DNA Sequences Compression using Repeat technique and Selective Encryption using modified Huffman's Technique

Syed Mahamud Hossein[1] Debashis De[2] Pradeep Kumar Das Mohapatra[3]
[1]mahamud123@gmail.com
[1] Department of Computer Science, Vidyasagar University, Midnapore-721102, West Bengal, India
[2]Department of Computer Science and Engineering,
Maulana Abul Kalam Azad University of Technology, Nadia-741249.
[3]Department of Microbiology, Raiganj University, Raiganj, West Bengal, India

Abstract
The DNA (Deoxyribonucleic Acid) database size increases tremendously transmuting from millions to billions in a year. Ergo for storing, probing the DNA database requires efficient lossless compression and encryption algorithm for secure communication. The DNA short pattern repetitions are of paramount characteristics in biological sequences. This algorithm is predicated on probing exact reiterate, substring substitute by corresponding ASCII code and engender a Library file, as a result get cumulating of the data stream. In this technique the data is secured utilizing ASCII value and engendering Library file which acts as a signature. The security of information is the most challenging question with veneration to the communication perspective. The selective encryption method is used for security purpose, this technique is applied on compressed data or in the library file or in both files. The fractional part of a message is encrypted in the selective encryption method keeping the remaining part unchanged, this is very paramount with reference to selective encryption system. The Huffman's algorithm is applied in the output of the first phase reiterate technique, including transmuting the Huffman's tree level position and node position for encryption. The mass demand is the minimum storage requirement and computation cost. Time and space complexity of Repeat algorithm are O(N^2) and O(N). Time and space complexity of Huffman algorithm are O(n log n) and O(n log n). The artificial data of equipollent length is additionally tested by this algorithm. This modified Huffman technique reduces the compression rate & ratio. The experimental result shows that only 58% to 100% encryption on actual file is done when above 99% modification is in actual file can be observed and compression rate is 1.97bits/base.

Keyword :
Sequence, Compression, decompression, Huffman, encryption & decryption .

1. Introduction

The whole DNA chains of many structures are already recognized and the entire human genome challenge is making regular progress. The statistics of DNA, RNA and amino-acid chains of proteins are stored in molecular biology databases. Now a day's data reliability is a tough mission, a way to shield the DNA facts from the hackers [1].The size of DNA database for both prokaryotes and eukaryotes are extremely increasing [2], complex, and have a few logical structures [3], so this large database required a systematic compression approach for storing[4-7]. However, if one implements standard software such as "compack", "pkzip" and "arj", these software enlarges the file size with increasingly above 8 bits per base, albeit these standard compression software are used in text files and structure & function in DNA chains are more precise.
It means that traditional compression algorithm m is inapplicable on DNA chains. To compress DNA content a better model is essential, higher statistics compression outcomes can be completed. The Huffman's code additionally is inapplicable also on genomic data both for the adaptive and static version, as the occurrence of probabilities of the 4 representatives are not extremely unlike [6]. In dictionary base compression, the genomic identity is not fully maintained, compromise genome identity. The use of reliability straight in the cellular DNA chain, obtain extremely small tag reliability because the DNA chain holds only 4 characters, by trial and error methods anyone can hack the data. The Data Encryption Excellence (DES), Advanced Encryption Excellence (AES), Rivest-Shamir-Adleman (RSA) and Escrowed Encryption are not systematic when the data size is large[8]. Also the problem is how to differentiate by naked eye the randomly generated artificial chains and Cellular DNA chains over communication point of view. Most of the available compression techniques are reducing the compression rate without considering the reliability concern. In selection- encryption, small part of the sequence is encrypted, other part is unencrypted, shown in fig.1.

This string matching is scanned left-to-right by the use of individual shift rule[8], for the encryption purpose exchange inside the Huffman's tree branches at a specific node & level on the same key required for decoding the encoded representatives using the particular modified Huffman's tree. This approach is applicable on compressed genomic data string and has to be compressed due to constraint of the network bandwidth before communicating. As per Shannon theory [9-10], source entropy is equal to the average bit rate when lossless compression occurs. If joined both the compression &

Manuscript received August 5, 2024
Manuscript revised August 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.8.9

selection encryption process, getting another property as shown in fig. 1

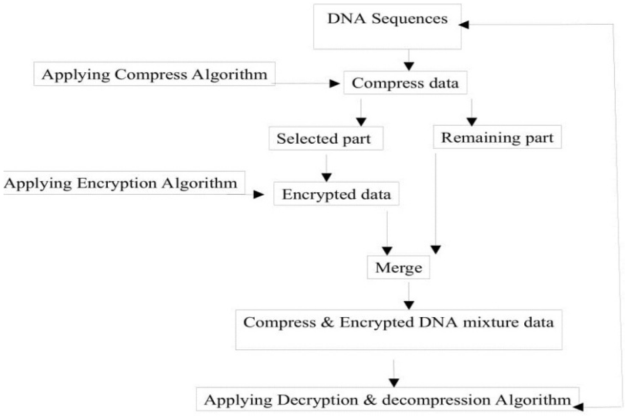

DNA Sequences
Applying Compress Algorithm
Compress data
Selected part
Remaining part
Applying Encryption Algorithm
Encrypted data
Merge
Compress & Encrypted DNA mixture data
Applying Decryption & decompression Algorithm

Fig.1 compression- encryption process

The selected encryption is done only in the match region of plain text and group of nucleotide is selected for encryption. The match region identification is the basic principal of this technique. If the subsequence match is high in a sequence, in that situation the security level is increased, also increase the time for encryption. In a plain text to find the match region is the benefit of the compression. The redundancy of the plain text is reduced by this technique and cipher text size is not larger than the plain text. So, the technique is user friendly. The plain text is reformed by selective encryption where match region is higher, as a result producing great effect on encryption through decompression. The remaining part where no match occurs produces no effect on encryption after decompression. The compression is done by the grouping of three/ four nucleotide sequence and replace by a single ASCII code, at the same time getting very high security by selection encryption. The nucleotides group are replaced by ASCII code, acting as key. This key is private, known only who encrypt and sequence is sending. If anyone tries to decrypt the sequence without proper grouping of sequence/key or library file, the sequence is changed. The decryption process is known to everyone and the private key is unknown to everyone so this process of cryptography is reliable.

To find out exact repeat is not an easy task in a very long sequence. The exact repeat searching algorithm requires more time for execution. Every compression algorithm has an aim to minimize the compression rate with operational time. It is based on dynamic repeat searching and repeat substring is placed in Library file and maximum repeat places is replaced by ASCII character.
The operating time is very less and it depends on the input file size. The evaluation of any encryption system depends on its speed and levels of security it provides. The operating time is minimum, required minute recollection and facilely utilizes this algorithm.

The Huffman [11] coding is used for lossless compression technique. It is a particular type of optimal prefix code and output can be viewed as a code of variable length. The Huffman's code is one sort of measuring code [12] and entropy coding [13-14] it allocates codes to images as to coordinate code lengths with the probabilities of the images.

Four phases of the proposed algorithm i) All exact repeat finding ii) finding un-match and match regions and encode match region iii) apply modified Huffman's technique on tree node position and label position for security purpose and iv) Selective encryption applied in a compressed, library file or in both.

Our developed algorithms as discussed with experimental results, are compared with standard available results [15-21]. To complete this work, also developed other related algorithms as file size measurement, calculating numbers of bases in a file, i,e file size, file mapping algorithm; is developed because no mathematical formula for proving the two files are same or not, this check input-output file character one by one, change the DNA sequences orientation for reverse, reverse complement the sequence and the same work on random string also. We observed that compression rate, ratio and run time on benchmark [22] DNA sequences is better than any standard technique, simultaneously. Also find out the run time performance of this algorithm.

2. Motivation and contribution

The primary objective of any compression techniques is to reduce the disk cost, by this techniques the order can be obtained independently of the exiting sequences as auxiliary storage as such the databases store compactly .These processes have need of much time for I/O compression-encryption and decryption decompression method. But in some examples, compression enlarges the over head like size of place for keeping records and processing time and so on. To prevent data loss during sending, many compression algorithms are used to get changed to other forms, the size of the facts during transmission. However, they are also deeply related to order and data mining and observations genomic facts compression, and the connected techniques related to theory of information, are often thought as centre of main aim of facts exchange and space for storing. The repeat identification is an important characteristic of compression technique. A chief trouble in repeat identification comes about from the facts that the repeat unit can be certain, errorless and given details of measure end to end.

This algorithm is an effective tool for compression-encryption of DNA sequences by using exact repeat and modified Huffman’s technique. The important feature of repetition in biological sequence has not been observed by all available algorithm. Our algorithm overcome this drawback & considered this issue.

3. Methods

3.1 Process diagram

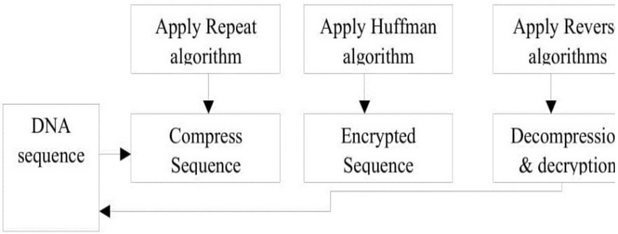


Fig.2 Process diagram

3.2 File format : The algorithm testing purpose use text file and end of file is indicated by the blank space. The result is also stored in a text file, the output file is the mixture of unmatched four base pair and ASCII notation.

3.3 Formation of substring / word of different size
Consider a DNA sequence is atggtagtaatgtacatg …… ...n. Where n is the number of a,t,g & characters present in a file. n is the size of file, required n byte to store this file. The sub-string formation method is in paper[23-24].

3.4 Merge Process :The Merge process is used for reducing the compression rate and other parameter of this work. The merge compression process is two pass, 1st pass followed by repeat and in 2nd pass uses excellent lossless compression techniques available in the market such as Huffman's technique. The first step consists of repeat coding (let each individual repeat process is called A, the output is O_1), and second step use excellence Huffman coding (let each individual process called B, the output is O_f) process. O_f is the final output, shown in fig.3.

The procedure for Multi step DNA Chains compression

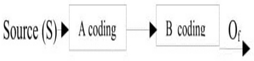


Fig. 3 Merge Process

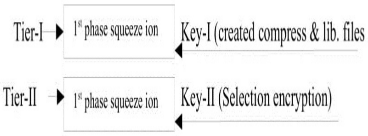

Tier-I
1st phase squeeze ion
Key-I (created compress & lib. files
Tier-II
1st phase squeeze ion
Key-II (Selection encryption)

Fig. 4 Two tier compression-encryption method

Define as - Let, S be source file to be coded.
Step 1 : O_1 1st pass of Repeat coding(s)
Step 2 : O_f Huffman’s’ coding (O_1)
For reliability purpose, introduced a new reliable method two tier selection encryption method as shown in fig. 4.

