GENBIT COMPRESS – ALGORITHM FOR REPETITIVE AND NON-REPETITIVE DNA SEQUENCES.

P.RAJA RAJESWARI [1]   Dr. ALLAM APPARAO [2]

(1) Acharya Nagarjuna University,Guntur.
(2)Vice-chancellor ,Jawaharlal Nehru Technological University, Kakinada.

ABSTRACT

The Deoxyribonucleic acid(DNA) constitutes the physical medium in which all properties of living organisms are encoded. Molecular sequence databases(e.g.,EMBL,Genbank, DDJB, Entrez, SwissProt, etc) currently collect hundreds of thousands of sequences of nucleotides and amino acids reaching to thousands of gigabytes and are under continuous expansion. Need for Compression arises because approximately 44,575,745,176 bases in 40,604,319 sequence records are there in the GenBank database (http://www.ncbi.nlm.nih.gov/Genbank/).Efficient compression may also reveal some biological functions and helps in phylogenic tree reconstruction etc.

We present a compression algorithm, “GenBit Compress” for DNA sequences based on a novel algorithm of assigning binary bits for smaller segments of DNA bases to compress both repetitive and non repetitive DNA sequence. our proposed algorithm achieves the best compression ratio for DNA sequences for larger genome. As long as 8 lakh characters can be given as input. Significantly better compression results show that GenBit Compress algorithm is the best among the remaining compression algorithms. While achieving the best compression ratios for DNA sequences(Genome),our new GenBit Compress algorithm significantly improves the running time of all previous DNA compression programs. We have also identified that it is a good idea to express the performance of an algorithm as a function of the input size. For the first time we have defined the Worst case, Average case and Best case for DNA compression using our proposed Algorithm. Assigning binary bits for fragments of DNA sequence is also a unique concept introduced in this algorithm for the first time in DNA compression.

Keywords: *BioCompress,GenCompress,Arithmetic coding,Huffman coding,LZ,LZW,worst,Best,Average.*

1. INTRODUCTION:

In modern molecular biology, the **genome** is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of virus, in RNA.The genome includes both the genes and the non-coding sequences of the DNA.(Ridley, M. (2006) *Genome).*

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 data in a local system. For a four-letter alphabet in DNA(A,C,G,T),an average description length of 8 bits can be assigned per fragment. Other algorithms specifically designed for DNA sequences compression did not manage to achieve average compression rate below 1.6 bits/base. Algorithms for Compressing DNA sequences, such as GenCompress[1] ,Biocompress[2] and Cfact[3] are available to compress DNA sequences. Their compression rate is about 1.74 bits per base i.e.,78% in compression rate[4]. Hence we present a new compression algorithm named ”GenBit Compress” whose compression rate is below 1.2 bits per byte(for Best case) even for larger genome(nearly 2,00,000 characters).

1.1 EXISTINGCOMPRESSION ALGORITHMS

Most of the compression methods used today including DNA compression fall into two categories.

- First is statistical method, which compresses data by replacing a more popular symbol to a shorter code.
- Second is dictionary-based scheme, which compresses data by replacing long sequences by short pointer information to the same sequences in a dictionary.

In statistical methods, arithmetic coding and CTW are known to compress the DNA data well [5] and Huffman coding is known to compress not very efficiently. The Burrows-Wheeler transform (BWT, also called block-sorting compression),(Michael Burrows and David Wheeler.) is an algorithm used in data compression techniques such as bzip2. Both the LZ77 and LZ78 algorithms work on this principle. In GS Compress,LZ77 scheme with reverse complement is introduced as a dictionary-based scheme. E. Rivals et al.[6] another compression algorithm Cfact, which searches the longest exact matching repeat using sux tree data structure in an entire sequence. Sadeh has proposed lossy data compression schemes based on approximate string matching and proved some asymptotic properties with respect to stationary sources. In spite of the good compression ratio, arithmetic coding and CTW have disadvantages such as low decompression speed.

2. PROPOSED ALGORITHM – GENBIT COMPRESS

In this paper we consider the problem of DNA compression both for repetitive and non repetitive DNA sequences.To improve the compression rate, a new technique named GenBit Compress has been devised, which is much effective with respect to compression rate. Here an encoding scheme containing 8 possible bits has been introduced. Since the DNA sequence contains only A,C,G,T letters, we named each character
as a "Base".
The input sequence is divided in to fragments, where each fragment = 4 characters. Thus in this coding scheme, 256(2 power 8 = 256)combinations can be represented. Hence every DNA segment containing four bases is replaced by a 8 bit binary number "00000000". 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 with out 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")

The Total number of bits per byte($\Re$) is calculated as :
$\Re = 9/4\ (n - \tau) + 2(\ \tau\ ) - 9\ (\ \Upsilon\ )$

Where n = length of the given sequence.
τ = (n mod 4), number of bases
excluded from (n mod 4).
Υ =Number of repetitive fragments (fragments = 4 bases{ACGT}) present in the given sequence.

