Genome sequences compression & security by subsequence method

Syed Mahamud Hossein[1]
[1]Lecturer (Selection Grade), Deptt. of CS&T, I.C.V.Polytechnic,Jhargram

Susanta Mandal[2]
[2] Student, Jalpaiguri Govt. Engg. College, Jalpaiguri.

Abstract

For genetic sequences, a lossless compression approach based on palindrome and exact repeat search is presented. The algorithm's compression results demonstrate that one of the primary hidden regularities in DNA sequences is exact duplicates, or palindromes. Based on repeat and palindrome substring pairings, the suggested DNA sequence compression approach generates an online library file that functions as a lookup table. Beginning with 33(!), the equivalent ASCII character is used to replace the repeated & palindrome substring. The user determines the length of this substring. The most difficult issue to safeguard data from unauthorized users is information security. This suggested approach might keep hackers away from the data. By employing ASCII code and an online library file serving as a signature, it can offer data security. The efficiency of their applications will be improved by compressing the genomic sequences. Benchmark DNA sequences, reverse, complement, and reverse complement benchmark DNA sequences, as well as synthetic DNA sequences, are used to test this approach. An approximate compression rate of 3.851273 bits/base can be achieved by the technique.

Keywords: DNA, Compression and Security

Introduction

One practical method for extracting information from biological sequences is biological sequence compression. Fundamental issues about the properties of these sequences emerge as more and more whole genomes of prokaryotes and eukaryotes become available, and the human genome project is soon to be finished. The compressibility of DNA sequences is one such fundamental subject that we investigate. Order is embodied in life. It is neither random nor chaotic [1]. Therefore, we anticipate that the DNA sequences encoding life will not be random. They ought to be quite compressible, of course. Strong biological evidence also backs up this assertion: DNA sequences are known to contain a large number of repeats and palindromes, particularly in higher eukaryotes. Many important genes, such as rRNAs, are known to have multiple copies. Only roughly a thousand fundamental protein folding configurations are thought to exist. Additionally, it has been hypothesized that genes occasionally repeat themselves for evolutionary or purely "selfish" reasons. All of them provide concrete evidence that DNA sequences ought to be fairly compressible. Only four nucleotide bases—a, c, g, and t—make up DNA sequences, and each base may be stored in eight bits. Even though all of these compression tools use universal compression methods, they all expand the file by more than 8 bits per base when typical compression software is used, such as Unix's "compress" and "compact" or MS-DOS's "pkzip" and "arj" programs. These programs are made to compress text [2], but DNA sequences have considerably more subtle regularities. Our goal is to investigate these nuances in DNA sequences. We will introduce an exact matching-based DNA compression algorithm that produces the greatest compression outcomes on common benchmark DNA sequences. Finding every exact repetition and palindrome in a lengthy DNA sequence is not an easy task, though. To locate approximation repeats and palindromes that are best for compression, this approach takes a long time (it's basically a quadratic time search or even longer). It's still difficult to achieve the greatest compression ratio and high speed at the same time. DNA sequences that have been proposed For benchmark DNA sequences, compression concurrently outperforms all other compression programs in terms of speed and compression ratio. The suggested approach is divided into two stages: i) locate all exact repeats and palindromes; and ii) encode regions that are repetitions and palindromes and non-repeat regions. Our exact repeat & palindrome search engine is designed for a quick and sensitive homology search [3]. DNA sequence compression is a very difficult endeavor. The fact that no commercial file-compression software can compress the benchmark DNA sequences used in this paper is evidence of this. Prior research elsewhere has generated a number of compression techniques specifically tailored for DNA sequences. In order to achieve the best compression results on common benchmark DNA sequences, we will introduce a DNA compression algorithm that is based on repeated and palindromic substrings and corresponding repeated and palindromic substrings placed in library files. This algorithm creates a dynamic Lookup Table and places ASCII characters in the right places in the source file. We will go over the technique's specifics, present experimental findings, and contrast them with the outcomes of the most successful compression algorithm for DNA sequences (gzip-9).

If the file segment at the end does not precisely match the pre-coded table, the base segment is still present in the Lookup Table Encoding time "Sub-sequence size-1." When we are unable to locate any arrangement in the lookup table, we simply put the original segment into the target file [4]. This approach increases the likelihood of compaction while reducing this issue. We discover that the compression rate and ratio provide different orientations for the input sequences, including the reverse, complement, and reverse complement. However, experimental results employing other orientations as input sequences indicate no discernible modifications. Additionally, we can determine the compression ratio and rate of an artificial DNA sequence that is randomly created and of equivalent length. Examine each outcome in relation to the others.

Methods

The paper describes how to generate a substring from an input sequence and evaluate a method.

