Abstract:
The databases of genomic sequences are growing at an explicative rate because of the increasing growth of living organisms. Compressing deoxyribonucleic acid (DNA) sequences is a momentous task as the databases are getting closest to its threshold. Various compression algorithms are developed for DNA sequence compression. An efficient DNA compression algorithm that works on both repetitive and non-repetitive sequences known as “HuffBit Compress” is based on the concept of Extended Binary Tree. In this paper, here is proposed and developed a modified version of “HuffBit Compress” algorithm to compress and decompress DNA sequences using the R language which will always give the Best Case of the compression ratio but it uses extra 6 bits to compress than best case of “HuffBit Compress” algorithm and can be named as the “Modified HuffBit Compress Algorithm”. The algorithm makes an extended binary tree based on the Huffman Codes and the maximum occurring bases (A, C, G, T). Experimenting with 6 sequences the proposed algorithm gives approximately 16.18% improvement in compression ration over the “HuffBit Compress” algorithm and 11.12% improvement in compression ration over the “2-Bits Encoding Method”.

Keywords: Compression and Decompression, Compression Ratio, Extended Binary Tree, HuffBit Compress, 2-Bits Encoding Method

DOI: 10.1515/jib-2017-0057

Received: July 23, 2017; Revised: January 2, 2018; Accepted: January 25, 2018

1 Introduction

R is one of the most popular languages used by statisticians, data analysts, researchers and marketers to retrieve, clean, analyze, visualize and present data [1]. It is an interactive environment providing a wide variety of statistical and graphical techniques. The R tool has become more popular among statisticians, mathematicians, computer scientists and researchers because of its excessive number of libraries and packages and ease of use. The R tool can also be applied in genetic applications. It is an implementation of the S programming language and has an estimated two million users [2], [3].

The genome contains all of the information needed to build and maintain an organism. In humans, a copy of the entire genome – more than 3 billion deoxyribonucleic acid (DNA) base pairs – is contained in all cells that have a nucleus [4] and more than 99% pairs are the same in all humans [5]. A DNA chain is made of four bases: adenine (A), guanine (G), thymine (T) and cytosine (C). These four chemical bases always bond with the same co-partner to comprise base pairs. For a complete genome a DNA sequence can be represented as a very long text containing only the bases (A, C, G, T) [6].

A text-based format known as FASTA format can be used to represent or store a DNA sequence. Some other formats include Plain sequence format, FASTQ format, EMBL format, GCG format, GCG-RSF format, GenBank format and IG format [7].

Thousands of nucleotides are sequenced every day. From 1982 to the present, the number of bases in GenBank has doubled approximately every 18 months [8]. Approximately 228719437638 bases are there in the GenBank database in February 2017 [8]. As a result, it becomes very hard to maintain, process, and store the large dataset in the database. Again, finding the characteristics and comparing Genomes is a major task [9], [10]. So it is clear that, DNA sequences should be compressible which is an analytical method created by bioinformatics.
Biological sequence compression is an effective apparatus to extract information from biological sequences. With respect to computer science data compression stands for curtailment of the size of memory used to store a data. From a mathematical point of view, compression implies better understanding and comprehension [11]. The compression of DNA sequences is not an easy task [6], [12]. General purpose compression algorithms fail to perform well with biological sequences. Most of the Existing software tools worked well for English text compression [13] but not for DNA Genomes. Lossless and lossy are two compression techniques. In lossless compression all of the information is completely restored after decompression. While in the lossy technique, we lose some of the data and do not recover the complete original data after decompression.

The proposed research is a lossless compression algorithm which builds an Extended Binary Tree for DNA sequences by assigning binary codes (0 and 1) to each base to compress both repetitive and non-repetitive DNA sequences. The algorithm is implemented using R. This paper is organized in 6 sections. Section 2 presents the background and previous work related to the research, Section 3 describes the proposed methodology and working principle, Section 4 presents the results, Section 5 discusses and analyzes the results, and last but not least Section 6 includes conclusion and future work.

