Compression Algorithm for all Specified bases in Nucleic Acid Sequences

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, India

ABSTRACT
Organizations such as IT industry, colleges and Scientists regularly encounter problems to handle large data sets for their different purpose in many areas as for example biological research. These limitations also affect internet search to fetch data, business for analysis etc. So it is simply needed generalized but special types of compression algorithm for dissimilar data to get utmost saving percentage. In this article Compression of biological data that is single and double strand DNA and single strand RNA have been considered. Since biological data are less random compare to any text data that means redundancy within the sequences are more but they have some special property as for example different types of repeat one of such repeat is called dinucleotide repeat .This type of repeat are more in any sequence. Here the two proposed algorithm are based on this repeat using static fixed length LUT for input file and output file mapping.

Keywords
Completely and incompletely specified nucleic acid bases, static LUT, dinucleotide repeats, base pair, sequence line length, compressed sequence length, compression factor, saving percentage.

1. INTRODUCTION
Till now in most of the nucleic acid sequences compression algorithm, only completely specified bases that are A, C, G and T/U [1-2]. That means if any sequence contains some of the incompletely bases, although it is a rare case then those techniques will not be adequate. So there is need of some different algorithm which has the generalized property. That means those techniques can handle only four completely bases [3-5] as well as eleven incompletely specified bases. The primary bases are adenine (A), cytosine (C), guanine (G) which are exist in both DNA and RNA sequence, thymine (T) only in DNA and uracil (U) in RNA respectively. The above said symbols are typically called bases in genome. Other than theses symbols there may exists some intermittent and incompletely specified bases in nucleic acid sequences these are K, M, R, S, W, Y, B, D, H, V, N respectively [6]. The incompletely specified bases can be represent in terms of A, C, G and T/U where Keto (K) may be G or T, Amino (M) A or C, Puine (R) A or G, S C or G, W A or T, Pyrimidine (Y) C or T all of these having 50% probability between two primary bases. The following bases which may have one of the three primary bases are B - C or G or T, D A or G or T, H A or C or T and V A or C or G respectively. The last one N may be A or C or G or T. The above said bases relevant for both deoxyribonucleic acids (DNA) and ribonucleic acids (RNA) respectively. The only difference is in T and U in case RNA it is U where T for DNA. So the deduction of RNA sequences from the subsequent DNA sequences is just a replacement of all occurrence of U by T in the corresponding incompletely specified bases. Since the difference between DNA and RNA is just by single bases so storing process are quite equivalent both for DNA and RNA. So programmer need not to make different data bank for DNA and RNA. Here no discrimination between lower and upper case letters have been considered also i.e. 'A' is equivalent to 'a' and so on. All symbols have their corresponding complement but only for DNA not for RNA they are A-T, C-G, K-M, M-K, R-Y, Y-R, B-V, D-H, H-D, and V-B. Some symbols are self complement they are S, W, and N.

2. BRIEF REVIEW
There have a lot of pre-existing DNA sequence compression. Here only few of them have been explained.

Some authors use the property approximate repeat [7]. Approximate repeats are subsequences of a sequence which can be transformed into a copy of the original previous subsequences using edit operations such as substitution, insertion and deletion. But the searching process is time consuming, to save time greedy approach misses long repeats which prohibits from receiving high compression.

The tandem repeat finder [8] which is a program to analyze DNA sequences. It is used to find the two or more contiguous exact or approximate pattern in a sequence. It helps to find which portions of a genomic sequence are similar and which are not.

Another compression, which divides the entirely scanned DNA sequence into factors of length four, is Hashbased (Ateet Mehta et al, 2010) [9] and as its name itself suggests, the algorithm initially builds a hash table and assigns a unique character to each of the factors which act as the hash key. Each factor of length four is assigned corresponding unique characters to each of the factors. But this algorithm doesn't consider any junk characters in the sequence.

In this article completely specified as well as all incompletely specified bases have been considered. So here the proposed compression techniques are more flexible to switch a wide variety of bases.

Rest of the article organized as follows. Section 3 two proposed generalized and specially designed algorithms which

have four sub-sections explain four important basics of the two algorithms, sections 4 results after applying this algorithm on a set of twelve biological data, section 5 conclusions and discussions, section 6 future work and finally section 7 references.

3. PROPOSED ALGORITHMS

This article contains two proposed algorithms B2 (Block Two) and B2DNR (Block Two of Dinucleotide Repeats) respectively as explained below. These two algorithms are based on four possible fundamentals. These are explained below respectively.