Encoding algorithm

Algorithm for compression with repeated & palindrome sequence using variable length

```
0.  CH=54, CH1=32
1.  Input the compression length l.
2.  Input the input file name FNAME.
3.  Suppose FNAME is a.txt then create a file name FLIB by appending 'lib' to the end of the
    FNAME like in this case alib.txt. FLIB will store the ascii character and its corresponding word
    which it replaces in the compressed file.
4.  Suppose FNAME is a.txt then create a file name FCOM by appending 'com' to the end of the
    FNAME like in this case acom.txt. FCOM will store the compressed file.
5.  Create an empty file TEMP.
6.  MAX=0
7.  MWORD=NULL
8.  Extract a word of length L from FNAME which only consists of a, t, g, c. Check whether it exist in
    TEMP or not. If it exist go to step 9 else go to step 10.
9.  If it is end of file go to step12 else go to step 8.
10. Append this word to TEMP. Count the number of times this word is repeated in the file. If it is
    greater than MAX do MWORD=this word and MAX=the count of this word.
11. If it is end of file go to step 12 else go to step 8.
12. If MAX >1 do step 13 to 17
13. CH=CH+1.if CH=a/t/g/c CH=CH+1
14. If CH=0 do CH1=CH1+1 and CH=54
15. If CH1==32 append to FLIB CH and MWORD else append to FLIB CH1 and CH and MWORD
    in this order.
16. Replace every word in FNAME which matches MWORD with the corresponding ascii character.
    Store it in FCOM.
17. Replace the content of FNAME with FCOM.
18. IF MAX>1 go to step 5
19. Remove FNAME and TEMP.
```

Decoding algorithm

DECOMPRESSION ALGORITHM

```
1. We accept the compressed file FCOM.
2. Suppose FCOM is 'acom.txt' we will write library file name FLIB as 'alib.txt' and original file
   name FNAME as 'a.txt'.
3. Read the compressed file FCOM character by character
4. If the character is a/t/g/c copy it to FNAME.
5. If the character is not a/t/g/c we will find the word matching to the character in FLIB and write that
   word in FNAME.
6. Do step 3 to 5 until end of file is reached.
7. Remove FCOM and FLIB
8. FNAME holds the original decompressed file.
```

Experimental Results

This algorithm tested on standard benchmark data used in [5]. For testing purpose use eight types of data.

			Cellular DNA Sequences					Artificial DNA Sequences				
Orientation	Sequence Name	Base pair/File size	Reduce file size Byte(C)	Compression ratio	Compression rate(bits /base)	Compare with gzip-9	Improvement	Reduce file size Byte(C)	Compression ratio	Compression rate(bits /base)	Compare with gzip-9	Improvement
Normal orientation	atatsgs	9647	4363	3.6181	-0.8090	4.6260	23.05%	4355	3.6114	-0.8057	4.6260	21.94%
	atef1a23	6022	2670	3.5469	-0.7734			2702	3.5895	-0.7947		
	atrdnaf	10014	4418	3.5294	-0.7647			4514	3.6061	-0.8030		
	atrdnai	5287	2275	3.4424	-0.7212			2387	3.6118	-0.8059		
	celk07e12	58949	26483	3.5940	-0.7970			26735	3.6282	-0.8141		
	hsg6pdgen	52173	23455	3.5964	-0.7982			23515	3.6056	-0.8028		
	mmzp3g	10833	4815	3.5558	-0.7779			4921	3.6340	-0.8170		
	xlxfg512	19338	8688	3.5941	-0.7970			8698	3.5983	-0.7991		
	Average			3.5596	-0.7798				3.6106	-0.8053		
Reverse Orientation	atatsgs	9647	4359	3.6148	-0.8074	4.6260	22.75%	4343	3.6015	-0.8007	4.62605	21.82%
	atef1a23	6022	2704	3.5921	-0.7960			2734	3.6320	-0.8160		
	atrdnaf	10014	4446	3.5518	-0.7759			4506	3.5997	-0.7998		
	atrdnai	5287	2315	3.5029	-0.7514			2383	3.6058	-0.8029		
	celk07e12	58949	26249	3.5622	-0.7811			26775	3.6336	-0.8168		
	hsg6pdgen	52173	23391	3.5866	-0.7933			23537	3.6090	-0.8045		
	mmzp3g	10833	4839	3.5735	-0.7867			4911	3.6266	-0.8133		
	xlxfg512	19338	8706	3.6016	-0.8008			8756	3.6222	-0.8111		
	Average			3.5732	-0.7866				3.6163	-0.8081		
Complement	atatsgs	9647	4359	3.6148	-0.8074	4.6260	23.01%	4355	3.6114	-0.8057	4.6260	21.94%
	atef1a23	6022	2678	3.5576	-0.7788			2702	3.5895	-0.7947		
	atrdnaf	10014	4418	3.5294	-0.7647			4520	3.6109	-0.8054		
	atrdnai	5287	2275	3.4424	-0.7212			2385	3.6088	-0.8044		
	celk07e12	58949	26483	3.5940	-0.7970			26735	3.6282	-0.8141		
	hsg6pdgen	52173	23455	3.5964	-0.7982			23515	3.6056	-0.8028		
	mmzp3g	10833	4821	3.5602	-0.7801			4921	3.6340	-0.8170		
	xlxfg512	19338	8688	3.5941	-0.7970			8698	3.5983	-0.7991		
	Average			3.5611	-0.7805				3.6108	-0.8054		
Reverse Complement	atatsgs	9647	4359	3.6148	-0.8074	4.6260	22.75%	4343	3.6015	-0.8007	4.6260	21.83%
	atef1a23	6022	2704	3.5921	-0.7960			2734	3.6320	-0.8160		
	atrdnaf	10014	4450	3.5550	-0.7775			4506	3.5997	-0.7998		
	atrdnai	5287	2317	3.5059	-0.7529			2383	3.6058	-0.8029		
	celk07e12	58949	26249	3.5622	-0.7811			26775	3.6336	-0.8168		
	hsg6pdgen	52173	23391	3.5866	-0.7933			23539	3.6093	-0.8046		
	mmzp3g	10833	4831	3.5676	-0.7838			4907	3.6237	-0.8118		
	xlxfg512	19338	8706	3.6016	-0.8008			8756	3.6222	-0.8111		
	Average			3.5732	-0.7866				3.6160	-0.9086		

