An Efficient Horizontal and Vertical Method for Online DNA Sequence Compression

Kamta Nath Mishra
Computer Sc. Department,
Birla Institute of Technology,
Ranchi, India (Allahabad Campus)

Dr. Anupam Aaggarwal
Computer Sc. Department,
I.I.I.T. Jhalwa,
Allahabad, India

Dr. Edries Abdelhadi
Computer Sc. Department,
Sebha University,
Libya

Dr. Prakash C. Srivastava
Computer Sc. Department,
Birla Institute of Technology,
Ranchi, India (Allahabad Campus)

ABSTRACT

DNA matching has become one of the most used biometric identification method during the last several years. DNA stores the information for creating and organizing an organism. It can be thought of as a string over the alphabets {A, C, G, T, N}, which makes four chemical components that make it up. Here, N represents an unknown nucleotide. This unknown nucleotide may be either A, or C, or G, or T. The size of each sequence is varying in the range of millions to billions of nucleotides.

Compression of DNA is interesting for both practical reasons (such as reduced storage and transmission cost) and functional reasons (such as inferring structure and function from compression models). We present a new Lossless Compression algorithm; which compresses data first horizontally and then vertically. It is based on substitution and statistical methods. We claim that our algorithm achieves one of the best compression ratios for bench mark DNA sequences in comparison to other DNA sequence compression methods.

General Terms
DNA Sequence Compression and Identification

Keywords
DNA Sequence, Lossless Compression, Horizontal compassion, Vertical compression, Substitution methods, Statistical methods, Genome structure

I INTRODUCTION

In the last few years, DNA evidence has started to play a big part in many nations criminal justice systems. It has been used to prove that suspects were involved in crimes and to free people who were wrongly convicted. Several countries, including United States and Britain, have built elaborate databases with hundreds of thousands of unique individual profiles [SSZR2005]. Currently many countries are establishing DNA database from individual with the history of violent and crimes. Those databases would allow the identification of suspects by simply cross checking the DNA profile of the evidence with those stored in the database. The Combined DNA Identification System (CODIS) connects local states and federal law enforcement agency data banks across the country USA (approximately 2 million profiles). As of march 2005 CODIS has produced 21000 criminal identifications and it has assisted in over 23000 investigations.

Today, increasing genome sequence data of organisms lead DNA database two or three times bigger annually. Thus, it becomes very hard to download and maintain such data in personal local system [CKL99]. The size of current biological databases is rapidly increasing due to continuous sequencing efforts. Some common databases need more than 160GB of disk space (e.g. DNA databank of Japan (DDBJ) [CKL99] and furthermore, additional disk space is needed for the index files of the different retrieval methods (e.g. BLAST [GT94]). Thus, there is a lot of disk space needed for redundant data. One could now think about efficient compression algorithms to solve this problem.

The problem with DNA based identification system is that it cannot be used for online identification of a person like iris based identification system or thumb print based identification system. But, DNA database of a person may be useful for further criminal investigation. Thus, we may either need to retrieve the DNA data from server for further processing and investigations or we may need to transfer the DNA data from one location to another location for further criminal investigations.

A single person's complete DNA structure may take more than 100 gigabytes of memory. Therefore if we transfer DNA database from one location to another location for further criminal investigations or for any other purpose then the transfer time will be very high. If we download the data from the server for further research work then its download time will be very high. Thus, we need an efficient lossless compression technique to compress the DNA sequence before sending it to another location so that after receiving and decompressing the data

at destination end the exactly same DNA sequence should be obtained.

Sometimes, we need to download the DNA data for pursuing further research work. If the compressed form of DNA data is stored on the server then its download time can be minimized and the researchers can use the same lossless compression technique to decompress the data after downloading it. Although lossless compression algorithms like BZip, Bio-Compress, Bio-Compress2, GeN, and GeNML are available for compression of DNA sequences but the compression result of these algorithms are not sufficient to compress the DNA sequence with high efficiently.