3.1 Static fixed length Look up table

Both algorithms are based on mapping implemented by static fixed length LUT.

Here all the input sequences are of frequent and rarely specified bases in nucleic acid sequences {A, C, G, T or U} and {K, M, R, S, W, Y, B, D, H, V, N} respectively. There are total 15^2=225 combinations are formed using the above 15 characters as a combination of two. Using input-output mapping process those 225 combinations are mapped into 225 ASCII character out of possible 256 ASCII character. The resultant mapping table is shown in table 1. Reverse mapping is done at decompression process.

3.2 ASCII characters

Here some chosen 8 bit ASCII characters are used.

In both algorithms, as the total number of ASCII character required is 225 so among 256 available ASCII character 225 can be selected easily. So the number of remaining characters are (256-225) = 31. Out of which 15 possible input characters not be used. Now the remaining characters are (31-15) = 16. Two characters gives new line and five characters do not give correct mapping, so the remaining (16-2-5) = 9 characters are used for repeat count.

Table 1: Static Look up table for all specified bases

Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character	Base	Character
AA	(char)0	TA	(char)26	CA	(char)41	GA	(char)56	RA	(char)80	YA	(char)101	WA	(char)	SA	(char)132	MA	(char)150	KA	(char)166	HA	(char)181	BA	(char)196	VA	(char)211	DA	(char)226	NA	(char)241
AT	(char)11	TT	(char)27	CT	(char)42	GT	(char)57	RT	(char)81	YT	(char)102	WT	(char)117	ST	(char)133	MT	(char)151	KT	(char)167	HT	(char)182	BT	(char)197	VT	(char)212	DT	(char)227	NT	(char)242
AC	(char)12	TC	(char)28	CC	(char)43	GC	(char)58	RC	(char)85	YC	(char)103	WC	(char)118	SC	(char)134	MC	(char)152	KC	(char)168	HC	(char)183	BC	(char)198	VC	(char)213	DC	(char)228	NC	(char)243
AG	(char)14	TG	(char)29	CG	(char)44	GG	(char)59	RG	(char)88	YG	(char)104	WG	(char)119	SG	(char)135	MG	(char)153	KG	(char)169	HG	(char)184	BG	(char)119	VG	(char)214	DG	(char)229	NG	(char)244
AR	(char)15	TR	(char)30	CR	(char)45	GR	(char)60	RR	(char)90	YR	(char)105	WR	(char)120	SR	(char)136	MR	(char)154	KR	(char)170	HR	(char)185	BR	(char)200	VR	(char)215	DR	(char)230	NR	(char)245
AY	(char)16	TY	(char)31	CY	(char)46	GY	(char)61	RY	(char)91	YY	(char)106	WY	(char)121	SY	(char)137	MY	(char)155	KY	(char)171	HY	(char)186	BY	(char)201	VY	(char)216	DY	(char)231	NY	(char)246
AW	(char)17	TW	(char)32	CW	(char)47	GW	(char)62	RW	(char)92	YW	(char) 107	WW	(char)122	SW	(char)138	MW	(char)156	KW	(char)172	HW	(char)187	BW	(char)202	VW	(char)217	DW	(char)232	NW	(char)247
AS	(char)18	TS	(char)33	CS	(char)48	GS	(char)63	RS	(char)93	YS	(char)108	WS	(char)123	SS	(char)139	MS	(char)158	KS	(char)173	HS	(char)188	BS	(char)203	VS	(char)218	DS	(char)233	NS	(char)248
AM	(char)19	TM	(char)34	CM	(char)49	GM	(char)64	RM	(char)94	YM	(char)	WM	(char)124	SM	(char)140	MM	(char)159	KM	(char)174	HM	(char)189	BM	(char)204	VM	(char)219	DM	(char)234	NM	(char)249
AK	(char)20	TK	(char)35	CK	(char)50	GK	(char)69	RK	(char)95	YK	(char) 110	WK	(char)125	SK	(char)142	MK	(char)160	KK	(char)175	HK	(char)190	BK	(char)205	VK	(char)220	DK	(char)235	NK	(char)250