Normal and artificial DNA Compression ratio and Compression rate shown in Table I . From top to bottom. Each column displays The result for a single algorithm showing the compression ratio for each sequence and the compression rate in bits per bases.

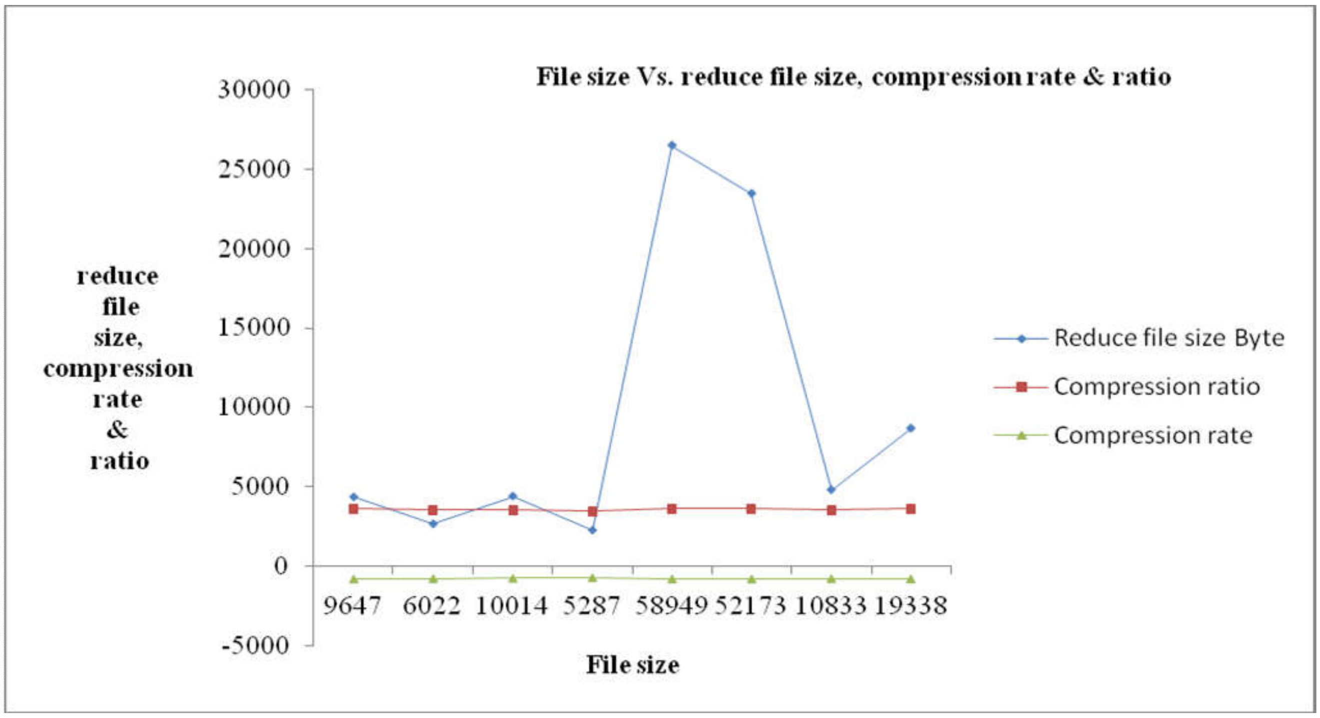

Fig.-1: File size versus compression size, ratio & rate of cellular sequences in normal orientation