Thus, our proposed algorithm DNASC (DNA Sequence Compressor) will be useful to compress the DNA data efficiently before sending it from one location to another location. The receiving computer will further use our DNASC algorithm to decompress the data after receiving it.

Therefore, our algorithm will be useful to compress the DNA sequence before transmitting it and our algorithm will be used to decompress the DNA data after receiving it without any data loss (lossless compression method).

The deoxyribonucleic acid (DNA) constitutes the physical medium in which all the properties of a living organism are coded [TG93]. A DNA sequence consists of four alphabets namely Adenine (A), Cytosine(C), Guanine (G) and Thymine (T).

The ribonucleic acid (RNA) sequence is also consisting of four alphabets Adenine (A), Cytosine(C), Guanine (G) and Uraceil(U). RNA's nucleotide are similar to DNA's nucleotide but Thymine (T) is replaced by Uraceil(U) in RNA nucleotides. DNA compression by standard methods such as Lempel and Ziv[LZ76] style or Huffman coding [Huf52] does not give positive compression result. It means if we compress DNA or RNA sequence with LZ76 method or with Huf52 method then the compressed data will need more space than the actual data.

Various algorithms have been proposed to compress the size of text such as LZ76, LH87, Sto88, ZL77, ZL78, Grumbach and Tahi[GT93], Grumbach and Tahi[GT94], Apostolico and Lonardi[AL2000]. To compress the DNA sequence either offline or online we use two main approaches, the statistical approach and the substitution approach. In the statistical approach, blocks of fixed length (generally letters) are encoded with respect to their probability of appearance [Huf52]. In the substitution approach, factors of different length are encoded using a pointer to the previous occurrences of the text [TG94]. The substitution algorithms have negative compression ratio in DNA sequence compression. It means, the size of the DNA sequence increases after compressing it by using substitution approach.

Therefore, Grumbach and Tahi proposed two compression modes: Horizontal and Vertical. In the horizontal mode, Grumbach and Tahi compressed the DNA sequence into a shorter form. In the vertical mode Grumbach and Tahi compressed the DNA sequence with respect to another sequence B. This DNA sequence compression is known as Biocompress. For some genomes the compression rate of Biocompress is higher than 30%.

Biocompress-2 was developed by Grumbach and Tahi in 1994. Biocompress-2 was based on the detection and encoding of factors and palindromes. Biocompress-2 gives good result if the DNA genome has large number of similar sequences.

If we see the actual DNA structure then we find that there are so many unknown nucleotides in any DNA structure. The unknown nucleotides are represented by N in DNA sequence representation. We have divided our research work into two parts: In the first part we have excluded unknown nucleotides in compressing the DNA sequences, and in the second part we have included unknown nucleotides at the time of compressing the DNA sequence.

DNA compress introduced by Chen, Li, Ma, and Tromp [Chen et al. 2002], which also employs a two pass strategy is based on substitution (Lempel Ziv style) compression method. In the first pass a specialized programme called *Pattern Hunter* is used as preprocessor for finding significant approximate repetitions. The second pass then encodes these by a pointer to their previous occurrences [KT2005].

This paper is organized as follow. In the next section (section-II) we will review the basic knowledge of genome structure. In section-III we will compare the results of universal text compression algorithms. In section-IV we will propose our *DNA Sequence Compression (DNASC) Algorithm.* In section 5 we will analyze our results.

II RESEARCH ANALYSIS OF BASIC GENOME STRUCTURE'S DATA

A chromosome contains two complementary strands of deoxyribonucleic acid or DNA. These are long polymers of nucleic acid (nucleotides)each consisting of phosphate, deoxyribose and one of four 'bases' which consist of Adenine(A), Cytosine(C), Guanine(G), and Thymine(T). These always form a base pair based on hydrogen bonds between complementary bases: A-T or C-G(The two strands are termed as anti parallel, in that they 'run' in opposite direction) [VVP].