## 2 Related Work

Due to not using special structures of biological sequences, the standard compression techniques cannot compress the biological sequences well. Recently, several algorithms have been proposed for the compression of DNA sequences based on DNA sequence special structures. Two lossless compression algorithms named BioCompress [14] and BioCompress-2 [6] were proposed by Grumbach and Tahi. These algorithms were based on Ziv and Lempel data compression method [15], [16]. They search the previously processed part of the sequence for repeats. BioCompress-2 is the extended version of BioCompress which uses the arithmetic coding of order 2 if no significant repetition is found. The results showed that both algorithms compressed the standard benchmark data with an average compression ratio of 1.850 bpb for BioCompress and 1.783 bpb for BioCompress 2 [17]. Decompression may demean the algorithm performance, as it requires reference to the starting of the sequence which requires more memory reference [18].

The Cfact [19] used a two-pass algorithm to search for the longest exact and reverse complement repeats. It builds the suffix tree of the sequence and then encodes using LZ. Non-repeat regions are also encoded by 2 bpb. There are no compression results about this algorithm; therefore it is difficult to compare [17]. It works similar to BioCompress, but takes more compression time, since it uses two passes, and constructs a suffix tree.

GenCompress is a one pass algorithm which is based on LZ77. This algorithm carefully finds the optimal prefix and uses order-2 arithmetic encoding whenever needed [13]. It searches for both approximate matches and the approximate complemented palindrome. There are two versions of GenCompress: GenCompress-1 uses the Hamming distance with replacement or substitution operations only. GenCompress-2 uses edit distance based on the insert and delete operation. GenCompress achieves 1.742 bpb which is a higher compression ratio compared to BioCompress-2 [17].

DNACompress [20] is the modified algorithm of GenCompress which is a two-pass algorithm. It also uses Lempel-Ziv compression scheme which finds all the approximate repeats including complemented palindromes using a software tool called Pattern Hunter [21]. It encrypts approximate repeat and non-repeat regions. DNACompress was able to deal with large sequences (e.g. *E. coli* with about 4.6 Megabases) in about a minute, where GenCompress required nearly about half an hour [17]. This algorithm achieves a compression rate of 1.72 bpb.

Huffman coding or 2 Bits Encoding Method is based on building a binary tree according to the bases frequencies. Each base is represented by 2 bits and assigned as A = 00, C = 01, G = 10, T = 11. As a result, it requires 2400 bits of space when the sequence length is 1200. That is, the storage of encoded sequence is almost the double of its original sequence length [6].

HuffBit Compress Algorithm [22] uses the concept of extended binary trees for compression. It assigns a zero for left child and one for right child. It is a two way process; in the initial phase it constructs an extended binary tree then replaces the sequences using the codes generated from the tree. The algorithm replaces a, c, g, t by a constant binary value where a is replaced by 0, c is replaced by 10, g is replaced by 110 and t is replaced by 111. It achieves a compression ratio of 1.006 bpb for the best case, 1.611 bpb for the average case, and 2.109 bpb for the worst case. In most of the cases it fails to achieve the best case compression ratio.

From the above discussion, it is clear that day by day research work on compressing DNA sequences is increasing and each algorithm comes with a better compression ratio. Each descendant algorithm comes reducing the disadvantages of previous algorithms. For example, BioCompress 2 gives better compression ratio than BioCompress1 but the storage complexity of them are high. Again, GenCompress gives better compression ratio than BioCompress2 but takes more time for large sequence compared to DNACompress. Cfact is more
complex in case of time. Huffman coding has a problem of storage complexity. For worst case Huffbit Compress compression ratio is not satisfactory. In this article, it has been tried to achieve a better compression factor than existing compression programs using the repeat and non-repeat sequence. Moreover, the proposed algorithm is simple, easy to implement, takes less memory space and most of the cases gives better compression ratio than HuffBit Compress Algorithm and 2 Bits Encoding Method.

3 Proposed Methodology

The proposed algorithm works on two phases. In the first phase it uses Huffman code to generate an extended binary tree based on the frequencies of the bases. In the second phase, each base is replaced by its corresponding binary bits generated from the tree. Compression and decompression of the DNA sequences are performed using R. Several steps are performed in this manner as described in the subsections through 3.1 to 3.6. A flowchart showing the proposed methodology including compression and decompression method is noticed in Figure 1.

Figure 1: Flowchart of proposed methodology.

3.1 DNA Sequence Download

The ‘National Center for Biotechnology Information’ (NCBI) maintains a huge database of all the DNA and protein sequence data. For the research project the ‘DENGUE’ DNA sequence with accession number “NC_001477” is retrieved from NCBI GenBank database using R and saved to a FASTA-format file using “write.fasta()” function.
3.2 Pre-Processing

The saved FASTA-format file is then read using read.fasta() function in R environment. The command reads the contents of the file into an R list object.

