Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Biological Data Compression Methods Based Upon Microsatellite

Subhankar Roy
CSE Department, Academy of Technology, G. T. Road, Aedconagar, Hooghly-712121, W.B., India

Sunirmal Khatua
Department of Computer Science and Engineer, University of Calcutta, 92, A.P.C. Road, Kolkata-700009, W.B, India

Abstract. Data Storage costs have an appreciable proportion of total cost in the creation and analysis of DNA or RNA sequences. Modern biological science produces vast amounts of genomic sequence data. This is fuelling the need for efficient algorithms for sequence compression, alignment and analysis. Data compression and the associated technique coming from information theory are often perceived as being of interest for data communication and storage. There are number of data compression algorithms, which are dedicated to compress different data formats. Even for a single data type there are number of different compression algorithms, which use different approaches. Now Compression of biological sequence of both DNA and RNA has been proposed by means of microsatellite or simple sequence repeats or short tandem repeats (STR).

Keywords: microelectronics, variable length LUT, lower case letter, upper case letter, saving percentage.

I. INTRODUCTION

Earlier, a lot of text compression techniques have been applied on biological data, which worked less efficiently. Nowadays huge biological data are available, and growing in size with the passages of time. Hence, the information must be stored and communicate efficiently. The standard techniques of text compression do less compress these sequences; in some sequences they expand the size of original file. It is known that the compression means mapping between source file and destination file, so all compression algorithm leads to find those relationship within the sequences. The objective of this paper is to build a variable length LUTs in which the combination of two, three and four, five, six, seven, eight and nine bases of A, C, G, T or U are taken as input and by mapping relationship during encoding process output correspond to key characters. The mapping is done using hash map table whose inputs corresponding to the above said block size of bases and output is a key value. Then the algorithm will search microelectronics in the sequences. There are different types of possible repeats in a sequence. Microelectronics is a repeating sequence of two to six base pair. In nucleic acid i.e. DNA or RNA repeated sequences are patterns that occur in multiple copies throughout the genome as for example human genome. Compression is done on the individual DNA sequences, not on the whole genome sequence [1] using the above said repeats [2] as a combination of bases of size two, three, four, five and six respectively. The methods will also search other size of repeats of size seven, eight and nine consecutive bases respectively. So microelectronics is a type of variable number tandem repeats. This will help to search gene duplication.

The human genome (23 chromosomes out of which 22 pairs of autosomes and one pair of sex chromosomes X & Y) are nearly 3,079,843,747 base pair long and contain 32,185 genes, out of which 99% are same in all humans. Thus, organisms are primarily defined with these bases. That means there have redundant data i.e. repeat. That is why normal text compressions are not suitable for DNA data.

II. SOME EXISTING COMPRESSION ALGORITHMS

There have several DNA data compression algorithm like Block3+Tandem repeats, Block4 [3] etc, the compression factor, saving percentages is not so good. These algorithms use two static fixed lengths LUT of length 64 and 256. Then it maps using static hash map table to a key value. It counts repeat of block size 3. As it use very short tandem repeats the saving percentage by these algorithms are not good as required for huge biological sequence. Another compression algorithm that compresses all possible nucleic acid bases i.e. four completely specified A, C, G, T/U and eleven incompletely specified bases K, M, R, S, W, Y, B, D, H, V, N respectively [4] using static fixed length LUT of length 225 as a block size of two bases also do not fulfil the modern requirements fast communication.

Some compression algorithm uses the similarity of genomic sequence database [5]. Complementary nature of DNA sequence also uses for compression [6]. Another compression tool is GBC [7]. Algorithm using identical motifs [8] and approximate repeat [9] also give good result. Compression algorithm using hash based data structure [10] that is implemented by static LUT. All of these algorithms can operate only on frequent occurrence bases in DNA and RNA sequences. If there have some infrequent bases then they cannot hold it.

In this article, it has been tried to survive the above said problem. That means the proposed algorithm can work with both completely and incompletely specified bases. Also focus is on very good saving percentage within a less time. So that communication of data through network would be very fast.

III. PROPOSED METHODOLOGY

Here six algorithms named B4-B2R (Block Four to Two with Repeats), B5-B2R (Block Five to Two with Repeats), B6-B2R (Block Six to Two with Repeats), B7-B2R (Block Seven to Two with Repeats), B8-B2R (Block Eight to Two with Repeats) and B9-B2R (Block Nine to Two with Repeats) have been proposed. The compression and decompression are just reverse process. The differences between all of these algorithms are only of block size and their corresponding microelectronics. The major calculations in the proposed algorithms are explains below.