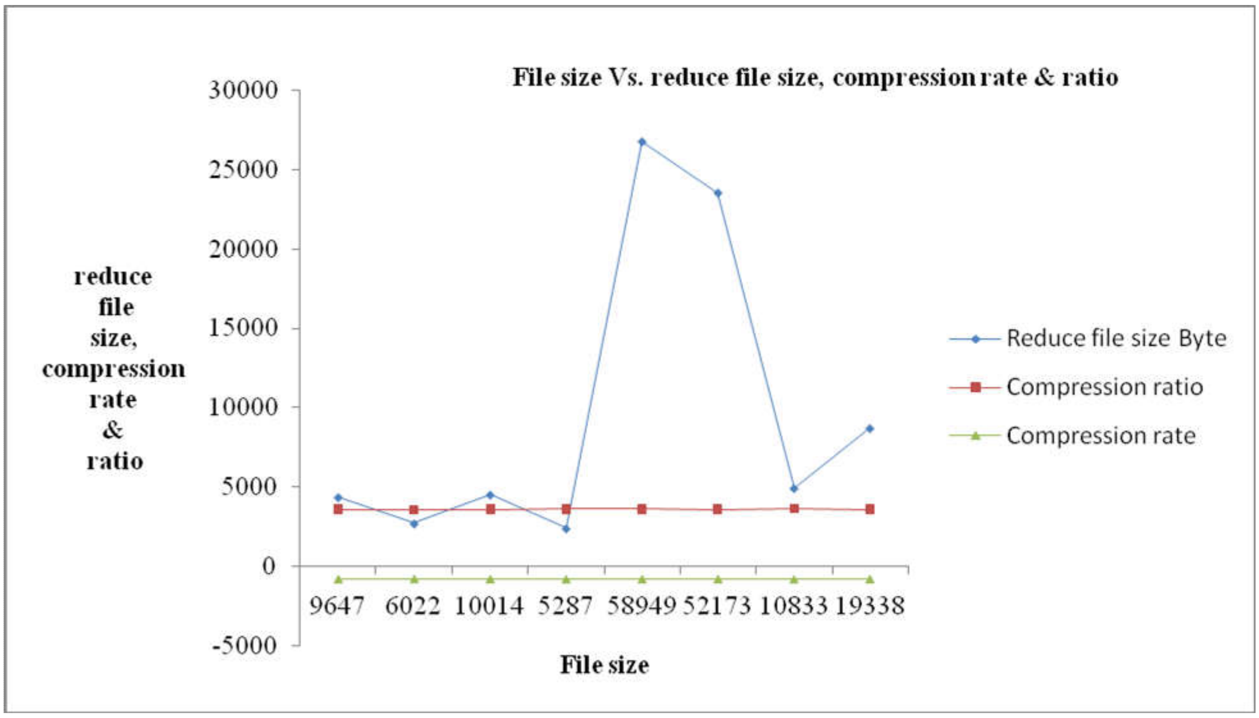

Fig-2: File size versus compression size, ratio & rate of artificial sequences in normal orientation

