DOI 10.29042/2017-1491-1497

Helix Vol. 8: 1491-1497

DNA Sequence Decompression Using Bitmap Matrix & Wavelet Transformation in Image Processing

[*1] Raju Bhukya, [2] B.Vamsi Viswanath, [3] Y.Mahendra Kumar, [4] D. Swathi Kiran, [5] Prashant Bagdia, [6]Girdhar L, [7] Neelesh.G

National Institute of Technology, Warangal

E mail: drrajunitw@gmail.com

Received: 20th April 2017, Accepted: 28th April 2017, Published: 1st May 2017

Abstract

Genomic science is currently encountering an unstable increment of information and quick improvement of sequencing innovation. Till date, various calculations have been created for compacting genomic groupings. The vast majority of which regard genomes as one-dimensional content strings and pack them in view of word references or likelihood models. Some Compression calculations can't pack DNA successions however just extends in size. This paper proposes a novel approach for genome compression, which changes the genomic successions to double grouping utilizing Genbit Algorithm and develops a grid from the encoded arrangement. On the off chance that the length of the Encoded grouping is not an impeccable square then we attach a few bits to the arrangement to make its length a flawless square. We include paired bits by checking the bit which showed up the most in the encoded grouping. On the off chance that both bits seemed same number of times we pick '0' and connect the bit over and again at its end to make its length an immaculate square for framework development. Further we take the framework as Grayscale Image and Compress by utilizing wavelet change Image Compression Technique. The proposed Compression Algorithm" accomplishes the best compression proportions for whole Genome DNA successions. It is compared with existing ones and is found to accomplish better compression results.

Keywords

Compression, Encode, Decode, DNA Succession

Introduction

The genome is an organism's complete set of DNA, let in all of its genes. The genome of a living thing comprise all ancestral info encrypt in DNA. Each genome contains all of the info required to physique and hold that living thing. The two fields in deoxyribonucleic acid sequencing which has seen a fast exploitation was speed of processing and inexpensive price which leads to tremendous gain in genomic collection. Future Genesis sequencing(NGS)[4] ,a Broad through put sequence Engineering and Idiosyncratic molecule sequencing[5] do it easy for galactic measure of one-on-one genomes to be processed in a calendar week or less period of time and costs less than US$10000[6]. The thought-provoking, wide-open investigation of problems have created the necessity to store and transfer rattling sizable amount of data. The starring problem colligate with vast data depository. For instance, the storehouse of one 2009 human reference genome needs 905 MB using the tar.gz compaction change [7]. This means that we have to expend more or less 30 minutes in downloading it via a communication system of 4Mb/s bandwidth, and about 5 hours for ten such genomes [1]. DNA sequences comprises of four bases {A, C, T, G} and each base (symbol) can be depicted by two bits. Many standardized text compression tools, such as compress (Lempel–Ziv–Welch "LZW"), gzip (Lempel–Ziv "LZ" + Huffman) and bzip2 (Burrows–Wheeler transform "BWT" + Move-to-Front "MTF" + Huffman), can't be used to constrict these DNA sequences [28]. According to standardized data, the average compaction ratio is 2.185 bpb "bits per base" for compress, 2.271 bpb for gzip, and 2.138 bpb for bzip2 except for "HUMGHCSA" sequence, where the mean compression ratio is 1.729 bpb [29].

The genome compression algorithms are categorized into two types .One type of genomes is that they don't use cite genomes like DNA Compaction [9], Gen Compress [8], GeNML [10] and so on, among which XM [11] is best one regarding compression proportion. It is a factual technique that uses a few "specialists" to foresee the likelihood circulation of the following image and encodes images by math coding. Despite the fact that XM performs best contrasted with other sans reference compression calculations[12]. Coming to the next kind of genome compression calculation, they pack different genomes by picking one of them as a source of perspective genome and look at the similitude between genomes. These reference based calculations are exceptionally helpful in specific situations where all genomes are indistinguishable like homosapiens which are 99% indistinguishable [14] and GRS [7] was likewise proposed in view of this reality. In any case, it declines to pack a few genomes that are excessively not quite the same as each other [15]. Another reference-based technique, named GReEn [16], finds the likelihood appropriations of the bases by a versatile model and a static model, and after that encodes them utilizing a number juggling coder. In 2010, Kuruppu et al.

proposed the RLZ [17] calculation, which keeps up a self-record for the reference arrangement and after that packs alternate successions utilizing a methodology like LZ77 [18]. RLZ-select [19] is an enhanced rendition of RLZ, which utilizes a non-insatiable parsing technique with the assistance of addition exhibit. In spite of the fact that the postfix cluster encourages non-ravenous parsing, the time has come devouring to construct. There are additionally some other DNA compression calculations going for supporting arbitrary get to, including COMRAD [12] and RLCSA [20]. In any case, a large portion of them are extremely tedious, and along these lines are not viable for information appropriation.

