REVISITING DNA COMPRESSION ALGORITHMS

Rexline S J
Associate Professor/Department of Computer Science, Loyola College, Chennai, India
Jasmine Jintha A
Associate Professor/Department of Computer Science, Loyola College, Chennai, India
Trujilla Lobo F
Creek front Road, Jacksonville, Florida-32256, USA.

Abstract: Genome contains the genetic information of biological organisms. Large amount of spaces are required to store DNA sequences in DNA databases like GenBank sequence database. Thus, reduce the Data storage for DNA sequences has become an essential. Standard general purpose compression algorithms are not competent enough to compress DNA sequences. The compressors such as "gzip", "bzip2", "winzip" expanded the DNA genome file instead of compressing them. Therefore special algorithms are specifically designed for DNA sequences compression. In this paper, the specially designed compressors for DNA sequences are analysed and their compression ratio are compared and concluded.

Keywords: DNA compress, DNA sequences, DNAPack, Genome Compress.

Introduction: Modern biological field generates huge amounts of genomic sequence. The GenBank database is of all publicly available nucleotide sequences. GenBank is the most significant database for researchers in the Modern biological field around the world [14]. Since the DNA sequence needs huge amount of space it is necessary to compress and store the sequences. So that, there is a need of competent algorithms in the research area of DNA sequence compression [25].The biological activity of every living organism is controlled by the deoxyribonucleic acid (DNA) that contains a complete set of information needed to direct the functioning of each and every one of the cells. The chemical composition of the DNA is the same for all living organisms. The use of DNA in genetic engineering, forensics and anthropology applications has been extensive. The DNA of all organisms has four components in common. They are the four nucleotide bases; namely Adenine, Cytosine, Guanine, and Thymine [4, 5].They are represented using the first character of their names; namely A, C, G and T respectively [9, 10]. There is another unknown nucleotide base element symbolized by the letter N. Therefore the DNA sequence is symbolized as a set of {A, C, G, T, and N}[4, 5]. The element N is still remains unknown and is yet to have pictorial representations but participates in the functionalities of DNA. The DNA sequences are characterized as they contain many tandem repeats, many of the strings are palindromes and some of them are reverse palindromes. Since DNA sequences consists of only for nucleotides bases {A, C, G, T}, two bits are enough to store each base [13, 24]. So that it is not an easy task to compression of DNA sequences. Standard general purpose compression algorithms not succeeded to get a good compression ratio. Combination of statistical and substitution methods are tried out by the researches to compress DNA files [11]. Pattern recognition based algorithms are also used in the field of DNA sequences compression [12]. Here all the existing DNA compression algorithms developed by the DNA sequence researchers are evaluated and compared for their suitability for compressing large collections of DNA sequences.

Existing Compression Methods: The standard general purpose compression algorithm such as "gzip", "bzip2", "winzip" enlarged the DNA sequences. Most of the existing algorithms worked well for English text only but not for DNA sequences. Downloading and maintaining the DNA sequences is the very much cost consuming factor due to the increasing amount of DNA sequences. So that compression algorithms are specifically designed for DNA sequences such as Ziv-Lempel compression algorithms Biocompress, Gencompress and DNAPACK compress. A straightforward way to store and manage DNA data in computers is to use ASCII-encoded characters, resulting in one byte for each base [23]. Since this

representation is inefficient in terms of space, many proposals for DNA compression algorithms are thrown by the researchers.

Biocompress2: Grumbach and Tahi proposed compression algorithms called *Biocompress*s and *Biocompress*2 based on Ziv and Lempel data compression method. *Biocompress*2 uses a window of size of the sequence to search for exact repeats or reverse complements and encodes them with length and position of it. Literal encoding and second order arithmetic encoding is also used. The adaptive arithmetic coding of order 2 has the best compression ratio on the DNA sequences; in literal encoding each symbol is encoded as a two bit number. Biocompress2 compares these three methods and chooses the most efficient one dynamically. At each step of the compression, the longest factor beginning at the current position, and matching a factor which starts in the part of the chain already encoded, is chosen. This factor may be either an identical factor or a palindrome. It is then encoded by a *copy* codeword consisting of a pair of integers *(l,p),* where I is the length of the factor and *p* the position of its first occurrence.