Tier one- the input order has within only 4 symbols (c, t. g & a), after compression is changed to the other form, get four characters to 256 notation with un-match c, t. g & a and one sub-string has in it three characters, replaced by single ASCII symbol. As a result the output file is safer than the input file.

In tier two- In this way of encryption process the file is encrypted either in compressed file or in the library file or in both. The process of encryption is done by exchanging of the branches of Huffman’s tree.

This technique bulwarks the DNA sequence information from hackers. The decoding time required the authentic encoding value, this value can provide the security, this security is applied in tier one. This technique uses only available ASCII code for encryption purpose and different pattern is utilized for cull encryption purport. The DNA sequences probing purpose the utilizer can send the encrypt compressed data to the receiver and the receiver decrypt the encrypted data by utilizing correct coded value. The transmission time is reduced over the Internet while the compressed file is decrypted followed by decompression at the client end. The utilizations of DNA sequence is incremented by applying compression and security techniques. This technique increases the efficiency of DNA uses.

3.5 Introduction of Repeat technique

In repeat technique, the highly repeated sub- sequence is replaced by a single ASCII code in source file and subsequence is placed into the library file dynamically. This dynamic library file work as a lookup table and act as security key, known only who encrypt the sequence. This substring length and ASCII code starting position depends on the user.
In two ways proposed algorithm work as first find out all the perfect match repeated substring. Second perfect match region is encoded by ASCII code and non match bases placed into the output file.

3.6 Methodology of Repeat Technique

Consider a DNA sequence as
s=atgtggtagtaatgtacatgcatgtgg……..n
In repeat technique, the principal idea is as the substring s_1=atg is repeated in how many places, is shown by red color. The s_2=tgg sub-string repeated in how many places is shown by the green color and so on.
First replace maximum repeated substring by the corresponding ASCII code in appropriate places.
The input string ***S***, assume that a part w_r (variable word/sub-sequence size) has been compressed , it is defined as S=w_{ri} (where i is denoted different word from 1 to n nos.) . The algorithm finds an optimal match position, stored in ascending order that can be encoded

economically. This left to right scanning process is search character by character, if no optimal match is found, left the character, moved the process forward at the end of file, shown in fig.5.

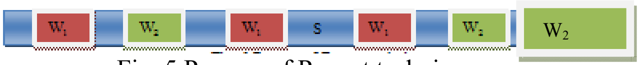


Fig. 5 Process of Repeat technique

3.7 Searching procedure

Searching for exact repetitions, encoding procedures of Repeat technique , the encoding analysis of Repeat technique and decoding procedure of Repeat technique are discussed in detail in paper[22]

3.8 Compression & decompression algorithm of Repeat technique

The DNA sequence compression algorithm based of Repeat technique

```
INPUTS:
i. DNA sequence & Artificial sequence in text format
ii. l is the length of word
iii. $S=w_{ri}$

OUTPUT :
i. Library file and compressed file

BEGINING
i. First start position of ASCII code is defined
ii. 1 to <10 is the word size and count
iii. Different  word is produced
iv. Hole DNA sequence is scan by sub word
v. Output store in two different files


ITERATION

while $w_r \neq$ end of file do
Search for an optimal postfix of different word with the DNA sequence
if   an optimal postfix is found , store in ascending order  then
    Encode the maximum repeat substring  by ASCII code, where i is
    Starting word position and l is the length of word.
Output the code.
else Set $w_{ri}$ in next step, encode and output it.
    Remove the temporary compressed file and Library file
End
```

The DNA sequence decompression algorithm based on Repeat technique

```
INPUTS INITIALIZE:
i.  First input library file and compressed file
OUTPUT :
i.  Original  sequence extracted

BEGINING
iv. DNA sub sequence is replace by ASCII value

ITERATION
1.  for( library file size check)
      flib[i]=fcom[i]
      for( match library file size with subsequence size)
      fname[i]=fcom[i]

2. Search character by character in the compressed file.

3. if( Sub sequence is match with ASCII value);
       produce original sequence
      Else repeat the process

4. Do step 2 to 3 until end of file is reached.

5. Generate original file.

7. End
```

3.9 Methodology of experiments performed in modified Hoffman's technique

In the first phase repeat experiment is done on different size of DNA sequences and Huffman's tree is generated using the output of statistical property of 1st phase compressed data. The main aim is to select the r part in the output of the 1st phase compressed data and on the basis of key swapping the Huffman's tree branches, this is called encoding, decoding required actual encoding key. This modified Huffman's technique is classified in the process I, II & III.

Process-I: Swapping the Huffman tree nodes at a particular level.

Process-II: Swap the Huffman tree particular nodes at different level. This process is done on character as well word.

Process-III : considering words instead of character

Process-I : First select the r part in the compressed text. On the key basis the Huffman's tree nodes is swapped at a particular level and decode using the encoding key of the modified Huffman's tree. The left and right nodes are interchanged at a particular level. Due to interchanging of node corresponding code is affected and remaining codes are also altered. Only interchanged nodes are affected other node is as usual and nodes related bit is altered.

The above process-I is explained below fig 6,7 & 8 where Huffman's codes are M=00, N=01, R=10, S=11.

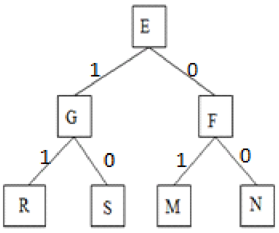


Fig. 6 example of process –I

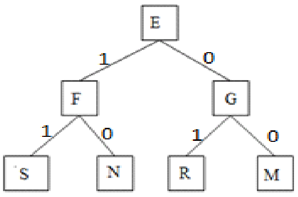


Fig. 7 level 1 swapping

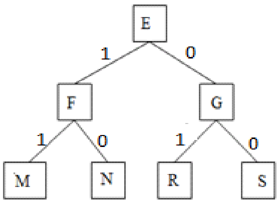


Fig. 8 level 2 swapping

Swapping is done at level 1 where E is single node is called root node and the level 0. Now change the child node position at level 1 with their sub tree is explained in fig. 8. Simultaneously sub tree position and value are changed as M=01, N=00, R=11, S=10. The actual text "MMNRNS" is encrypted as "010100110010". Decoding without actual encoding key the text is "RRSMSN". Here the Lavenstein distance is 6.

Next swapping at level 2, the interchange left node S with right node M explain in fig. 7, codes are also changed and remaining code is same. The actual text is "MMNRNS" and corresponding encrypted value is "000010011011". Decoding without actual encoding key the text is "SSNRNM". Here the Lavenstein distance is 3. The corresponding binary code is shown in table 1.

Table 1 Huffman code before and after encryption

Character	Before Encrypt	After Encrypt	
		Swapping at Level 1	Swapping at Level 2
M	11	01	00
N	10	00	10
R	01	11	01
S	00	10	11

To find Lavenstein distance on Modified Huffman techniques faced some problem on interchanging file. Interchanging of nodes are not applicable in all the cases for example, if the frequency is E=2, F=1 and G=1 then tree is explained in fig.9, interchanging binary node value is E=0, F=10, G=11 and assume string is 'EEFFG' it would be encoded as 00101011.

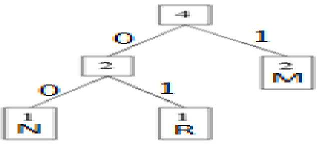


Fig. 10 Huffman tree after apping at level 0

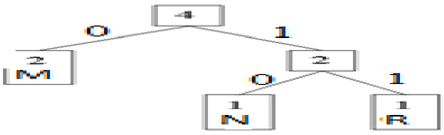


Fig. 9 Huffman tree after swapping at level 1

The Huffman tree will look like as in the fig. 10, when the swapping is done at level 0 causing codeword as M=1, N=00 & R=01 and the string would be encoded as 11000001.

Finding the effect on the actual text and calculate the Lavenstein distance. Then decode the encoded string without swapping technique, the string will look like as table 2 with respect to fig.8.

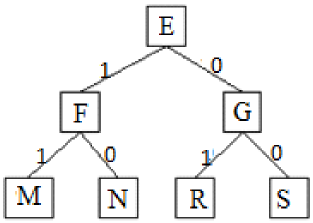
Table 2 Huffman code after swapping

1	1	0	0	0	0	0	1
R	R	M	M	M	M	M	-

There is no corresponding character for 1 in the last column of the table-3. For accuracy purpose the actual text size is altered in the above cases.

Proceed-II :Two different node at specified level of swapping

The r part is selected by swapping two notes in specified level. This technique is useful for interchanging any two nodes of the Huffman's tree at its subtree level. This technique modifies the actual Huffman concept and enhance the security aspect. This technique required two level value with their corresponding binary codes. The corresponding binary codes are equivalent to their level value with respect to nodes. This process uses two key values for execution, it enhances the security aspect than process-I. This process is depicted as in fig. 11 with their code as M=11, N=10, R=01, S=00.



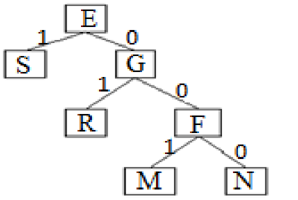


Fig. 12 swapping two nodes in level 1

Fig. 11 Huffman Tree

Now, interchange in between F & S, as in fig. 12 of level 1, in this case the binary value of S is 1 and F is 00. Also the interchange in between N and G is depicted as in fig.13. The code value of M, N, R, S are changed as in table 3. If the actual text is as "MMNRNS", the binary value is encrypted as "111110011000". If decrypt is done without considering the changed value, the string is as "SNRSSNS" and in this case the D_{SID} is 10. The corresponding binary code shown in table 3.

Table 3 Huffman code before & after encryption

Character	Before Encrypt	After Encrypt	
		Swapping Between F and S	Swapping Between G and N
M	11	001	11
N	10	000	0
R	01	01	101
S	00	1	100

Process-III : considering words instead of character

In the previous section applied selective encryption considering each character as a symbol in a text document. The characters alone does not possess any meaning but words do have. Generally it is found that in any text document file, there are few vital words. It amends the protection of the whole document. So by swapping the Huffman tree at a lower level (i.e. encrypt a small % of the original file) can encrypt all the keywords. With this idea, new process-III is now illustrating below.