Compression Rate = Number of Bits/Total number of Bytes.

Separate analysis for the proposed algorithm is given for all the three cases(worst case ,best case and average case).
Worst case: In the worst case there are no repetitive fragments and individual bases are maximum and (τ<4)
In this algorithm , the worst-case compression rate is the highest.(Compression Rate: 2.238)

Best Case: In this algorithm ,maximum repetitive fragments are n/8 and τ=0. The Best-case efficiency is proved in this GenBit algorithm since its compression Rate = 1.125, which is the best among other cases.

Average Case:
The Average case efficiency of this algorithm defines the compression rate of a typical input or a random input which is not given by neither the worst case nor the best-case efficiency. Number of fragments= n/16 and τ=2 ,Compression Rate = 1.727.

2.1: ANALYSIS FOR VARIOUS CASES.

<u>CASE 1:</u> DNA sequence with same fragments.

Example 1:
Given DNA input sequence = aaaa aaaa
length of n = 8.

Assigned Unique bit number = "00000000 1"
The nineth bit "1" is the specific bit since the Consecutive fragments "aaaa aaaa" match with each other.

CASE 2: DNA SEQUENCE WITH DIFFERENT FRAGMENTS.

Example 2:

Given DNA input sequence = acgt atgc

Length of n = 8
Assigned unique bit number = “00000000 0”
The nineth bit “0” is the specific bit since the consecutive fragments “acgt atgc” does not match with each other.

CASE 3: DNA SEQUENCE WITH $\tau = 0$

Example 3:
Given DNA input sequence = agct aaaa
Length of given sequence = 8.
Number of individual bases(τ) = n mod 4 = 8 mod 4 =0.

Assigned unique bit number = “001010000 000000000”.

CASE 4: DNA SEQUENCE WITH(n mod 4 $\neq$ 0) $\tau \neq 0$.

Example 4:
Given DNA input sequence = agct aaaa tt

Length of input sequence = 10.
Assigned unique bit number = “001010000 000000000 1111”
The individual bases which are excluded after fragmentation , allocates “2” bits for “t t” i.e.,t=”11”.

2.2: ENCODING ALGORITHM

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

Output: Encoded String (OUTSTRING)

PROCEDURE ENCODE
Begin

1: Divide the given DNA sequence in to fragments, where each fragment consists of 4 characters.

2: Generate all possible combinations of DNA sequence (A,C,G,T).(Since the sequence contains 4 different bases, there will be 4^4 = 256 combinations).

3: Assign unique 8 bit number(“0” & “1”) to each fragment.

4: If the consecutive fragments are same,assign a specific bit “1” to the 8 bit unique number as a 9th bit.

5: If the consecutive fragments are different ,assign a specific bit “0” to the 8 bit unique number as a 9th bit.

6: Repeat the steps 4 and 5 until the length of sequence is “n- τ”.(where n = length of the given sequence and τ = n mod 4)

7: Allocate unique “2” bit number to individual bases if $\tau > 0$.

8: Transfer the 9 bit binary number to the output String
(OUTSTRING).

End

The Decryption algorithm involves the same procedure as Encryption in the reverse form.

2.3: DECODING ALGORITHM

Input: Input String

Output: Decoded String(DECSTRING)

PROCEDURE DECODE

Begin

1: Generate all possible combinations for {A,C,G,T}.

2: Allocate unique 8 bit number to each combination.

3: Divide given binary code in to 9 bit segments.

4: If 9th bit is equal to “1” , the corresponding combination is taken two times , otherwise normal.

5: Repeat step 4, until the end of the input sequence is reached.

6: If there are any individual bases(τ>0), the corresponding binary code gets transformed.

(Assigned values for bases are :a=”00”, G=”01”,c=”10”,t=”11”).

End

3: EXAMPLE AND COMPARISON

The Total number of bits per byte ($\Re$)is calculated as :

$$\boxed{\Re=9/4(n-\tau)+2(\tau)-9(\Upsilon)}$$

where n = length of the given sequence.

τ= (n mod 4) , number of bases excluded from (n mod 4).

Υ = Number of repetitive fragments(fragments = 4 bases{ACGT}) present in the given sequence.

3.1: Worst case:

Consider there is no repetitive fragments(4 bases - {A,C,G,T}) and individual bases are maximum.i.e.,(τ< 4).

$\Re = 9/4 (n - \tau) + 2(\tau) – 9 (\Upsilon)$.

Example:

let n=67

τ = n mod 4

= 3

= 9/4(64)+2(3)-0.

Total bits($\Re$) = 150.