There is another kind of nucleic acid in the cell which is called Rebo-Nucleic Acid (RNA). The RNA contains the same nucleotide but where the thymine(T) is replaced by Uracil(U). The molecules of RNA are folded up in a complicated way. It is their three dimensional structure which determines their activity. There are pairs of complementary subsequences of RNA which are mapped together. Such pairs of subsequences are called palindromes [TG94]. The algorithms for detecting palindromes were studied by Apostolico, Breslauer, and Galil in 1992 [AB92].

Fig (i) Conversion of DNA to mRNA

DNA
TTTTCGAATTNAACCTCGGTTTNCCTGC
CTAACCTCCCAAGTAGCTGGGACTACA
GGCGCCTGCCCGCGCACCCGGCTAATT

TTTAGTAGAGACCGTGTTTCACCGTGTT
AGCCAGGATGGTCTCGATCTCCTGAC
mRNA
UUUUCGAAUUNAACCUCGGUUUNCCU
GCCUAACCUCCCAAGUAGCUGGGACU
ACAGGCGCCUGCCCGCGCACCCGGCUA
AUUUUUAGUAGAGACCGUGUUUCACC
GUGUUAGCCAGGAUGGUCUCGAUCUC
CUGAC

The complete DNA sequence of a living organism is called its genome. The RNA plays an important role to translate DNA into proteins. A section of the DNA coding for a protein is called genome structure [TG93].

DNA can be converted into RNA just by replacing thymine (T) with uracil(U). In the above Fig (i), we have converted DNA into messenger ribonucleic acid.

Sequence	Size	Bzip2	Bio2	Gen 2	CTW	DN A	GeNML
CHMPXX	121024	2.12	1.68	1.67	1.67	1.67	1.66
CHNTXX	155844	2.18	1.62	1.61	1.61	1.61	1.61
HEHCMV-CG	229354	2.17	1.85	1.85	1.84	1.85	1.84
HUMDYSTROP	38770	2.18	1.93	1.92	1.92	1.91	1.91
HUMGHCSA	66495	1.73	1.31	1.10	1.10	1.03	1.01
HUMHDABCD	58864	2.07	1.88	1.82	1.82	1.80	1.71
HUMHPRTB	56737	2.09	1.91	1.85	1.84	1.82	1.76
MPOMTCG	186608	2.17	1.94	1.91	1.90	1.89	1.88
MTPACG	100324	2.12	1.88	1.86	1.86	1.86	1.84
VACCG	191737	2.09	1.76	1.76	1.76	1.76	1.76

Fig (ii) Comparison of the compression results in bits per base obtained from the algorithms Bzip2, Bio2,Gen2, CTW, DNA, and GeNML

For converting DNA sequence if we take window size equals to 2^{18} then we find very good compression results. But we can take block size equals to 24 or 32 or 40 or 48 or 56 or 64 or 72 or 80 or 88 or 96 characters.

The sequence on which we perform compression tests are the DNA samples which are obtained from the finch TV tool. We are using the sample data obtained from finch TV tool because it is the actual DNA sequence data and it is easily available in the form of A, C, G, T, and N(space in DNA sequence). Similar results can be obtained for the data of MIPACGA, MPOMTCG, CHNTXX, MPOCPCG, YSCCHRIII, VACCG, HUMGHCSA, HUMHBB, and HS5HCMVCG.

III COMPARISON OF DNA COMPRESSION ALGORITHMS

If we compare the results obtained by algorithms suggested in the previous research work of Bzip2, Biocompress2, Gen2, CTW, DNA, and GeNML for the DNA sequences of CHMPXX, CHNTXX, HEHCMVCG, MTPACCG, and VACCG for different size of DNA sequences in terms of Bits per base, then we find the above results of Fig (ii)[KT2005]:

The results of above Fig (ii) show that the GeNNML model gives the best compression result. The GeNML algorithm has combined the statistical and substitution method together for window size = 2^{18} and block size=24 to 96.

