DNA DATA COMPRESSION ALGORITHMS BASED ON REDUNDANCY

Subhankar Roy¹ and Sunirmal Khatua²

¹Department of Computer Science and Engineering, Academy of Technology, G. T. Road, Aedconagar, Hooghly-712121, W.B., India
²Department of Computer Science and Engineer, University of Calcutta, 92, A.P.C. Road, Kolkata-700009, India

ABSTRACT

Carl Jung said, 'Collective unconscious' i.e. we are all connected to each other in some way or the other via our DNA. In frequent cases there are four bases in a DNA. They are a (Adenine), c (Cytosine), g (Guanine) and t (Thymine). Each of these bases can be represented by two bits as $2^2 = 4$ i.e. a – 00, c – 01, g – 11 and t – 10 respectively, although this choice is random. So redundancy within a sequence is more likely to exist. That’s why in this paper we have explored different types of repeat to compress DNA. These are direct repeats, palindrome or reverse direct repeat, inverted exact repeats or complementary palindrome or exact reverse complement, inverted approximate repeats or approximate complementary palindrome or approximate reverse complement, interspersed or dispersed repeats, flanking repeats or terminal repeats etc. Better compression gives better network speed and save storage space.

KEYWORDS

Direct repeats, Inverted repeats, Complementary palindrome, Interspersed repeats and Flanking repeats.

1. INTRODUCTION

Today, more and more DNA sequences are becoming available [1]. The information about DNA sequences are stored in molecular biology databases. The size and importance of these databases getting bigger with time; therefore this information must be stored or communicated efficiently [2]. Furthermore, sequence compression can be used to define similarities between biological sequences. The only way to handle this situation is to encoding these sequences. Better compression ratio depends upon the DNA sequences [3].

However if one applies the standard text compression software such as GZIP [4], they cannot compress DNA sequences but only expand the file with more than two bits per symbol. There are some reasons pointed out. This software is designed mainly for English text compression, while the regularities in DNA sequences are much subtler and these tools do not make use of the special characteristics of a DNA sequence. For example, it is well known to us that all DNA sequences have very long-term correlation [5] in that the subsequences in different regions of any DNA sequence of same genome or of different genome are quite similar to each other. State-of-the-art DNA encoding schemes rely on exploiting this long-term correlation. In particular, similarity within the DNA sequence is searched so that similar subsequences of different fragment can be encoded with reference to its earlier section.

Thus DNA sequences are important as a new challenge for study of compression algorithms. Eclipse uses Cp1252 as the default for its encoding. But UTF-8 can represent every character in Unicode ($2^{16} = 65536$) and has a much bigger library of recognized symbols. The console in
eclipse uses the default encoding of OS i.e. inherited from container (CP1252). The size of each character in Unicode is two byte whereas ASCII code character size is one byte. When compression is done on DNA sequence or on some plain text, for a line break in a file each line feed character (‘\n’) and carriage return (‘\r’) as a whole takes four bytes. So if a text file contains more line compressed file using Unicode character give bad compression factor. But in ASCII code \(2^8=256\) the size of each character is one byte so a line break that is a combination of line feed (‘\n’) and carriage return (‘\r’) as a whole takes two bytes.

If number of lines is 1000 then compressed file using Unicode 1000*4 = 4000 bytes and using ASCII code 1000*2 = 2000 bytes that is half in size compare to Unicode.

That is why most of the efficient compression algorithm coding is done using C language rather than using Java because Java character set is called UNICODE character set and C follows ASCII code character set.

2. LITERATURE REVIEW

It is true that the compression of DNA sequence is a difficult task for general compression algorithms, but at the same time, from the viewpoint of compression theory it is an interesting subject for understanding the properties of various compression algorithms.

Generally the windows of the methods based on dictionary [6-7] have a fixed width of small size. The use of small windows is efficient on plain text whose redundancy is very fewer and local. However, in the case of DNA sequence, redundancies may occur at very long distances and factors can be very long.

