GenomeCompress: A Novel Algorithm for DNA Compression

Umesh Ghoshdastider[1], Banani Saha[2]
[1]*Department of Physiology, Rammohan College, Kolkata,*
Email:coolunmesh@gmail.com
[2]*Department of Computer Science and Engineering,*
University of Calcutta, 92, APC Road Calcutta-700009,
email:bsaha_29@yahoo.com

Abstract

The genome of an organism contains all hereditary information encoded in DNA. So it is extremely important to sequence the genome which determines how the organisms survive, develop and multiply. Since three decades, due to massive efforts on DNA sequencing, complete genome sequence of a large number of organisms including humans are now known and the genomic databases are growing exponentially with time. Also for the huge size of the genomes, an efficient algorithm is required to compress them. General text compression algorithms don't utilize the specific characteristics of a DNA sequence. DNA specific compression algorithms exploit the repetitiveness of bases in DNA sequences. A repetitive DNA sequence can be best compressed using dictionary based compression algorithm. Non-repetitive parts of the DNA are generally compressed using dynamic programming, by dividing the sequences in square matrices which contain common repeat of a single base and then substituting the matrix with the base and putting the order of the matrix in a string. In this paper, a novel algorithm for DNA compression is proposed in order to compress both repetitive and non repetitive DNA sequence. The algorithm is also compared with existing ones and is found to achieve better compression ratio than the others.

Index terms—*BioCompress, GenCompress, LZ, LZW, Huffman Coding, Arithmetic Coding, transposon, junk DNA, LINE, SINE.*

I. Introduction

Life represents organization. It is not chaotic or random. Thus, it is expected that the DNA sequences that encode life is to be nonrandom. The central dogma of life is hidden in the DNA. DNA transcribes mRNA which is translated to proteins.[1] Proteins play a mojor role in regulating all the biological functions.

It is well-known that DNA sequences, especially in higher eukaryotes, contain many tandem repeats; and also segments that produce noncoding RNA molecules like tRNA, rRNA. Genome may contain several copies of the same gene. Although human genome contains about 3 billion base pairs, only 3 % of it encodes protein. There are only about 25000 genes in human genome which encode about 100000 proteins by alternative splicing. Other 97% which doesn't encode proteins is called "junk DNA" and if often associated to the promoter region and regulates gene expression. Repeated sequences are of two basic types: unique sequences that are repeated in one area; and repeated sequences that are interspersed throughout the genome. [2]There are interspersed sequences are tandem repeats, with sequences that are found interspersed across the genome. They can be classified based on the length of the repeat as: SINE: Short interspersed sequences. The repeats are normally a few hundred base pairs in length. These sequences constitute about 13% of the human genome with the specific Alu sequence accounting for 5%. LINE: Long interspersed sequences. The repeats are normally several thousand base pairs in length. These sequences constitute about 21% of the human genome. [3]

Hence DNA sequences should be reasonably compressible. However, such regularities are often blurred by random mutations like point mutation, inversion, translocation, cross-over, and reversal events, as well as sequencing errors. It is well recognized that the compression of DNA sequences is a very difficult task.[4, 5, 6, 7]. Subsequent sections describe about the DNA related work on DNA compression. Section V deals with the proposed algorithm and the structure to define the algorithm. The algorithm has been analysed at section VI Lastly the paper ends with an example and comparison with existing methods and paving ways to future work.

II. DNA SEQUENCE

A DNA sequence only contain succession of A, C, G, and T, representing the four nucleotide subunits - adenine, cytosine, guanine, thymine bases covalently linked to phosphate backbone. DNA sequencing is the process of determining the nucleotide order of a given DNA fragment, called the DNA sequence. DNA sequencing has been achieved using the chain termination method, developed by Frederick Sanger in 1975.[8] Recently Pyrosequencing and 454 Sequencing are used for this purpose. [9] The size of the genome varies greatly in different organisms which are shown in the following table: [10]