For instance, COMRAD needs a few disregards the genomes to be compacted, which is an exceptionally costly process. As of late, [6] proposed a compression calculation that backings coordinate calculation (e.g. Impact and BLAT) on the compacted genomes. In any case, its compression proportion is not ideal. Take note of that a few calculations were proposed for packing genomes of particular organizations. For instance, TGC [21] packs the VCF arrange [22] documents that contain data of variations (e.g., SNP, inclusion, cancellation and basic variations). BEETL-fastq [23] packs FASTQ [24] records and backings k-mer inquiries over the compacted arrangements. NGC [25] was created for SAM organize [26] documents. As both SAM and FASTQ records contain quality scores for nucleotides, compression techniques for quality scores were additionally explored as of late [27]. All the above calculations go for packing the genome arrangement as a one dimensional string however we propose another compression calculation for genome grouping as a two dimensional string by changing over it into a bitmap picture and further compacting by wavelet change of picture compression strategy.

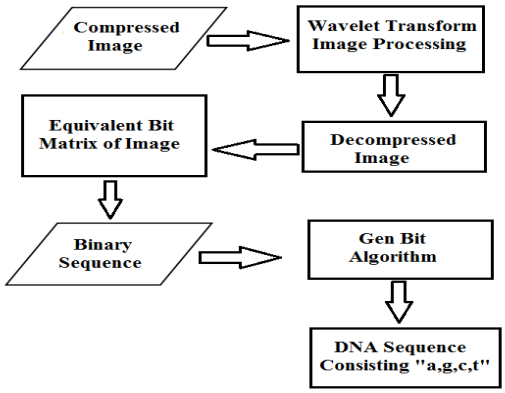
There are two parts of this algorithm first one being the Compression and second one Decompression. We First take the Input DNA sequence consisting of “A, G, C, T” We decode it by two bit encoding by replacing a = “00”, g = “01”, c = “10”, t = “11”. We encrypt the DNA sequence by Encryption algorithm.

Our Encrypt Algorithm converts Original DNA Sequence to Binary Encoded Sequence. This Algorithm takes Original DNA Sequence as Input and generates the Binary Sequence. The DNA Sequence is divided into chunks of size ‘8’. Then it compares left half and right half of the chunk. If both halves are same we encode left half and then we add ‘1’ at its end. If both halves are not equal we encode left half and append ‘0’ then we encode right half and concatenate this encoded right half to output along with ‘0’ at its end. Then it repeats the same process with next chunk after Encrypting the DNA sequence we transform the linear bit sequence into Matrix. We add extra 0’s and 1’s (depending on which count is maximum in the sequence) to the linear bit stream to make the length of the DNA sequence as a nearest perfect square so that we can form the Square Matrix. We compress the matrix into Image by Wavelet Transform technique.

II. DECOMPRESSION USING IMAGE PROCESSING

In this Decompression first we take the available compressed Gray Scale Image as Input. We get the DNA sequence as output which consists {a ,g ,c, t}.Our Input Gray-scale Image undergoes Discrete Wavelet Transform of Image Decompression. From the Compressed image we get the Decompressed original image. From this original gray-scale image the proposed algorithm retrieves the equivalent binary matrix. The binary Matrix is converted into linear sequence of 0’s and 1’s.To this binary sequence we again apply another decryption algorithm (algorithm 2) which is a part of Genbit algorithm. The output of the decryption algorithm gives us the DNA sequence consisting of “A, G, C, T ” letters. This Decrypt Algorithm converts Binary Encoded Sequence to Original DNA Sequence. This Algorithm takes Binary Sequence as Input and generates the Original DNA Sequence. The Binary Sequence is divided into chunks of size ‘9’ and checks its 9th bit. If it’s ‘1’, it decodes the first 8 characters and appends it two times to the output sequence. If it’s ‘0’, it appends the decoded sequence only once.