AH	(char)21	TH	(char)36	CH	(char)51	GH	(char)70	RH	(char)96	YH	(char)111	WH	(char)126	SH	(char)145	MH	(char)161	KH	(char)176	HH	(char)191	BH	(char)206	VH	(char)221	DH	(char)236	NH	(char)251
AB	(char)22	TB	(char)37	CB	(char)52	GB	(char)73	RB	(char)97	YB	(char)112	WB	(char)127	SB	(char)146	MB	(char)162	KB	(char)177	HB	(char)192	BB	(char)207	VB	(char)222	DB	(char)237	NB	(char)252
AV	(char)23	TV	(char)38	CV	(char)53	GV	(char)74	RV	(char)98	YV	(char)113	WV	(char)128	SV	(char)147	MV	(char)163	KV	(char)178	HV	(char)193	BV	(char)208	VV	(char)223	DV	(char)238	NV	(char)253
AD	(char)24	TD	(char)39	CD	(char)54	GD	(char)76	RD	(char)99	YD	(char)114	WD	(char)113	SD	(char)148	MD	(char)164	KD	(char)179	HD	(char)194	BD	(char)209	VD	(char)224	DD	(char)239	ND	(char)254
AN	(char)25	TN	(char)40	CN	(char)55	GN	(char)79	RN	(char)100	YN	(char)115	WN	(char)131	SN	(char)149	MN	(char)165	KN	(char)180	HN	(char)195	BN	(char)210	VN	(char)225	DN	(char)240	NN	(char)255

3.3 Line length

Line length in the sequence is the number of bytes in a line.

If the length of a line in the sequences is even then there will be no extra character, if it is odd then there will be extra character. In both algorithms if there are any extra character exist then we keep it as it is.

E.g. let us consider line length (L) = 70bp where bp=base pair, a unit of length in nucleic acid chains.

Then L%2 = 0 so there are L/2 = 35 block each of size two and there are no extra base at the end of line.

If L = 71bp, then L%2 = 1 so there are L/2 = 35 block each of size two and there are one extra base at the end of line. Even number line length sequences give better result compare to odd length.

3.4 Dinucleotide repeats

One of the properties in the sequences is dinucleotide repeats.

In B2DNR this feature has been used by counting the number of dinucleotide repeats occur in the sequence. But at a time not more than nine repeats have been counted. If there are more than nine repeats then recount the repeats.

E.g. ACACACACACACACACACACACAC

Here total twelve blocks are same. So from index eight recounts started.

3.5 B2 and B2DNR compression algorithm

The compression and decompression are just reverse process.

3.5.1 B2 Compression algorithm

Input: Text file contains completely and incompletely specified bases in the biological sequences respectively.

Output: Compressed file of ASCII characters.

Function Compression (input.txt)

Begin

Step1: A List which adds 15 possible bases in the sequence.

Step2: A HashMap map string of length two (all possible) to chosen ASCII characters.

Step3: An input file of bases is taken to read data.

Step4: An output file is opened to write the compressed data.

Step5: Read the input file line wise.

Step6: Calculate the line length.

Step7: While str <> null do

/* str is String type variable. It store a line at time */

While str.length()>2 do

/* used to take block size of two characters */

subStr = str.substring(0,2)

/* subStr is a string type variable to store block of two bases */

Add these bases block to cbufList.

/* cbufList is an ArrayList of String type*/

End While

For cbuf : cbufList do

/* cbuf is a String variable*/

Write the mapped value to output file

End For

End While

End

3.5.2 B2DNR Compression algorithm

Input: Original file of completely and incompletely specified bases.

Output: Compressed file.

Function Compression (input.txt)

Begin

Step1: A List which adds 15 possible bases in the sequence.
Step2: A HashMap map string of length two (all possible) to chosen ASCII characters.

Step3: An input file of bases is taken to read data.

Step4: An output file is opened to write the compressed data.

Step5: Read the input file line wise.

Step6: While str <> null do

/* str is String type variable. It store a line at time */

While str.length()>2 do

/* used to take block size of two characters */

subStr = str.substring(0,2)

/* subStr is a string type variable to store block of two bases */

Add these bases block to cbufList.

/* cbufList is an ArrayList of String type*/

End While

For cbuf : cbufList do

/* cbuf is a String variable*/

If cbuf.length() = 2 do

If cbuf.equals(temp) do

/* temp is used to previous block */

count++;

If count=9

Write the count to output file

count=0

End If

/* count is used to count number of dinucleotide repeats*/

Else

If count>0 do

Write the count to output file

count=0

Write the mapped value

Else

Write the mapped value

Enf If

End If

temp=cbuf;

End If

End For

End While

End

4. RESULTS

Here the above said techniques are applied on twelve sequences [10] of size in bp (base pair).