Shannon-Fano [8-9] or Huffman Coding [8-9] does not give good compression factor for DNA data compression. Huffman’s code fails badly on DNA sequences both in the static and adaptive model [8-9], because there are only four kind symbols in DNA sequences and the probabilities of occurrence of the symbols are close. Both follow statistical modeling by creating variable length codes for bases in DNA sequence. Base with higher probability of occurrence in a sequence gets shorter codes in binary form as an integral multiple. Code assignment is done by making a binary tree that is called the Huffman tree. As the most frequent bases in DNA sequence are 4 in number. They are a, c, g and t respectively. Let us consider a part of a sequence “actgtgtgatgatgacataggaaccaacccccaaaaa … accat” and table 1 shows their frequencies in that sequence.

<table>
<thead>
<tr>
<th>Nucleotides</th>
<th>Frequencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>100</td>
</tr>
<tr>
<td>c</td>
<td>100</td>
</tr>
<tr>
<td>g</td>
<td>50</td>
</tr>
<tr>
<td>t</td>
<td>50</td>
</tr>
</tbody>
</table>

So if we build the Huffman tree using these symbols then the binary code for these four symbols is shown in Table 2.

<table>
<thead>
<tr>
<th>Nucleotides</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0 / 00</td>
</tr>
<tr>
<td>c</td>
<td>10 / 01</td>
</tr>
</tbody>
</table>
So total number of bits required is \((100 \times 1 + 100 \times 2 + 50 \times 3 + 50 \times 3) = (100+200+150+150) = (100 \times 2 + 100 \times 2 + 50 \times 2 + 50 \times 3) = (200+200+100+100) = 600\) bits.

Using simplest two bit encoding technique it requires same number of bits. As Huffman code does not consider any properties within a sequence such as repeats or similarity that is why it cannot give better result. If we apply Shannon-Fano technique then again it needs 600 bits to compress the above file.

Concerning compression ratio, Prediction by Partial Matching (PPM) [10] is one of the best compression algorithms in practice developed by Cleary and Witten is accomplished of very high compression rates, encoding English text in as little as 2.2 bits/character. It predicts according to the most recent symbols seen. However it cannot compress DNA sequences less than two bits per symbol either.

Context Tree Weighting (CTW) [11] method introduced by Willems, Shtarkov, and Tjalkens can compress DNA sequences less than two bits per symbol. It compresses the data by branch prediction. But this algorithm does not use special structures of biological sequences.

DNA sequences are more redundant in nature due to only four frequent bases a, c, g and t without considering eleven rare bases. So it is obvious that the similarity between subsequence within a particular sequence will be more. Now if instead of considering a particular sequence if a number of sequences of a particular genome considered then the similarity between those sequences are more due to redundancy. So it would be advantageous to compress different chromosomes together to take into account both self-chromosomal similarity and cross-chromosomal similarities within chromosome [12]. A statistical study reveals that the similarity within a sequence is 4.5% of total sequence length. Whereas similarity within two chromosomes is 10% in which 8% comes from cross-chromosomal similarity and 2% comes from self-chromosomal similarity. Among 16 chromosomes it is 18% in which cross-similarity is 16.8% and rest is self-similarity. There is a software tool called PatternHunter are used to search both self and cross-chromosomal similarity. It can search both exact, approximate repeat and complementary palindrome. So the compression algorithm using self and cross chromosomal similarity is local in nature because here chromosomes of a particular genome has been considered. This one is like a dictionary based encoding algorithm because each chromosomes is compared with other using PatternHunter. So there have some reference chromosomes with which target chromosomes are compared and the differences are compressed. There are some pointer which points the common region and length of the common region of the reference chromosomes. In the decompression time the compressed difference and reference sequences are used to recover the original sequence. Instead of compressing individual genome sequence of an organism, here entire genome sequence of an organism has been compressed [13-14]. This would give better saving percentage due to redundancy nature of DNA sequence. Here each genome sequence of an organism is compared with other genome sequence of that organism and only difference are compressed. The common region needs not to compress again. This one is like a dictionary based encoding algorithm because each different genome sequence is compared with different specific genome sequence. Therefore, the compressed data are poised of reference sequence, differences sequence, and the locations of differences, instead of storing each DNA data sequence individually. A better result of this approach is depends on the similarity between sequences. It is shown that mitochondria dataset give 195-fold compression rate.