Fig 1: The flowchart for decompression technique using Image Processing



Algorithm 1: The Decrypt Algorithm(Genbit algorithm)

Input: S - Input Binary Sequence;
n - length of Binary Sequence derived from S ;

Output: b- DNA Sequence ;

```
1 l=0 , b="";
2 for i=0 to n/9 do
3    if (l+8)<n then
4          if (S.charAt(l+8)) = '1' then
```

  ISSN 2319– 5592 (Online)

```
5            b+ = decode(S.substring(l,l+8));
6            b+ = decode(S.substring(l,l+8));
7            l=l+9;
8        else
9            b+ = decode(S.substring(l,l+8));
10           l=l+9;

11 if l<n then
12   b+ = decode(S.substring(l));
13 return b;
```

III. EXPERIMENTAL RESULTS

Our User Interface consists of an Input Box for DNA Sequence .It has "Encode" button, on clicking it encodes the Input DNA Sequence to Binary Sequence. It displays the Binary Sequence and Compression ratio and then "Decode" Button appears. On clicking it decodes the Binary Sequence to Input DNA Sequence. The genomes can be downloaded from the website (http://www.ncbi.nlm.nih.gov/Genbank/). The genome sequence can be of any size**.** Here on giving the DNA sequence as Input we get the encoded sequence as output along with calculated compression ratio.

Example 1:

Consider an Input DNA Sequence
"TTAAGGACCCCATGCCCTCGAATAGGCTTG
AGCTTGCCAATTAACGCGCACGGGCTGGCC
GGGCGTATAAGCCAAGGTGTAGTGAGGTTG
CATTATACATGCCGGCTTGTGATTAACGCAT
GCCATAGGACGGTTAGGCTCAGAACCCGCA
ACCAATACACGTGATTTTCTCGTC "

After Encoding the given DNA Sequence data, we get the Binary Sequence as shown below which consists of 0's and 1's.
"11110000001010010010101000011011010010111
00100000110000101101101101000101011110101 0
10000001111000001001100101000100100101101 1
00101101000101011000111001100000011001000 0
00100111011100001110100001011101101100001 1
11001100010001100110100100110111100111010 0
01111000001001100001101101000011000100100 1
00100111110000101101101000010000010101000 1
10000001010000001100100001001110100011111 1
01110111000111100"

Compression Ratio Calculated is 2.25.It is calculated by the formula .Total number of Bits(R) = 9/4 (n-t) + 2(t) - 9 (¡).
Compression Rate = R/n

Matrix generated

Fig. 2: Matrix generated from the encoded sequence of given DNA sequence

1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1
1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1
0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0 1 0 1 1 1 1 0 1 0 1 0 1
0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0
1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 1 0
1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0
0 1 0 0 1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 1 1
1 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 1
0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 0 0
1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0
0 1 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1
0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1
0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1
1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0

To form a square matrix first find the length of the sequence and count no of 1's and 0's if the sequence has maximum no of 1's add 1's to the output sequence until the length of the sequence is equal to its nearest perfect square and vice-versa.
Now convert the obtained sequence whose length is a perfect square to a square matrix of order sqrt(n)*sqrt(n),where n is the length of new binary sequence. Now by using Wavelet Transform of Image compression compress this matrix again which is beneficial for further compression is takes place.