Compression Rate = $\Re$/n

= 150/67

= 2.238 bits/bytes.

3.2: Best case:

Consider there is a probability of occurance of ,a maximum of n/8 repetitive fragments in the given sequence. i.e., (τ = 0)

Then

Total number of Bits($\Re$) = $9/4 (n - \tau) + 2(\tau) – 9 (\Upsilon)$.

Example:

let n=64

τ = 0 Υ=8.

$\Re$ = 9/4(64)+2(0)-9(8)

= 144-72

= 72.

Compression Rate = $\Re$/n.

= 72/64 = 1.125 bits/bytes.

3.3: Average Case:

Consider τ = 2 and number of fragments = n/16.

$\Re$ = 9/4(64)+2(2)-9(4)

= 144+4-36

= 114

Compression Rate = $\Re$/n

= 114/66 = 1.727 bits/bytes.

4: METHODOLOGY OF GENBIT COMPRESS

Assume that n is the length of the sequence and fragments τ = n (mod 4) .Each fragment(ACGT)is replaced with 9 bits binary code(0 or 1). For any fragment followed by different fragment the bits assigned are “000000000”. When there is a repetitive segment followed by any segment then the value of the 9th bit (specific bit) raises to “1” otherwise “0”.Then the total number of bits required to encode the sequence of n byte can be obtained as follows:

$\Re=9/4(n-\tau)+2(\tau)–9(\Upsilon)$.

Approximately **1.125 bits per byte** is required to encode each base.

Let us consider the sequence:

GAAT TTGC AAAA AAAA GCTA ATGC CTAG GGTT TTTG CCCC CCCC AAAA TCAG TTGC ATAG GACG .

SequenceLength	= 64.
Bytes to store in a text file	= 64 bytes.
Windows XP zip size	= 163 bytes.
Biocompress	= 14 bytes.
GenBit Compress algorithm	= 9 bytes.

Thus our proposed algorithm GenBit compress has the following advantages:

i) Compression ratio of 1.125 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.

5: CONCLUSION

A simple DNA compression algorithm which is completely new in its design is proposed to compress DNA sequences which are repetitive as well as non repetitive in nature. DNA sequence analysis i.e. single and multiple alignments are areas of active research in bioinformatics. If the sequence is compressed using GenBit Compress algorithm, it will be easier to compress large bytes of DNA sequences with the compression ratio of 1.125 bits per base which will be very useful in sequence comparisons and Multiple sequence Alignment analysis.
The simplicity and flexibility of **GenBit Compress algorithm** could make it an invaluable tool for DNA compression in clinical research.

6: LIMITATIONS

Summative evaluation of learning outcomes such as testing the real biological sequences on this alogrithm, performance with the tools, or the transfer of knowledge to similar tasks could not be performed.

7: FUTURE WORK

(1)The compression algorithm can be improved by incorporating dynamic programming technique to the GenBit Compress algorithm.

(2)Developing a java based tool for GenBit compress algorithm.

ACKNOWLEDGEMENTS

Authors are thankful for the support rendered by C.K.Krishna and V.K .Kumar during its development phase**.**

REFERENCES

[1] Chen,X.,Kwong,S.,and Li,M.,A compression algorithm for DNA sequences and its applications in genome comparison,Genome Informatics ,10:52 –61,1999.

[2] Grumbach,S.and Tahi,F.,A new challenge for compression algorithms:genetic sequences, Information Processing & Management 30:875 –886,1994.

[3] 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.

[4] Matsumoto,T.,Sadakane,K.,and Imai, H., Biological sequence compression algorithms, Genome Informatics 11:43 –52,2000.

[5] Matsumoto, T., Sadakane, K., and Imai,H.,Biological sequence compression algorithms,Genome Informatics 11:43 –52,2000.

[6] 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.

P.Raja Rajeswari received her post graduate degree in Computer Applications in 1999 and M.Tech[IT] in 2003. She is working as Assistant Professor in DMSSVH college of Engineering ,Machilipatnam since 2000 to till date.She is pursuing her Ph.D from Acharya Nagarjuna University in Computer Science under the guidance of Dr. Allam Appa Rao. Her research interests includes Bioinformatics, compression techniques, design and analysis of Algorithms,development of software tools.

Dr. Allam Appa Rao has received PhD in Computer Engineering from Andhra University, Visakhapatnam, Andhra Pradesh, India.He has worked as the Professor in Bioinformatics & Computational Biology, Department of Computer Science and Systems Engineering &Principal, Andhra University College of Engineering (AUTONOMOUS). Currently he is Vice Chancellor to Jawaharlal NehruTechnological University, Kakinada. His research interest includes Bioinformatics, Software Engineering and Network Security. He is a member of professional societies like IEEE, ACM and a life member of CSI and ISTE. www.allamapparao.net.