Name of the organism	**Size of genome**
Bacterium, Escherichia coli	4×10^6
Amoeba, Amoeba dubia	6.7×10^11
Plant, Arabidopsis thaliana	1.2×10^8
Plant, Fritillaria assyrica	1.3×10^11
Plant, Populus trichocarpa	4.8×10^8
Fungus,Saccharomyces cerevisiae	2×10^7
Nematode, Caenorhabditis elegans	8×10^7
Insect, Drosophila melanogaster aka Fruit Fly	1.3×10^8
Insect, Bombyx mori aka Silk Moth	5.30×10^8
Insect, Apis mellifera aka Honey Bee	1.77×10^9
Mammal, Homo sapiens	3×10^9

Duplications shapes the genome. Duplications may range from extension of short tandem repeats, to duplication of a cluster of genes, and all the way to duplications of entire chromosomes or even entire genomes. Such duplications are fundamental to the creation of genetic speciality. Transposons are sequences of DNA that can move around to different positions within the genome of a single cell, a process called transposition. In the process, they can cause mutations and change the amount of DNA in the genome. About 45% of the human genome is composed of transposons and their defunct remnants. [11]

The DNA sequences only consist of 4 nucleotide bases A, C, G, T. 2 bits are enough to store each base. However, if one applies standard compression software such as the Unix compress and MS-DOS archive programs like pkzip and arj, they all expand the DNA with more than 2 bits per base.[12]

Today, increasing genome sequence data of organisms lead DNA database size two or three times bigger annually. Thus, it becomes very hard to download and maintain such data in a personal local system. So it requires effective compression techniques. Different algorithms have been proposed in this domain. Algorithms like DNACompress [13] , GenCompress[12], Biocompress[14] etc are developed using the characteristics of DNA sequences like point mutation or reverse complement. It gains a compression of about 1.76 bits per base (~22% compression ratio).[15] Although many algorithms are proposed to compress DNA, they only use the presence of only four bases A, T, G, C and the repetitive nature of DNA sequence. There exist general compression algorithms based on Context Tree Weighting (CTW) and Arithmetic Coding. Also algorithms like LZ77, LZ78 are used for this purpose. But they are inefficient as far as DNA compression is concerned. Order 2 coding which is a modification of Huffman coding to compress data is also used.

III. GENERAL COMPRESSION ALGORITHMS

Most lossless compression programs use two different kinds of algorithms: one which generates a statistical model for the input data, and another which maps the input data to bit strings using this model in such a way that "probable" (e.g. frequently encountered) data will produce shorter output than "improbable" data. Often, only the former algorithm is named, while the latter is implied (through common use, standardization etc.) or unspecified.

Statistical modeling algorithms for text (or text-like binary data such as executables) include:

* Burrows-Wheeler transform (block sorting preprocessing that makes compression more efficient)
* LZ77
* LZW

Encoding algorithms to produce bit sequences are:

* Huffman coding
* Arithmetic coding

The Burrows-Wheeler transform (BWT, also called block-sorting compression), is an algorithm used in data compression techniques such as bzip2. It was invented by Michael Burrows and David Wheeler.

When a character string is transformed by the BWT, none of its characters change value. The transformation rearranges the order of the characters. If the original string had several substrings that occurred often, then the transformed string will have several places where a single character is repeated multiple times in a row. This is useful for compression, since it tends to be easy to compress a string that has runs of repeated characters by techniques such as move-to-front transform and run-length encoding.
Arithmetic coding is a method for lossless data compression. It is a form of entropy encoding, but where other entropy encoding techniques separate the input message into its component symbols and replace each symbol with a code word, arithmetic coding encodes the entire message into a single number, a fraction n where ($0.0 \leq n < 1.0$). A dictionary coder, also sometimes known as a substitution coder, is any of a number of lossless data compression algorithms which operate by searching for matches between the text to be compressed and a set of strings contained in a data structure (called the 'dictionary') maintained by the encoder. When the encoder finds such a match, it substitutes a reference to the string's position in the data structure. Both the LZ77 and LZ78 algorithms work on this principle.

IV. RELATED EXISTING ALGORITHMS

Grumbach and Thai introduced DNA specific compression algorithm. Two common algorithms, BioCompress and BioCompress 2 are based on LZ77 and LZ78.