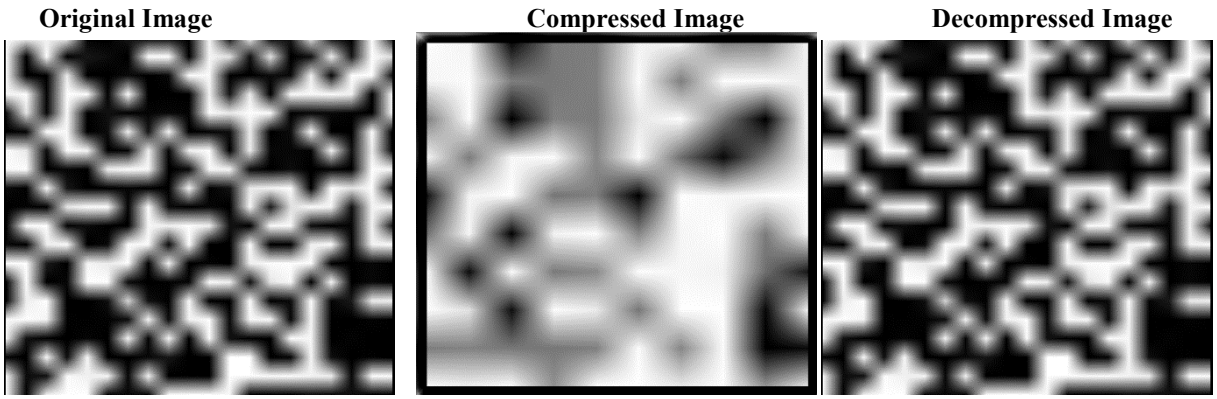


Fig .3(a): The original Image obtained from the above generated matrix (image is magnified due to it's micro size).
The equivalent bitmap image of the above generated matrix. Any Matrix containing only 0's and 1's can be converted to image by using Image Processing Techniques.
Fig .3(b) : The Compressed Image obtained by compressing the above generated matrix using Wavelet Transform

Image Compression Technique.(image is magnified due to it's micro size)
Fig.3(c) : Decompressed Image obtained from Decompressing the compressed Image using Image Processing (image is magnified due to it's micro size). The original Image is compressed using the Wavelet Transform Image Compression Technique.

Compressed Matrix

Fig 4: Compressed Matrix obtained from compressing the above Matrix

1.50000000000000 1.50000000000000 0 0.500000000000000 0.500000000000000 1.50000000000000 1.00000000000000
0.500000000000000 0.500000000000000 1.50000000000000 1.50000000000000 1.00000000000000 0.500000000000000
0.500000000000000 0.500000000000000 1.00000000000000 0.500000000000000 1.50000000000000 1.50000000000000
1.00000000000000 0.500000000000000 1.50000000000000 0 0.500000000000000 0.500000000000000
1.00000000000000 1.50000000000000 0.500000000000000 0 1.00000000000000 2.00000000000000 0.500000000000000
1.00000000000000 1.50000000000000 0.500000000000000 1.50000000000000 0.500000000000000 0 0.500000000000000
1.50000000000000 1.00000000000000 1.00000000000000 0.500000000000000 0.500000000000000 0 1.50000000000000
1.50000000000000 1.50000000000000 1.50000000000000 0.500000000000000 1.50000000000000 0 1.50000000000000
1.50000000000000 0.500000000000000 1.00000000000000 1.00000000000000 0.500000000000000 1.50000000000000
1.50000000000000 2.00000000000000 0.500000000000000 0.500000000000000 1.00000000000000 1.50000000000000
1.50000000000000 0.500000000000000 0 1.50000000000000 1.00000000000000 0 1.00000000000000
1.00000000000000 0.500000000000000 1.50000000000000 1.00000000000000 0 1.00000000000000
0.500000000000000 0.500000000000000 0.500000000000000 0.500000000000000 0.500000000000000 1.00000000000000
0.500000000000000 1.50000000000000 0 0 1.00000000000000 1.50000000000000 1.50000000000000
0.500000000000000 1.00000000000000 1.50000000000000 1.00000000000000 1.00000000000000 0.500000000000000
1.00000000000000

The compressed Image is converted to Matrix using Image Processing Technique. It gets converted to a Matrix containing decimal numbers. Here the original size of the uncompressed Matrix was 706kb and the compressed image is 332kb.Now we apply the decompression Algorithm for the above compressed image and we retrieve the original matrix using analog Discrete Wavelet Transform of Image Processing. The entire execution process is performed using Matlab and results were analyzed.