Mapping implemented by variable length Look up table of length equals to four to the power corresponding microelectronics size plus number of repeats of microelectronics size:

1. ***BnB2R Encoding Algorithm:*** *It is a generalised encoding algorithm. Here n greater than equals to four and less than equals to nine i.e. 4<=n<=9.*

   Input: DNA or RNA data file.
   Output: Encoded file.
   Step1: A Collection of bases.
   Step2: Two variable lengths LUT1 and LUT2 are created to map string of length n to a key value.
   Step3: Input file is taken to form variable length (n) LUT.
   Step4: Both LUT are formed to map unique block to key value ignoring repeated block.
   Step5: Again open the same sequence file.
   Step6: Output file is open to write encoded data.
   Step7: Read the sequence as a block of size n.
   Step8: Map using LUT1 of size 4^n if there is no repeat.
   Step9: If there are repeats then a counter variable will count the number of repeats.
   Step10: When a new block appear then stop counting and write the corresponding key value from LUT1.
   Step11: At the end of line if there are some remaining characters of size less than n, then using LUT2 of 4^2<=size<= $4^{(n-1)}$ map them to the corresponding key value.
   Step12: Continue until EOF

2. ***BnB2R Decoding Algorithm:*** *It is a generalised decoding algorithm. Where 4<=n<=9.*

   Input: Encoded file.
   Output: DNA or RNA data file.
   Step1: Open encoded file.
   Step2: Another file to store original recovered data.
   Step3: Read the encoded file character wise.
   Step4: Using reverse mapping by LUT1 and LUT2 of same size write original data to output file.
   Step5: Continue until EOF.

3. ***Best case, Average case and Worst case Saving percentage of the above algorithms:*** *For all algorithms it is depend on the input sequences.*

   3.1. B4-B2R algorithm:

   Best Case:
   E.g. if the input sequence is: ATGCATGCATGCATGCATGCATGCATGC…
   Output = }D…
   Saving Percentage = (1 – 2 / (8*4)) *100 = 93.75%

   Average Case:
   E.g. if the input sequence is: ATGCATGCATGCATGCACGCACGCACGCACGC…
   Output = H_IO…
   Saving Percentage = (1 – 4 / (8*4)) *100 = 87.50%

   Worst Case:
   E.g. if the input sequence is: ATGCACGCAGTCAAGCATGCATTGCAGGCATGC...
   Output = MNKJL:,{…
   Saving Percentage = (1 – 8 / (8*4)) *100 = 75.00%

   3.2. B5-B2R algorithm:

   Best Case:
   E.g. if the input sequence is: ATGCTATGCTATGCTATGCTATGCTATGCTATGCTATGCT…
   Output = ab…
   Saving Percentage = (1 – 2 / (8*5)) *100 = 95%

   Average Case:
   E.g. if the input sequence is: ATGCTATGCTATGCTATGCTACGCTACGCTACGCTACGCT…
   Output = abcd…
   Saving Percentage = (1 – 4 / (8*5)) *100 = 90%

   Worst Case:
   E.g. if the input sequence is: ATGCTACGCTAGGCTAAGCTATGCTACGCTAGGCTAAGCT...

Output = abcdabcd…
Saving Percentage = (1 – 8 / (8*5)) *100 = 80%

3.3 B6-B2R algorithm:

Best Case:

E.g. if the input sequence is: ATGCTTATGCTTATGCTTATGCTTATGCTTATGCTTATGCTTATGCTT…
Output = ef…
Saving Percentage = (1 – 2 / (8*6)) *100 = 95.83%

Average Case:

E.g. if the input sequence is: ATGCTTATGCTTATGCTTATGCTTACGCTCACGCTCACGCTCACGCTC…
Output = efgh…
Saving Percentage = (1 – 4 / (8*6)) *100 = 91.67%

Worst Case:

E.g. if the input sequence is: ATGCTAACGCTCAGGCTGAAGCTTATGCTAACGCTCAGGCTGAAGCT G...
Output = lmnolmno…
Saving Percentage = (1 – 8 / (8*6)) *100 = 83.33%

3.4. B7-B2R algorithm:

Best Case:

E.g. if the input sequence is:
ATGCTTAATGCTTAATGCTTAATGCTTAATGCTTAATGCTTAATGCTTAATGCTTA…
Output = 12…
Saving Percentage = (1 – 2 / (8*7)) *100 = 95.83%

Average Case:

E.g. if the input sequence is:
ATGCTTAATGCTTAATGCTTAATGCTTAACGCTCCACGCTCCACGCTCCACGCTCC…
Output = efgh…
Saving Percentage = (1 – 4 / (8*7)) *100 = 92.86%

