012).
[11] Prasad VH & Kumar PV, "A New Revised DNA Cramp Tool Based Approach of Chopping DNA Repetitive and Non- Repetitive Genome Sequences", *International Journal of Computer Science Issues (IJCSI)*, Vol.9, No.6,(2012), pp.448-454.
[12] Rajeswari PR & Apparao A, "DNABIT Compress-Genome compression algorithm", *Bioinformatics*, Vol.5, No.8,(2011), pp.350-360.
[13] Rajeswari PR & Apparao A, "GenBit Compress Tool (GBC): A Java-Based Tool To Compress DNA Sequences and Compute Compression Ratio (BITS/BASE) Of Genomes", *International Journal of Computer Science and Information Technology*, Vol.2, No.3,(2013), pp.181-191.
[14] Afify H, Islam M, Wahed MA & Kadah YM, "Genomic sequences differential compression model", *International Journal of Computer Science and Information Technology*, Vol.3, (2011), pp.145-154.
[15] Cao MD, Dix TI, Allison L & Mears C, "A simple statistical algorithm for biological sequence compression", *Proceedings of the Data Compression Conference*, (2007), pp.43-52.
[16] Myung JI, Navarro DJ & Pitt MA, "Model selection by normalized maximum likelihood", *Journal of Mathematical Psychology*, Vol.50, No.2, (2006), pp.167-179.
[17] Behzadi B & Le Fessant F, "DNA compression challege revisited: a dynamic programming approach", *Proceedings of the Annual Symposium on Combinatorial Pattern Matching*, (2005).
[18] Abel J & Teahan W, "Universal Text Preprocessing for Data Compression", *IEEE Transactions On Computers*, Vol.54, No.5, (2005).
[19] Ma B, Tromp J & Li M, "PatternHunter: fast and more sensitive homology search", *Bioinformatics*, Vol.18, No.3, (2002), pp.440-445.
[20] Chen X, Li M, Ma B & Tromp J, "DNACompress: fast and effective DNA sequence compression", *Bioinformatics*, Vol.18, no. 12, (2002), pp.1696-1698.
[21] Chen X, Kwong S & Li M, "Compression algorithm for DNA sequences and its applications in genome comparison", *Proceedings of the 4th Annual International Conference on Computation Molecular Biology*, (2000).
[22] Matsumoto T, Sadakane K & Imai H, "Biological sequence compression algorithms", *Genome Informatics*, (2000), pp.43-52.
[23] Loewenstern D & Yianilos PN, "Significantly lower entropy estimates for natural DNA sequences", *Journal of Computational Biology*, Vol.6, No.1, (1999), pp.125-142.
[24] Grumbach S & Tahi F, "A new challenge for compression algorithms: genetic sequences", *Information Processing & Management*, Vol.30, No.6,(1994), pp.875-886.
[25] Grumbach S & Tahi F, "Compression of DNA sequences", *Proceedings of the IEEE Symposium on Data Compression*, (1993).