Here is a high-level description of the Biocompress2 algorithm [7,8]:
1. begin
2. i= 1;
3. *bool = F;*
4. while $(i < n)$ do begin
5. find the longest factor *f*
6. starting at a position smaller than *i*
7. matching the factor at the current position *i*
8. let l_1, be the length and p_1, be the position of *f*
9. find the longest palindrome *f'*
10. starting at a position smaller than *i*
11. matching the factor at the current position *i*
12. let f_2 be the length and p_2 be the position of *f'*
13. if $|(l_1,p_1)|+ c > 2l_1$, then
if $|(l2,p2)|+ c > 2l_2$, then
output the character at position *i* in *buffer;*
bool = T;
$i=i+1$;
14. elseif *bool = T* then output arithmetic or literal encoding of *buffer;*
output the codeword (l_2,p_2);
empty *buffer:*
bool = F;
$i=i+l_2$;
15. *else*if *bool = T* then output arithmetic or literal encoding of *buffer;*
empty *buffer;*
bool = F;
16. *if* $|(l_1 > l_2)|\&|l_2|or|(l_2,p_2)|) + c > 2l_2$ then
output the codeword (l_1,p_1)
$i = i + l_1$;
17. else
output the codeword (l_2,p_2);
$i = i + l_2$;
18. end
19. end

CFACT: CFACT was proposed by E.Rivals et al. [21] which is a two-pass algorithm. First pass builds the Suffix tree to locate the repeated longest segment in a sequence and second pass encodes the repetitions based on their guaranteed gain; In addition two-bit per base encoding will be used.

Gen Compress: Gen Compress proposed by Chen et al [2] the performance of which is based on reference sequence selection as approximate matching with edit operations uses this reference sequence for compression. It uses both approximate repeats and reverse complements, and encodes it with length, position and the errors. If approximate repeats and approximate reverse generate errors in the encoding process, then the second order arithmetic encoding is used. Both GenCompress uses the greedy approach for selection of the repeat segments. Three standard edit operations are used in this approximate matching algorithm [2]. These are:

1. ***Replace****:* The *Replace* operation is used as (*R; p; char*) which indicates that the character at position *p* by character *char*.
2. ***Insert:*** This operation is used as (*I; p; char*), which indicates that the character *char* which inserts at the position *p*.
3. ***Delete****:* This operation is used as (*D; p:char*), which indicates that the character *char* which deletes from the position *p*.

Here is a high-level description of the GenCompress algorithm:
Input: A DNA sequence *w*
Output: compressed DNA sequence *v*
Initialization: u = *w* and *v* = €

1. while u ≠ € do
2. Find an optimal prefix *p* of *u*
3. If an optimal prefix *p* with *q* number of repeat in *v* is found then Encode the repeat *(i, |q|),* where i is the position of *q* in *v*, together with the edit operations λ*(q, p*). The edit operations λ(q, p) that transforms q into p. Output the code.
4. else fix the optimal prefix *p* is the first character of *u*,Encode and output it.
5. eliminate the prefix *p* from *u* and affix it to *v*
6. *end*

CTW+LZ: CTW+LZ is a non-greedy program which searches exact and approximate reverse complements and exact and approximate repeats. This program encodes these structures using LZ77-like function and edit operations are encoded by arithmetic coding. Symbols which are not encoded in a repeat are encoded by order-32 Context-Tree Weighting (CTW) algorithm. Context Tree Weighting (CTW) method has on average a good compression ratio for an unknown model.

Genome Compress: Genome Compress [6] compresses both repetitive and non repetitive sequences. The algorithm divides the given input string into partitions of length four and assigns a five bit binary sequence for four DNA bases and for eight repeated sequence of each bases also. Four used for encoding repetitive sequence and remaining for encoding partition's bases. It is simple and uses less execution time and memory.
The following is the pseudo code incorporating in Genome Compress compressor.