Worst Case:

E.g. if the input sequence is:
ATGCTAGACGCTCAAGGCTGTAAGCTTCATGCTATACGCTCCAGGCTGAAAGCT AG...
Output = lmnolmno…
Saving Percentage = (1 – 8 / (8*7)) *100 = 85.71%

3.5. B8-B2R algorithm:

Best Case:

E.g. if the input sequence is:
ATGCTTACATGCTTACATGCTTACATGCTTACATGCTTACATGCTTACATGCTTACATGCTTAC…
Output = XY…
Saving Percentage = (1 – 2 / (8*8)) *100 = 96.88%

Average Case:

E.g. if the input sequence is:
ATGCTTAGATGCTTAGATGCTTAGATGCTTAGACGCTCCGACGCTCCGACGCTCCGACGCTCCG…
Output =PQRS…
Saving Percentage = (1 – 4 / (8*8)) *100 = 93.75%

Worst Case:

E.g. if the input sequence is:
ATGCTAGAACGCTCACAGGCTGTGAAGCTTCTATGCTATAACGCTCCTAGGCTGAGAAGCTACG...
Output = MNLKRTHI…
Saving Percentage = (1 – 8 / (8*8)) *100 = 87.50%

3.6. B9-B2R algorithm:

Best Case:

E.g. if the input sequence is:
ATGCTTACGATGCTTACGATGCTTACGATGCTTACGATGCTTACGATGCTTACGATGC TTACG…
Output = mn…
Saving Percentage = (1 – 2 / (8*9)) *100 = 97.22%

Average Case:

E.g. if the input sequence is:
ATGCTTAGAATGCTTAGAATGCTTAGAATGCTTAGAACGCTCCGTACGCTCCGTACGCTCCGTACG CTCCGT…

Output = prqo…

Saving Percentage = (1 – 4 / (8*9)) *100 = 94.44%

Worst Case:

E.g. if the input sequence is:

ATGCTTAGGATGCTTAGAATGCTTAGTATGCTTAGAACGCTCCGTACGCTCCGAACGCTCCGTACG CTCCGT…

Output = zfdcvenh…

Saving Percentage = (1 – 8 / (8*9)) *100 = 88.89%

4. ***Best case, Average case and Worst case due to line length in the sequences:*** *Here we have used 4*1, 5*1, 6*1, 7*1, 8*1 and 9*1 mapping using variable length LUT.*

Let us consider L be the line length of a sequence in bp.

There have three possible cases.

Best Case:

When L%4=0, L%5=0, L%6=0, L%7=0, L%8=0 and L%9=0 that means at the end of a line there is no unmapped base reamians.

Average Case:

When L%4=1, L%5=1, L%6=1, L%7=1, L%8=1and L%9=1 so at the end of a line there is one base remains.

Worst Case:

When L%4=2-3, L%5=2 - 4, L%6=2 - 5, L%7=2 - 6 , L%8=2 - 7 and L%9=2 - 8 i.e. at the end of a line there are 2 to n-1 unmapped bases remains where n=4, 5, 6, 7, 8, 9 respectively.

IV. RESULTS and DISCUSSIONS

Sequences have been taken from: http://www.ncbi.nlm.nih.gov/Genbank. Then saving percentage = (1- Compressed sequence length / Original sequence length) * 100 by different method have been calculated.

Two much well known lossless text compression technique like WinZIP and WinRAR give worst saving percentage on biological sequence compression process.

Table 1: Sequence Length (bp) before and after compression by different methods

Sequence Name	Original Seq. Length	ID, LL & BZIP2	LZSS & PPM	B2 R	B3 B2 R	B4-B2 R	B5-B2 R	B6-B2 R	B7-B2 R	B8-B2 R	B9-B2 R
ZM_BFc0038A03	765	422	396	367	246	208	149	127	107	96	85
ZM_BFc0025I23	811	417	386	380	268	203	158	136	113	102	91
ZM_BFb0203I02	787	425	391	383	262	197	154	132	110	99	88
ZM_BFb0129K09	682	385	355	331	226	171	133	114	95	86	76
MOPT	712	377	351	341	235	177	139	119	100	89	80
RPL19	689	378	347	336	230	173	135	115	96	87	77
ILP4-1	660	402	355	370	216	163	129	110	92	83	74
ILP4-2	610	379	331	282	198	149	117	101	85	76	68
DRT2	691	414	350	334	231	173	135	116	97	87	77
TP53AIP1	616	393	336	294	206	152	120	103	86	77	69