These are six Homo sapiens, sequences: IL4, transcript variant 1 (IL4, 1); IL4, transcript variant 2 (IL4, 2); MT1F; Rattus norvegicus; SERF2 and TP53AIP1, transcript variant 3 (TP53AIP1) respectively. Three Zea mays, sequences: HDZIV13_OCL13, ZM_BFc0038A03 and ZM_BFb0129K09 respectively. Three Mus musculus sequences: 496.1h6-3, 496.1H6-5S and IST15114H3 respectively. We have calculated compression factor and saving percentages by different techniques.

Table 2: Comparison of Compress Sequence Length by different techniques

Sequence Name	Original Seq. Len.	ID, LL & BZIP2	LZSS & PPM	B2	B2DNR
IL4, 1	660	402	358	321	307
IL4, 2	610	379	333	297	282
MT1F	468	363	281	228	217
Rattus norvegicus	442	332	256	215	207
SERF2	536	355	295	261	252
TP53AIP1	616	393	336	300	294
HDZIV13_OCL13	391	280	231	191	184
ZM_BFc0038A03	765	422	396	373	363
ZM_BFb0129K09	682	385	355	332	321
496.1h6-3	687	414	350	333	320
496.1H6-5S	384	322	250	186	176
IST15114H3	506	328	293	245	238

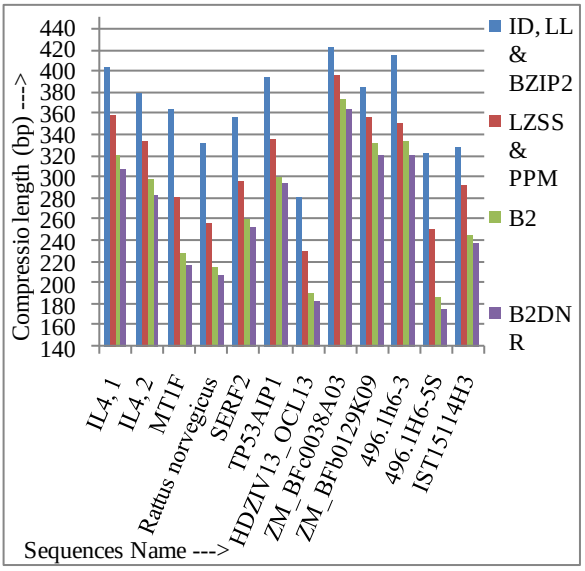

Fig 1: The graphical representation of table 2.

Table 3: Comparison of Compression Factor by different techniques

Sequence Name	ID, LL & BZIP2	LZSS & PPM	B2	B2 DNR
IL4, 1	1.642	1.859	2.056	2.150
IL4, 2	1.609	1.843	2.054	2.163
MT1F	1.289	1.665	2.053	2.157
Rattus norvegicus	1.331	1.727	2.056	2.135
SERF2	1.510	1.817	2.054	2.127
TP53AIP1	1.567	1.833	2.053	2.096
HDZIV13_OCL13	1.397	1.693	2.047	2.125
ZM_BFc0038A03	1.813	1.932	2.051	2.108
ZM_BFb0129K09	1.772	1.921	2.054	2.125
496.1h6-3	1.660	1.963	2.063	2.147
496.1H6-5S	1.193	1.536	2.065	2.182
IST15114H3	1.543	1.727	2.065	2.126

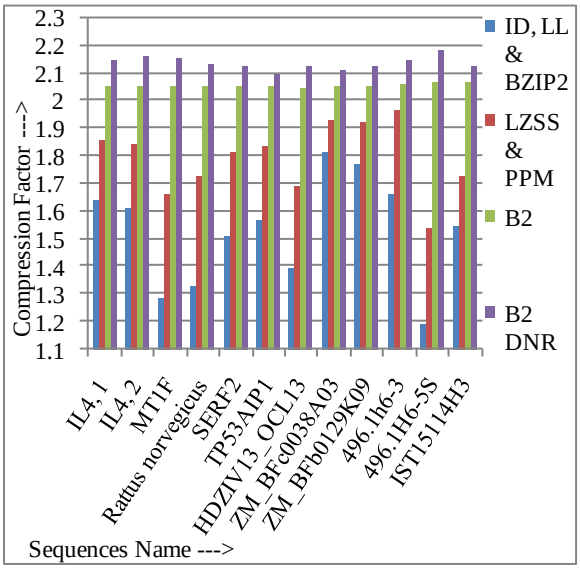

Fig 2: The graphical representation of table 3.