IV THE ALGORITHM DNA SEQUENCE COMPRESSOR (DNASC)

In human DNA structure there are so many unknown nucleotides. These unknown nucleotides are represented by N (space). On April 14 2003 complete human genome structure was released on the NCBI website. If we see the DNA sequence of human genome then we find that the human genome structure consists of five characters A, C, G, T, and N where N represent unknown nucleotides (space).

Finch TV software tool is a tool which converts chromosomes of human genome in the form of DNA sequences (A, C, G, T, N). We have taken the human DNA sequence which is obtained from finch TV software tool for our research and HUMDYSTROP. The study reports reveal that in the human DNA structure certain characters are repeated again and again. The method which we have developed for compressing human DNA sequence is known as DNA Sequence Compressor (DNASC). This method can be used to compress the DNA and RNA sequence of human genome. But it is not applicable for compressing proteins.

Biocompress-2[GT94], and GeNML[KT2005] uses four characters of alphabet for DNA sequences in the form of characters A, C, G, and T. We take 5 characters A(Adenine), C(Cytosine), Guanine(G), Thymine(T), and N(Unknown Nucleotides) in our research work. We are including unknown nucleotides (N) in our research work because these unknown nucleotides

are occurring again and again in the human genome structure.

GeNML[KT2005] method is the combination of substitution and statistical methods and Biocompress-2[GT94] is based on substitution method. GeNML and Biocompress-2 both can compress the online textual data [SSZR2005].

OFF-LINE$_3$ [AL2000] compresses the offline textual data of DNA sequence. Since, the block size providing the best compression may vary as per the local changes in the DNA and RNA sequences. In our research work we have taken window size equals to 2^7 = 128 characters and we may take block size equals to 4, 6, 8, 10, 12, 14, or 16.

Let us take following Fig (iii) of human genome DNA sequence of 128 characters which is obtained from Finch TV software:

Fig (iii): Sample of human DNA sequence for 128 character data obtained from Finch TV software Tool

TTTTCGAATTNAACCTCGGTTTNCCTGCCTAACCT
CCCAAGTAGCTGGGACTACAGGCGCCTGCCCGCG
CACCCGGCTAATTTTTAGTAGAGACCGTGTTTCAC
CGTGTTAGCCAGGATGGTCTCGAT

If we see the repetition of the characters A, C, G, T or N in the human genome DNA sequence HUMDYSTROP, HUMGHCSA, HUMHDABCD, and HUMHPRTB then we find that no character is repeated continuously for more than 9 times. Therefore, in our research work we have assumed that no character of the human genome is repeated continuously for more than 9 times.

The human DNA sequence characters are represented by following equations:

S_1= {A, C, G, T, N} (equation I)

S_2= {1, 2, 3, 4, 5} (equation II)

If character A is repeated continuously for 5 times, character C is repeated for 3 times, character G is repeated for 4 times, T is repeated for 3 times and character N is repeated for 2 times in the DNA sequence then in Lempel-Ziv (LZ) style representation our data will be represented by following equations:

S_3= {AAAAA, CCCC, GGGG, TTT, NN}

(equation III),

and S_4= {15, 23, 35, 43, 52}

(equation IV)

We can't use 2 bits to represent the digits of equation IV because the largest two digit number which may exist in human DNA sequence as per our assumption will be 59. To represent 59 we need at least 7 bits. Thus, we will use 6 bits to represent each two digits number of equation-IV. If we represent each two digits number of equation IV by 7 bits then we get the following equation:

S_5 = {0001111, 0010111, 0100010, 0101011, 0110100} (equation V)

Therefore, if we represent the sample data of Fig (iii) in extended Lempel-Ziv (LZ) style by considering the equations I, II, III, IV, and V after including unknown nucleotides (N) then we find the following sequence of digits:

Fig(iv): Lempel-Ziv style (LZ) representation of data after including unknown nucleotides (N) and considering equation I, II, III, IV.