Table 2: Comparison of Saving Percentage (%) by different methods

Sequence Name	ID, LL, & BZIP 2	LZSS & PPM	B2 R	B3 B2 R	B4-B2 R	B5-B2 R	B6-B2 R	B7-B2 R	B8-B2 R	B9-B2 R
ZM_BFc0038A03	44.84	48.24	52.03	67.84	72.81	80.52	83.40	86.01	87.45	88.89
ZM_BFc0025I23	48.58	52.40	53.14	66.95	74.97	80.52	83.23	86.07	87.42	88.78
ZM_BFb0203I02	45.99	50.31	51.33	66.71	74.97	80.43	83.23	86.02	87.42	88.82
ZM_BFb0129K09	43.55	47.94	51.47	66.86	74.92	80.50	83.28	86.07	87.39	88.86
MOPT	47.05	50.70	52.11	66.99	74.14	80.48	83.28	85.96	87.50	88.76
RPL19	45.14	49.63	51.23	66. 62	74.89	80.40	83.30	86.07	87.37	88.82
ILP4-1	39.09	46.21	43.94	67.27	75.30	80.45	83.33	86.07	87.42	88.79
ILP4-2	37.87	45.73	53.77	67.54	75.57	80.81	83.44	86.08	87.54	88.85
DRT2	40.09	49.34	51.66	66.57	74.96	80.46	83.21	85.96	87.41	88.86
TP53AIP1	36.20	45.45	52.27	66.56	75.32	80.52	83.28	86.04	87.50	88.80

More the value of the saving percentage, then computer can store huge amount of data using less memory. The lengths of the sequences are in base pair (bp) or bytes instead of only bytes because when a sequence is a DNA sequence then

sequence length is bp but when a sequence is a RNA sequence then sequence length is in bytes because RNA has single strand where as DNA has dual strand.