In this scheme take a small text file and using the statistical property of the words of the text, encode input text file. Then made swapping on the basis of process-I and compute the damage occurred due to the swapping with respect to the original text file and got the original text file decoding by modified Huffman tree. Here illustrate this with an example. Now take a simple text

"L.AA.ATG.ATGC.ATGCA.ATGATG.ATGCA.AA"

It contains words and also some punctuation marks. These punctuation marks are also considered as words. Frequencies of words are given in table 4.

Table 4 word frequency

Distinguishable words	Frequency
L	1
.	7
AA	2
ATG	1
ATGC	1
ATGCA	2
ATGATG.	1

The above string corresponding Huffman's tree is depicted in the fig.14.

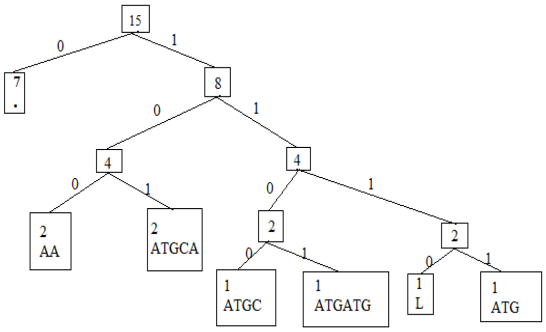


Fig.14 Huffman tree using word

Now generate Huffman's code from Huffman's tree which are given in table 5.

Table 5 word base Huffman code

Distinguishable words	Code
L	1110
.	0
AA	100
ATG	1111
ATGC	1100
ATGCA	101
ATGATG.	1101

So after encoding text message will be 111001000111101100010101101010100. For this approach perform a swapping method at a specified level (same as performed in case of considering characters).

Now apply swapping method at level 2. Fig. 14 shows after swapping.

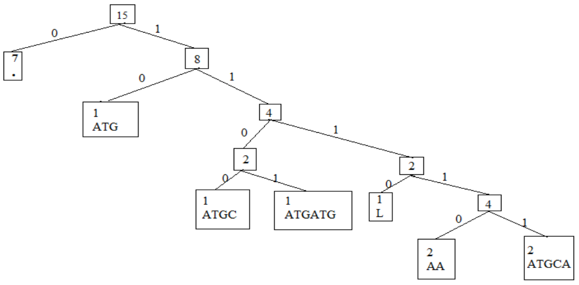


Fig. 15 Huffman tree after swapping

So corresponding codes of words are changed selectively, i.e. since in fig. 15, where 2 and 4 interchange their positions as "ATG", "AA.ATGCA" are only changed, other should be unchanged. After modifying the tree codes corresponding to distinguishable words are changed which are shown in table 6

Table 6 Huffman code after swapping

Distinguishable words	Code
L	1110
.	0
AA	11110

ATG	10
ATGC	1100
ATGCA	11111
ATGATG.	1101

So after encoding text message will be encrypted like 1110011110010011000111110110101111101110
If not consider changed of level in decoding time the text will be look like "L.ATG.AA.ATGC. ATG. ATGCAATGCA. ATG. ATGCA.L". D_{SID} in this case is 19
.

3.10 Encoding algorithm of modified Huffman's technique

```
Compressed DNA sequence encryption process using Swapping the Huffman tree nodes at a particular level (process-I)

INITIALIZATION OF INPUTS:
i. The output of Reverse technique is input of this technique
ii. Enter the level of 1st node
iii. Enter the binary path of 1st node
iv. Enter the level of 2nd node
v. Enter the binary path of 2nd node

ESTIMATED OUTPUT :
i. Input file size
ii. Output file
iii. Output file size
iv. Compression rate
v. Encoding  time

START
i. Generate Huffman tree
ii. Swapping the Huffman tree nodes at a particular level
iii. Request to store the output file

ITERATION
1. The key basis of Huffman tree nodes are swapped at a specific level and  modified Huffman tree is use for decoding.
2. The r part selection swaps two nodes at specified levels or same levels. To exchange right most node with left most node at specified level. The swapping nodes are only affected also other codes also  altered including bits. Except swapping nodes, the remaining nodes are as usual.
3. End
```

```
Compressed DNA sequence decryption process using Swapping the Huffman tree nodes at a particular level (process-I)

INITIALIZATION OF INPUTS:

i.  Enter the compressed text file
ii. Enter the level of 1st node
iii. Enter the binary path of 1st node
iv. Enter the level of 2nd node
v. Enter the binary path of 2nd node

ESTIMATED OUTPUT :
i.  Exact original  sequence
ii. Decoding time

START
i. Generate Reverse Huffman tree

ITERATION
1.  for(sort the Huffman tree)
            for(decoding the swapping tree)
                  if(frequency match)
                     reconstruct the code
                     else repeat the step
                     End
2. File decompressor for files compressed with HuffEnc.
3. End
```

```
Compressed DNA sequence encryption process using Swap the Huffman tree particular nodes at different level. This process also done on character as well word (Process-II).

INITIALIZATION OF INPUTS:
i. The output of Reverse technique is input of this technique
ii. Enter the lebel to change
iii. Enter the binary value

ESTIMATED OUTPUT :
i. Input file size
ii. Output file
iii. Output file size
iv. Compression rate
v. Encoding  time

START
i. Generate Huffman tree
ii. Swap the Huffman tree particular nodes at different level
iii. Request to store output file

ITERATION
1:  This process is done in any two particular nodes and including their subtree of Huffman tree. For interchanging purpose use more than one key of modified Huffman tree.
2: This process first identify two particular node of their corresponding level  and their binary value, also kept in mind that the binary value is same as level
```

```
value. This process uses two key value for two nodes, it
improves the security than process-II.
3. End
```

Compressed DNA sequence decryption process using Swap the Huffman tree particular nodes at different level. This process is also done on character as well word (Process-II).

```
INITIALIZATION OF INPUTS:
i.  Enter the compressed text file
ii. Enter the level to change
iii. Enter the binary value

ESTIMATED OUTPUT :
i.  Exact original  sequence
ii. Decoding time

START
i. Generate reverse Huffman tree

ITERATION
1.  for(sort the Huffman tree)
        for(decoding the swapping tree)
            if(frequency match)
              reconstruct the code
              else repeat the step
              End
2. File decompressor for files compressed with
HuffEnc.
3. End
```

4. Results and discussion

The benchmark DNA data[22] are used for testing this al described in different fig. from 16 to 44, throughput result is described in table 8, improvement results described in table 9 and comparison results describe in table 10 & 11.

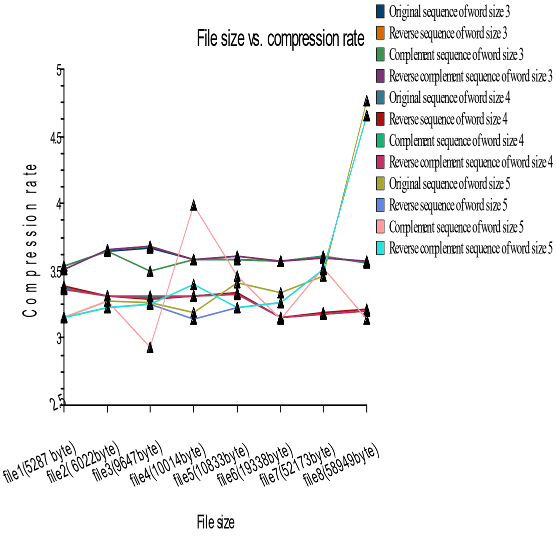


Fig.16 compression rate versus file size among original, reverse, complement and reverse complement sequences using Repeat technique of data set-I

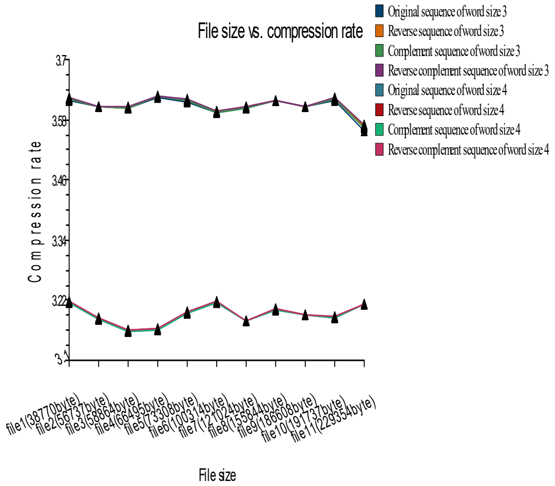


Fig.17 compression rate versus file size among original, reverse, complement and reverse complement sequences using Repeat technique of data set-II

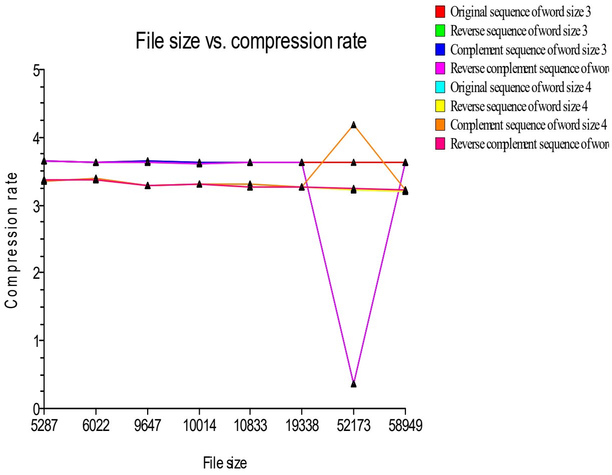


Fig.18 the file size versus compression rate of artificial data(data set-I)

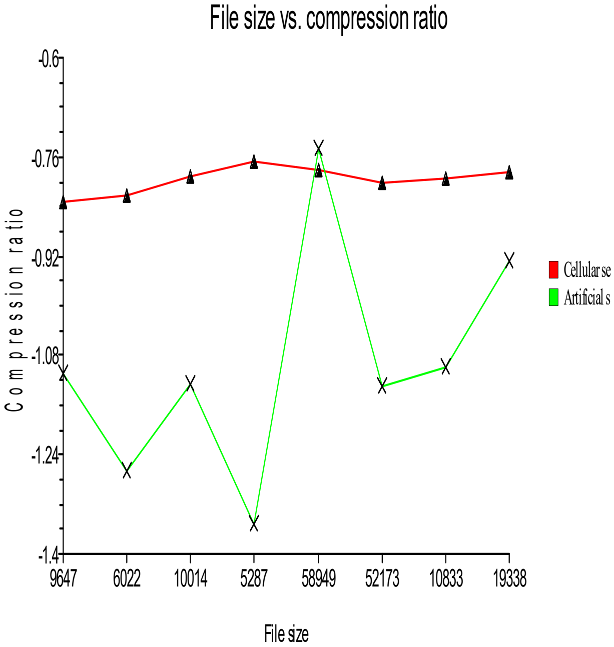


Fig. 19 compression rate vs file size of cellular DNA sequences and artificial data(data set-I)