The length() function is used to obtain the length of the sequence and table() function is used to count the frequencies of the bases (a, c, g, t). The R-queries to read and pre process the DNA sequence is given below.

1. dengue ← read.fasta(file = “dengue.fasta”)
2. dengueSeq ← dengue[1]
3. length(dengueSeq)
4. table(dengueSeq)

3.3 Extended Binary Tree Generation

Huffman tree is a full binary tree consisting of external and internal nodes. Each internal node has exactly 2 children and each external node acts as a leaf. The External nodes are labeled by the base characters in the sequence whereas the internal nodes are labeled by their frequencies or probabilities. As a DNA sequence has only 4 bases (a, c, g, t) there will be 3 internal nodes and 4 external nodes in the tree. The algorithm for generating an extended binary tree is given below and the generated binary tree is displayed in Figure 2.

Figure 2: Generated extended binary tree.

Algorithm

1. START
2. Count the frequencies of each base in the DNA sequence.
3. Find the max1, max2, max3, max4 frequency base from the sequence.
4. Combine the two lowest frequencies bases as max3, max4; max2, (max3, max4); max1, (max2, (max3, max4)) to form a tree.
5. Assign ‘0’ to maximum frequency base and ‘1’ to minimum frequency base of each pair of the generated tree as shown in Figure 2.
6. Thus from Figure 2 max1 is assigned to 0, max2 is assigned to 10, max3 is assigned to 110 and max4 is assigned to 111.
7. END

3.4 Compression of the Sequence

To compress the DNA sequence, the bases are replaced by their respective codes generated from the tree. Thus the compressed sequence is found. But max1, max2, max3 and max4 bases are different for different sequences. So, during decompression of the sequence, some clue must be given to identify them from the compressed sequence. For this reason, six extra bits are added after the compressed sequence which can be accomplished in two steps.
1. Globally assign a 2-bit binary number for each base like A = 00, C = 01, G = 10, T = 11.

2. For max1, max2 and max3 base add the corresponding bits after the encrypted sequence in descending order to recognize which base has the maximum frequency.

Now save the encrypted sequence to get the compressed file. The pseudo code of the proposed compression method is displayed below.

**Pseudo Code**

1. START
2. Find max1, max2, max3, max4 frequency base.
3. Replace max1 by 0, max2 by 10, max3 by 110 and max4 by 111.
4. Assign A = 00, C = 01, G = 10, T = 11 in the code.
5. Based on max1, max2, max3 base add the corresponding 6 bits from step 4, after the sequence found from step 3.
6. Save the compressed sequence.
7. END

In the ‘HuffBit Compress Algorithm’ always, ‘a’ is replaced by 0, ‘c’ is replaced by 10, ‘g’ is replaced by 110 and ‘t’ is replaced by 111. Thus, if ‘g’ or ‘t’ becomes the maximum frequency base then the worst case compression ratio results because both ‘g’ and ‘t’ are replaced by 3 bits binary number. To overcome this limitation, here a ‘Modified Huffbit Compress Algorithm’ is proposed. The algorithm replaces the bases according to their frequency that is the max1 frequency base is replaced by a 1 bit binary number, the max2 base is replaced by a 2 bit binary number and the max3 and max4 bases are replaced by a 3 bit binary number. So, the proposed algorithm will always achieve the best case compression ratio compared to the ‘HuffBit Compress Algorithm’.

3.5 Decompression of the Sequence

Decompression is important to get the original file from the compressed file. The proposed algorithm is a lossless compression algorithm. So, without losing any data the original file will be retrieved. To decompress the previously compressed file, first it needs to be read in R environment. Then we compare the last 6 bits with the assigned 2-bit number to detect max1, max2, max3 and max4 frequency base. After finding the max1, max2, max3, max4 bases we remove the last 6 binary bits, then we replace 0 by the base which has the maximum frequency base (max1), replace 10 by the second maximum base (max2), replace 110 by the third maximum base (max3) and 111 by the minimum base (max4). The sequence is now decompressed. We write or save the sequence into a FASTA-format to get the original sequence back. The pseudo code of the decompression method is depicted below.