Procedure Encode:

1. *Begin:* Divide the given input string into partitions of length 4 if two consecutive partitions only contain eight A, T, G or C, then encode consecutive sequence with 5 bit binary numbers. Transfer the five bit binary number to the output string.
2. *Else:* For each partition
3. *Begin:* Find out the corresponding 5-bit binary number.
   Replace the partition by five bit binary number.
   Transfer the five bit binary number to the output string.
   End
4. 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.
5. *End:*

Procedure Decode:

1. *Begin:* Divide the input binary string into partitions of length 5
2. For each binary partition of length 5

3. *Begin:* Convert each partition into corresponding A, T, G, C sequence. Transfer the sequence to the source.
4. *End*
5. *End*

DNA Compress: DNA Compress is also a two pass algorithm designed by Chen et. al [3] and uses special software tool Pattern Hunter for finding the repeats. Pattern Hunter finds complementary palindromes and approximate repeats with highest score in the first pass and encodes them in the second pass. It checks each repeats to see whether it saves bits while encoding, if not it will be discarded. At the end all the non-repeats are concatenated together and encoded by sending as input to a two-order arithmetic coder. DNA Compress uses almost the same encoding as Gen Compress.
Here is a high-level description of the DNA Compress Algorithm:

1. Use the Pattern Hunter tool to generate all approximate repeats and approximate reverse complements into a list A in the descending order.
2. The selection of which repeats are more optimal for sequence compression can be rescheduled at the end of Pattern Hunter homology search.
3. Extract a highest score repeat from list A and add it into another repeat list B;
4. Process each repeat in A so that there's no overlap with the extracted repeat ;
5. Go to steps 2 if the highest score of repeats in A is still higher than a pre-defined threshold; otherwise exit.

DNA Pack: Behshad Behzadi et al. [1] used Dynamic Programming techniques to identify the repeating sequences called *DNA Pack*.
The following is the pseudo code incorporating in *DNAPack* compressor.
Let s be the input DNA sequence. Let Best Comp[i] be the smallest compressed size of the prefix s[1..i]. The following simple recurrence is the general scheme of DNAPack dynamic programming.
Initialization:
Best Comp[0] = 0
Recurrence: $\forall_i > 0$

$$\text{Best Comp}[i] = \min \begin{cases} \text{Best Comp}[j] + \text{Copy Cost}(j, i, k) \\ \quad \forall k\ \forall o > j < i \\ \text{Best Comp}[j] + \text{Palin Cost}(j, i, k) \\ \quad \forall k\ \forall o > j < i \\ \text{Best Copy}[j] + \text{Min Cost}(j+1, i) \\ \forall o > j < i \end{cases}$$

Copy Cost(j, i, k) is the number of bits needed to encode the substring of size k starting at position i if it is an approximate repeat of the substring of size k starting at j. The Palin Cost is similarly defined for reverse complementary substrings. Min Cost(j + 1, i) is the number of bits required for the compression of the segment s[j + 1, i]. It depends on the substring size as well as the compression ratio attained for the algorithm by arithmetic coding or CTW. Min Cost allows creating a repeat segment only if it yields a benefit in the compression ratio. These three functions are estimations of the real cost, since the efficiency of some optimizations done during the encoding cannot be computed at this point.

PRDNAC: Panneer Arokiaraj S et.al used the Pattern Recognition method [15, 16] to identify the repeating sequences in the DNA file and designed the compressor called PRDNAC. The idea behind the PRDNAC algorithm is that this compressor finds the longest repeating patterns and their reverse are identified and the symbol tables are generated with a unique sequence code for the identified repeating patterns of varying Sizes. Based on the patterns identified the bits required to represent the patterns are determined and encoded. To indicate the reverse or regular pattern additional bit is used apart from the required bits.
The following is the pseudo code incorporating in PRDNAC compressor [17, 18].

Procedure Encode:

1. Read the DNA Sequence repeatedly and generate the symbol tables with sequence codes [000-110].

2. Calculate the number of bits required to represent the patterns identified.
3. Using the symbol tables generated in step 1, build the compressed sequenced file using the combination of sequence codes [000-110] and the bits required for representing the patterns identified. To represent the uncompressed sequences, sequence code [111] is used with the two bit code [00, 01, 10, and 11].
4. Repeat the step3 till the end of the file is reached.
5. Attach the index required to retrieve the DNA sequences with the compressed file.

Procedure Decode

1. Extract the index required to retrieve the DNA sequences from the compressed file to obtain the size of the file header.
2. Extract the size of the blocks to be read, bit patterns and the sequence code to recollect the patterns from the compressed file.
3. Read the blocks of compressed patterns and encode it from the beginning to the end of the DNA file.