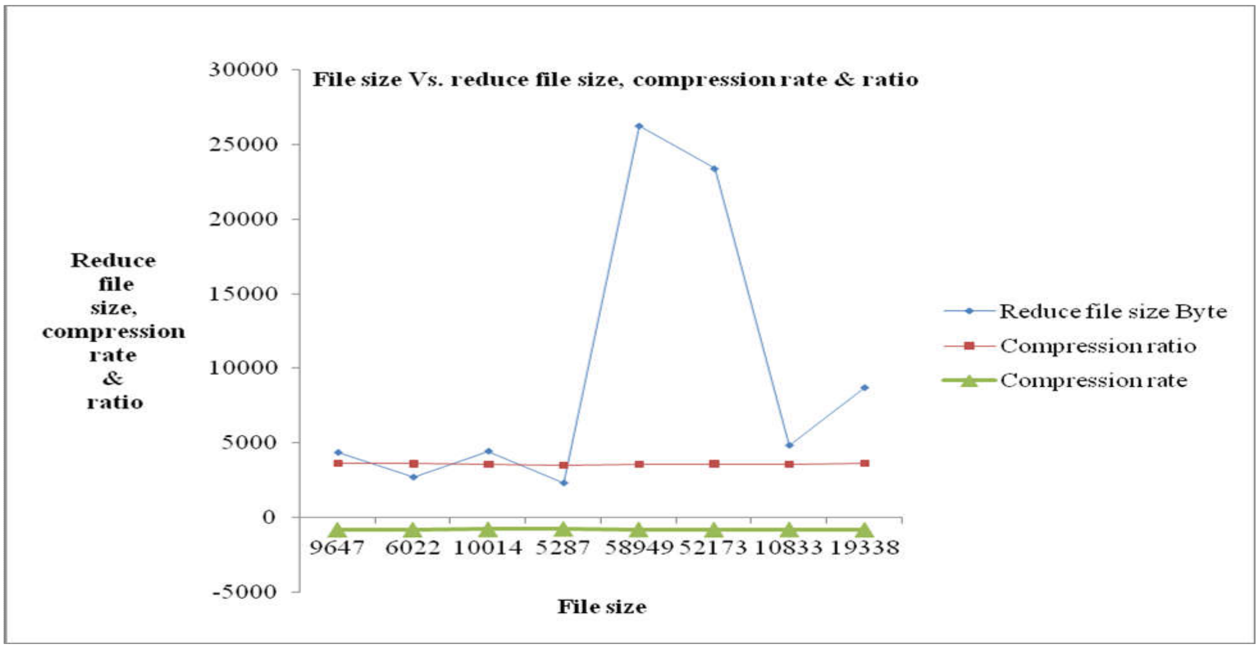

Fig.-3: File size versus compression size, ratio & rate of cellular sequences in reverse orientation

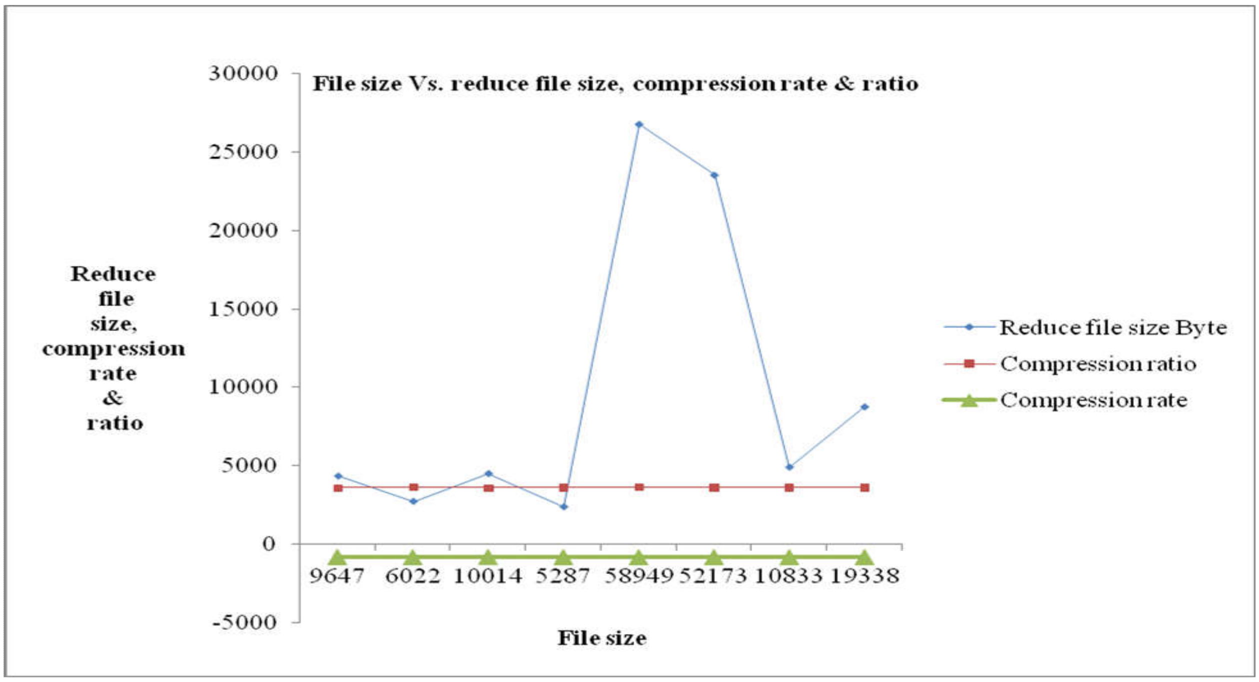

Fig.-4: File size versus compression size, ratio & rate of artificial sequences in reverse orientation

