The Scientific Temper (2025) Vol. 16 (6): 4456-4467
E-ISSN: 2231-6396, ISSN: 0976-8653
Doi: 10.58414/SCIENTIFICTEMPER.2025.16.6.19
https://scientifictemper.com/

RESEARCH ARTICLE

Pattern-driven Huffman encoding and positional encoding for DNA compression

Arunachalaprabu G[1*], Fathima Bibi K[2]

Abstract

Researchers from bioinformatics, biology, biotechnology, and medical sciences who are engaged in genetic data analysis face significant challenges in the manipulation and storage of large datasets. Compression algorithms are essential for increasing storage capacity and reducing the number of bits required to represent nucleotide bases. The pattern-driven Huffman encoding and positional encoding for DNA compression (P2DNAComp) algorithm is designed to compress both non-repetitive and repetitive pattern bases within DNA sequences. This demonstrates the algorithm's adaptability across various pattern types in genomic data. P2DNAComp employs a systematic approach to efficiently compress DNA sequences. It reads the sequences and constructs a symbol table to maintain the positional values of repeated patterns. Using Huffman coding, the algorithm determines the optimal bit representation for each repeated pattern to maximize storage efficiency. For non-repetitive patterns, a coded table is created to store positional values. Subsequently, a positional encoding technique is applied to minimize the number of bits needed for efficient representation. The maximum positional value is set as the upper limit, and the minimum number of bits required is computed using a binary logarithm function. The final compressed sequence is generated by encoding both repetitive and non-repetitive patterns. Using standard datasets from the GenBank database, the performance of the P2DNAComp algorithm was evaluated based on compression ratio, compression/decompression time, and compression gain. The algorithm achieved an average compression ratio of 1.09 bits per base (bpb), an average compression gain of 86.279%, and average compression and decompression times of 0.547 and 0.563 seconds, respectively.

Keywords: Compression ratio, Deoxyribonucleic acid, Huffman coding, Positional encoding technique, Binary logarithm function.

Introduction

The structure of genetic material is defined by three fundamental biomolecules:

- Proteins,
- Deoxyribonucleic Acid (DNA),
- Ribonucleic Acid (RNA).

Proteins govern cellular behavior, concentration, and morphology, thereby determining the unique characteristics of each cell type—be it hair, nerve, or blood cells. DNA, often regarded as the molecular architect, directs the synthesis of specific proteins, although its activation depends on protein interactions. RNA, which shares structural similarities with DNA, plays a complementary role in regulating various cellular processes. Genetic material is transmitted across generations and forms the basis of heredity and cellular identity. DNA, located within the cell nucleus, is composed of long chains made up of four nucleotide bases: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). These bases pair specifically—A with T, and C with G—to ensure structural stability and the accurate transmission of genetic information. The DNA sequence, a long string of these base pairs, encodes the essential instructions for all cellular functions. The double-helix structure of DNA emphasizes the importance of base pairing in preserving genetic fidelity. Notably, (1) the precise order of these bases is vital for decoding the genetic code, and (2) unraveling the encoded information is critical for understanding the organism's traits and functions. DNA serves as the blueprint for an organism's genetic makeup. Despite advancements in databases and

Research Scholar in Computer Science, Thanthai Periyar Government Arts & Science College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, Tamilnadu, India.

Assistant Professor in Computer Science, Thanthai Periyar Government Arts & Science College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, Tamilnadu, India.

***Corresponding Author:** Arunachalaprabu G, Research Scholar in Computer Science, Thanthai Periyar Government Arts & Science College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, Tamilnadu, India, E-Mail: guruarun12@gmail.com

How to cite this article: Arunachalaprabu, G., Bibi, F.K. (2025). Pattern-driven Huffman encoding and positional encoding for DNA compression. The Scientific Temper, **16**(6):4456-4467.

Doi: 10.58414/SCIENTIFICTEMPER.2025.16.6.19

Source of support: Nil

Conflict of interest: None.