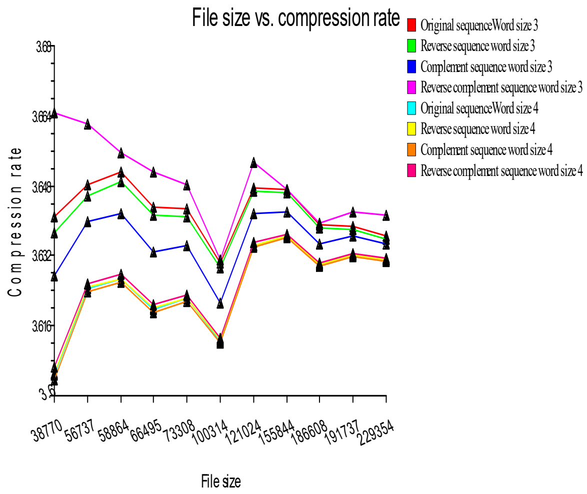


Fig. 20 compression rate vs. file size of artificial data (data set-II)

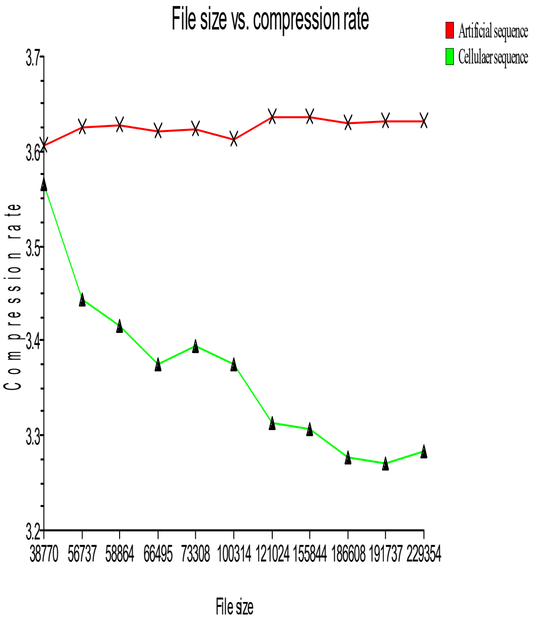


Fig. 21 file size versus compression rate of cellular DNA sequences and artificial data(data set-II)

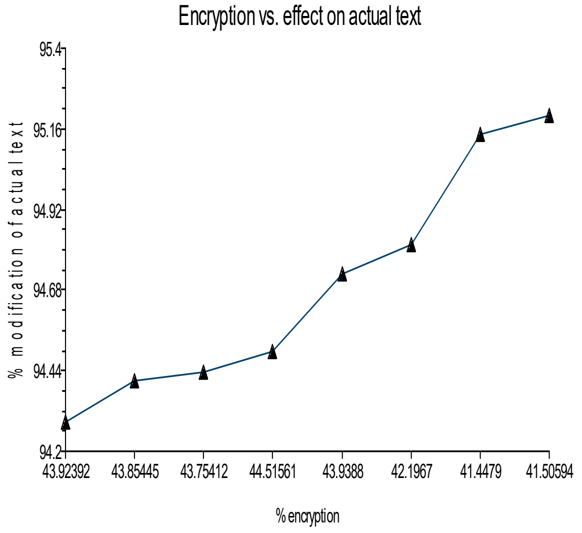


Fig. 22 % encryption vs. % modification for the actual text of data set-I using Repeat technique

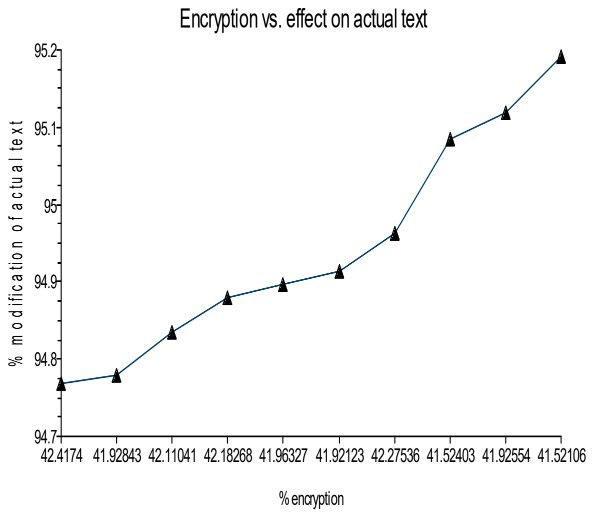


Fig.23 % encryption vs. % modification for the actual text of data set-II

Table 7 relative frequency

	Using Repeat			Using RHUFF			
Sequence Name	File size (byte)	Relative frequency for input file	Relative frequency for output file	File size (byte)	Relative frequency for input file	level	Relative frequency for output file
Hehc mvcg	229354	57339	1244	92077	1244	1	233
						2	389
						3	242
						4	252
Humd y strop	38770	9693	168	15585	166	1	40
						2	66
						3	41
						4	43
Celk 07e12	58949	14737	310	23585	310	1	61
						2	100
						3	66
						4	65
Atrd naf	10014	2504	30	4146	30	1	11
						2	17
						3	12
						4	11

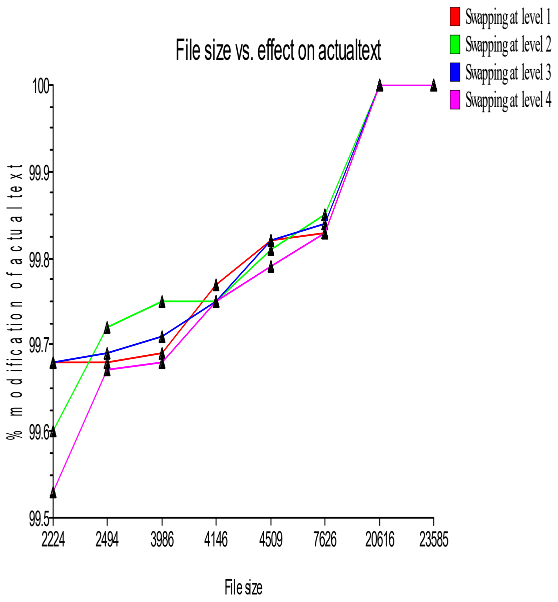


Fig.24 file size vs. % modification of the actual text of different level in process-I(data set-I)

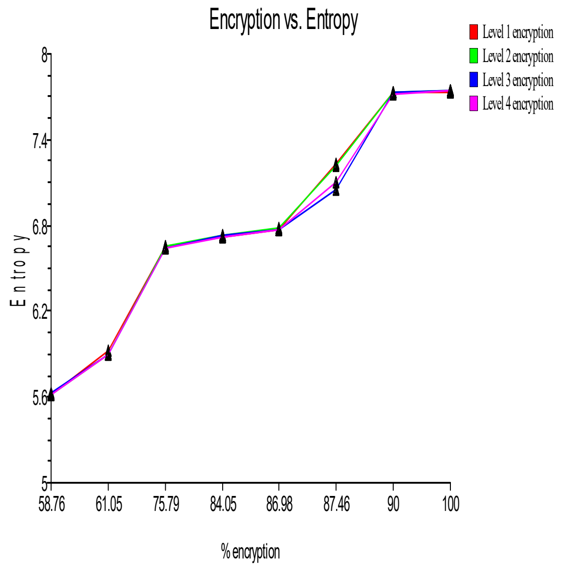


Fig. 25 file size vs. entropy of different level in process-I (data set-I)

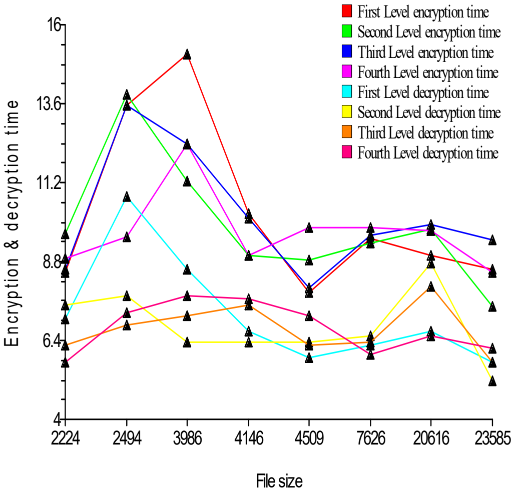


Fig. 26 different level time in encryption & decryption vs. file size in process-I (data set-I)

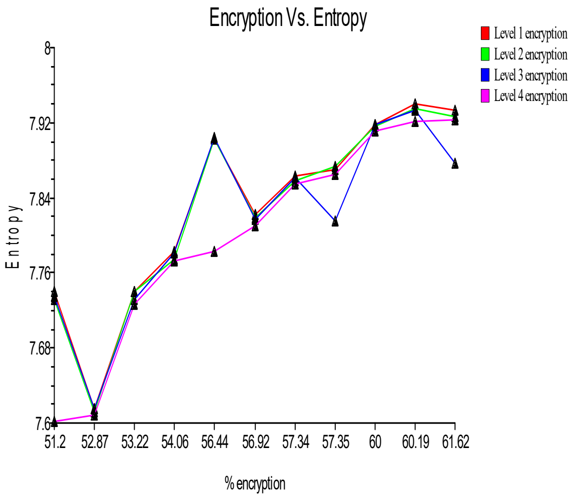


Fig. 28 % encryption vs. entropy of different level in process-I (data set-II)

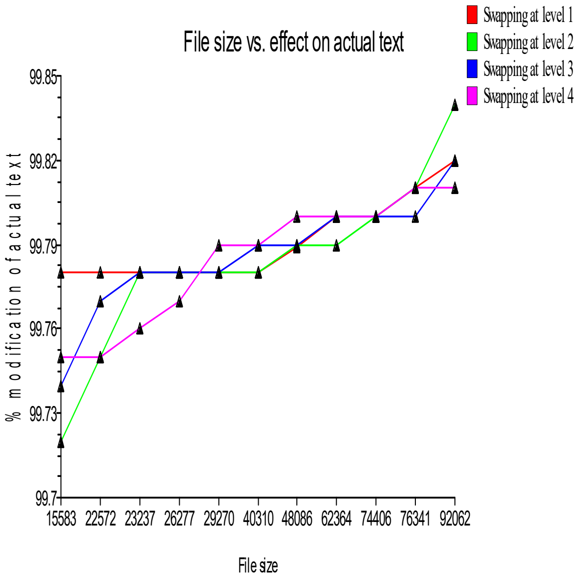


Fig. 27 file size vs. % modification of actual file of different level in process-I (data set-II)