Original sequence = Reference sequences + The difference within sequence
Genbit compression algorithm [15] considered both for repetitive and non-repetitive DNA sequences. In this scheme all possible combination has been introduced. The input sequence is divided in to block, where each block is of four bases. Thus in this coding scheme, 2 power 8 = 256 combinations can be represented. Hence every DNA segment containing four bases is replaced by an 8 bit binary number for e.g. “01010101”. If the consecutive fragments are same, then a specific bit “1” is introduced as a 9th bit. If the consecutive fragments are different, then a specific bit “0” is introduced as a 9th bit to the 8 bit unique number. GenBit Compress is a simple algorithm without Dynamic programming approach. It takes an input of a DNA sequence of length n, and divides into n/4 number of fragments. The left out individual bases (fragment length<4) is assigned 4 unique “2” bits. (a=”00”, g=”01”, c=”10”, t=”11”).

3. PROPOSED METHODOLOGY

The general encoding methodology is before encoding the next symbol in a DNA sequence by mapping technique, algorithms search for direct repeats, reverse direct repeats and complementary palindrome which are explain below in details. If they found with desired sequence length then our algorithm count the number of occurrence and represents it by the mapped symbol followed by the number of repeat. These algorithms are completely different from others existing in the way that it search for best result from different compression technique using different technique and give the final output with least bpb.

3.1 Direct exact or approximate repeats in DNA sequence

Algorithm FS4DR uses this property. 
Repeat in DNA sequence means exact and approximate matches. Approximate matches are done by operations such as substitution or Replacement, insertion and deletion.
An exact appearance means two substrings of a string in a sequence consist of indistinguishable bases along the chosen subsequence.

For example if a subsequence is “attcgtgtattcgtgt”. The first eight bases i.e. “attcgtgt” and last eight bases i.e. “attcgtgt”, the second subsequence is exact copy of the first one.
Exact repeat also called tandem repeat. When a pattern of two or more nucleotides is repeated and the repetitions are adjacent to each other directly or inverted.

An example is: atcgatcgatcgatcg, in which the sequence “atcg” is repeated four times.
When the number is not known, variable, or irrelevant, it is sometimes called a variable number tandem repeats (VNTR). The term minisatellite has been interchangeably used with variable number tandem repeats (VNTRs).

When between 10 and 60 nucleotides are repeated, it is called a minisatellite. These occur at more than 1,000 locations in the human genome. Those with fewer are known as microsatellites or short tandem repeats (STR). STRs are also repeated sequences, but they are usually 2–10 nucleotides long.

When exactly two nucleotides are repeated, it is called a dinucleotide repeat (for example: tctctctc…). The microsatellite instability in hereditary nonpolyposis colon cancer most commonly affects such regions.

When three nucleotides are repeated, it is called a trinucleotide repeat (for example: tagtagtag…), and abnormalities in such regions can give rise to trinucleotide repeat disorders.
An approximate appearance means two substrings of a string in a sequence consist of indistinguishable bases after one of the following operations along the chosen subsequence. For example if a subsequence is “attgttatgctgt”. The first eight bases i.e. “attgtt” and last eight nucleotides i.e. “atggctgt”. The second subsequence can be obtained from the first subsequence if the 4th base “T” in the first subsequence is substituted or replaced by “c”.