**Pseudo Code**

1. START
2. Compare last 6 bits with assigned binary number and get max1, max2, max3 and max4.
3. Delete the last 6 bits.
4. Replace 0 by max1 base, 10 by max2 base, 110 by max3 base and 111 by max4 base.
5. Save the decompressed sequence.
6. END
3.6 Measurement of Compression Ratio with Examples

Compression ratio is denoted by,

\[
\text{Compression ratio} = \frac{\text{Number of bits encoded}}{\text{Number of bases}} \tag{1}
\]

where,

\[
\text{Number of bits encoded} = \sum_{i=1}^{n} L(b_i) \times F(b_i) + 6 \text{ bits} \tag{2}
\]

Here,
- \(L(b_i)\) = length of path in the tree (number of edges in the generated tree) from root to external node labeled \(i\) = 1 to max \(n = 3\).
- \(F(b)\) = Frequency of bases in the sequence.

The following three examples show the calculation procedure.

**Example 1**

Given,

Sequence = AAACTGAAGGGTAAGAGAAGCAATAGGAAAA

\[F_A(b) = 17, \ F_G(b) = 10, \ F_T(b) = 3, \ F_C(b) = 2\]

Since, A is max1, G is max2, T is max3 and C is max4, a tree will be created according to the topic 3.3 by combining (T, C) then (G, (T, C)) and then (A, (G, (T, C))) as like Figure 3 and \(L(b)\) for A is 1(0), G is 2(10), T is 3(110) and C is 3(111).

From (2),

\[
\text{Number of bits encoded} = 1 \times 17 + 2 \times 10 + 3 \times 3 + 3 \times 2 + 6 = 58 \text{ bits}
\]

From (1),

\[
\text{Compression ratio} = \frac{58}{32} \text{ bits/base} = 1.81 \text{ bits/base}
\]

Similar procedure is applied in Example 2 also.

**Example 2**

Given sequence = ACTGCCCTACCGACCTTTCA ..............

\[F_C(b)=80, \ F_T(b)=20, \ F_A(b)=6, \ F_G(b)=4 \text{ and } L(b) \text{ for C is } 1(0), \ T \text{ is } 2(10), \ A \text{ is } 3(110) \text{ and G is } 3(111)\]
The generated tree for this sequence is shown in Figure 4. From (2),

\[
\text{Number of bits encoded} = 1 \times 80 + 2 \times 20 + 3 \times 6 + 3 \times 4 + 6 \\
= 156 \text{ bits}
\]

From (1),

\[
\text{Compression ratio} = \frac{156}{110} \text{ bits/base} \\
= 1.42 \text{ bits/base}
\]

Figure 4: Generated tree for Example 2.

Example 3

Given sequence = GGGGACTAAACCGGAGTAGC...
\[F_C(b) = 800, \ F_A(b) = 120, \ F_C(b) = 60, \ F_T(b) = 20 L(b) \text{ for } G \text{ is } 1(0),\]
\[A \text{ is } 2(10), \ C \text{ is } 3(110) \text{ and } T \text{ is } 3(111).\]

The generated tree for this example is displayed in Figure 5. From (2),

\[
\text{Number of bits encoded} = 1 \times 800 + 2 \times 120 + 3 \times 60 + 3 \times 20 + 6 \\
= 1286 \text{ bits}
\]

From (1),

\[
\text{Compression ratio} = \frac{1286}{1000} \text{ bits/base} \\
= 1.286 \text{ bits/base}
\]
4 Result

The proposed DNA compression algorithm is a modified version of the original ‘HuffBit Compress Algorithm’ and is developed using R. The algorithm is designed to achieve the best case compression ratio always, compared to the ‘HuffBit Compress Algorithm’. Several subsections are used here to describe the result of the proposed algorithm through 4.1 to 4.3.

4.1 Downloaded DNA Sequence

DNA sequence is a sequence with only four bases (a, c, g, t) of large numbers. For this analysis ‘DENGUE’ DNA sequence is downloaded and saved in FASTA-format using R. The downloaded DNA sequence containing 10735 bases is then read in R for pre-processing and further analysis. The top 630 bases of the sequence are displayed in Figure 6.
4.2 Pre-processing and Compression Method

Before starting the compression method, some operations must be performed on the DNA sequence such as finding out the length of the sequence; count the frequency of bases etc. Then the compression method begins with findings of the max1, max2, max3 and max4 base and replacing them with the binary value generated from the extended binary tree. Assigning A = 00, C = 01, G = 10 and T = 11 and at the last step appending 6 extra bits after the encrypted sequence.