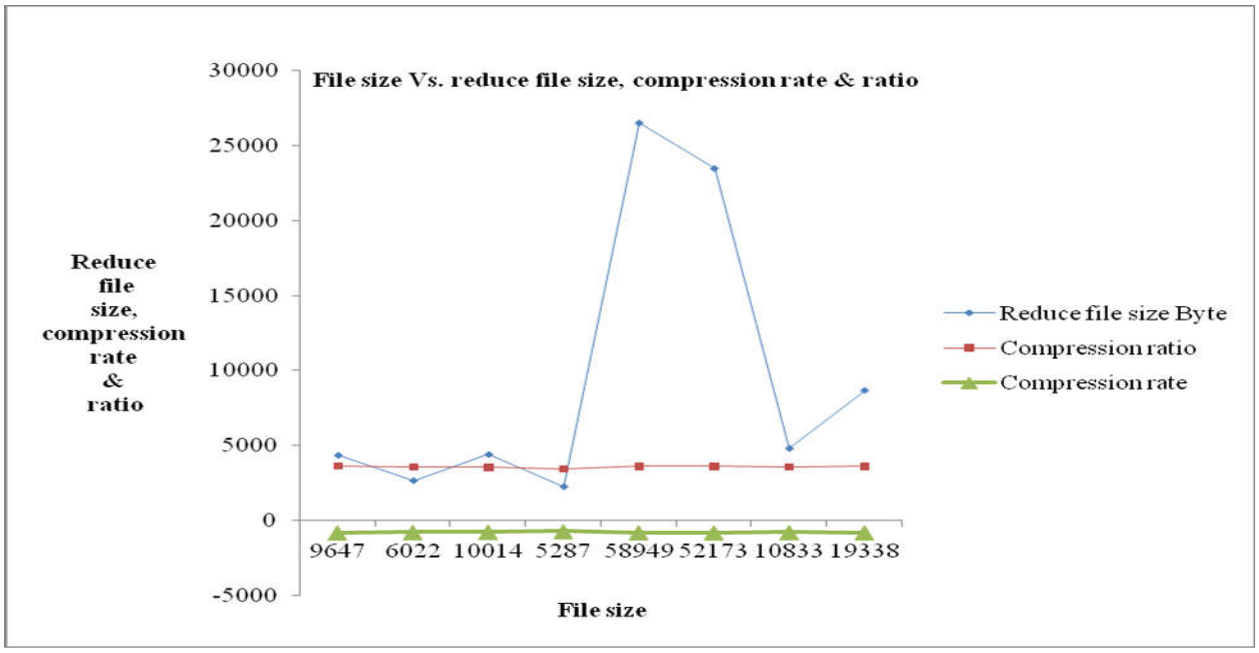

Fig.-5: File size versus compression size, ratio & rate of cellular sequences in complement orientation

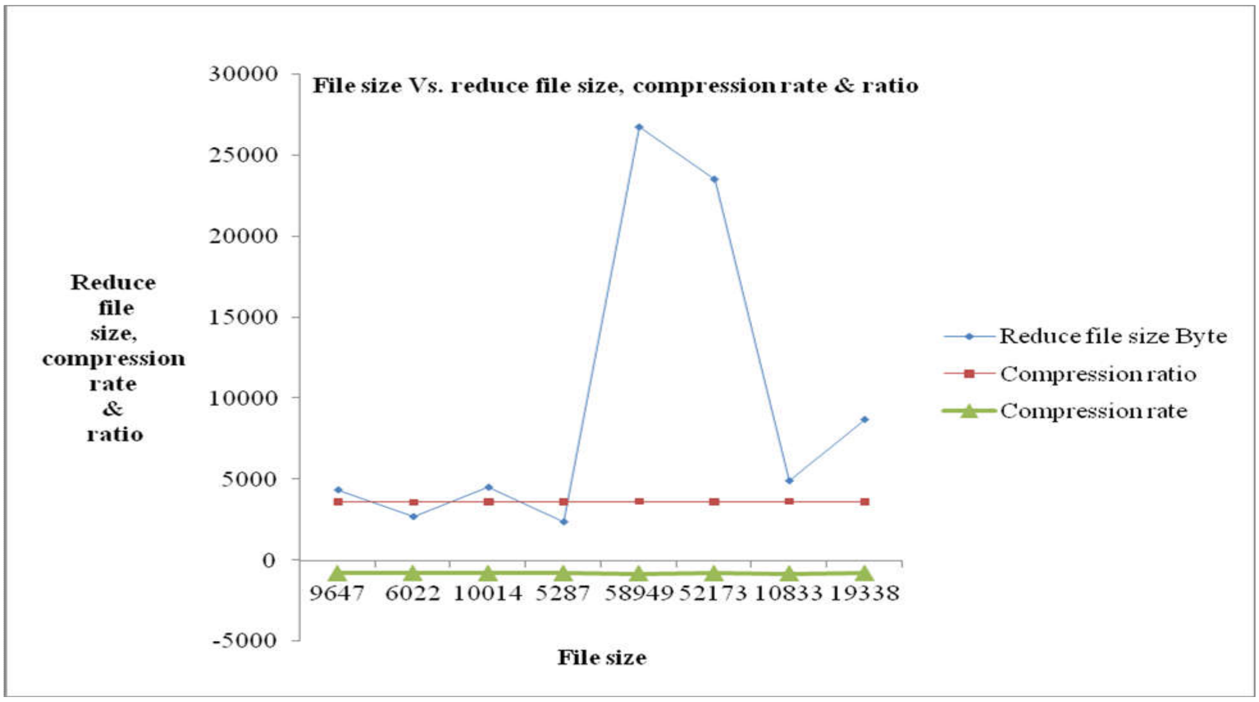

Fig.-6: File size versus compression size, ratio & rate of artificial sequences in complement orientation