“Formatdb” compresses the sequence with a Huffman-like coding method but deals minor symbols very efficiently and can uncompress fast (http://sapiens.wustl.edu/blast/blast/ncbi20ntfmt.html)F or dictionary-based methods, LZ77 scheme is known to be the best method for compressing DNA data so far. Several DNA-oriented algorisms have been tried to make the best of the haracteristics of DNA such as reverse complement and point mutation in order to apply LZ77 scheme more efficiently. In GSCompress, we employed LZ77 scheme with reverse complement as a dictionary-based scheme.

E. Rivals et al. [7] give another compression algorithm Cfact, which searches the longest exact matching repeat using sux tree data structure in an entire sequence. The idea of Cfact is basically the same as Biocompress-2 except that Cfact is a two-pass algorithm. It builds the sux tree in the 1st pass. In the encoding phase, the repetitions are coded with guaranteed gain; otherwise, two-bit per base encoding will be used. This is similar to the codeword encoding condition in Biocompress-2 except that the order-2 arithmetic coding is not used in Cfact. E. Rivals et al. [16] also designed a compression algorithm as a tool to detect the approximate tandem repeats in DNA sequences.
Sadeh [18] has proposed lossy data compression schemes based on approximate string matching and proved some asymptotic properties with respect to ergodic stationary sources. However, we are not interested in lossy compression.

GenCompress is a one-pass algorithm. It proceeds as follows: For input w, assume that a part of it, say v, has already been compressed, and the remaining part is u, i.e. $w = vu$. *GenCompress fi*nds an optimal prefix of u such that it approximately matches some substring in v so that this prefix of u can be encoded economically. After outputting the code of this prefix, remove the prefix from u, and append it to the suffix of v. Continue the process till u = €. [12]

If we know the string u and an edit operation sequence _($u; v$) from u to v, then the string v can be constructed correctly using lamda. There are many ways to encode one string given another. Using the above example, we describe four ways to encode “gaccgtca" using string “gaccttca".
1. Two bits encoding method. In this case, we can simply use two bits to encode each character,i.e. 00 for *a*, 01 for *c*, 10 for *g*, 11 for *t*. Thus “10 00 01 01 10 11 01 00" encodes "gaccgtca". It needs 16 bits in total.
2. Exact matching method. We can use (repeat position, repeat length) to represent an exact repeat. This way, for example, if we use three bits to encode an integer, two bits to encode a character, and use one bit to indicate if the next part is a pair (indicating an exact repeat) or a plain character, then the string “gaccgtca" can be encoded as $f(0; 4); g; (5; 3)g$, relative to “gaccttca". Thus, a 17 bits binary string “0 000 100 1 10 0 101 011" is required to encode the $f(0; 4); g; (5; 3); g$.

3. Approximate matching method. In this case, the string “gaccgtca" can be encoded as $f(0;\ 8);\ (R;\ 4;\ g)g$, or “0 000 111 1 00 100 10" in binary, with R encoded by 00, I encoded by 01, and D encoded by 11, and 0/1 indicating whether the next item is a doubleton or triple. A total of 15 bits is needed.

4. For approximate matching method, if we use the edit operation sequence (I,4,g),(D,6). Then the string “gaccgtca" can be encoded as $f(0;\ 8);\ (I;\ 4;\ g);\ (D;\ 6)g$, or “0 000 111 1 01 100 10 1 10110", in total 21 bits.[12]

V. Proposed Algorithm

In the previous section, various DNA compression techniques have been mentioned. But they achieve a compression of only 1.76 bits per base(~22% compression). To improve the parameter, a new technique named GenomeCompress has been devised which is much effective from storage and time point of view. Here an encoding scheme containing 5 possible bits has been introduced. Thus in this coding scheme $2^5 = 32$ characters can be represented. Hence every DNA segment containing four bases is replaced by a 5 bit binary number.

GenomeCompress is an one pass algorithm. It takes an input of a DNA sequence of length n, and divides into (n-r)/4 segments where n = r mod 4. It also employs four unique five bit binary numbers for each of AAAAAAAA, CCCCCCCC, GGGGGGGG and TTTTTTTT. Whenever the eight repeated sequence of A, C, G, and/or T are found in the sequence, they are replaced by an unique five bit binary number. This is continued until length of the segment = r. Four 5 bit binary numbers are given to represent bases A, C, G and T which fall in the segment r.