Received: 24/06/2025 **Accepted:** 25/06/2025 **Published**: 30/06/2025

Table 1: GenBank database

Release	*Month/Year*	*Bases*	*Sequence*	*Growth Rate Bases %*	*Growth Rate sequence%*
241	12 / 2020	723003822007	221467827	–	–
242	02 / 2021	776291211106	226241476	+7.37	+2.15
243	04 / 2021	832400799511	227123201	+7.14	+0.39
244	06 / 2021	866009790959	227888889	+3.46	+0.34
245	08 / 2021	940513260726	231982592	+8.66	+1.79
246	10 / 2021	1014763752113	233642893	+7.23	+0.72
247	12 / 2021	1053275115030	234557297	+3.92	+0.39
248	02 / 2022	1173984081721	236338284	+11.43	+0.76
249	04 / 2022	1266154890918	237520318	+7.12	+0.50
250	06 / 2022	1395628631187	239017893	+9.54	+0.63
251	08 / 2022	1492800704497	239915786	+8.15	+0.37
252	10 / 2022	1562963366851	240539282	+2.93	+0.26
253	12 / 2022	1635594138493	241015745	+4.66	+0.20
254	02 / 2023	1731302248418	241830635	+6.13	+0.34
255	04 / 2023	1826746318813	242554936	+5.51	+0.30
256	06 / 2023	1966479976146	243560863	+7.66	+0.41
257	08 / 2023	2112058517945	246119175	+7.39	+1.05
258	10 / 2023	2433391164875	247777761	+15.19	+0.67
259	12 / 2023	2570711588044	249060436	+5.66	+0.52

data processing technologies, researchers continue to face challenges due to the overwhelming volume of genetic data. Collecting complete DNA sequences and annotating genomic features are complex and resource-intensive tasks. Ongoing genome projects are now generating trillions of base pairs, supported by a wide variety of biomedical devices and data acquisition methods.

For instance, Release 259 of the GenBank database (December 2023) reported approximately 2,570,711,588,044 bases across 249,060,436 sequences (National Library of Medicine, 2024) (Table 1). The database has nearly doubled in size every 18 months, presenting storage and computational bottlenecks. Accumulating such large volumes of DNA sequences has become a primary challenge in bioinformatics, often leading to memory overflow and network congestion during data transfers. Efficient access to DNA sequences remains a major concern for the scientific community. To address these issues, various algorithms have been developed to reduce DNA file sizes—mitigating challenges related to storage, transmission, and accessibility. These algorithms aim to optimize storage efficiency and improve the usability of genetic data in the rapidly expanding bioinformatics landscape.

Overview of Compression Techniques

The task of minimizing the number of bits required to represent DNA bases is known as compression. Compression techniques are broadly categorized as lossy or lossless. Lossy compression reduces file size effectively but cannot precisely recover the original data, making it unsuitable for DNA data. In contrast, lossless compression reduces the file size while ensuring the original data can be fully restored, making it ideal for applications in genomics where data integrity is crucial.

Impact of DNA Mutations on Compression Strategies

DNA mutations—such as base insertions, deletions, or substitutions—pose significant challenges for compression. These changes can disrupt sequence regularity, making lossy techniques unsuitable. The necessity to preserve data accuracy highlights the importance of lossless compression strategies that maintain sequence fidelity despite genomic variation.

Advances in Lossless Compression for Genomic Data

Lossy compression may result in the loss of essential bases during sequence encoding, which is unacceptable in bioinformatics. Consequently, researchers have shifted focus toward lossless compression techniques, which preserve the complete genetic sequence while reducing storage requirements.

A universal storage system that employs lossless compression is essential for efficient data exchange between databases—enabling faster uploads, downloads, and cross-platform compatibility. Compression quality is generally determined by how well algorithms handle repetitive and non-repetitive patterns in DNA sequences.

This study proposes a solution using lossless compression strategies tailored for genomic data. Several existing lossless algorithms have aimed to reduce storage space and improve data transmission efficiency. Among them are: BioCompress (Grumbach & Tahi, 1993), BioCompress2 (Grumbach & Tahi, 1994), GenCompress (Chen, Kwong, & Li, 1999), DNACompress (Chen, Li, Ma, & Tromp, 2002), Normalized Maximum Likelihood (NML) (Tabus, Korodi, & Rissanen, 2003), GeNML (Korodi & Tabus, 2005), DNASC (Mishra, Aaggarwal, & Abdelhadi, 2010). These earlier algorithms demonstrated efficacy for repetitive DNA sequences, but their performance degrades with non-repetitive patterns, limiting their generalizability. This dependency on regularity in DNA sequences is a core limitation of many traditional models.