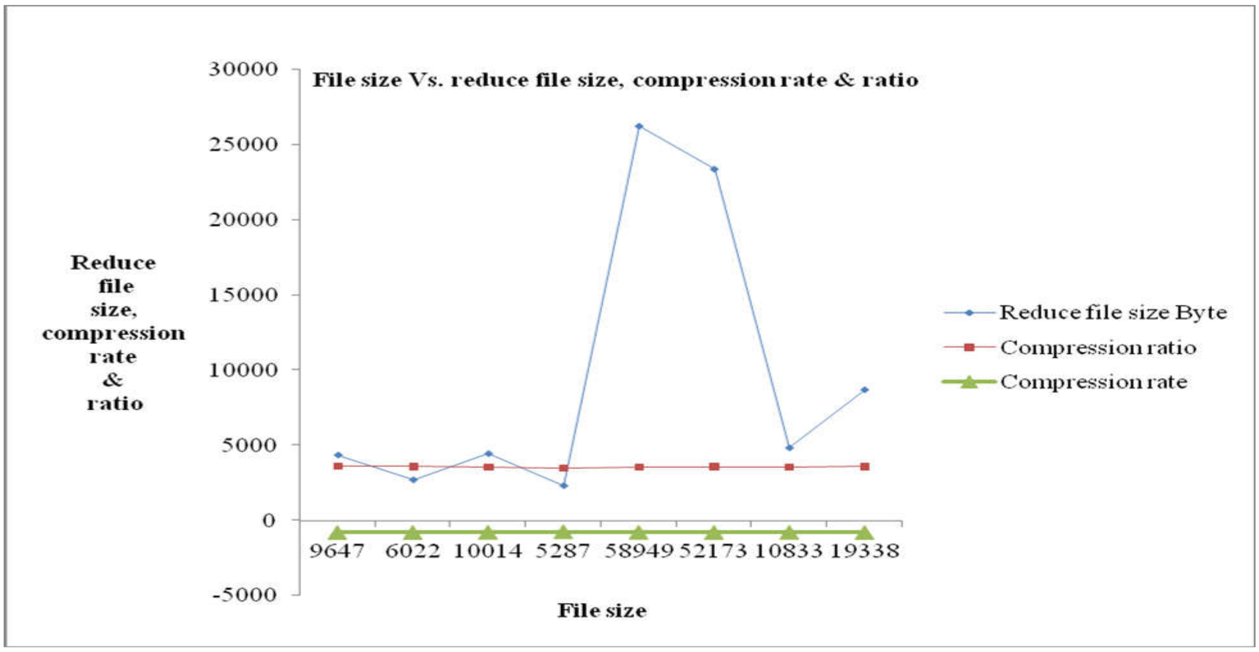

Fig.-7: File size versus compression size, ratio & rate of cellular sequences in reverse complement orientation

