Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 6 (2017) pp. 1571-1576

http://www.ripublication.com

A Modified Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Look up Table (LUT)

Govind Prasad Arya
Ph.D Scholar, Uttarakhand Technical University, Dehradun
&
Assistant Professor, Sikkim Manipal University, Gangtok, India

Dr. R.K. Bharti
Assistant Professor, BTKIT Dwarahat, Dist-Almora, Uttarakhand, India

Devendra Prasad
Ph.D Scholar, Uttarakhand Technical University, Dehradun, India.

Vishal Garg
B.Tech Student, Shivalik College of Engineering, Dehradun

Abstract

DNA sequences consist of sequentially connected nucleotides, A (Adenine), C (Cytosine), G (Guanine) & T (Thymine). The DNA sequences are useful for diseases detection, criminal identification etc. Human DNA consists of about 3 billion bases and about 20,000 genes, and more than 99% of those bases are similar in every human being. These DNA sequence are very much large in size. This large amount of data causes difficulty to store, analyse and process DNA sequences. Therefore, DNA sequence compression has become the topic of research for the researchers worldwide. In this paper we are presenting an efficient and fast compression algorithm based on differential direct coding and variable look up table (LUT).

Keywords: Differential Direct Coding, Look Up Table, LUT, Compression, Decompression, Nucleotide Data Compression, DNA Compression Algorithm.

INTRODUCTION

As DNA sequence compression has become wide area of research for researchers, therefore in labs research is continuously going on for analysing these DNA sequences. For this purpose DNA sequences need to be stored somewhere and transmitted from one place to another, but because DNA sequences are of very large size, it results in very high transmission cost. From 1982 to present, the numbers of bases in GenBank are getting doubled approximately in every 18 months. So we require a efficient algorithm to compress these DNA sequences. There is a direct coding algorithm which uses 2 bits for representing each of the 4 nucleotides. As DNA sequence consists of 4 nucleotide bases A, C, G & T called exons(i.e. coding regions or protein synthesis) or introns(i.e. non-coding regions or no protein synthesis), 2 bits are sufficient to represent each of the 4 bases.

LIMITATIONS OF EXISTING ALGORITHMS

Some standard compression algorithms like GZIP, COMPRESS, BZIP2, WinRAR or WinZip requires more than 2 bits per byte. This is because of the fact that these algorithms do not use the common properties available in DNA sequences.

Algorithms like GenCompress, BioCompress utilizes properties available in DNA sequences for compression & their compression rate is around 1.74 bits per base i.e. 78% in compression rate. But these algorithms have very high running time.

Shortcomings of Existing Algorithm -"A Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Lookup Table (LUT)**"** [8]

- **a)** This algorithm stores all the possible combinations of {A, G, T, C, U}, however some of the combinations are not acceptable so of no use, therefore some non-printable ASCII characters are left unused.
- **b)** It only concentrates over the triplets available in DNA sequences, even there may be a possibility that other combinations may lead to better compression ratio.
- **c)** This algorithm uses ASCII code ranging from -65 to -2 for fixed sub-strings of length 3 and -128 to -66 for variable length sub-strings which is multiple of 3 & other ASCII codes are not used.
- **d)** Sometimes there might be an ambiguity in identifying the nucleotide; it could either be an A or a C, or an A or G and so on. To identify all these possible combinations, the extended DNA alphabet is used where all the 15 possible combinations of the standard 4 nucleotides are given unique symbols. We identify the extended DNA alphabet as Σ= {A, B, C, D, G, H, K, M, N, R, S, T, V, W, Y}. The above algorithm has not used extended DNA alphabet.

PROPOSED Algorithm

In this paper we have presented an algorithm which is the modification of "A Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and

Variable Length Lookup Table (LUT)**"** [8] in following ways-

a) The previous algorithm used only negative ASCII codes for representing the fixed and variable length sub-strings. While our algorithm uses 240 ASCII characters excluding extended DNA alphabet & -1 for representing the end of file.

b) The previous algorithm used only 4 nucleotide bases while our algorithm uses extended DNA alphabet.

When we compress biological sequences then we know that at a time only DNA or mRNA sequence would be compressed. Hence, if we consider the case of formation of triplets with combination of four symbols {A, C, G, T} or {A, C, G, U} for DNA or mRNA respectively, there would be maximum 64 combinations. These 64 triplet combinations require only 64 ASCII codes, whereas there exist total 127 non-printable ASCII characters & 128 printable characters. Therefore, the remaining characters would be used to store some other combinations of size more than 3, which would yield to result in a better compression. We have divided the modified look-up table into two parts: fixed length LUT and variable length LUT. The fixed length LUT will always contain 64 combinations of triplets, but the variable length LUT can be of variable size containing the combination of bases of the size which is in multiple of 3. This technique will utilize whole ASCII character set (excluding DNA alphabet & -1 for representing EOF) and would yield a better compression ratio.

Model: We consider the ASCII characters [2] between the ranges -128 to 127. The range between 49 to 127 is used to adjust fixed size LUT and remaining ASCII codes except -1(used for EOF) is used to accommodate variable size LUT.