Table 4: Comparison of Saving Percentage (%) by different techniques

Sequence Name	ID, LL & BZIP2	LZSS & PPM	B2	B2 DNR
IL4, 1	39.091	45.758	51.364	53.485
IL4, 2	37.869	45.409	51.312	53.771
MT1F	22.436	39.957	51.282	53.634
Rattus norvegicus	24.887	42.081	51.358	53.168
SERF2	33.769	44.963	51.306	52.985
TP53AIP1	36.201	45.454	51.299	52.273
HDZIV13_OCL13	28.389	40.921	51.151	52.941
ZM_BFc0038A03	44.837	48.235	51.242	52.549
ZM_BFb0129K09	43.548	47.947	51.320	52.933
496.1h6-3	39.738	49.054	51.539	53.421
496.1H6-5S	16.146	34.896	51.563	54.167
IST15114H3	35.178	42.095	51.581	52.965

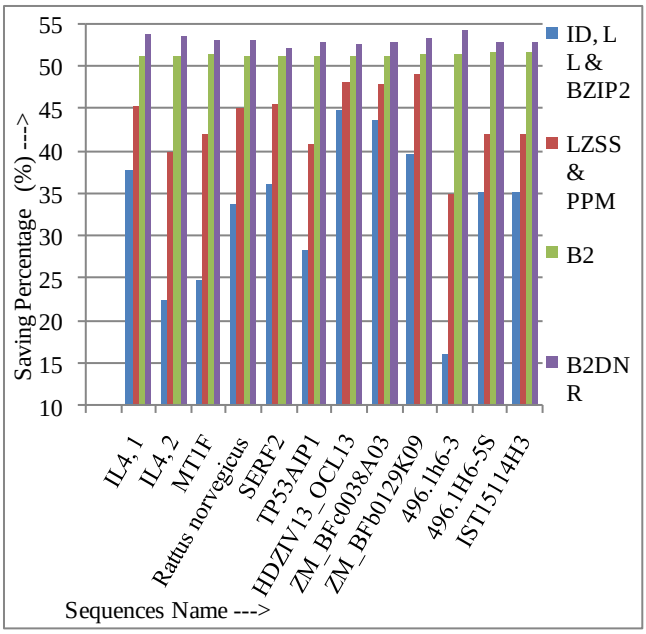

Fig 3: The graphical representation of table4.

5. CONCLUSIONS AND DISCUSSIONS

There are different compression techniques available usually like WinZIP, WinRAR, Gzip, Bzip2 etc but they are not appropriate for the compression of nucleic acid sequences due to some special property of those sequences. Therefore need of some specialized compression algorithms for those sequences. Although there have been different algorithm developed using the property of tandem repeat, approximate repeat, Interspersed repeats and complementary nature of DNA sequences. But those algorithms can handle only completely specified bases.

The proposed algorithms can hold both completely and all incompletely specified bases as well as use the best possible redundant property i.e. dinucleotide repeat in the sequence. Also the memory requirement, compression factor, saving percentage and compression time are good compare to the existing algorithms as shown by the above experiential results and graph. More the compression factor and saving percentage means sequence length is very less after compression

6. FUTURE WORK

Develop a compression algorithm to compresses any sequence level wise. That means input sequences will pass through more than one compression techniques to produce final compress data.

7. REFERENCES

[1] 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.

[2] R.K. Bharti and Prof. R.K. Singh, "A Biological Sequence Compression based on Look up Table (LUT) using Complementary Palindrome of Fixed Size", ICJA (0975–8887), Volume 35– No.11, December 2011.

[3] Heba Afify, Muhammad Islam and Manal Abdel Wahed, "DNA lossless differential compression algorithm based on similarity of genomic sequence database", IJCSIT, Vol. 3, No 4, August 2011.

[4] R. K. Bharti and Prof. R.K. Singh, "A Biological sequence compression Based on Approximate repeat Using Variable length LUT", International Journal of Advances in Science and Technology, Vol. 3, No.3, PP: 71-75, 2011.

[5] Suman Chakraborty, Sudipta Roy**,** Prof. Samir K. Bandyopadhyay, "Image Steganography Using DNA Sequence and Sudoku Solution Matrix", International Journal of Advanced Research in Computer Science and Software Engineering(IJARCSSE), Volume 2, Issue 2, February 2012.

[6] Department of Chemistry, Queen Mary University of London, "Nomenclature for Incompletely Specified Bases in Nucleic Acid Sequences".

[7] 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.

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

[9] 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.

[10] Sequences are taken from: httpncbi.nl://www.m.nih.gov/Genbank.