Fig. 5: Decompressed Matrix obtained from Image Processing

1.00000000000000 1.00000000000000 1.00000000000000 1.00000000000000 0 0 0 0 0 0
1.00000000000000 1.11022302462516e-16 1.00000000000000 0 0 1.00000000000000 0 0 1.00000000000000
1.11022302462516e-16 1.00000000000000 1.11022302462516e-16 1.00000000000000 1.11022302462516e-16 0 0 0
1.00000000000000 1.00000000000000 0 1.00000000000000 1.00000000000000 0 1.00000000000000 0
1.00000000000000 0 1.00000000000000 1.00000000000000 1.00000000000000 1.11022302462516e-16 0
1.00000000000000 0 0 0 0 0 1.00000000000000 1.00000000000000 0 0 0 1.11022302462516e-16
1.00000000000000 1.11022302462516e-16 1.00000000000000 1.00000000000000 0 1.00000000000000
1.00000000000000 0 1.00000000000000 1.00000000000000 0 1.00000000000000 0 0 0 0 1.00000000000000 0
1.00000000000000 0 1.00000000000000 1.00000000000000 1.00000000000000 1.00000000000001.00000000000000
0 1.00000000000000 1.11022302462516e-16 1.00000000000000 0 0 0 0 0 0 1.00000000000000
1.00000000000000 1.00000000000000 1.00000000000000 0 0 0 0 1.00000000000000 0 0
1.00000000000000 1.00000000000000 0 0 1.00000000000000 0 1.00000000000000 0 0 0
1.00000000000000 1.11022302462516e-16 0 1.00000000000000 0 0 1.00000000000000 0 1.00000000000000
1.00000000000000 0 1.00000000000000 1.00000000000000 0 1.11022302462516e-16 1.00000000000000 0
1.00000000000000 1.00000000000000 1.11022302462516e-16 1.00000000000000 0 0 0 1.00000000000000 0
1.00000000000000 1.11022302462516e-16 1.00000000000000 1.00000000000000 0 0 0 1.00000000000000
1.00000000000000 1.00000000000000 0 0 1.00000000000000 1.00000000000000 0 0 0 0 0 0
1.00000000000000 1.00000000000000 1.00000000000000 1.11022302462516e-16 1.00000000000000 1.00000000000000
1.00000000000000 1.11022302462516e-16 0 0 0 1.00000000000000 1.00000000000000 1.00000000000000 0
1.00000000000000 0 0 0 0 1.00000000000000 1.11022302462516e-16 1.00000000000000 1.00000000000000
1.00000000000000 1.11022302462516e-16 1.00000000000000 1.00000000000000 0 1.00000000000000
1.00000000000000 1.11022302462516e-16 0 0 1.11022302462516e-16 1.00000000000000 0 0 0 1.00000000000000
1.00000000000000 1.00000000000000 0 0 1.00000000000000 1.00000000000000 0 0 0 1.00000000000000
1.11022302462516e-16 0 0 1.00000000000000 1.00000000000000 0 0 1.00000000000000 1.00000000000000
1.11022302462516e-16 1.00000000000000 0 0 1.00000000000000 0 0 1.00000000000000 1.00000000000000
0 1.00000000000000 1.00000000000000 1.00000000000000 1.00000000000000 0 0 1.00000000000000
1.00000000000000 1.00000000000000 0 1.00000000000000 0 0 0 1.00000000000000 1.00000000000000
1.00000000000000 1.00000000000000 0 0 0 1.11022302462516e-16 1.00000000000000 0
1.00000000000000 1.00000000000000 1.00000000000000 0 0 0 0 1.00000000000000 0 0
1.00000000000000 1.00000000000000 0 0 0 0 1.00000000000000 1.00000000000000 1.11022302462516e-16
1.00000000000000 1.00000000000000 1.11022302462516e-16 1.00000000000000 0 0 0 1.11022302462516e-16
1.00000000000000 1.00000000000000 0 0 1.00000000000000 0 0 1.00000000000000 0 0
1.00000000000000 1.11022302462516e-16 0 1.00000000000000 0 0 1.00000000000000 1.00000000000000
1.00000000000000 1.00000000000000 1.00000000000000 0 0 0 0 1.00000000000000 0 0 1.00000000000000
1.00000000000000 0 0 0 0 1.00000000000000 1.00000000000000 0 1.00000000000000 0 0 0
1.00000000000000 0 0 1.00000000000000 1.00000000000000 0 0 0 0 0 0 0 0
0 0 1.00000000000000 1.00000000000000 0 1.11022302462516e-16 1.00000000000000 0 0 0 0
1.00000000000000 0 1.11022302462516e-16 1.00000000000000 1.00000000000000 1.00000000000000 0
1.00000000000000 0 0 1.11022302462516e-16 1.00000000000000 1.00000000000000 1.00000000000000
1.00000000000000 1.00000000000000 1.00000000000000 0 1.00000000000000 1.00000000000000
1.00000000000000 0 1.00000000000000 1.00000000000000 1.00000000000000 1.11022302462516e-16 0 0
1.00000000000000 1.00000000000000 1.00000000000000 0 0 0 0 0 0 0 0

The above obtained matrix should be equal to the original matrix which is given as the input dataset. Considering e-16 value as 0 we get matrix containing zeros and ones. The matrix is almost same as the bitmap image. The performance of our algorithm is determined by similarity between the above decompressed image and the original Image given as Input.

Fig.6: Difference matrix of original and the decompressed matrix, in short this is our error matrix.