To address this, the present research introduces the P2DNAComp method, which effectively compresses both repetitive and non-repetitive DNA patterns using pattern matching and an improved Huffman coding approach. The method also relocates non-repetitive patterns to a separate working file, where positional encoding and binary logarithmic optimization techniques are applied. This approach significantly enhances storage efficiency, even under high-throughput conditions where non-repetitive sequences are prevalent.

Organization of the Paper

This paper is organized as follows:

- Section 2 reviews recent advances in DNA compression algorithms.
- Section 3 defines the performance metrics used to evaluate the proposed method.
- Section 4 describes the P2DNAComp algorithm in detail.
- Section 5 presents the experimental results.
- Section 6 concludes the study with key findings and future directions.

Related Works

Recently, researchers have implemented lossless compression techniques for genetic data (DNA sequences), marking a significant advancement in the field of bioinformatics. This development highlights the need for strong interdisciplinary collaboration—particularly among the domains of computer science, bioinformatics, biology, biotechnology, and medical science. This section presents a comprehensive literature review of recently developed lossless DNA sequence compression algorithms, examining both theoretical foundations and practical applications. It also addresses ongoing research challenges in managing large-scale genomic datasets.

Krishnamoorthy and Karthikeyan (2022) proposed a technique called Hybrid Streamlining of Hospitalization–Subordinate DNA Compression (HOARDNAComp), which uses a firefly algorithm and an auto-regression strategy. The technique integrates an even-mode statistical method with autoregressive modeling. Modified firefly optimization is employed to set model parameters dynamically. This approach resolves computational efficiency issues and achieved an average compression ratio of 1.39 bits per base (bpb). In comparison with DNA Compression using Particle Swarm Optimization (DCPSO) (Arya & Bharti, 2017), HOARDNAComp demonstrated superior performance (Krishnamoorthy & Karthikeyan, 2022).

Murugan and Punitha (2021) introduced an innovative algorithm called Small Pattern Matching (S_Pattern) for DNA sequence compression. In this method, input DNA sequences are divided into segments of size 2 to 6, and each matching segment is encoded using ASCII symbol representation. These encoded sequences are then compressed using the LZ77 algorithm (Ziv & Lempel, 1977). An average compression ratio of 93% was achieved across various datasets from the UCI repository. However, the limitation lies in its constrained segment size range (2–6), which restricts flexibility (Murugan & Punitha, 2021).

Rosario Gilmary and Murugesan (2021) proposed a bit-reduction technique involving three stages: Bit reduction, Binary-to-hexadecimal conversion, and Huffman coding of the hexadecimal values.

Compared to existing algorithms, this method achieved better compression ratios and reduced storage requirements. The technique utilizes multiple transformations to reduce complexity in compressing DNA datasets, thereby emphasizing storage efficiency and lossless performance (Rosario Gilmary & Murugesan, 2021).

Mansouri et al. (2020) presented Single–Block Encoding (DNAC–SBE), a technique based on One-Bit encoding. In this method: a) Frequently occurring bases are substituted with 1, b) Less frequent bases with 0, and c) The output is encoded using Single–Block Encoding (SBE) with dynamically assigned short codewords. DNAC–SBE effectively identified previously unrecognized DNA bases and achieved a strong compression ratio. However, the use of a fixed 7-bit block size limited its compression efficiency when applied to diverse datasets (Mansouri & Yuan, 2018).

Murugesan (2020) introduced a Codon-Based Compression Algorithm (CBCA) that performs both compression and decompression without the use of a dictionary, thus reducing the need for additional storage space. CBCA achieved a compression ratio of 1.59 bpb with a decompression time of 0.18 seconds. However, the algorithm uses fixed-length binary strings (1, 2, 4, 5, or 6 bits), which may constrain adaptability when encoding variable DNA sequences.