If a DNA sequence is “attgttatgctgt”. The second subsequence “atggctgt” can be obtained from the first one “attgtt” if we insert “g” within first sequence between 6th and 7th character. This is called approximate matches with insertion.

For approximate matches with deletion operation, let us consider a DNA sequence is “attgttatgctgt”. The second subsequence “atggctgt” can be obtained from the first one “attgtt” if we delete 7th base i.e. “g” within first sequence.

Substitution can be used as a combination of insertion and deletion operations in some cases. For example if a subsequence is “attgttatgctgt” then the second subsequence “atggctgt” of length eight can be obtained from the first one “attgtt” by inserting “c” between 3rd and 4th bases then by deleting base “t” from the 5th position. This one can be also obtained by simply replacing character as described above.

```
1 atgaatgaatgaatgaatgaatga 24
 1 atga 4
 5 atga 8
 9 atga 12
13 atga 16
17 atga 20
21 atga 24
```

Figure 1. An example of direct repeat of four nucleotides. The sequence is a part of DNA sequence.

### 3.2 Palindrome or reverse direct repeat, Inverted exact repeat or complementary palindrome and inverted approximate repeat or approximate complementary palindrome or approximate reverse complement repeats

Algorithm FS4RDR and FS4CP uses these property.

The palindrome sequence is “acgttgca”, the fragment “tgca” can be obtained from fragment “acgt” by reversing the second one.

An inverted repeat, reversed repeat, complemented inverted repeat or reverse complement repeat is a sequence of bases that is the reversed after complementing of another sequence further downstream to the sequence.

The original sequence of nucleotides is written in the 5’ to 3’ direction.
For e.g. 5’-atgc-3’

The complementary sequence is written, matching bases ‘a’ with ‘-t’ and ‘g’ with ‘c’. The resulting sequence is in the 3’ to 5’ orientation.

3’-tacg-5’

If the new sequence “3’-tacg-5’ “, is reversed, resulting in the complementary sequence in the correct orientation i.e. 5’-gcat-3’.

If a sequence is “gcatatgc” so second subsequence “atgc” of length four is reverse complement of first subsequences “gcat” of length four.
If there are no nucleotides or bases intervene between the subsequence and its downstream complement, it is called a palindrome. So in the above case the second subsequence is complementary palindrome or inverted exact repeat of the first subsequence. Inverted repeats also indicate regions capable of self-complementary base pairing (regions within a single sequence which can base pair with each other).

If a sequence is “gcttatgc” so second subsequence “atgc” is reverse complements of first subsequences “gctt”. But it is an approximate reverse complement. Complement of the second subsequence is “tacg” in 3’ to 5’ direction. So in the original direction i.e. 5’ to 3’ it is “gcat”. So if the 3rd base “t” of reference subsequence is replaced by “a” then it will be similar to the compared subsequence.

![Figure 2. An example of palindrome of four nucleotides. The sequence is a part of DNA sequence.](image)

![Figure 3. An example of complementary palindrome of four nucleotides. The sequence is a part of DNA sequence.](image)

### 3.3 Interspersed or Dispersed Repeats

It is the copy of repetitive sequences that are interspersed throughout the genome. If a genome sequence that consists of two or more repeats of a specific subsequence. Two identical (or nearly identical) nucleotide bases sequences sometimes separated by a sequence of non-repeated DNA. For example if a sequence is “3’ tagtacgttagt 5’”. The last four subsequences “tagt” is exact repeat of first four subsequences i.e. “tagt” but the intermediate sub sequence “acgt” in a non repeated region. It can present in multiple copies in the genome. This can be exact and approximate repeat. It is classified as Short Interspersed Repeat (SIR) and Long Interspersed Repeat (LIR).

### 3.4 Flanking Repeats or Terminal Repeat

The type of repeats is sequences that are repeated on both ends of a sequence. Direct terminal repeats are in the same direction and inverted terminal repeats are opposite to each other in direction.