The error we got in this algorithm is -5.4734e-14

-4.44089209850063e-16 -4.44089209850063e-16 -4.44089209850063e-16 -4.44089209850063e-16 0
0 0 0 0 0 -2.22044604925031e-16-1.11022302462516e-16 -2.22044604925031e-
16 0 0 -2.22044604925031e-16 0 0 -2.22044604925031e-16 -1.11022302462516e-
16 -2.22044604925031e-16 -1.11022302462516e-16 -2.22044604925031e-16 -1.11022302462516e-16 0
0 0 -2.22044604925031e-16 -2.22044604925031e-16 0 -4.44089209850063e-
16 -4.44089209850063e-16 0 -2.22044604925031e-16 0 0 -2.22044604925031e-16 0
-4.44089209850063e-16 -4.44089209850063e-16 -2.22044604925031e-16 -1.11022302462516e-
16 0 -2.22044604925031e-16 0 0 0 0 0 -2.22044604925031e-16-
2.22044604925031e-16 0 0 0 -1.11022302462516e-16 -2.22044604925031e-16 -
1.11022302462516e-16 -2.22044604925031e-16 -2.22044604925031e-16 0 -4.44089209850063e-16 -
4.44089209850063e-16 0 -2.22044604925031e-16 -2.22044604925031e-16 0 -2.22044604925031e-
16 0 0 0 -2.22044604925031e-16 0 -2.22044604925031e-16 0 -
4.44089209850063e-16 -4.44089209850063e-16 -4.44089209850063e-16 -4.44089209850063e-16 0
-2.22044604925031e-16 0 -2.22044604925031e-16 -1.11022302462516e-16 -
2.22044604925031e-16 0 0 0 0 0 0 -2.22044604925031e-16 -
2.22044604925031e-16 -4.44089209850063e-16 -4.44089209850063e-16 0 0 0 0
0 -2.22044604925031e-16 0 0 -4.44089209850063e-16 -4.44089209850063e-16 0
0 -2.22044604925031e-16 0 -2.22044604925031e-16 0 0 0 -
2.22044604925031e-16 -1.11022302462516e-16 0 -2.22044604925031e-16 0 0 -
2.22044604925031e-16 0 -2.22044604925031e-16 -2.22044604925031e-16 0 -2.22044604925031e-
16 -2.22044604925031e-16 0 -1.11022302462516e-16 -2.22044604925031e-16 0 -
2.22044604925031e-16 -2.22044604925031e-16 -1.11022302462516e-16 -2.22044604925031e-16 0
0 0 -2.22044604925031e-16 0 -2.22044604925031e-16 -1.11022302462516e-16 -
2.22044604925031e-16 -2.22044604925031e-16 0 0 0 -2.22044604925031e-16 -
4.44089209850063e-16 -4.44089209850063e-16 0 0 -4.44089209850063e-16 -
4.44089209850063e-16 0 0 0 0 0 0 -4.44089209850063e-16 -
4.44089209850063e-16
0 0 -2.22044604925031e-16 0 0 0 0 0 0 -
2.22044604925031e-16 0 0 -4.44089209850063e-16 -4.44089209850063e-16 -2.22044604925031e-
16 -1.11022302462516e-16 -4.44089209850063e-16 -4.44089209850063e-16 -
2.22044604925031e-16 -1.11022302462516e-16 0 0 0 -2.22044604925031e-16 -
2.22044604925031e-16 -2.22044604925031e-16 0 -2.22044604925031e-16 0 0 0
0 -2.22044604925031e-16 -1.11022302462516e-16 -4.44089209850063e-16 -
4.44089209850063e-16 -2.22044604925031e-16 -1.11022302462516e-16 -4.44089209850063e-16 -
4.44089209850063e-16 0 -2.22044604925031e-16 -2.22044604925031e-16 -1.11022302462516e-16 0 0
-1.11022302462516e-16 -2.22044604925031e-16 -4.44089209850063e-16 -
4.44089209850063e-16 -2.22044604925031e-16 0 0 -2.22044604925031e-16 -
2.22044604925031e-16 0 0 0 -2.22044604925031e-16 -1.11022302462516e-16 0 0
-4.44089209850063e-16 -4.44089209850063e-16 0 0 -4.44089209850063e-
16 -4.44089209850063e-16 -1.11022302462516e-16 -2.22044604925031e-16 0 0 -
2.22044604925031e-16 0 0 -2.22044604925031e-16 -2.22044604925031e-16 0 -
4.44089209850063e-16 -4.44089209850063e-16
-4.44089209850063e-16 -4.44089209850063e-16 0 0 -2.22044604925031e-16 -
2.22044604925031e-16 -2.22044604925031e-16 0 -2.22044604925031e-16 0 0 0
-4.44089209850063e-16 -4.44089209850063e-16 -4.44089209850063e-16 -
4.44089209850063e-16 0 0 0 0 -1.11022302462516e-16 -2.22044604925031e-16 0
0 -2.22044604925031e-16 -2.22044604925031e-16 0 0 0 0 -
2.22044604925031e-16 -2.22044604925031e-16 -1.11022302462516e-16 -2.22044604925031e-16 -
2.22044604925031e-16 -1.11022302462516e-16 -2.22044604925031e-16 0 0 0 -1.11022302462516e-
16 -2.22044604925031e-16 -2.22044604925031e-16 0 0 0 -2.22044604925031e-
16 0 0 -2.22044604925031e-16 0 0 -2.22044604925031e-16 -
1.11022302462516e-16 0 -2.22044604925031e-16 0 0 -2.22044604925031e-16 -
2.22044604925031e-16 -4.44089209850063e-16 -4.44089209850063e-16 -2.22044604925031e-16 0
0 0 0 -2.22044604925031e-16 0 -2.22044604925031e-16 -
2.22044604925031e-16 0 -4.44089209850063e-16 -4.44089209850063e-16 0 -
2.22044604925031e-16 0 0 0 0
-2.22044604925031e-16 0 0 0 0 0 -2.22044604925031e-16 0 -
2.22044604925031e-16 0 -2.22044604925031e-16 0 0 0 -4.44089209850063e-16 -
4.44089209850063e-16 0 0 0 0 0 0 -2.22044604925031e-16 0 -