Figure 7 displays the bottom of the compressed sequence. From Figure 7, it is clearly viewed that each of the DNA sequence base is replaced by their corresponding code. As each base is of 8 bits and each binary unit is of 1 bit, it is obvious that the file size will be reduced to a great extent after decompression. In Figure 7, the last 6 bits are each of a 2 bits binary number. Comparing these 6 bits with the assigned binary number a decision can be made in these ways.

1. The 1st 2 bits of the last 6 bits is 00. A is assigned to 00. So, ‘a’ is the max1 base.
2. The 2nd 2 bits of the last 6 bits is 10. G is assigned to 10. So, ‘g’ is the max2 base.
3. The 3rd 2 bits of the last 6 bits is 11. T is assigned to 11. So, ‘t’ is the max3 base.
4. Now the remaining base which is ‘c’ is max4.

4.3 Decompression Method

During decompression the binary codes will be replaced by their respective DNA bases. For this reason, at first max1, max2, max3 and max4 base need to be identified. To identify them, read the decompressed sequence in R and compare the last 6 bits of the sequence with the assigned binary bits. Based on this comparison, find the max1, max2, max3 and max4 base of the sequence. Then remove the last 6 binary bits and replace the binary codes with their corresponding bases such as, 0 by max1, 10 by max2, 110 by max3 and 111 by max4. Thus the original DNA sequence will be found. Save and store the sequence. The decompressed sequence will look like the downloaded DNA sequence as in Figure 6.

5 Discussion

In this section, Comparison of the proposed algorithm with previous algorithms (tree based) and complexity analysis of the proposed algorithm are shown through subsections 5.1 and 5.2.
| DNA Sequence length | Base | Base frequency | Binary bits | Compression ratio for Modified HuffBit Compress (bpb) | (%) Compression ratio for Modified HuffBit Compress | Compression ratio for HuffBit Compress (bpb) | (%) Compression ratio for HuffBit Compress | (%) Improvement of Modified HuffBit Compress over HuffBit Compress | Compression Ratio for 2-Bits Encoding Method (bpb) | (%) Compression Ratio for 2-Bits Encoding Method | (%) Improvement of Modified HuffBit Compress over 2-Bits Encoding Method |
|---------------------|------|----------------|-------------|------------------------------------------------------|-------------------------------------------------|---------------------------------|---------------------------------|----------------------------------------------------------|---------------------------------|---------------------------------|----------------------------------------------------------|
| 1. Sequence: 100    | A    | 45             | 0           | 1.71                                                 | 21.375                                          | 2.05                             | 25.625                          | 16.59                                                      | 2                               | 25                              | 14.5                                                      |
|                     | T    | 45             | 10          |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | C    | 5              | 110         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | G    | 5              | 111         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
| 2. Sequence: 600    | A    | 275            | 0           | 1.635                                                | 20.4375                                         | 2.04                             | 25.520                          | 19.92                                                      | 2                               | 25                              | 18.25                                                     |
|                     | T    | 275            | 10          |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | C    | 25             | 110         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | G    | 25             | 111         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
| 3. Sequence: 500    | C    | 300            | 0           | 1.612                                                | 20.15                                           | 2.1                              | 26.25                           | 23.24                                                      | 2                               | 25                              | 19.4                                                      |
|                     | T    | 100            | 10          |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | A    | 75             | 110         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | G    | 25             | 111         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
| 4. Sequence: 64     | T    | 40             | 0           | 1.625                                                | 20.3125                                         | 2.75                             | 32.031                          | 36.58                                                      | 2                               | 25                              | 18.75                                                     |
|                     | C    | 14             | 10          |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | A    | 7              | 110         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | G    | 3              | 111         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
| 5. Sequence: 200    | A    | 80             | 0           | 1.98                                                 | 24.75                                           | 1.95                             | 24.375                          | −1.54                                                     | 2                               | 25                              | 1                                                          |
|                     | C    | 50             | 10          |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | G    | 40             | 110         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
| 6. Sequence: 10735  | A    | 3426           | 0           | 2.10                                                 | 26.30                                           | 2.15                             | 26.91                           | 2.27                                                      | 2                               | 25                              | −5.2                                                      |
|                     | G    | 2770           | 10          |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | T    | 2299           | 110         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
|                     | C    | 2240           | 111         |                                                     |                                                 |                                  |                                |                                                            |                                  |                                  |                                                           |
5.1 Comparison with Prior Algorithms

The most natural benchmark of the proposed algorithm is a comparison with the other compression algorithms designed to compress DNA sequences.