DNA Bit: Raja Rajeswari et al. [19] developed *DNA Bit Compress,* an algorithm that compresses repetitive and non-repetitive sequences using the ideas of extended binary tree. Raja Rajeswari et al. [20] also proposed GENBIT Compress. 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 an eight bit binary number "00000000". 9^{th} bit "1" is used to indicate the consecutive fragments are same. Otherwise the bit "0" is used as 9^{th} bit. DNA sequence of length n is taken and divided into n/4 number of fragments. The remaining individual bases is assigned 4 unique"2"bits. (A="00",g="01",c="10",t="11").
The following is the pseudo code incorporating in DNA Bit compressor.

Procedure Encode:

Begin

1. Divide the given DNA sequence in to fragments, where each fragment consists of 2 characters, 4 characters.
2. Generate all possible combinations of DNA sequence (A, C, G, T).
3. Apply Even Bit Technique if the simultaneous bases do not match with each other. (The DNA sequence is assigned two bits for every individual base of non-repeat regions.)
4. If there exists two or three similar bases next to one another, the 3 Bit technique is applied.
5. If there exists more than 3 repeats upto 8 repeats (4,5,6,7,8) Three to eight similar bases next to one another, the 5 Bit technique is applied. The encoded string is represented as 5 Bit CODE.
6. In the given string if there is 2 characters repeat more than 1 time upto 8 times, it is represented as 7 bit code. In this 7 bit code, first 3 bits represent the number of repeats of that character. (The other 4 bits represent the code for that character.)
7. If the consecutive 4 bases are same, then the encoded string is taken to be 9 bit CODE. The first significant bit either represents as "0"or "1". "0" indicates that the repeat is exact repeat. "1" indicates that the repeat is reverse repeat.
8. Transfer the binary bits to the output String (OUTSTRING).
9. *End*

Procedure Decode:

Begin

1. Generate all possible combinations for {A, C, G, T}.
2. Allocate unique binary bit number (0 and 1) to each combination.
3. Divide given binary code in to segments.
4. According to the Binary code (either 3 BIT CODE, 5 BIT CODE, 7 BIT CODE, or 9 BIT CODE) assign appropriate base {a, c, g, t}.
5. Repeat step 4, until the end of the input sequence is reached.
6. If there are any individual bases (non repeat regions) the corresponding binary code gets transformed. (Assigned values for bases are: a="00", g="01", c="10", t="11").

End

Table 1.Comparison Results of General Compressors

Sequence Name	File Size in Bytes	gzip	bzip	WinRAR	Gzip--4	Bzip-4
chmpxx	121024	2.28	2.12	2.25	1.86	1.97
chntxx	155844	2.33	2.18	2.24	1.95	2.01
humhbb	73308	2.25	2.15	2.22	1.90	2.00
humdystrop	38770	2.36	1.18	2.37	1.95	2.07
mpomtcg	186609	2.33	2.17	2.30	1.97	2.01
humhdabcd	58864	2.24	2.07	2.19	1.91	1.99
humhprtb	56737	2.27	2.09	2.23	1.92	2.00
hehcmvcg	229354	2.33	2.17	2.32	1.98	2.01
vaccg	191737	2.25	2.09	2.23	1.87	1.95
Average		2.29	2.02	2.26	1.92	2.00

Sequence Name	Bio-Compress	Genome Compress	CTW +LZ	DNA Compress	DNA PACK	DNABIT	PRDNAC
chmpxx	1.68	1.67	1.67	1.67	1.66	1.52	1.47
chntxx	1.62	1.61	1.61	1.61	1.61	1.58	1.49
humhbb	1.88	1.82	1.81	1.79	1.78	1.61	1.60
humdystrop	1.93	1.92	1.92	1.91	1.91	1.57	1.69
mpomtcg	1.94	1.91	1.90	1.89	1.89	1.57	1.65
humhdabcd	1.88	1.82	1.82	1.80	1.74	1.61	1.59
humhprtb	1.91	1.85	1.84	1.82	1.79	1.57	1.62
hehcmvcg	1.85	1.85	1.84	1.85	1.83	1.57	1.63
vaccg	1.76	1.76	1.76	1.76	1.77	1.65	1.61
Average	1.83	1.80	1.80	1.79	1.78	1.58	1.60