2.22044604925031e-16 0 0 0 0 0 0 -2.22044604925031e-16 -2.22044604925031e-16 0 -1.11022302462516e-16 -2.22044604925031e-16 0 0 0 0 -2.22044604925031e-16 0 -1.11022302462516e-16 -2.22044604925031e-16 -4.44089209850063e-16 -4.44089209850063e-16 0 -2.22044604925031e-16 0 0 -1.11022302462516e-16 -2.22044604925031e-16 -2.22044604925031e-16 -2.22044604925031e-16 -2.22044604925031e-16 -2.22044604925031e-16 -2.22044604925031e-16 0 -2.22044604925031e-16 -2.22044604925031e-16 -2.22044604925031e-16 0 -4.44089209850063e-16 -4.44089209850063e-16 -2.22044604925031e-16 -1.11022302462516e-16 0 0 -2.22044604925031e-16 -2.22044604925031e-16 -4.44089209850063e-16 -4.44089209850063e-16 0 0 0 0 0 0 0 0

As the error is relatively very small we just ignore the decimal point values and take the values of matrix ignoring the decimal values and convert it into linear sequence and apply the decryption algorithm of Genbit algorithm to generate the DNA sequence consisting of "A, G, C, T".

CONCLUSION AND FUTURE WORK

The algorithm discussed above gives the best compression ratio over GenBit algorithm. But the problem comes with the Image Compression Algorithm Where the Decompressed Image cannot be brought back to the exact Original Image. The preparation of Matrix from the encoded sequence is a big challenge here as what should be its dimensions in getting the better output. Coming to our future work, we would like to determine the Image Processing Algorithm other than wavelet transform so that the compression ratio can be improved. Determining the dimensions of Matrix instead of a square matrix and retrieving the original Matrix with greater Accuracy from the Compressed Matrix is also a scope for further better results.

References

[1]Xie, X.; Zhou, S.; Guan, J. CoGI: Towards compressing genomes as an image. IEEE/ACM Trans. Comput. Biol. Bioinform. ***2015****, 12, 1275–1285.*

[2]Rajeswari, P. R., and Apparao, A., 2010, 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, 2(3), 181-191.

[3]Grumbach,S and Tahi,F A new Challenge for Compression Algorithms ;genetic sequences Information Processing and management 30;875-886,1994.

[4] D. S. Horner, G. Pavesi, T. Castrignan` o, P. D. De Meo, S. Liuni,M. Sammeth, E. Picardi, and G. Pesole, "Bioinformatics approaches for genomics and post genomics applications of next-generation sequencing," Briefings in Bioinformatics, vol. 11,no. 2, pp. 181–197, 2010.