This comparison can be done by using the compression ratio. The ratio of the original, uncompressed data file to the compressed file is referred to as the compression ratio. The comparison among Modified HuffBit Compress, HuffBit Compress and 2-Bits Encoding Method are given in Table 1.

In the table,

\[
\text{Percentage Compression ratio} = \left(\frac{\text{Number of byte encoded}}{\text{Number of bases}}\right) \times 100 \%
\]

\[
\text{Improvement of Modified HuffBit Compress over HuffBit Compress} = \frac{\text{Percentage Compression ratio for HuffBit Compress} - \text{Percentage Compression ratio for Modified HuffBit Compress}}{\text{Percentage Compression ratio for HuffBit Compress}} \times 100 \%
\]

\[
\text{Improvement of Modified HuffBit Compress over 2 Bits Encoding Method} = \frac{\text{Percentage Compression ratio for 2 Bits Encoding Method} - \text{Percentage Compression ratio for Modified HuffBit Compress}}{\text{Percentage Compression ratio for 2 Bits Encoding Method}} \times 100 \%
\]

From the table, it is observed that the “Modified Huffbit Compress Algorithm” performs better than “Huffbit Compress” algorithm and “2 Bits Encoding Method” for sequence 1, 2, 3, 4 but for sequence 5, “HuffBit Compress” and for sequence 6, “2 Bits Encoding Method” gives better result. Therefore, the average improvement of using the proposed algorithm is 16.18 % and 11.12 % compared to “HuffBit Compress” and “2 Bits Encoding Method” respectively. The article by Ghoshdastider and Saha [23] mentioned that for a sequence of length 64, GenCompress, BioCompress and other software takes about 14 bytes for compressed sequence while in “Modified Huffbit Compress Algorithm” sequence 4 with a length of 64 needs 13 bytes for compressed sequence.

Thus the proposed algorithm performs well for most of the cases but it has some limitations also.

1. If the sequence length is small then the extra 6 bits in the compressed file cannot be neglected and compression ratio increases. But, as in most cases, DNA sequences are too large; so that issue might not be a problem.
2. Compression ratio depends on max1, max2, max3 and max4. So, if there exists a large number of max3 and max4 then the compression ratio also increases.

Nevertheless, the proposed algorithm performs better in most cases.

5.2 Complexity Analysis of Proposed Algorithm

Complexity of an algorithm is a measure of the amount of time and/or space required by an algorithm for an input of a given size (n) [24]. It indicates how fast or slow an algorithm actually performs. The time complexity is usually expressed using big O notation. The time complexity of an algorithm is a measure of the amount of time taken by an algorithm to run. The time complexity of proposed algorithm and ‘HuffBit Compress Algorithm’ can be shown as in Table 2.

| Proposed algorithm compression method | Time complexity  | HuffBit Compress Algorithm |
|--------------------------------------|------------------|-----------------------------|
| 1. Counting the frequency of each base | O(n)             | Counting the frequency of each base |
| 2. Finding max1, max2, max3, max4 base | O(4)             | Encryption operation |
| 3. Encryption Operation               | 4*O(n)           |                             |
| 4. Assigning binary bits              | O(1)             |                             |
| 5. Appending last 6 bits              | O(4)             |                             |
| Total = O(n) + O(4) + 4*O(n) + O(1) + O(4) = 5*O(n) + O(9) | Total = O(n)+4*O(n) = 5*O(n) |
|                                      | ~ 5*O(n)         |                             |
From Table 2, it is ensured that the time complexity of the proposed ‘Modified HuffBit Compress Algorithm’ is approximately same as ‘HuffBit Compress Algorithm’. Also proposed algorithm always gives best case compression ratio than ‘HuffBit Compress Algorithm’. So, using ‘Modified HuffBit Compress Algorithm’ will be more fruitful for DNA sequence compression.