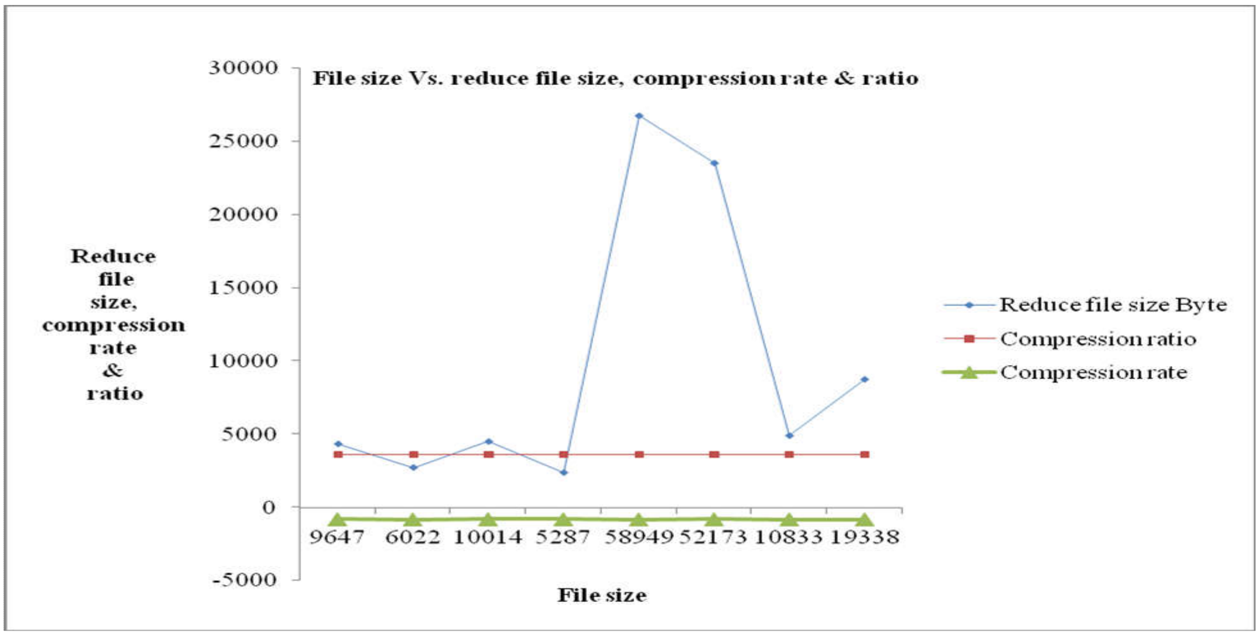

Fig.-8: File size versus compression size, ratio & rate of artificial sequences in reverse complement orientation

Table 1 displays the experimental results for data sets I and II. Set I comprises cellular DNA sequences that are normal, reverse, complement, and reverse complement, whereas Set II comprises artificial DNA sequences that are normal, reverse, complement, and reverse complement. The average cellular sequence compression rate for data sets I and II is 4.90853 bits/base and 4.49969 bits/base, respectively. This result is visually represented in Figures 1, 3, 5, and 7 for data set I and 2, 4, 6, and 8 for data set II. The graph's nature is heterogeneous since each sequence is logically ordered and belongs to a distinct species.

The cellular sequence and generated data results are shown graphically in Figures 18 to 20 for Data Set I and 21 to 23 for Data Set II. The sharp contrasts between the biological sequence and the artificial data nature were illustrated by this graph.

Result Discussion

These tests lead us to the conclusion that internal repeats and palindrome matching patterns are the same across all kinds of sources and are essential for identifying patterns or similarities in DNA sequences. Since the output file contains ASCII characters that do not match a, u, g, and c, it can offer information security, which is crucial from the perspective of data protection during transmission. The nucleotide sequence in a specific source can be protected with high security using these strategies. Better security than static LUT is available here.

Our technique is highly helpful for saving data in databases. Rather than storing sequences as files, we can store them as records in a database. Users can obtain original sequences in an intangible time by simply employing the exact repeat and palindrome. Furthermore, our algorithm is simple to implement.

Conclusion

This article talks about a novel DNA compression method that uses internal repeats and palindromes as its main concept. A decent model for compressing DNA sequences that reveals their true properties is provided by this compression technique. The compression findings of DNA sequences with repeats and palindromes also show that our approach works better than many others. This technique uses this finding to get the best compression results and is able to identify more regularities in DNA sequences, such as crossover and mutation. This method offers very strong information security, but it is unable to obtain a better compression ratio than other typical methods.

Important observations include:

a) The substring length of repeats and palindromes ranges from two to five, and if the substring length reaches six or more, no match is detected.

b) The substring length of three is quite compressible compared to the substring lengths of four and five, meaning it is more repeated than the substring lengths of four and five.

c) Compared to revere, complement, and reverse complement sequences, normal sequences are far more compressible.

d) The compression rate and compression ratio of synthetic DNA sequences are constant across all data sets, whereas those of cellular DNA sequences can be distinguished from one another since they originate from diverse sources.

References

1. **M. Li and P. Vitányi (1997)**, An Introduction to Kolmogorov Complexity and Its Applications, 2nd ed. New York: Springer-Verlag.

2. **Bell, T.C., Cleary, J.G., and Witten, I.H.(1990)** , Text Compression, Prentice Hall.

3. **Ma,B., Tromp,J. and Li,M. (2002)** PatternHunter—faster and more sensitive homology search. Bioinformatics, 18, 440–445.1698.

4. **S. Grumbach and F. Tahi (1994)**, “A new challenge for compression algorithms: Genetic sequences,” J. Inform. Process. Manage., vol. 30, no. 6, pp. 875-866.

5. **Xin Chen, San Kwong and Mine Li(2001)**, “A Compression Algorithm for DNA Sequences Using Approximate Matching for Better Compression Ratio to Reveal the True Characteristics of DNA”, IEEE Engineering in Medicine and Biology,pp 61-66.

6. **T.Matsumoto,K.Sadakame and H.Imani (2000),** ”Biological sequence compression algorithm”, Genome Informatics 11:43-52.

7. ASCII code. [Online]. Available: http://www.asciitable.com

8. National Center for Biotechnology Information,http://www.ncbi.nlm.nih.gov