Hui Chen (2020) developed an Entropy Coding Technique (ECT) based on context modeling. The ECT method: a) Segments input DNA into coding sequences, residual clusters, RNA, and introns, b)

Assigns attributes based on sequence features, and c) Applies entropy-based encoding.

It achieved an average compression ratio of 1.72 bpb, although it suffered from high computational time. Despite this drawback, ECT is noted for its effective entropy-based compression of DNA sequences (Chen, 2020).

Syed Mahamud Hossein et al. (2020) proposed a novel method named GP2R, which integrates Genetic Palindrome (GP), Palindrome (P), and Reverse (R) algorithms. The method involves:

Stage 1: Identification of all substrings,
Stage 2: Encoding unmatched and palindrome regions,
Stage 3: Encoding compressed files using a modified RSA technique.

GP2R outperformed standard algorithms in terms of compression ratio, showcasing its potential for efficient DNA data compression (Hossein et al., 2020).

Performance Evaluation Metrics

Table 2 outlines the key metrics used to evaluate the performance of DNA sequence compression algorithms. These metrics help in quantifying the efficiency, effectiveness, and practicality of the proposed method.

Proposed Algorithm

Computational modeling of DNA sequencing techniques has generated voluminous data in the form of DNA sequences. The rapid proliferation of these DNA sequences has attained prominent pace. These genetic data has been popular as well as easily accessible for homology searches, intricate modeling and sequence mining. The need for advanced storage solution in bioinformatics is motivated by the large size, the great complexity and diversity of genetic data in databases. It is an essential need for researchers to store extensive amount of genomic data as well as efficiently analyze them.

Within the intricate landscape of the bioinformatics community, several prominent challenges demand meticulous attention from researchers (Table 3).

The pursuit of addressing pressing challenges in genomic data management is intricately tied to several pivotal objectives (Table 4).

The main objective of lossless compression algorithms is to maintain integrity of information during compression. This work achieves good compression ratio and commendable compression gain as well as reduce demands for time related with compression and decompression. P2DNAComp algorithm compresses both repetitive pattern and non–repetitive pattern bases of DNA sequences. The uniqueness of the algorithm depends on the combination of advanced positional encoding methodologies and improved Huffman coding techniques which enhance the genetic data compression efficiency. P2DNAComp provides a novel solution to challenges of compression and holds promise for efficiency of lossless DNA sequence compression. The important suggestions are also offered for development in analysis and storage of genetic sequences.

The input for P2DNAComp consists of discrete set of nucleotide bases within DNA sequence. The P2DNAComp algorithm works as follows. First, all genetic information (as a sequence) of individual elements is recognized for analysis of the sequences. Then, formulate a symbol table for repeated patterns that inherent in the sequence. The positions and occurrences of recurrent bases are systematically maintained in this structured table. Next, Huffman coding technique is applied to efficiently optimize the frequently occurring patterns representation and storage. This

Table 2: Key Performance Metrics

Compression Ratio (CR)	
Definition	Establishes the ratio of compressed file size to original file size in bits per base (bpb) or bits per character (bpc).
Formula	CR = Compressed file size / Original file size
Compression Factor (CF)	
Definition	Represents the ratio of original file size to compressed file size.
Formula	CF = Original file size / Compressed file size
Saving Percentage (SP)	
Definition	Depends on the difference between original and compressed file sizes, expressed as a percentage.
Formula	SP =(Original file size –Compressed file size) / Original file size
Compression Time (CT)	
Definition	Time needed for file compression
Decompression Time (DT)	
Definition	Time required reconstructing the file to its original state, both measured in seconds.

Table 3: Challenges and Solutions in Genomic Data Storage and Management

	Challenge	*Significance*	*Research Focus*
Escalating DNA Sequences and Storage Imperatives	Need for substantial disk storage capacity	Genetic information with its complex sequences requires advanced storage solutions to handle the huge volume of the DNA sequences	Improve the capacity of storage device and provide a platform capable to handling the huge volume of the DNA sequences efficiently
Logistical Complexities in Genomic Data Transfer	Transferring the genomic data(DNA sequences) from one node to another node which makes difficulties, resulting in time–intensive procedures	Transfer of genetic data (DNA sequences) is a laborious task hindering the efficiency of collaborative research endeavours	Aims to provide the seamless transfer of large volume of genetic data (DNA sequences) among databases, institutions and researchers
Genomic Data Compression Challenges and Pathway Inference	Non–repetitive bases in DNA sequences occupy more storage space during compression	Non-repetitive bases (space–intensive nature) within the DNA sequences require more space which leads to challenges for data compression	Emphasizes the requirement of enhanced compression approaches which ensures the reliability of information while efficiently compressing DNA sequences