There are only four bases i.e. A,T, G, C found in DNA sequence. So there are 4!= 24 DNA segments each containing four bases. So 24 five bit binary numbers are required to encode them. Out of eight five bit binary numbers, four are utilized to encode repetitive DNA sequence and the remaining is used to encode each bases of segment r. Thus 32 five bit binary numbers are utilized to encode DNA. The algorithm has been shown below:

Encoding Algorithm:

Input: Input String(INSTRING) Containing A, T, G and C

Output: Encoded String (OUTSTRING)

```
PROCEDURE ENCODE
Begin
  Divide INSTRING into segments of length 4
    if two consecutive segments only contain eight A, T, G or C
      encode consecutive sequence with 5 bit binary numbers
      transfer the five bit binary number to the output string(OUTSTRING)
  else
    for each segment
      Begin
        find out the corresponding 5-bit binary number.
        replace the segment by five bit binary number
        transfer the five bit binary number to the output string(OUTSTRING)
      End
    for the remaining sequence of r where n = r mod 4 ,
        encode each A, T, G and C with unique 5 bit binary number
        transfer the five bit binary number to the output string(OUTSTRING)
  End
```

Decoding is carried out in the reverse manner of encoding. During decoding each five bit binary number is replaced by the complementary DNA segment, either containing four or eight bases or just one base. The DNA segments are stored in an array of length 32 where the address of the each element is denoted by a five bit binary number. During decoding process whenever the program finds a five bit binary number, it replaces it with the DNA sequence stored in its address. The decoding procedure is as under:

Decoding Algorithm

Input: Input String

Output: Decoded String(DECSTRING)

```
PROCEDURE DECODE
Begin
 Divide the input binary string into segments of length 5
  For each binary segment of length 5
    Begin
      Convert each segment into corresponding A, T, G, C sequence.
      Transfer the sequence to the DECSTRING
    End
End
```

VI. ANALYSIS

Assume that n is the length of the sequence and n=r (mod 4) .Suppose a,b,c,d denotes the number of 8 repetitive A, T, G, C sequences (e.g., in AAAAAAAACCCCCCCC GGGGGGGGTTTTTTTT, a=b=c=d=1). If each of these eight repetitive sequence requires five bits(binary) then the total number of bits (B) required to encode the sequence of n byte can be obtained as follows:

B=5/4*(n-r) + 5*r- (a+b+c+d)*5/8= 1.25n+ 3.75r – 0.625*(a+b+c+d)

Here 5 bits are required to encode 4 bytes. So 5/4 or 1.25 bits are required to encode 1 byte or one base. Hence 5/4*(n-r) bits are required to encode a sequence of length (n-r). Because each of the bases of the segment "r" of DNA is represented by a 5 bit binary number, 5*r bits are needed to encode it. As 0=<r<=3, 0=<5*r<=15, so the value of 5*r is negligible when the value of the n increases. In other words the length of the DNA sequence is huge. As 8 repetitive A, T, G, C are represented by four unique five bit binary numbers (a+b+c+d)*5/8 bits can be saved to encode the sequence. This encoding scheme exploits the repetitive nature of the DNA sequence often pronounced in short tandem repeats or inverted repeats. However the other encoding strategy utilizes the non repetitiveness of a DNA sequence. Therefore for a non repetitive huge sequence i.e. a = b = c = d = 0, approximately 1.25 bits is required to encode each base.

VII. EXAMPLE AND COMPARISON

Let us consider the sequence GAAT TTGC AAAA AAAA GCTA ATGC CTAG GGTT TTTG CCCC CCCC AAAA TCAG TTGC ATAG GACG . This sequence is of length 64 and 64 bytes are required to store it in a text file. If it is compressed using zip program included in Windows XP, the size becomes 163 bytes which is not intended at all. This proves that it is very difficult of compress a DNA sequence and general compression programs just increase its size while compressing. By *GenomeCompress* every segment of length four is replaced by a 5 bit number and eight repetitive A,T, G and C are replaced by four unique 5 bit numbers. Two AAAAAAAA and CCCC CCCC are found in this sequence. So 5*12+5*2 bits or 70 bits or just 9 bytes are required for compressed sequence by *GenomeCompress*. GenCompress, Biocompress, DNACompress and other softwares take about 14 bytes for compressed sequence because they give about 22% compression in general. Therefore *GenomeCompress* performs significantly better than other existing algorithms for DNA compression.