### 3.5 Complementary Nature of Nucleotide Sequences

DNA double strand is complementary in nature. If one strand is known, then other strand can be formed from that one just by replacing ‘a’ with ‘t’ and ‘g’ with ‘c’ and vice versa. The polymer forms in a unidirectional manner from the 5’ carbon that is bound to a phosphate group to the 3’ carbon that is bound to the hydroxyl group. DNA sequences are often written in a shorthand notation using the one-letter name for each nucleotide, and listing the nucleotides in the 5’ to 3’ direction.
3.6 Compression Technique

Original file
e.g. humdystrop.txt

Compression technique
FS4

Compressed file
e.g. humdystrop_cmp.txt

Figure 4. The compression technique.

3.7 Decompression Technique

Compressed file
e.g. humdystrop_cmp.txt

Decompression technique
RFS4

Reconstructed file
e.g. humdystrop_rec.txt

Figure 6. The decompression technique.
4. SIMULATION RESULTS

Table 3 shows the information about the ten standard benchmark genome sequence [17]. Table 4 shows the result in bytes by fragment size four (FS4), fragment size four with direct repeat (FS4DR), fragment size four with reverse direct repeat (FS4RDR) and fragment size four with complementary palindrome (FS4CP). RFS4 means reverse fragment size four. Table 5 shows the resultant bits per base (bpb) used of our proposed compression algorithm in compressing each genome sequence separately.

Table 3. Information of ten standard benchmark DNA sequences.

<table>
<thead>
<tr>
<th>Sequences Name</th>
<th>Length(Bytes)</th>
<th>Source</th>
<th>File Size(KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmpxx</td>
<td>121,024</td>
<td>Chloroplasts</td>
<td>118.1875</td>
</tr>
<tr>
<td>chntxx</td>
<td>155,844</td>
<td></td>
<td>152.1914</td>
</tr>
<tr>
<td>humdystrop</td>
<td>38,770</td>
<td>Human</td>
<td>37.8613</td>
</tr>
<tr>
<td>humghcsa</td>
<td>66,495</td>
<td></td>
<td>64.9365</td>
</tr>
<tr>
<td>humhdabcd</td>
<td>58,864</td>
<td></td>
<td>57.4844</td>
</tr>
<tr>
<td>humhprtb</td>
<td>57,737</td>
<td></td>
<td>55.4072</td>
</tr>
<tr>
<td>mpointcg</td>
<td>186,608</td>
<td>Mitochondria</td>
<td>182.2344</td>
</tr>
<tr>
<td>mtpacg</td>
<td>100,314</td>
<td></td>
<td>97.9629</td>
</tr>
<tr>
<td>hchmvng</td>
<td>229,354</td>
<td>Viruses</td>
<td>223.9785</td>
</tr>
<tr>
<td>vaccg</td>
<td>191,737</td>
<td></td>
<td>187.2431</td>
</tr>
</tbody>
</table>

Table 4. The sequences length in bytes by different compression techniques. The bits per base (bpb) are used by the below compression methods along with their properties.

<table>
<thead>
<tr>
<th>Sequences Name</th>
<th>FS4</th>
<th>FS4DR</th>
<th>FS4RDR</th>
<th>FS4CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmpxx</td>
<td>31,945</td>
<td>30,045</td>
<td>30,403</td>
<td>30,203</td>
</tr>
<tr>
<td>chntxx</td>
<td>40,960</td>
<td>38,960</td>
<td>38,900</td>
<td>38,700</td>
</tr>
<tr>
<td>humdystrop</td>
<td>10,006</td>
<td>10,083</td>
<td>10,011</td>
<td>10,020</td>
</tr>
<tr>
<td>humghcsa</td>
<td>18,160</td>
<td>16,576</td>
<td>16,776</td>
<td>16,976</td>
</tr>
<tr>
<td>humhdabcd</td>
<td>15,295</td>
<td>14,576</td>
<td>14,376</td>
<td>14,376</td>
</tr>
<tr>
<td>humhprtb</td>
<td>14,984</td>
<td>14,153</td>
<td>14,053</td>
<td>14,053</td>
</tr>
</tbody>
</table>
Table 5. The bpb by different compression techniques.