442131	124251	122241	233243	522241
312241	331231	411131	214133	122141
112111	322131	234131	233121	312111
233221	411246	314111	451131	411131
113111	223141	314321	112231	413142
113122	113211	413241	214121	311141

If we represent each 2 digits number of Fig (iv) into binary form by taking 7 bits then we will get the following result of Fig (v):

Fig (v): 7-Bit binary conversion of extended LZ style data of Fig - iv

0101100001010101011111
0001100010101001100 11
0001100000101100101001
0010101010000001010 11
0110011001011001010 01
0011111001011001010 01

0100001000110000111 11
0101001000101100111 11
0010101010010100001
0001011001010101001
0001011001010100010 11
0100000001010100111 11

0010110010100100111 11
0010111010100100111 11
0101001001111100010 11
0010111010000000101 01
0101001000110001011 10
0011111010100100010 11

0101101000101100101 11
0101001000101100101 11
0001011001111100000 11
0010110000111110100 01
0011111010101100111 01
0001011001011000101 11

0101001001111101000 10
0001011001111100101 10
0001011010000000010 11
0101001010000000101 01
0010101010100100101 01
0011111000101101010 01

and N. In our work we have represented it by S={1, 2, 3, 4, 5}.

Let C_b is the current block of 6 digits in which each 2 digit is represented by 7-bit binary number. Now each block will be encoded with respect to the current block[TG93]. If we represent each 2 digit of a block by using 7 bits then the maximum number of possible combinations will be $2^7 = 128$.

Let us suppose that the next block is represented by N_b. Now, following conditions may occur [EAHKNM2008]:

CONDITION 1: If next block is similar to the current block

Then $N_b = C_b$

CONDITION 2: If next block is complement of the current block

Then N_b = Complement of C_b

CONDITION 3: If next block is the reverse of the current block

Then N_b = Reverse of C_b.

CONDITION 4: If next block is complement of reverse of the current block

Then N_b = Complement of Reverse of C_b.

CONDITION 5: If next block is 2's complement of the current block

Then N_b = 2's Complement of C_b

CONDITION 6: If next block is the reverse of 2's complement of the current block

Then N_b = Reverse of 2's complement of C_b.

CONDITION 7: If next block is 2's complement of reverse of the current block

Then N_b = 2's Complement of Reverse of C_b.

CONDITION 8: If next block is 9's complement of the current block

Then N_b = 9's Complement of C_b

CONDITION 9: If next block is the reverse of 9's complement of the current block

Then N_b = Reverse of 9's complement of C_b.

CONDITION 10: If next block is 9's complement of reverse of the current block

Then N_b = 9's Complement of Reverse of C_b.

CONDITION 11: If next block is 10's complement of the current block

Then N_b = 10's Complement of C_b

CONDITION 12: If next block is the reverse of 10's complement of the current block

Then N_b = Reverse of 10's complement of C_b.

CONDITION 13: If next block is 10's complement of reverse of the current block

Then N_b = 10's Complement of Reverse of C_b.

CONDITION 14: If next block is equal to 1-bit to 6-bits of right shift or left shift of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of C_b.

CONDITION 15: If next block is equal to 1-bit to 6-bits of right shift or left shift of 1's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of 1's complement of C_b.

CONDITION 16: If next block is equal to 1-bit to 6-bits of right shift or left shift of 2's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of 2's complement of C_b.

CONDITION 17: If next block is equal to 1-bit to 6-bits of right shift or left shift of 9's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of 9's complement of C_b.

CONDITION 18: If next block is equal to 1-bit to 6-bits of right shift or left shift of 10's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of 10's complement of C_b.

CONDITION 19: If next block is equal to 1-bit to 6-bits of right shift or left shift of reverse of 1's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of reverse of 1's complement of C_b.

CONDITION 20: If next block is equal to 1-bit to 6-bits of right shift or left shift of reverse of 2's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of reverse of 2's complement of C_b.

CONDITION 21: If next block is equal to 1-bit to 6-bits of right shift or left shift of reverse of 9's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of reverse of 9's complement of C_b.