Coding: Here we have encoded DNA sequence using method described in the above model. As we would start scanning DNA sequence, firstly we would scan triplet from DNA sequence and search for this triplet from the fixed size LUT. As triplet would already be available in fixed size LUT so it would further scan next triplet from DNA sequence. Therefore now the total scanned characters will be of length 6. If this subsequence of 6 characters is available in the variable size LUT then we would scan next triplet, otherwise we will write the ASCII character corresponding to the last matched group in the output file and also insert the current combination/group into the variable size LUT and assign one respective ASCII code to it if available. This process will repeat till the whole DNA sequence is encoded. But once all the ASCII characters are allocated (i.e. variable size LUT is filled, we would not store more patterns and just search for the combinations in DNA sequence from both parts of the table (i.e. fixed size LUT & variable size LUT). And this process will be followed by writing the output corresponding to that combination to the output file. Once the whole DNA sequence has been scanned then final output file would be the required compressed file. While scanning the DNA sequence there might be a situation (i.e. at the end of file 1 or 2 characters are left which can't form a pair) where a triplet cannot be formed. In that case we would write that single character or a group of two characters into the output file as it is. Whenever an

unexpected symbol is obtained then we will stop scanning the triplets and write that unexpected character's ASCII equivalent into the output file.

Algorithm
Initialize: s=NULL Initialize: st=NULL Initialize: t=NULL
Step1: Read first three unprocessed characters (t). If t! =NULL, goto **step2.**
Else: process the last one or two characters by **step3**.

Step2: If st exists in the LUT* then s=st
Else {

End if

Output the code (character) for s
//signed Byte code (character) that is mapped in LUT (From -128 to 127). Add st to the LUT table;
s=t;
}

Step3: Write single or group of characters into a output file directly without any modification whose length is less than 3. After that **goto step4**.
Step4: Return to step1 and repeat all process until EOF is reached.

RESULTS
a) Our proposed algorithm has achieved higher compression ratio in comparison to other existing DNA sequence compression algorithms. This algorithm would require less amount of memory as compared to the other algorithms and it requires less amount of time in comparison to other algorithms as well as it is easy to implement.
b) Our proposed algorithm compress Nucleotide sequences like DNA as well as RNA.
Most of the other compression algorithms mostly depend on the other properties of
sequences such as repeated and non- repeated patterns. If the sequence is compressed using our proposed algorithm then it would be easier to make sequence analysis between compressed sequences. It will be easier to make multi sequence alignment as well. The compression results for differential direct coding using variable length LUT & differential direct coding using modified variable length LUT are shown in Table 3.

CONCLUSION
If we use variable length LUT then compression of the DNA sequence would be more than 1/3 of its original size. But now we are able to achieve more compression ratio by using modified variable length LUT. In this research paper, we have purposed some modification in differential direct coding with variable

length LUT. Our result shows that the proposed algorithm is much better than the existing one. The proposed algorithm would provide much better compression because longer sequences are found frequently in big DNA sequences.

Table 1. The 2D Coding With Modified Variable Length LUT Data Model

Type of Data	Description	Range	Look-UpTable
Auxiliary Symbol	ASCII	-128 to 127	
Triplets	Set of three base characters	49 to 127	Fix Length LUT
Multiple of Triplet	Set of **6,9,12…** base characters	-128 to 48	Variable Length LUT

Table 2. The 2D Encoding Process with Modified Variable Length LUT

Step	Input Sequence	Triplet (t)	Multiple of Triplet(st)	Look-Up Table		Encoded Sequence(s)
				Status of st	Entry	
1	ACTGCTACTGCTACTGCT					
2	ACT	ACT	ACT	Found	ACT=# GCT=+	
3	GCT	GCT	ACTGCT	Not Found	Add with ACTGCT=$	#
4	ACT	ACT	GCTACT	Not Found	Add with GCTACT=@	#+
5	GCT	GCT	ACTGCT	Found	ACTGCT=$	#+
6	ACT	ACT	ACTGCTACT	Not Found	ACTGCTACT = ^	#+$
7	GCT	GCT	ACTGCT	Found	ACTGCT = $	#+$$

Table 3. Compression Results

S.N	Type of Sequence	Original size of sequence before compression	Size of Sequence After Compression	
			Using Existing Algorithm	Using Proposed Algorithm
1	atatsgs	9647	3101	2951
2	atef1a23	6022	1957	1858
3	atrdnaf	10014	3276	3165

4	atrdnai	5287	1734	1700
5	chmpxx	15180	4874	4489
6	chntxx	155844	50540	48011
7	hehcmvcg	229354	74736	72397
8	humdystrop	105265	34347	33249
9	humhdabcd	58864	19201	18731
10	vaccg	47912	15374	14672
Average		**53812.4**	**20914**	**20122.3**

REFERENCES

[1] https://ghr.nlm.nih.gov/primer/basics/dna

[2] ASCII code.[Online]. Available: http://www.LookupTables.com

[3] Gregory Veyetal., Differential direct coding: a compression algorithm for nucleotide sequence data, Database,(2009), doi:10.1093/database/bap013.

[4] J. Ziv & A. Lempel., A universal algorithm for sequential data compression, (1977) *IEEE TransactionsonInformation Theory*, vol. IT-23.

[5] X. Chen & M. Lip, DNA compress: fast and effective dna sequence compression, (2002), *Bioinformatics*,vol.18.

[6] Bao, S. etal., A DNA sequence compression algorithm based on LUT and LZ77, (2005)

[7] Ateet Mehta, (2010), etal., "DNA Compression using Hash Based Data Structure", IJIT & KM, Vol2No.2,pp.383-386.

[8] Govind Prasad Arya, R.K. Bharti, (2012), "A Compression Algorithm for Nucleotide Data Based on Differential Direct Coding and Variable Length Lookup Table (LUT)", (IJCSIT) vol. 3(3) Page 4411-4416.

[9] Li Tan, Jifeng Sun, (2015), "K-means clustering based compression algorithm for the high-throughput DNA sequence", Publisher: IEEE, DOI: 10.1109/ICALIP.2014.7009935.