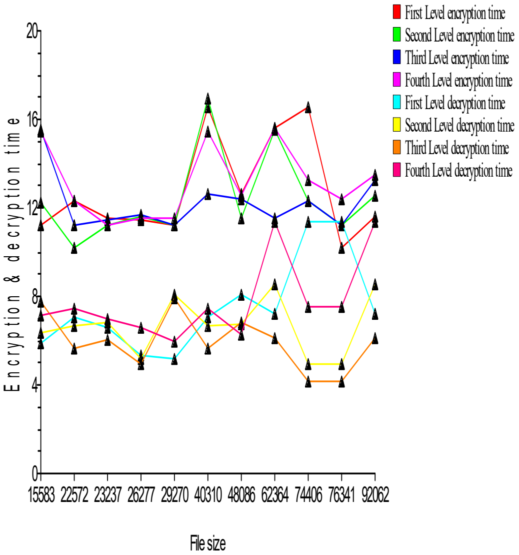


Fig. 29 different level time in encryption & decrypt vs. file size in process-I (data set-II)

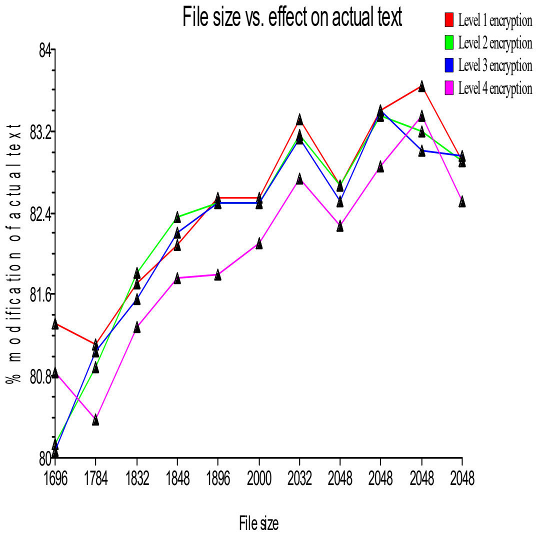


Fig. 30 file size vs. % modification of library file in process-I (data set-II)

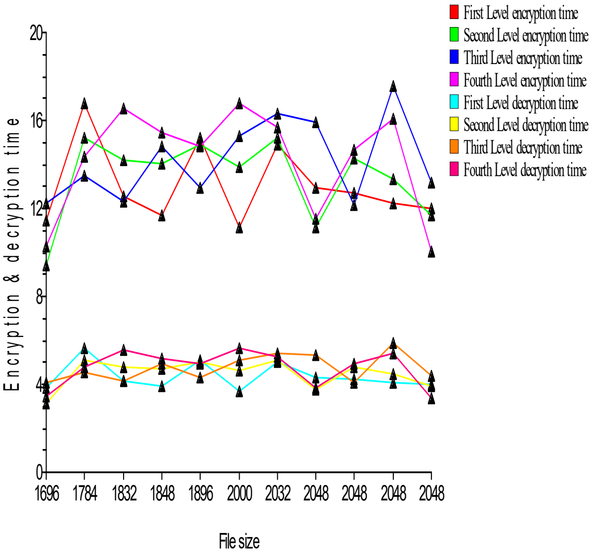


Fig.32 different level time in encryption & decryption vs. library file in process-I (data set-II)

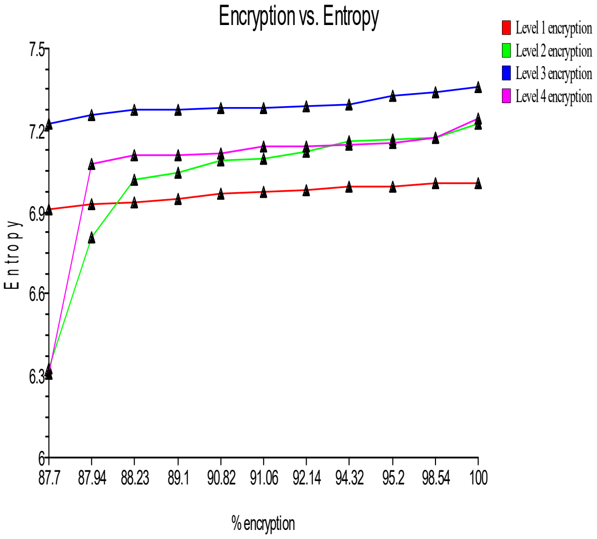


Fig. 31 % encryption vs. entropy of different level of library file in process-I (data set-II)

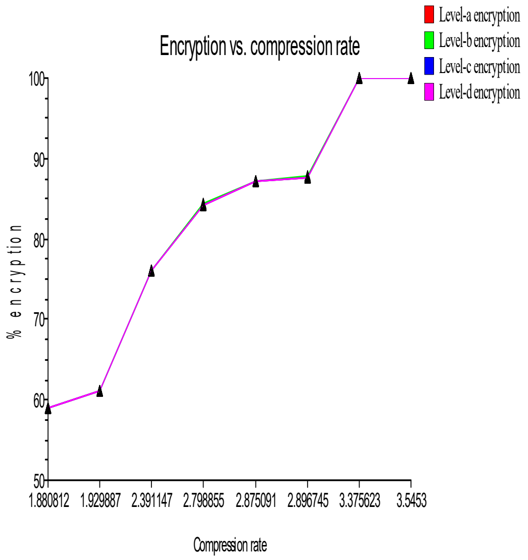


Fig. 33 % encryption vs. compression rate of different level of process-II(data set-I)

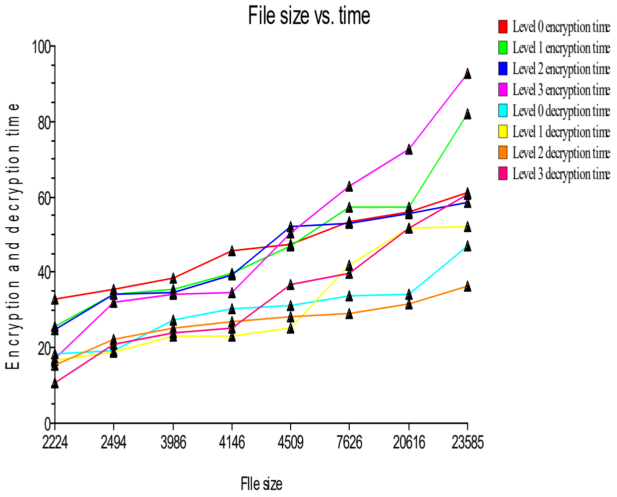


Fig.34 different level time in encryption & decryption vs. file size of process-II(data set-I)

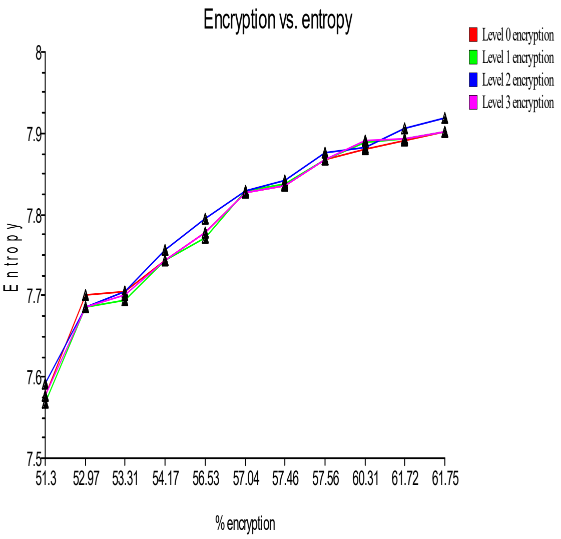


Fig.36 entropy vs. encryption of different level of process-II(data set-II)

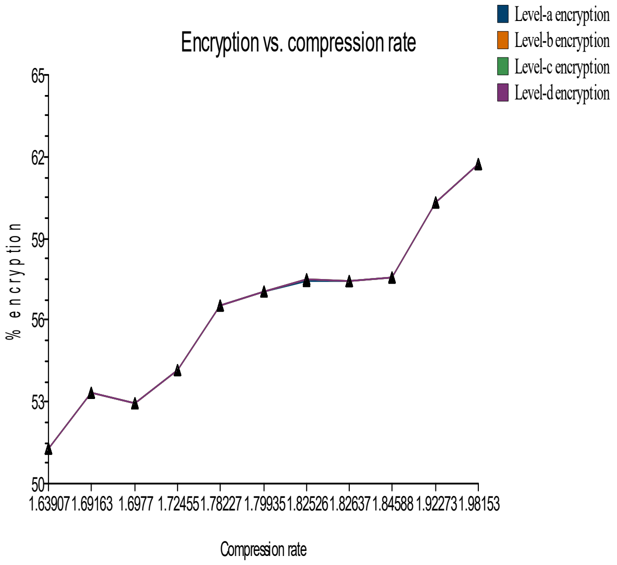


Fig.35 the % encryption vs. compression rate of different level of process-II(data set-II)

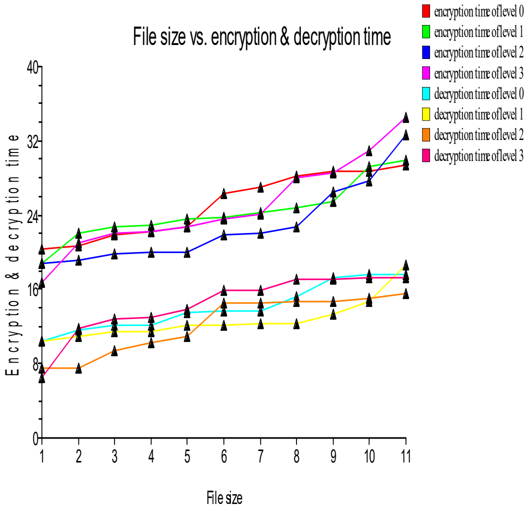


Fig. 37 different level time in encryption & decryption vs. file size of process-II(data set-II)

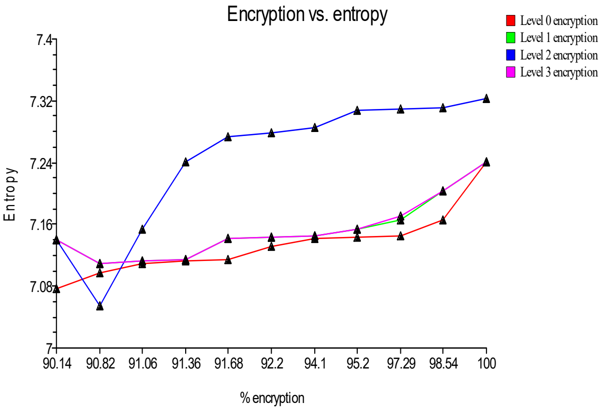


Fig.38 % encryption vs. entropy of different level of process-II(data set-II) on library file