CONDITION 22: If next block is equal to 1-bit to 6-bits of right shift or left shift of reverse of 10's complement of the current block

Then N_b = 1- bit to 6 Bits of right shift or left shift of reverse of 10's complement of C_b.

These conditions can be represented by either capital case letters from A TO Z or by small case letters from a to z or by numerals from 0 to 9 or by special symbols.

Thus, we will get following cases for all the above stated conditions:

CASE 1 : $P_1(C_b / N_b)$ = A - Z

CASE 2 : $P_2(C_b / N_b)$ = a -z

CASE 3 : $P_3(C_b / N_b)$ = 0 to 9

CASE 4 : $P_4(C_b / N_b)$ = Special symbol

Therefore, we can represent the data of DNA sequence by following mathematical equation:

$$P(C_b / N_b) = P_1(C_b / N_b) + P_2(C_b / N_b) + P_3(C_b / N_b) + P_4(C_b / N_b) \quad \text{(equation vi)}$$

If we take window size equals to 128 bytes means 128*8 = 1024 bits and block size equals to 21

bits then we will have total 48 blocks in which the last block will be having little unused space.

After replacing the values of capital case letters (A - Z), small case letters (a – z), digits (0 – 9), and special symbols for the conditions 1 to 128 in Fig(v), we will get the compressed data of DNA sequence.

If we compare the compression result of DNASC algorithm with the existing compression algorithms then we find that DNASC algorithm gives the best result. We will compare the compression result of our algorithm with other algorithms in the next section (section 5).

STEPS OF DNASC ALGORITHM:

STEP 1: Initialize the values of parameters: block_size(b), Window_size(w).

STEP 2: Let the value of DNA sequence A is 1, C is 2, G is 3, T is 4, and N is 5.

STEP 3: Read the DNA sequence from the file and convert it in the form of extended LZ style by using a counter variable (initialize counter=0). If a character of the DNA sequence is repeated more than one time continuously then the value of the counter will be increased by 1.

STEP 4: Convert the extended LZ data into different blocks. (For our case block_size(b)=6 digits).

STEP 5: Convert every 2-digits data of all the blocks in 7 bits binary code.

STEP 6: Initialize, current_block(W)=first block

STEP 7: Compute complement_of_first_block, reverse of first block, complement of reverse of fist block, 2's complement of current block, reverse of 2's complement of current block, 2's complement of reverse of current block, 9's complement of current block, Reverse of 9's complement of current block, 9's complement of reverse of current block, 10's complement of current block, Reverse of 10's complement of current block, 10's complement of reverse of current block, left shift of first block from 1-bit to 6 bits, and right shift of first block from 1-bit to 6-bits.

STEP 8: Compress all the blocks of the window in the form of A – Z, a – z, 0 – 9, and special symbols.

STEP 9(a): Display the compressed DNA sequence result.

STEP 9(b): Prepare a table which represents the conversion of bits into capital case letters (A-Z), small case letters(a-z), digits(0-9), and special symbols.

SPET 10: Decompress the DNA sequence by using the table of STEP 9, and display the decompressed result.

STEP 11: END.

The DNASC algorithm can be implemented in C or C++ language.

V ANALYSIS OF DNASC ALGORITHM'S RESULT

The biocompress and biocompress-II algorithms have used Lempel-Ziv style of data representation for DNA nucleotides A, C, G, and T. They did not include unknown nucleotide N (space) of the DNA sequence. It compresses the DNA sequence first horizontally and then vertically. Biocompress and biocompress-II uses 2 bits for encoding a DNA sequence alphabet.