[5] D. Pushkarev, N. F. Neff, and S. R. Quake, "Single-molecule Sequencing of an individual human genome," Nature Biotechnology,vol. 27, no. 9, pp. 847–850, 2009.

[6] P.-R. Loh, M. Baym, and B. Berger, "Compressive genomics," Nature Biotechnology, vol. 30, no. 7, pp. 627–630, 2012.

[7] C.Wang and D. Zhang, "A novel compression tool for efficient storage of genome resequencing data," Nucleic Acids Research, vol. 39, no. 7, pp. e45–e45, 2011.

[8] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences and its applications in genome comparison," in Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, 2000, p. 107.

[9] X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: fast and effective DNA sequence compression," Bioinformatics, vol. 18, no. 12, pp. 1696–1698, 2002.

[10] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," ACM Transactions on Information Systems (TOIS), vol. 23, no. 1, pp.3–34, 2005.

[11] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A simple statistical algorithm for biological sequence compression," in Data Compression Conference, 2007. DCC'07, 2007, pp. 43–52.

[12] S. Kuruppu, B. Beresford-Smith, T. Conway, and J. Zobel, "Iterative dictionary construction for compression of large DNA data sets," IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), vol. 9, no. 1, pp. 137–149, 2012.

[13] S.-M. Ahn, T.-H. Kim, S. Lee, D. Kim, H. Ghang, D.-S. Kim, B. C. Kim, S.-Y. Kim, W.-Y. Kim, C. Kim et al., "The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group," Genome Research, vol. 19, no. 9, pp. 1622– 1629, 2009.

[14] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHughet al., "Initial sequencing and analysis of the human genome," Nature, vol. 409, no. 6822, pp. 860–921, 2001.

[15] S. Deorowicz and S. Grabowski, "Robust relative compression of genomes with random access," Bioinformatics, vol. 27, no. 21, pp. 2979–2986, 2011.

[16] A. J. Pinho, D. Pratas, and S. P. Garcia, "GReEn: a tool for efficient compression of genome resequencing data," Nucleic Acids Research, vol. 40, no. 4, pp. e27–e27, 2012.

  ISSN 2319– 5592 (Online)

[17] S. Kuruppu, S. J. Puglisi, and J. Zobel, "Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval," in String Processing and Information Retrieval, 2010, pp. 201–206

[18] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Transactions on Information Theory, vol. 23, no. 3, pp. 337–343, 1977.

[19] S. Kuruppu, S. J. Puglisi, and J. Zobel, "Optimized relative lempel-ziv compression of genomes," in Proceedings of the Thirty-Fourth Australasian Computer Science Conference-Volume 113, 2011, pp. 91–98.

[20] V. M¨akinen, G. Navarro, J. Sir´en, and N. V¨alim¨aki, "Storage and retrieval of highly repetitive sequence collections," Journal of Computational Biology, vol. 17, no. 3, pp. 281–308, 2010.

[21] S. Deorowicz, A. Danek, and S. Grabowski, "Genome compression: a novel approach for large collections," Bioinformatics, vol. 29, no. 20, pp. 2572–2578, 2013.

[22] P. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry et al., "The variant call format and VCFtools,"
Bioinformatics, vol. 27, no. 15, pp. 2156–2158, 2011.

[23] L. Janin, O. Schulz-Trieglaff, and A. J. Cox, "BEETL-fastq: a searchable compressed archive for DNA reads," Bioinformatics, 2014.

[24] P. J. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, "The sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants," Nucleic Acids Research, vol. 38, no. 6, pp. 1767–1771, 2010.

[25] N. Popitsch and A. von Haeseler, "NGC: lossless and lossy compression of aligned high-throughput sequencing data," Nucleic Acids Research, vol. 41, no. 1, pp. e27–e27, 2013.

[26] H. Li, B. Handsaker, A.Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin et al., "The sequence alignment/ map format and SAMtools," Bioinformatics, vol. 25,no. 16, pp. 2078–2079, 2009.

[27] R. C´anovas, A. Moffat, and A. Turpin, "Lossy compression of quality scores in genomic data," Bioinformatics, 2014.

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

[29] Matsumoto, T., Sadakane, K., Imai, H., et al., 2000, Can General-Purpose Compression Schemes Really Compress DNA Sequences?, Computational Molecular Biology, Universal Academy Press, 76–77.

  ISSN 2319– 5592 (Online)