6 Conclusion

DNA compression is an important topic in bioinformatics which helps in storage, manipulation and transformation of large DNA sequences. If the sequence is compressed using Modified HuffBit Compress algorithm, it will be easier to compress large bytes of DNA sequences with better compression ratio. An advantage of the proposed algorithm is, it works well for large sequences. So, it can be helpful to save storage problem greatly. Moreover, it uses less time, memory and easy to implement using R. This research also creates a new dimension of using R in DNA compression which is the main potentiality of the research work. The future work of the research is to overcome the limitations of the proposed algorithm and to come with a better outcome.

Acknowledgments

The authors are grateful to the participants who contributed to this research. No financial support is provided from any organization during the research project.

Conflict of interest statement: Authors state no conflict of interest. All authors have read the journal’s publication ethics and publication malpractice statement available at the journal’s website and hereby confirm that they comply with all its parts applicable to the present scientific work.

References

[1] PROGRAMIZ. https://www.programiz.com/r-programming. Accessed September 19, 2017.
[2] Vance A. Data analysts captivated by R’s Power. NY Times, 2009. URL http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=1.
[3] Timothy Prickett Morgan. Open source R in commercial revolution. The Register, 2010. URL http://www.theregister.co.uk/2010/05/06/revolution_commercial_.r.
[4] Genetics Home Reference. https://ghr.nlm.nih.gov/primer/hgp-genome. Accessed March 15, 2017.
[5] Bharti RK, Singh RK. A biological sequence compression based on look up table (LUT) using complementary palindrome of fixed size. Int J Comput Appl. 2011;35:0975–8887.
[6] Grumbach S, Tahi F. A new challenge for compression algorithms: genetic sequences. Inf Process Manage 1994;30:875–86.
[7] Genomatix. https://www.genomatix.de/online_help/help/sequence_formats.html. Accessed March 15, 2017.
[8] GenBank and WGS Statistics (NCBI). https://www.ncbi.nlm.nih.gov/genbank/statistics/. Accessed March 15, 2017.
[9] Koonin EV. The emerging paradigm and open problems in comparative genomics. Bioinformatics 1999;15:265–6.
[10] Wooley JC. Trends in computational biology: a summary based on a RECOMB plenary lecture, 1999. J Comput Biol. 2004;6:459–474.
[11] Bennett CH, Gacs P, Li M, Vitanyi PMB, Zurek WH. Information distance. IEEE Trans Inf Theory. 1998;44:4.
[12] Rivals E, Delahaye J-P, Dauchet M, Delgrange O. A guaranteed compression scheme for repetitive DNA sequences. LIFL I University, technical report 1995; IT-285.
[13] Rajarajeswari P, Apparao A. DNABIT compress — genome compression algorithm. Bioinformation 2011;5:350–60.
[14] Grumbach S, Tahi F. Compression of DNA sequences. In: IEEE Symposium on the Data Compression Conference, DCC-93; Snowbird, UT, 1993:340–50.
[15] Ziv J, Lempel A. A universal algorithm for sequential data compression. IEEE Trans Inf Theory. 1977;IT-23:337.
[16] ZIV J. Coding theorems for individual sequences. IEEE Trans Inf Theory. 1978;IT-24. [PMID: 20157474].
[17] Bakr NS, Sharawi AA. DNA lossless compression algorithms: review. Am J Bioinform Res 2013;3:72–81.
[18] Bharti RK, Harbola D. State of the art: DNA compression algorithms. IJARCSSE 2013;3:397.
[19] Rivals E, Dauchet M. Fast discerning repeats in DNA sequences with a compression algorithm. In: The 8th Workshop on Genome and Informatics, (GIW’97) 1997; 8: 215–26.
[20] Chen X, Li M, Ma B, Tromp J. DNACompress: fast and effective DNA sequence compression. Bioinformatics 2002;18:1696–8.
[21] Ma B, Tromp J, Li M. PatternHunter — faster and more sensitive homology search. Bioinformatics 2002;18:440–5.
[22] Raja Rajeswari P, Apparao A, Kiran Kumar R. HUFFBIT COMPRESS — algorithm to compress DNA sequences using extended binary trees. J Theor Appl Inf Technol. 2010;13:101–6.
[23] Ghoshdastider U, Saha B. GenomeCompress: a novel algorithm for DNA compression, 2007.
[24] Complexity. http://www.dcs.gla.ac.uk/~pat/s23233/complexity.html. Accessed August 17, 2017.