Fig vi: Performance comparison of DNASC with GeNML and other algorithms

Sequence	Size	Bzip 2	Bio2	Gen2	CTW	DN A	GeN ML	DNASC
CHMPX X	121024	2.12	1.68	1.67	1.67	1.67	1.66	1.50
CHNTX X	155844	2.18	1.62	1.61	1.61	1.61	1.61	1.51
HEHCM V-CG	229354	2.17	1.85	1.85	1.84	1.85	1.84	1.80
HUMDY S-TROP	38770	2.18	1.93	1.92	1.92	1.91	1.91	1.89
HUMGH C-SA	66495	1.73	1.31	1.10	1.10	1.03	1.01	0.91
HUMHD A-BCD	58864	2.07	1.88	1.82	1.82	1.80	1.71	1.61
HUMHP R-TB	56737	2.09	1.91	1.85	1.84	1.82	1.76	1.71
MPOMT C-G	186608	2.17	1.94	1.91	1.90	1.89	1.88	1.88
MTPAC G	100324	2.12	1.88	1.86	1.86	1.86	1.84	1.80
VACCG	191737	2.09	1.76	1.76	1.76	1.76	1.76	1.70

GNML and GeNML algorithms use the combination of substitution method and statistical method. In GeNML method Korodi and Tahi first used the substitution method to compress the data horizontally and

vertically, and then KT2005 have used probability and statistics to find the occurrences of a particular sequence in the DNA database.

KT2005 have used window size equals to 2^{18} and block size equals to 24 or 32 or 40 or 48.

In our algorithm DNASC we have also included unknown nucleotides N (space) with other symbols A, C, G, and T of DNA sequence.

In our DNASC algorithm we have used extended Lempel-Ziv[LZ76] style representation for 5 basic symbols A, C, G, T, and N. Here, N represents unknown nucleotides which is either A, or C, or G, or T. These unknown nucleotides can be either A, or C, or G, or T. But in our representation we have taken unknown nucleotide (N) as a separate symbol. Thus, in our representation we have used five basic symbols.

The above Fig (vi) shows the performance comparison between Bzip-2, Bio-2, Gen-2, CTW, DNA, GeNml, and our algorithm DNA Sequence Compressor (DNASC) for DNA sequences after including unknown nucleotides as a separate entity N. The practical evaluation of the performance of DNASC was done in such a way that it is easily comparable with the published results explaining the performance of Biocompress-2[GT94], Gencompress-2 [Chen et al 2001], DNA Compress [Chen et al 2002], and GeNML [Grambach and Tahi 2005], which all are using the same set of DNA sequences. These DNA sequences are available as a DNA database and these DNA databases are modified and updated usually. For each file of DNA database we have taken window size is equal to 128 and block size equal to 21. Encoding all the files in a computer of dual core processor with 512MB RAM and 1.6GHz processing speed the algorithm has taken approximately the same time which Grambach and Tahi algorithm takes to compress the whole set of DNA sequences of Fig(ii). But our algorithm gives better compression result in terms of bits per base in comparison to other existing algorithms. To decompress the DNA sequence data set our algorithm has taken approximately half of the time which was taken by the processor to compress it.

We have found in our research work that the cases where our method cannot improve the known result then for these cases none of the existing algorithm could provide an improved result. The results of our algorithm are an improvement over the best known results to date on all the files in which repetitions may occur again and again.

In our algorithm we have compressed the DNA sequence first horizontally and then vertically in our algorithm. To compress the data vertically we have taken block size equals to 6 and window size equals to 128. To compress each 2 digits of a block we have used 7 bits. Therefore, each block is compressed by using only 21 bits because each block has 6 digits and each 2 digit is represented by 7 bits.

Two cases of GeNML model:

CASE 1: If the next block is same as the first block then the value of $P(C_b / N_b)$ will be 1.

CASE 2: Otherwise the value of $P(C_b / N_b)$ will be 0.

But, in our algorithm's statistical part we have used 22 basic cases which includes all the possible combination and these cases are described in section- iv. Further we have represented the conditions from 1 to 26 by capital case letters(A-Z), conditions from 27 to 52 by small case letters(a-z), and conditions from 53 to 62 by digits (0 – 9), and other conditions by special symbols. Now these numbers are further represented in 7 bits binary numbers.