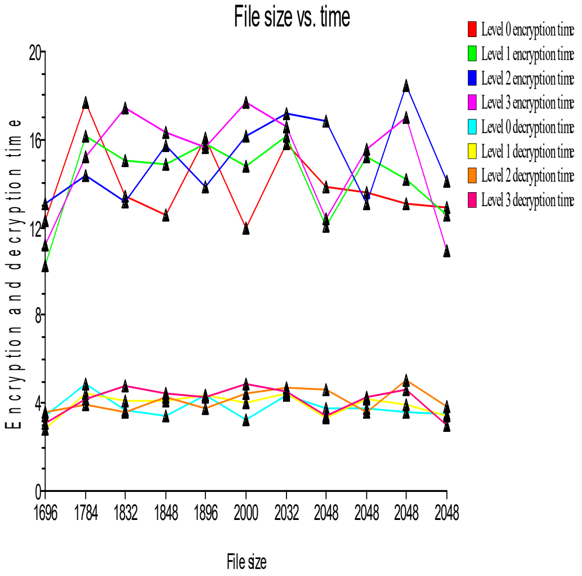


Fig. 39 different level time in encryption & decryption vs. library file of process-II(data set-II)

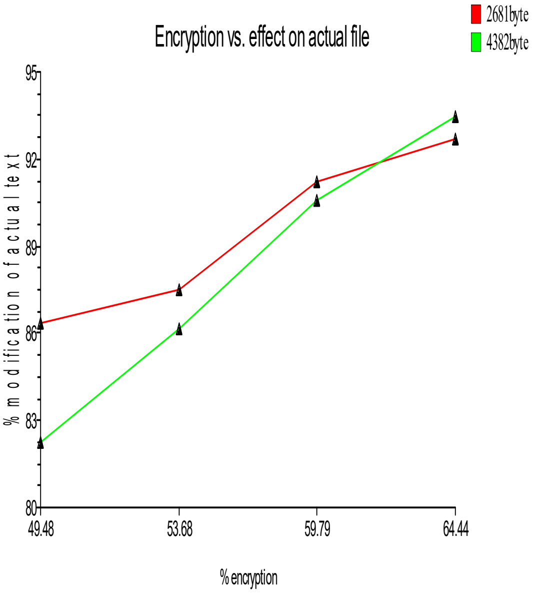


Fig. 40 % encryption Vs. % modification of original file

Table 8 data encryption & decryption throughput (Byte/See)

Data set	Process-I		Process-II	
	Encryption throughput(Byte/See)	Decryption throughput(Byte/See)	Encryption throughput(Byte/See)	Decryption throughput(Byte/See)
Data set-I	862.9911	869.3415	184.656	195.9417
Data set-II	3665.994	3672.331	1908.53	1916.386

Table 9 improvement of Repeat & Huffman Technique over Gzip

Data set	**Sequence**	**File Size byte**	**Using Repeat algorithm.**		**Using RHUFF algorithm.**		**Improvement over gzip**
			Compression ratio	Compression rate (bits /base)	Compression ratio	Compression rate (bits /base)	
D t se	atatsgs	9647	-0.65274	3.30548	0.016896	1.966207	**12.01**

	atef1a23	6022	-0.65659	3.31318	0.014945	1.97011	
	atrdnaf	10014	-0.65608	3.31216	-0.016577	2.033154	
	atrdnai	5287	-0.68262	3.36523	-0.005485	2.01097	
	celk07e12	58949	-0.60037	3.20073	0.007413	1.985174	
	hsg6pdgen	52173	-0.586108	3.17221	-0.007724	2.015449	
	mmzp3g	10833	-0.66491	3.32982	-0.001754	2.003508	
	xlxfg512	19338	-0.57741	3.15482	-0.007343	2.014686	
	Average			3.26920		1.999907	
Data set-II	MTPACGA	100314	-0.68710	3.374205	0.07705804	1.84588	**21.02%**
	MPOMTCG	186608	-0.63881	3.27763	0.13772186	1.72455	
	CHNTXX	155844	-0.65324	3.306486	0.18046251	1.63907	
	CHMPXX	121024	-0.65646	3.31293	0.15418429	1.69163	
	HUMGHCSA	66495	-0.68800	3.376013	0.10032333	1.79935	
	HUMHBB	73308	-0.69705	3.394118	0.03863153	1.92273	
	HUMHDABCD	58864	-0.70786	3.415738	0.1088611	1.78227	
	HUMDYSTROP	38770	-0.78271	3.565437	0.00923394	1.98153	
	HUMHPRTB	56737	-0.72162	3.443256	0.08736803	1.82526	
	VACCG	191737	-0.63534	3.270688	0.08681162	1.82637	
	HEHCMVCG	229354	-0.64130	3.282611	0.15114626	1.6977	
	Average			**3.365374**		**1.794213**	

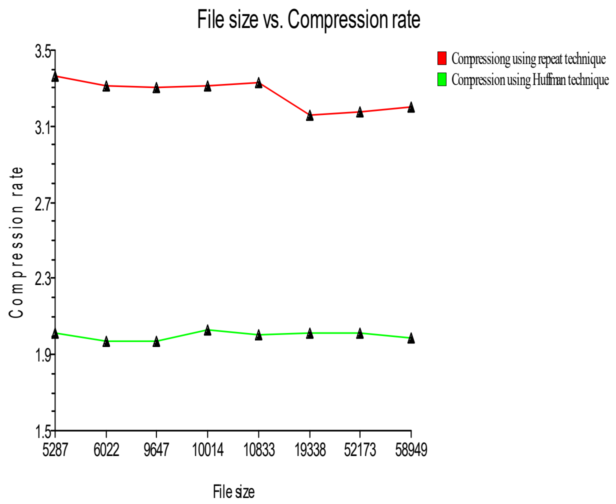


Fig. 41 compression rate vs. file size for data set-I using Repeat & Huffman's method

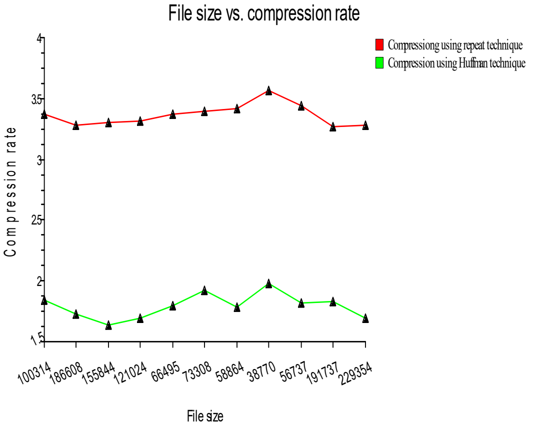


Fig.42 compression rate versus file size for data set-II using Repeat & Huffman's methods

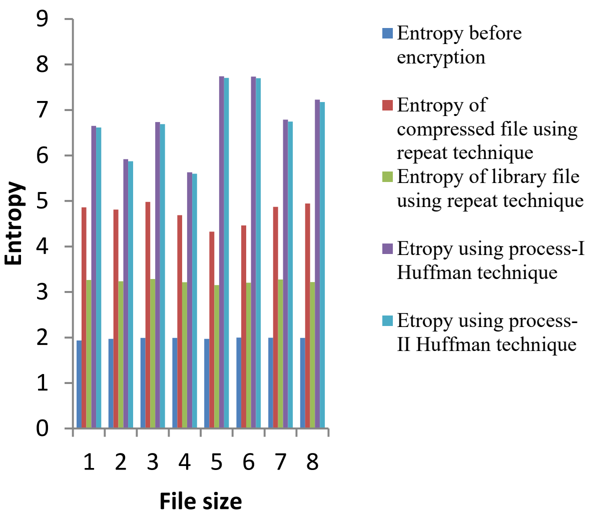


Fig. 43 entropy vs. file size for data set-I

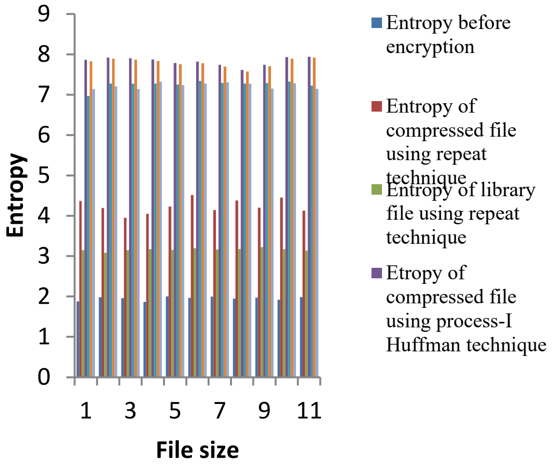


Fig.44 the entropy vs. file size for data set-II

Table 10 comparison our results with others standard results (data set-I)

Sequence	File Size byte	GZip	BZip2	Using RHUFF
atatsgs	9647	2.1702	2.15	1.966207
atef1a23	6022	2.0379	2.15	1.97011
atrdnaf	10014	2.2784	2.15	2.033154
atrdnai	5287	1.8846	1.96	2.01097
hsg6pdgen	52173	2.2444	2.07	2.015449
mmzp3g	10833	2.3225	2.13	2.003508
xlxfg512	19338	1.8310	1.80	2.014686
Average		2.109857	2.058571	1.999907

Table 11 comparison our results with others standard results (data set-II)