<table>
<thead>
<tr>
<th>Sequences Name</th>
<th>FS4</th>
<th>FS4DR</th>
<th>FS4RDR</th>
<th>FS4CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmpxx</td>
<td>2.111647</td>
<td>1.986052</td>
<td>2.009717</td>
<td>1.996497</td>
</tr>
<tr>
<td>chntxx</td>
<td>2.102615</td>
<td>1.999949</td>
<td>1.998669</td>
<td>1.986602</td>
</tr>
<tr>
<td>humdystrop</td>
<td>2.064689</td>
<td>2.080578</td>
<td>2.065721</td>
<td>2.067578</td>
</tr>
<tr>
<td>humghcsa</td>
<td>2.184826</td>
<td>1.994255</td>
<td>2.018317</td>
<td>2.042379</td>
</tr>
<tr>
<td>humhdabcd</td>
<td>2.07869</td>
<td>1.980973</td>
<td>1.953792</td>
<td>1.953792</td>
</tr>
<tr>
<td>humhrpb</td>
<td>2.112766</td>
<td>1.995594</td>
<td>1.981494</td>
<td>1.981494</td>
</tr>
<tr>
<td>mpomtcg</td>
<td>2.207794</td>
<td>1.988682</td>
<td>1.980108</td>
<td>1.975821</td>
</tr>
<tr>
<td>mtpacg</td>
<td>2.151883</td>
<td>1.994378</td>
<td>1.986403</td>
<td>1.978428</td>
</tr>
<tr>
<td>hehemvcg</td>
<td>2.120948</td>
<td>1.981426</td>
<td>1.998866</td>
<td>1.995378</td>
</tr>
<tr>
<td>vaccg</td>
<td>2.17669</td>
<td>1.900741</td>
<td>1.975706</td>
<td>1.967362</td>
</tr>
<tr>
<td><strong>Average bpb</strong></td>
<td><strong>2.131255</strong></td>
<td><strong>1.990263</strong></td>
<td><strong>1.996699</strong></td>
<td><strong>1.994533</strong></td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

Our proposed compression algorithm was tested on ten standard benchmark DNA sequences. The experimental results showed that the bpb is less than two bits per base when we compress these sequences using any of the three simple properties. After applying all techniques our algorithm searches for optimal solution to get the final result and will store that result. When none of the above characteristics is used the compression factor is approximately equals to 2, because each nucleotides base is represented by two bits.

ACKNOWLEDGEMENTS

This research work is supported in part by the Computer Science and Engineering Department of University of Calcutta, Bioinformatics Centre of Bose Institute and Academy of Technology under WBUT. We would like to thank Dr. Zumur Ghosh and Asst. Professor Utpal Nandi for their valuable suggestion.

REFERENCES


Authors

Subhankar Roy is currently working as an Assistant Professor in the Department of Computer Science & Engineering, Academy of Technology, India. He received his B.Tech degree and M.Tech degree in Computer Science and Engineering in 2012 both from University of Calcutta. He is doing research work from University of Calcutta under the guidance of Mr. Sunirmal Khatua. His research interests include Bioinformatics and compression techniques. He is a member of IEEE.

Sunirmal Khatua is currently an Assistant Professor in the Department of Computer Science and Engineering, University of Calcutta, India. He has received the M.E. degree in Computer Science and Engineering from Jadavpur University, India in 2006. He is also pursuing his PhD in Cloud Computing from Jadavpur University. His research interests are in the areas of Distributed Computing, Cloud Computing and Sensor Networks. He is a member of IEEE.