Table 2.Comparison Results of DNA Compression Algorithms

Experimental Results: In this section we focus our attention to comparing the performance of the existing DNA compression algorithms. In the case of lossless compression algorithm, there are several criteria such as encoding and decoding speed, compression ratio and memory requirements are taken under consideration to evaluate the excellence of good compression algorithm. The compression ratios are expressed in terms of average BPB (bits per base) in DNA compression scheme.
The set of DNA sequences used for the purpose of analysis are: Two chloroplast genomes (CHMPXX and CHNTXX) Five human genes (HUMDYSTROP, HUMGHCSA, HUMHDABCD, HUMHBB and HUMHPRTB), two mitochondria genomes (MPOMTCG and MTPACG) and two virus genomes (HEHCMVCG and VACCG).

The above sets of DNA sequences are used by DNA researchers to compare the compression algorithms of DNA sequence compression. The DNA sequences are made available in FASTA file format in DNA databases which can also retrieved by any text processor such as Notepad [22]. The structure of a DNA sequence file is that a single word having no white spaces, soft return or an end of line marker, with a constraint that a nucleotide may appear only nine consecutive times.
Table 1 shows the Comparison results of general compressors like gzip, bzip, *WinRAR,*gzip-4, bzip-4.Table 2 shows the comparison results of DNA compressors like BioCompress2, Genome Compress, CTW+LZ, DNA Compress, DNAPACK, PRDNAC, DNABIT Compress. Two bits is enough to store each base, in spite of this fact, the standard compression algorithm like "GZIP", "BZIP2", "WinZip" uses more than 2 bits per base [24]. The drawback of the general purpose compression algorithm is that instead of reducing the space required to store the DNA sequences, it is increasing the bits required to store the DNA files.

DNA Compress, Gen Compress and CTWLZ obtain the approximately same results among the existing algorithms. *DNABIT Compress* and *PRDNAC* obtain the best results among the existing algorithms. CTW+LZ algorithm is not practically used for compressing large sequences because of the very slow execution time.

C fact algorithm operates in O (n2) time and space. C fact is slower to compress than Bio compress, since two passes over the input is required. Gen Compress produces good compression results; it is not suitable for compressing large sequences or collections. DNA Compress takes less execution time than Gen Compress. The biggest improvement made by the DNA Compress algorithm was in terms of compression speed. DNA Compress was able to compress even the largest sequence in under a minute.

Conclusion: In this paper, recent progresses relating to DNA compression are reviewed. Furthermore, the important differences between DNA compression algorithms are highlighted. These novel approaches for DNA compression algorithms discussed here are only slight variations of each other, which further helps to stimulate the extremely heterogeneous background of different approaches to get better performance in the area of DNA compression. The comparisons presented in this paper are based on the original papers. As the high quality of sequence data is growing faster than ever, the research on novel DNA compression algorithms are highly encouraged to continue to get good compression ratio. However, the researchers are believed that not only higher compression ratio and memory requirement but also faster compression and decompression should also be addressed. This is an inconclusive research which leaves lots of space for the researchers in lossless DNA sequence compression regions.

References:

1. Behzadi, B. and Le Fessant, F., "DNA Compression Challenge Revisited", Symposium on Combinatorial Pattern Matching (CPM2005, June 2005), pp.190-200.
2. Chen, X., Kwong, S. and Li, M., "A compression algorithm for DNA sequences and its applications in genome comparison", the 10th workshop on Genome Informatics (GIW-99, Tokyo, Japan, 1999), pp.51–61.
3. Chen, X., Li, M., Ma, B. and Tromp, J., "DNACompress: Fast and effective DNA sequence compression", Bioinformatics, Vol. 18(12, 2002), pp. 1696–1698.
4. Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, "Cross chromosomal similarity for DNA sequence compression", Bioinformation 2(9), 2008 pp. 412-416.
5. Choi-Ping Paula Wu, Ngai-Fong Law and Wan-Chi Siu, "Analysis of cross sequence similarities for DNA multiple sequence compression", International journal of Computer Aided Engineering and Technology, 2009.
6. U.Ghoshdastider, et al,"GenomeCompress: A Novel Algorithm for DNA Compression", ISSN 0973-6824, 2005.
7. Grumbach, S. and Tahi, F., "Compression of DNA Sequences", In Proc. IEEE Symp. On Data Compression, 1993, pp. 340-350.
8. Grumbach, S. and Tahi, F., "A new challenge for compression algorithms: Genetic Sequences", Journal of Information Processing &Management, Vol. 30, 1994, pp. 875-886.
9. Heba A_fy, Muhammad Islam, and Manal Abdel Wahed. "DNA lossless differential compression algorithm based on similarity of genomic sequence database", CoRR, abs/1109.0094, 2011.
10. Heba A_fy, Muhammad Islam, and Manal Abdel Wahed. "Genomic sequences differential compression model", International Journal of Computer Science and Information Technology, 3:145, 2011.
11. Kamnath Mishra, Dr.Anupam Agarwal, Dr.EdriesAbdelhadi and Dr. Prakash C. Srivasatava, "An Efficient Horizontal and Vertical Method for Online DNA Sequence Compression", IJCA, Vol. 3(1), June, 2010, pp.39-46.
12. Lei Chen, Shiyong Lu, and Jerey Ram. , "Compressed pattern matching in DNA sequences", In Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference, CSB'04, 2004, pages 62-68.
13. Manzini, G. and Rastero, M., "A simple and fast DNA Compressor, Software: Practice and Experience", MIUR support project, Vol. 34(14), 2004, pp.1397-1411.

14. Minh Duc Cao, T.I. Dix, L. Allison, and C. Mears. "A simple statistical algorithm for biological sequence compression", In Proceedings of the 2007 Conference on Data Compression, DCC'07, 2007, pages 43-52.
15. Panneer Arokiaraj S, Robert L and Arunachalaprabu, "An Improvised Run Length Encoding Based DNA Sequence Compressor", IJET, 2012, pp.679-683.
16. Panneer Arokiaraj S, Robert L "Pattern Recognition based DNA Sequence Compressor", Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on Date 18-20 Dec. 2012, Published in: Coimbatore, 2012, Page(s):1 – 5.
17. Panneer Arokiaraj S, Robert L, "Parallelized Pattern Recognition Based DNA Sequence", Computing and Communication Technologies (WCCCT) 2014 World Congress, Trichirappalli, 2014,Page(s):13 – 16.
18. Panneer Arokiaraj S, Robert L, "An Improvised DNA Sequence Compressor Using Pattern Recognition", International Journal of Engineering and Technology (IJET), Vol 5 No 6 Dec 2013-Jan 2014.
19. Pothuraju Rajeswari and Allam Apparao,"DNAbit compress genome compression algorithm", Bioinformation, 5(8):350, 2011.
20. P. Raja Rajeswari, Allam Apparo, and V. K. Kumar. "GENBIT compress tool (gbc): A javabased tool to compress DNA sequences and compute compression ratio (bits/base) of genomes". CoRR, abs/1006.1193, 2010.
21. Rivals,E., Jean Paul Delahaye, M., Dauchet and Delgrange,O.,"A Guaranteed Compression Scheme for Repetitive DNA Sequences", In Proc. Data Compression Conf. (DCC-96), Snowbird, UT. p453, 1996.
22. Sebastian Deorowicz and Szymon Grabowski. "Compression of DNA sequence reads in fastq format", Bioinformatics, 27(6):860-862, 2011.
23. Sebastian Wandelt, Marc Bux, and Ulf Leser,"Trends in Genome Compression", Knowledge Management in Bioinformatics, Institute for Computer Science, Humboldt-Universit□at zu Berlin, Germany June 4, 2013.
24. Subhankar Roy,Akash Bhagot,Kumari Annapurna Sharma and Sunirmal Khatua," SBVRLDNACOMP:An Effective DNA Sequence Compression Algorithm", International Journal on Computational Science & Applications (IJCSA) Vol.5, No.4, August 2015.
25. Tungadri Bose, Monzoorul Haque Mohammed, Anirban Dutta And Sharmila Smande , "BIND – An algorithm for loss-less compression of nucleotide sequence data", 37(4), September 2012, 785–789, Indian Academy of Sciences, Published online: 13 August 2012.

***