DNA sequence name	MTPACGA	MPOMTCG	CHNTXX	CHMPXX	HUMGHCSA	HUMHBB	HUMHDABCD	HUMDYSTROP	HUMHPRTB	VACCG	HEHCMVCG	Average
Size →	100314	186609	155844	121024	66495	73308	58864	38770	56737	191737	229354	----
off-line	1.915	1.986	1.998	1.902	1.5993	1.969	1.9740	2.068	1.983	1.907	2.015	1.937
dna0	1.993	1.956	1.675	1.832	1.3860	1.939	1.9441	2.003	1.969	1.842	1.881	1.856
dna1	1.995	1.959	1.676	1.833	1.3946	1.945	1.9512	2.005	1.976	1.844	1.881	1.860
dna3	1.873	1.931	1.622	1.678	1.3750	1.880	1.9130	1.953	1.919	1.764	1.846	1.795
gzip	2.2919	2.3288	2.3345	2.2818	2.0648	2.245	2.2389	2.3618	2.2662	2.2518	2.3275	2.272
gzip-4	1.8827	1.9727	1.9519	1.8635	1.7372	1.8963	1.9141	1.9473	1.9207	1.874	1.9817	1.903
gzip -9	2.232	2.280	2.291	2.220	1.551	2.228	2.209	2.377	2.232	2.190	2.279	2.189
lz (32K)	2.249	2.289	2.300	2.234	1.580	2.255	2.241	2.427	2.269	2.194	2.286	2.211
WinRAR	2.23	2.30	2.24	2.25	1.38	2.22	2.19	2.37	2.23	2.23	2.32	2.19
lz (1M)	2.285	2.326	2.352	2.276	1.513	2.286	2.264	2.432	2.287	2.245	2.344	2.237

arith	1.880	1.984	1.957	1.867	2.001	1.969	1.999	1.949	1.972	1.919	1.985	1.952
arith+ (32k)	1.873	1.972	1.957	1.866	1.488	1.913	1.951	1.948	1.943	1.862	1.985	1.887
arith+ (1M)	1.873	1.961	1.956	1.866	1.438	1.911	1.950	1.948	1.942	1.862	1.985	1.881
normal PPMD+	2.018	2.075	2.062	1.977	2.077	2.116	2.130	2.237	2.130	2.002	2.053	2.079
adapted PPMD+	1.872(1)	1.966(2)	1.934(1)	1.840(1)	1.694(11)	1.921(2)	1.948(2)	1.921(1)	1.932(2)	1.910(2)	1.965(3)	1.900
PPMD+ escape	1.869(3)	1.964(3)	1.935(3)	1.839(3)	1.514(11)	1.923(3)	1.938(3)	1.931(3)	1.926(3)	1.908(3)	1.959(3)	1.882
normal CTW	1.902	1.989	1.974	1.879	1.376	1.917	1.909	1.960	1.922	1.897	1.997	1.883
CTW-4	1.866	1.962	1.933	1.838	1.363	1.892	1.897	1.920	1.913	1.857	1.958	1.854
Compress	2.12	2.20	2.19	2.09	2.19	2.20	2.21	2.23	2.23	2.14	2.20	2.18
bzip	2.1225	2.1701	2.1845	2.1218	1.7289	2.1481	2.0678	2.1802	2.0944	2.0949	2.1685	2.098
bzip-4	1.9847	2.0117	2.009	1.9667	1.8697	1.9957	1.9921	2.0678	2.0045	1.952	2.0091	1.987
bzip-2	2.12	2.17	2.18	2.12	1.31		2.07	2.18	2.09	2.09	2.17	2.05
ac-o2	1.8723	1.9654	1.9333	1.8364	1.9377	1.9176	1.9422	1.9235	1.9283	1.904	1.9647	1.920
ac-o3	1.8761	1.9689	1.9399	1.8425	1.9416	1.930	1.9466	1.9446	1.9352	1.906	1.9619	1.926
RHUFF	1.84588	1.72455	1.63907	1.69163	1.79935	1.92273	1.78227	1.98153	1.82526	1.826	1.6977	1.794
dna2	1.869	1.927	1.616	1.673	1.3668	1.867	1.9036	1.932	1.910	1.763	1.848	---
GenCompress	1.8624	1.9058	1.6146	1.673	1.0969	1.8204	1.8192	1.9231	1.8466	1.7614	1.847	---
CTW+LZ	1.8555	1.9000	1.6129	1.6690	1.0972	1.8082	1.8218	1.9175	1.8433	1.7616	1.8414	---
DNACompress	1.8556	1.8920	1.6127	1.6716	1.0272	1.7897	1.7951	1.9116	1.8165	1.7580	1.8492	---
Biocompress-2	1.8752	1.9378	1.6172	1.6848	1.3074	1.88	1.877	1.9262	1.9066	1.7614	1.848	---

This algorithm takes variable length sub-sequence size, starting from size 3 because less than 3 sub- sequence size is meaningless. Find out the results on normal & artificial sequences as well as reverse, genetic palindrome and invert complement of the sequences.

For **cellular sequences**, the data set-I results are presented in graphically in fig. 16 and data set-II is presented in fig.17. The fig. 16 and 17 shows that the compression rate is dependent on word size and independent of the file size. In case of data set-I, the average compression rate is 3.26920 bits/base and in data set-II the average compression rate is 3.190246 bit/base where sequence orientation is complement and word size is 4. If the word size increases simultaneously the compression rate is increased. The word size increases from 3 to 4 also improve compression rate/ratio, the compression rate decrease from 3.57189 bits/ byte to 3.327587 bits/base i,e 6% decreased, the library file increases about three to four times of word size 3 to 4 library size. The word size increases from 3 and onward, processing time is highly increases. So, the compression rate is minimum when the sub sequence size is 3, is better than sub sequence size is 4. For both the data sets, the fig. 16 & 17 shows that the graph nature is heterogeneous because sequences come from different species. In comparison library file size with compressed file size is too small. After applying Huffmans' technique the average compression rate of data set-I is 1.9999 bit/base and in case of data set-II the rate is 1.7942 bit/base. The result of data set-I is presented in fig.-41 and data set-II result is presented in fig. 42, shows that the increase in file size decreases the compression rate and Repeat plus modified Huffman technique is more acceptable.

For **artificial data**, the results of data set-I are presented graphically in fig. 18 and data set-II is presented in fig. 20. The dependence of the word size and the file size on the compression rate is shown in fig. 18 & 20. We can get minimum compression rate when the sub sequence size is 3 and reverse is the sequence orientation and compression rate is 3.22323bit/base and 3.62444 bit/base in case of data set-I and II. As the sequences are generated randomly, the fig. 18 & 20 shows graphical nature is homogeneous. Now draw a fig. on the basis of data set-I is fig. 19 and data set-II present in fig. 21 on the basis of cellular sequences versus artificial data, getting district fig. 19 & 21 where as graph characteristic can be seen by naked eye. The random sequence is unstructured and cellular sequences are non random, have logical organization, systematic and structure. It is also seen that in case of cellular sequence the library file varies in size whereas it is constant in size in case of artificial sequence. In comparison library file size with compressed file size is too small.

It is observed that the compression ratio in case of artificial sequence calculated follow by the formula {1-Output/input} and in case of cellular sequence compression ratio is calculated follow by the formula {1-2*Output/input}, where number of bit is the output file size. The result shows that the compression rate and compression gain is inversely proportional.

We have calculated percentage modification of actual text and percentage of encryption, shown in fig. 22 & 23 for the data set-I & II , on the basis of three/four characters secret key. It is observed that in data set-I the average 41%-43% & data set-II the average 41%-42% of actual text encryption and 95% modification is observed in actual file.

It is proved that after compression the two or three times increased the entropy as shown in fig. 43 & 44. The randomness is increased in both the output file of library and compressed, as a result it is hard to hack the sequence by the hacker. Data set-I is presented fig. 43 and data set-II is presented in fig. 44 shows the entropy of compressed file and library file before and after compression.

This repeat compression technique produces two separate text file over DNA sequence and applied modified Huffman method on it. If changing the level, percentage of encryption and effect on actual file is observed.

The table 7 shows the relative frequency for the different input file and encrypted output file, also shows the ratio is maintained in between input and output is 48:1 that mean possibility of frequency attack is minimized. The percentage of encryption did not vary significantly with compression rate, presented the data set-I result in fig. 24 and data set-II result is presented in fig. 27. The percentage of encryption varies due to sub-sequence/word consisting different characters with respect to file size & % modification of actual text also data set-I is presented in fig. 24 and data set-II is presented in fig. 27. The major effectiveness is found in the sequences. But decrypt the sequences without actual sub-sequence/word value or entered an incorrect sub-sequence the sequence will be different.

This is to observe that in case of repetition of input text having same frequency counting is not more. For key purpose use level number. The same frequency character ratio is approx 4:1 with respect to input and output text, in that situation frequency analysis attack is reduced. But if decrypt is done without applying key value or entered an incorrect key the text will be different.

In process –I : The result shown for data set-I & for data set-II that above 99% modification of actual file can be observed in data set-I & II by only when encrypting 58% to 100 for data set-I & 51% to 61% for data set-II of the actual file.

If top level is interchanged, we can get the higher effect on original file on the basis of Lavenstein distance highest level and in case of lower level, the effect is proportional. This result of data set-I is presented in fig.24 to 26 and data set-II result is presented in fig.27 to 32 on the basis of different file size, % effect on the original file, % encryption and compression rate. The encryption depends on the effectiveness of the output text. If the input file size is increased, the percentage effect on actual file also increases

In process-II : now taking the same text which was considered for previous experiment and do the same job i.e. first calculate frequency of each character and then measure the relative frequency to analyze probability of attack. Before encryption the text use two key values in binary form. These are actually specified two nodes at different levels. The results are shown in fig.33 for data set-I & fig.34 to 38 for data set-II.

Now measure relative frequency in case of encrypted text in the same manner when calculating this for input text. For encrypt text the redundancy value is 389. But in case of input text this value is 1244. The attack is low when output and input sequences relative frequency ratio is nearly 4:1. The results in the table shown that different input text and encrypted output value relative frequency are different at different level.

Now using appropriate key, we can generate the actual file. Without proper key value or entered an incorrect key the message will different.

The results of data set-I is presented in graphically in fig. 33 and data set-II is presented in fig. 34; shown that 99% original text will affect and reduce percentage of encryption when swapping at lower level. If encrypt the sequence on the basis of two binary values as a key getting higher security. Also increase the encryption & efficiency of the sequences.

In process –III : Testing purpose use two text files of size 2681 & 4382 bytes, find out the actual effect on the text by interchanging the different nodes based on calculating % encryption and % modification on the actual text, result shown in table 4.24.

Based on data, drawn a graph of fig.38, observed that 88% actual text effected when 57% encryption is done on the actual text. Since number of distinguishable words are huge so it need more %of encryption, but in case of considering character if apply minimum 42% encryption of real text it will affect more than 90% of real text case.

The throughput of encryption and decryption is shown in table 8.It is seen that encryption throughput is higher than decryption throughput, so the throughput of data set-II is better than data set-I. We may conclude the encryption of process-I is much better than process-II.

Table 9 shows the improvement results on Gzip technique , the improvement is 12.01% on data set-I and 21.02% on data set-II, graphically shown in fig. 38.

The encryption & decryption time is presented in graphically of fig. 26,29,32,35 & 35 of process-I & II. The decryption is always less than encryption time. The time in decryption and encryption is proportional to the size of file. The above process of encryption is very useful from security point of view because our principal and AES/DES/RSA/DNA encryption are same.