Other specialized DNA compression programs are either optimized of non repetitive nature of DNA sequence or repetitive nature of DNA sequence. Compared to other algorithms which give compression nearly 1.76 bits per base on benchmark sequences GenomeCompress compresses up to 1.25 bits per base(15.625 % compression ratio). Because this algorithm is relatively simple compared to other algorithms like BioCompress, GenCompress etc it takes lesser time to execute.

Thus our proposed algorithm *GenomeCompress* has the following advantages:
i) Compression ratio of 1.25 bits per base compared to 1.76 bits per base for the other algorithms.

ii) Because the method doesn't use dynamic programming technique which was used by other methods e.g., BioCompress, GenCompress etc, it is simple and takes less execution time.

iii) It uses less memory compared to the other algorithms.

VIII. Concluding Remarks

A new algorithm which is completely new in its design is proposed to compress DNA sequences which are repetitive as well as non repetitive in nature. All other existing algorithms are based on either statistics based or dictionary based.

DNA sequence analysis i.e. single and multiple alignments are areas of active research in bioinformatics. If the sequence is compressed using GenomeCompress it will be easier to make sequence analysis between compressed sequences. It'll also be easier to make multiple sequence alignment. High compression ratio in this algorithm also suggests a highly repetitive sequence. The compression method can be improved by incorporating dynamic programming technique.

REFERENCES

[1] Crick, F., 1970, Central Dogma of Molecular Biology. Nature 227, 561-563
[2] Voet & Voet, Biochemistry, 3rd Edition, 2004
[3] Pierce, B. A. (2005). Genetics: A conceptual approach. Freeman. Page 311
[4] Curnow, R. and Kirkwood, T., Statistical analysis of deoxyribonucleic acid sequence data { a review, *J. Royal Statistical Society*, 152:199{220, 1989.
[5] Grumbach, S. and Tahi, F., A new challenge for compression algorithms: genetic sequences, *J. Information*
Processing and Management, 30(6):875-866, 1994.
[6] Lanctot, K., Li, M., and Yang, E.H., Estimating DNA sequence entropy, to appear in *SODA '2000.*
[7] Rivals, _E., Delahaye, J.-P., Dauchet, M., and Delgrange, O., A Guaranteed Compression Scheme for Repetitive DNA Sequences, LIFL Lille I University, technical report IT-285, 1995.
[8] F. Sanger, S. Nicklen, and A. R. Coulson, DNA sequencing with chain-terminating inhibitors, Proc Natl Acad Sci U S A. 1977 December; 74(12): 5463–5467
[9] Ronaghi M, Pyrosequencing sheds light on DNA sequencing, Genome Res. 2001 Jan;11(1):3-11.
[10] Gregory, TR (ed) (2005). The Evolution of the Genome. Elsevier.
[11] Lewin B, Genes VIII, Oxford University Press
[12] Xin Chen et al, A Compression Algorithm for DNA Sequences
and Its Applications in Genome Comparison
[13] Xin Chen et al, *DNACompress: fast and effective DNA sequence Compression***,** Bioinformatics
Applications Note Vol. 18 no. 12 2002
Pages 1696–1698
[14] S Grumbach, F Tahi, LC INRIA,Compression of DNA sequences, Data Compression Conference, 1993. DCC'93., 1993
[15] T Matsumoto, K Sadakane, H Imai , Biological sequence compression algorithms , Genome Inform. Ser. Wokrshop Genome Inform, 2000
[16] Rivals, _E., Delgrange, O., Delahaye, J.-P., Dauchet, M., Delorme, M.-O., H_enaut, A., and Ollivier, E., Detection of signi_cant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences, *CABIOS*, 13(2):131{136, 1997.