In order to illustrate the practical strength of our algorithm we have tried to compress the complete human genome structure which was released in April 2004. Here, we found that the human genome consists of a large number of unknown nucleotide which are represented by N. These unknown nucleotides can be either A, or C, or G, or T. Our algorithm also compressed the human genome structure successfully. The parameter set for the compression of human genome structure was Window size equal to 512, and block size equal to 64. Here, we took block size took block size in the multiples of window size to minimize memory waste.

VI CONCLUSIONS AND FUTURE WORK

In this paper we have introduced an efficient DNA sequence compression algorithm called DNASC. This algorithm has following six parts:

Part I: Lempel-Ziv style representation of data,

Part II: Dividing the Lempel-Ziv style data into different blocks of same size (block size = 6) for a window size=2^7,

Part III: Converting each block into 7-bits binary code.

Part IV: Encrypt the data, by using the conditions (1 to 22) discussed in section IV, in the form of capital case letters (A-Z), small case letters (a-z), digits (0 – 9), and special symbols.

PART V: Prepare the encryption and decryption table.

PART VI: Finally decrypt the data by using decryption table.

DNASC algorithm is the combination of substitution and statistical methods. The performance of the algorithm DNASC depends upon 128 conditions explained in section IV.

Finally, we have analyzed and compared the performance of DNASC algorithm with other algorithms. Taking different block size and window size in DNA sequence compression is an important area of research and we need to do further research work in this area.

REFERENCES:

1. [AL2000] Alberto Apostolico and Stefano Lonardo, Compression of Biological Sequences By Greedy Off-Line Textual Substitution, 2000.
2. [AB92] A. Apostolico, D. Bresauer, and Z. Galil. Optimal Parallel algorithms for periods, palindromes, and squares. In unpublished 1992.
3. [LZ76] A. Lempel and J. Ziv, On the complexity of finite sequences, IEE Transaction Inform. Theory, 22(1): 75-81, 1976.

4. [CKL99] Chen, X, Kwong, S., and Li, M., A compression algorithm for DNA sequences and its application in genome comparison, Genome Informatics, 10:52-61, 1999
5. [Huf52] D.A. Huffman, A method for construction of minimum redundancy codes, In proc. IRE, volume 40, page 1098 – 11101, Sept 1952.
6. [LH87] D.A. Lelewer and D.S. Hirshberg, Data Compression. ACM computing Surveys, 19(3):261-287, 1987.
7. [EAHKNM2008] E.A. Hadi, K. N. Mishra, DNA Sequence Compression Algorithm, National Conference on Communication and Information Technology, Tripoli, May 19 – 21 2008.
8. [TG93] Fariza Tahi, Stephen Grumbach, A New Challenge for Compression Algorithms: Genetic Sequence, 1993
9. [TG94] Fariza Tahi, Stephen Grumbach, Compression of DNA Sequences: Extended Abstract, 1994
10. [KT2005] Gregely Korodi, Ioan Tabus, An Efficient Normal Maximum Likelyhood Algorithm for DNA sequence compression, 2005.
11. [Sto88] J.A. Storer. Data Compression methods and theory, Computer Science Press, 1988.
12. [ZL77] J. Ziv and A. Lempel, A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 22(3): 337-343, May 1977.
13. [ZL78] J. Ziv and A. Lempel, Compression of individual sequences via variable rate coding. IEEE Transactions on Information Theory, 24(5): 530-536, Sept 1978.
14. [SSZR2005] Sheng Bao, Shi Chen, Zhi-Qiang Jing, Ran Ren, DNA Sequence Compression Algorithm Based on LUT and LZ77, Proceeding of the fifth International Symposium on Signal Processing and Information Technology, Volume, Issue, 18 – 21 Dec. 2005 Page(s): 23 – 28.
15. [GT94] S. Grumbach, and H. Tahi, A New Challenge for compression Algorithms: genetic sequences, Information Processing and Management, 30:875-886, 1994.
16. [VVP] V.V. Pillay, Textbook of Forensic Medicine and Toxicology, Paras Publication, India
17. www.ebi.ac.uk/embl/Documentation/User_manual/usrman.html, Jan 2008