The Table 10 & 11 shows the comparison result on standard available techniques. This combine technique RHUFF (Repeat + Modified Huffman's) shown by red color is better than 23 nos. techniques (mention 1st coloum of table 11), also below RHUFF red color results are better than column 1 standard techniques.

This RHUFF technique is far better than gzip techniques with respect to compression rate and information security.

The gzip technique is Lempel-Ziv "LZ"+Huffman [6] without any security concept. This method is the combination of method of Repeat with modified Huffman for compression as well as security.

Important observations are :

- The cellular DNA sequences have logical organization, structure, systematic and non random where as artificial data are random and unstructured.
- The substring measure end to end different from 2 to 5 and no match is discovered if the substring measure end to end becoming more than 6.
- The cellular DNA sequence encode amino acid/protein that why sub-sequence of repeat/reverse/palindrome/genetic complement are found in the original sequence, more exact match are found in the repeat search method, other orientation the exact match are found in less number over repeat method.
- The results are showing that cellular DNA sequence are reasonable compressible in any orientation (cellular DNA sequence, reverse sequence, complement sequence and reverse complement sequence) result is homogeneous in nature also where as artificially (random sting) generated sting of same length compression rate is heterogeneous in nature.
- All compression rate are similar also suggests a highly similar sequences
- This technique are also apply on corresponding other orientation of cellular DNA sequences like Reverse, Complement & reverse complement of cellular DNA sequence, the better result found on normal i.e cellular DNA sequence performance is better.
- Best encryption result is found when modified Huffman's technique apply on library file
- The output text having ASCII code and non match DNA bases, containing more than four character than the input text. So, after first pass Huffman's and two bit encoding technique is easily applicable and overcome the drawback of using Huffman's & two bit encoding technique.

5. Conclusion

The compression rate of Cellular DNA sequences lies between 3.25 bits/bases & 3.3 bits/base, where as in case of artificial data value lies in between 3.3 bits/bases & 3.5 bits/base. This combine technique, the compression rate lies between 1.69 bit/base to 1.92 bits/base. In process-I & II, the percentage of encryption rate is 81.76, 56.23 and 81.89, 56.35 with respect to data set-I & II. This compression and percentage encryption rate is better in my earlier paper [25]. The nature of graph is homogeneous in case of cellular sequence where as artificial sequences the nature of graph is heterogeneous in nature.

The lowest compression rate is found when repeat technique run on the bench mark DNA sequences in complement orientation and the sub sequence size is 4, lowest compression rate is 3.26920 bit/base. Also the output is secured than input sequences in transmission point of view. The output contains 256 characters include a, t, g & c. So, the resultant test is highly secure than input text. The substring measure end to end different from 2 to 5 and no match discovered if the boat able to go under water line measure end to end becoming 6 or more.

The results show that the compressed pattern matching algorithm for DNA order is in competition among the best algorithm.

The results are showing that the compression rate & ratio are different to each other in case of reverse, complement and reverse complement, it make certain that the point of comparison facts are a part of same family.

If consider library with compress file, the overall compression rate is slightly increase. So better result find only when compression rate is calculated on compressed file size. In comparison to compressed file the library life is too minuscule.

In case of selective encryption of compressed text, as compression follows the encryption process, the effect on text based on statistical properties, plain text will not be possible because of the reduction of redundancy due to process of compression. This approach of selective encryption has got some advantage due to constraint of the bandwidth in network before communicating and also it needs to be encrypted so confidentiality is maintain or to protect the digital rights.

This is a static Huffman coding method applied on compressed text (1st pass output use here as input) based on selection encryption and effectiveness compare by dissimilarity in original file. If anyone can decrypt without key, the cipher text resistance from the attacks based on statistical property of the plain text

For applying selection encryption, the result found that the encryption effectiveness is increases at where interchanging is done. So the attack is very low on the basis of probability of frequency analysis.

Another conclusion is that if consider word instead of characters then workspace is increased. But in this case, it is found the percentage of encryption is 57 then the actual effect on the text is 88%, so the different number of word is more and their frequency is less on character frequency. It is observed that relative frequency ratio of input text and encrypt text is nearly 4:1, so same frequency character ration in between input and output is approx 4:1.

In case of word consideration word's frequency are high but other word have very lower frequency. These lower frequency words are representing by higher bit. So percentage of compression decreases. In terms of security word encryption is more effective than character encryption. In case of character encryption, There is only 256 characters are available and since workspace is short. So here is a possibility to break the security. But in case of word encryption, numbers of distinguishable words are huge, not known by all, so that workspace is also increased and breaking the security is not possible.

This techniques provide the better facts safety than other techniques. Also biological order compression is an useful apparatus for recovering useful knowledge from biological orders. This lossless compression technique achieve a moderate compression rate & ratio than that of existence DNA order compression algorithm and provides the better knowledge safety with encryption least decompression time, the execution time of this algorithms is fraction of second and optically points the different in between cellular DNA order and not natural DNA of equal measure end to end.

The process-II encryption techniques requires optically higher number of bits for encoding the data and repeat technique requires optically less number of bits, indicating process-II Huffman's requires transmission of higher bandwidth.

The entropy increased from 1.98 to 7.90 per byte of encryption. The information entropy is measure by a degree of randomness. Randomness is an important and desirable property of compression-encryption algorithm. It is impossible to attack the output file by the attacker because the library & compressed file produced high degree of randomness.

References

[1]Deloula Mansouri, Xiaohui Yuan and Abdeldjalil Saidani, A New Lossless DNA Compression Algorithm Based on A Single-Block Encoding Scheme, Algorithms, pp 1-18,2020

[2]Deorowicz, S., and Grabowski, S., 2011, Robust relative compression of genomes with random access, Bioinformatics, 27(21), 2011, pp 2979–2986.

[3] Schrodinger, E. What is Life; Cambridge University Press: Cambridge, UK, 1944.

[4]B. Saada," DNA Sequence Compression Technique Based on Nucleotides Occurrence" Proceedings of the International MultiConference of Engineers and Computer Scientists 2018 Vol I IMECS 2018, March 14-16, 2018

[5]A. Jahaan, Dr. T. N. Ravi, Dr. S. Panneer Arokiaraj" Bit DNA Squeezer (BDNAS) : A Unique Technique for Dna Compression" International Journal of Scientific Research in Computer Science, Engineering and Information Technology,pp-512-517,2017

[6] Nour S. Bakr, Amr A. Sharawi, 'DNA Lossless Compression Algorithms: Review ', American Journal of Bioinformatics Research, 2013 pp 72-81

[7] Kirill Kryukov, Mahoko Takahashi Ueda, So Nakagawa and Tadashi Imanishi, 'Nucleotide Archival Format (NAF) enables efficient lossless reference-free compression of DNA sequences' ,Bioinformatics, pp 1-3,2019

[8] William stallings, "cryptography and network security principles and practise", 5th edition 2011

[9] A. Amir, and G. Benson, "Effcient two-dimensional compressed matching", In Proc. CC'92, pp. 279-288,2002.

[10] C. E. Shannon, "Communication theory of secrecy systems," Bell Systems Technical Journal, v. 28, October 1949, pp. 656-715.

[11] D. A. Huffman, "A method for the construction of minimum-redundancy codes,"Proc. IRE, vol. 40, pp. 1098-1101,1952.

[12]Kryukov K, Ueda MT, Nakagawa S, Imanishi T (2019) Nucleotide Archival Format (NAF) enables efficient lossless reference-free compression of DNA sequences. Bioinformatics I, 2019, pp 1–3

[13]K. Kryukov, M. T. Ueda, S. Nakagawa, and T. Imanishi, ``Nucleotide archival format (NAF) enables efficient lossless reference-free compression of DNA sequences,'' *Bioinformatics*, vol. 35, no. 19, pp. 3826-3828,Oct. 2019.

[14]Diogo Pratas,Morteza Hosseini,Jorge M. Silva and Armando J. Pinho, A Reference-Free Lossless Compression Algorithm for DNA Sequences Using a Competitive Prediction of Two Classes of Weighted Models, Entropy, pp 1-18,2019

[15] Xin Chen, Ming Li, Bin Ma and John Tromp, DNACompress: fast and effective DNA sequence Compression, Bioinformatics Applications Note, Vol. 18 no. 12 2002, Pages 1696–1698

[16]Maleeha Najam, Raihan Ur Rasool , Hafiz Farooq Ahmad,Usman Ashraf, and AsadWaqarMalik, Pattern Matching for DNA Sequencing Data Using Multiple Bloom Filters, BioMed Research International, pp 9,2019

[17] Ashutosh Gupta and Suneeta Agarwal, A Novel Approach For Compressing DNA Sequences Using Semi-Statistical Compressor, International Journal of Computers and Applications, Vol. 33, No. 3, 2011,pp 1-7

[18] Nahida Habib, Kawsar Ahmed, Iffat Jabin and Mohammad Motiur Rahman, Modified HuffBit Compress Algorithm – An Application of R, Journal of Integrative Bioinformatics, pp 1-13. 2018

[19] Jahaan A, Ravi TN, Panneer Arokiaraj S (2017) Bit DNA Squeezer (BDNAS): a unique technique for DNA compression. Int J Sci Res Comput Sci Eng Inf Technol 2, 2017, pp 512–517

[20]Deloula Mansouri, Xiaohui Yuan and Abdeldjalil Saidani, " A New Lossless DNA Compression Algorithm Based on A Single-Block Encoding Scheme"Algorithms 2020,pp 1-18

[21]Giovanni Manzini and Marcella Rastero, " A Simple and Fast DNA Compressor", Dipartimento di Informatics, University del Piemonte Orientale,Italy, February 17,2004

[22] S. Grumbach and F. Tahi, "A new challenge for compression algorithms: Genetic sequences," J. Inform. Process. Manage., vol. 30, no. 6, pp. 875-866, 1994.

[23] Md. Syed M.H. ,D.Roy , S.Saha, A Compression Algorithm for DNA Sequences based on repeat sequences and its applications in Genome comparison with Information security, International Journal HIT Transactions on ECCN (Electronics, Communication, Computer & Networks), 0973-6875, Vo. 2, No. 6, April-June 2007, INDIA

[24] Syed Mahamud Hossein,P. K. Das Mohapatra and Debashis De," DNA Compression & Security based on Reverse Technique", Journal of Bionanoscience, 2013, pp 1-5

[25] Syed Mahamud Hossein, Debashis De , (Senior Member, IEEE),Pradeep Kumar Das Mohapatra, Sankar Prasad Mondal,Ali Ahmadian, Ferial Ghaemi and Norazak Senu "DNA Sequences Compression by GP^2 R and Selective Encryption Using Modified RSA Technique" IEEE Access,2020,pp 76880-76895