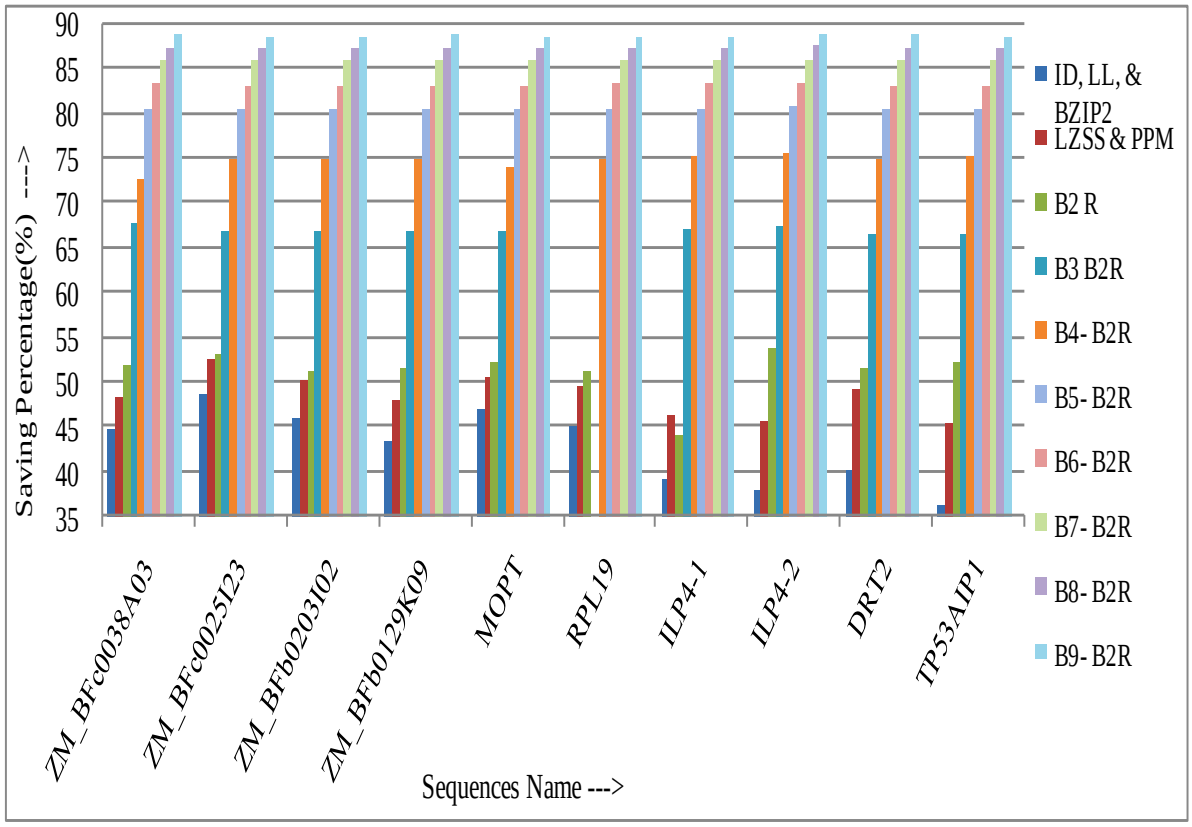

Figure 1: The graphical representation of Table 2

V. CONCLUSIONS

The above result shows the saving percentages by the specially designed different biological data compression technique are very good and consistent especially for upper values of microelectronics where as if we apply plain text compression algorithm to those sequences they are inconsistent and give very poor saving percentages. If technique is consistent then human can predict the outcome size of the input file. This will help a lot. Seeing the above result we can conclude that as the normal plain text contains 256 characters so randomness within a text is more. Whereas biological data contains mainly A, C, G and T or U etc i.e. four characters so randomness in data is less. That is why famous compression techniques WinZIP, WinRAR etc for plain text, image, video etc do not give good result for DNA/RNA data. Compression algorithm B2R and B3B2R also give poor saving percentage because they are implemented by static fixed length LUT. The output result is actually in between best and worst case saving percentage. Although the higher values of microelectronics gives very good results, but with the increasing of block size repeats become less so repeat counts are more in B4-B2R compare to the others methods and it is very less in B9-B2R.

ACKNOWLEDGMENT

The authors would like to thank the referees for their valuable suggestions. The authors also like to thank some research scholar specially Ranjan Kumar Maji of Bioinformatics Centre**,** A.J.C. Bose Centenary Building, P-1/12, CIT Scheme - VII M, Kolkata - 700 054, India. Our gratitude goes to Dr. Zumur Ghosh, Bose Institute for her valuable suggestion.

REFERENCES

[1] Hyoung Do Kim and Ju-Han Kim. : *"DNA Data Compression Based on the Whole Genome Sequence. Journal of Convergence Information Technology ",* Vol. 4, No. 3, September 2009.

[2] Gary Benson, *"Tandem repeats finder: a program to analyze DNA sequences"*, Oxford University Press, Nucleic Acids Research, Vol. 27, No. 2, pp. 573-580, 1999.

[3] Subhankar Roy, Sunirmal Khatua, Sudipta Roy and Prof. Samir K. Bandyopadhyay, *"An Efficient Biological Sequence Compression Technique Using LUT and Repeat in the Sequence"*, IOSRJCE, Vol. 6, Issue 1, pp. 42-50, Sep-Oct. 2012.

[4] Subhankar Roy, Sunirmal Khatua, *"Compression Algorithm for all Specified Bases in Nucleic Acid Sequences"*, IJCA, Vol. 35, No. 13, pp. 46-50, August. 2013

[5] Heba Afify, Muhammad Islam and Manal Abdel Wahed, *"DNA lossless differential compression algorithm based on similarity of genomic sequence database"*, International Journal of Computer Science & Information Technology (IJCSIT), Vol. 3, No. 4, pp. 145-154, August 2011.

[6] R.K. Bharti, Prof. R.K. Singh. *: "A Biological Sequence Compression based on Look up Table (LUT) using Complementary Palindrome of Fixed Size"*, ICJA, Vol. 35, No.11, pp. 56-59, December 2011.

[7] P.Raja Rajeswari and Dr. Allam AppaRao. : *"GENBIT COMPRESS TOOL (GBC): A java-based tool to compress DNA sequences and compute compression ratio (bits/base) of genomes"*, IJCSIT, Vol. 2, No. 3, pp. 181-191, June 2010.

[8] Ashish Kishor Bindal, R. Sabarinathan, J. Sridhar, D. Sherlin, and K. Sekar, *"An Algorithm to Find All Identical Motifs in Multiple Biological Sequences"*, pp. 137-148, Springer-Verlag Berlin Heidelberg, 2010.

[9] Xin Chen, Sam Kwong and Ming LiA*, "Compression Algorithm for DNA Sequences, Using Approximate Matching for Better Compression Ratio to Reveal the True Characteristics of DNA"*, pp. 61-66, IEEE Engineering in Medicine and Biology, July/August 2001.

[10] Ateet Meheta & Bankim Patel, *"DNA compression using hash based data structure"*, International Journal of Information Technology and Knowledge Management, pp. 383-386, Vol. 2, No. 2, July-December 2010.