technique determines the optimal number of bits needed for representation of repeated pattern. Here, shortest binary codes are assigned to patterns encountered most frequently which ensures encoding patterns efficiently. To represent non–repetitive patterns with respective positional values a coded table is constructed. It is helpful for encoding the positional values of the patterns and facilitates the way for reconstruction of the sequence in an efficient manner during decompression. Using information theory techniques, minimum required bits are determined to store positional values in the coded table. The key objectives are:

- Minimum redundancy
- Optimal bit representation
- Information content preservation

Following this step, the binary representation of the repetitive patterns is written into the work file. With the help of previously constructed symbol table and Huffman codes this operation is performed with exactitude. In this algorithm, the features of Huffman coding (for repeated patterns) and coded table representation (for non–repetitive patterns) are used. This helps to achieve better DNA sequence compression that encapsulates all essential genetic information and significantly reduce storage requirements.

The novelty of P2DNAComp lies in its transformative integration of information techniques such as systematic positional encoding; Huffman coding that collectively builds a paradigm shift in compression of sequences. It takes the input sequences as a discrete set of bases. Symbol table is generated for recurrent patterns to organize them in a systematic manner. Thereby the algorithm offers a sophisticated comprehensive and structured method to recognition of patterns in the DNA sequences. The process of determining the number of bits required for repeated patterns is performed using the application of Huffman coding and principles of information theory. Another important feature of P2DNAComp algorithm is that generation of a coded table for non–repetitive

Table 4: Objectives in Genomic Data Management–Compression, Storage and Data Transfer

	Objective	*Significance*
Innovative Compression Algorithm for Genomic Data	Propose a novel lossless DNA sequence compression algorithm to minimize the number of bits need to represent genomic data with better compression ratio in DNA sequences (both repetitive bases and non–repetitive bases)	Design an algorithm that compress the size of DNA sequences, understanding of the complexities within DNA sequences
Efficient Storage and Capacity Expansion	The algorithm provides the way to store huge volume of genomic data without compromising the efficacy of the storage medium Improves the capacity of storage medium	Optimize the capacity of storage device(storage efficiency) Enhancing capacity of storage device
Scalability for Varied Dataset Sizes	The algorithm should be adapt to compress DNA sequences with different sizes	Ensure the scalability of algorithm across various dataset with different sizes
Optimized Data Transfer and Reduced Network Traffic	The algorithm ensures to transferring genomic datasets(DNA sequences) from one point to another point and minimize network traffic	Reduce network traffic streamlines the data transfer process optimizing overall genomic data exchange

Table 5: Illustrative Symbol Table

Sequence identifier	*Repeated patterns*
RP0	"ACGT"
RP1	"ATCG"
RP2	"TTTT"
RP3	"ATTA"
RP4	"TTC"
RP5	"GTT"
RP6	"CC"

patterns to incorporating positional values. This proposed technique not only minimizes the redundancy but also offers efficient decompression which shows the uniqueness of the algorithm and underscoring the significant advancement in lossless DNA sequence compression.

Symbol Table for Recurrent DNA Sequences

For effective data representation of repetitive patterns and to eliminate the overhead of occupying more storage space, the symbol table is designed for recurrent sequences. The symbol table provides tremendous flexibility to organize the repetitive DNA sequences and consequently document the sequences. It collects the details of sequence identifier and repeated sequences in Table 5.

- Sequence Identification: The recurring sequences in DNA dataset are identified by the P2DNAComp algorithm.
- Building Symbol Table: After identification of the repeated sequences, the symbol is generated into two parts. a) The sequence identifier b) The repetitive sequences.

Table 6 presents a list of repeated patterns identified with sequence identifier (unique label). These elements provide users with clear reference to enhance the compression and decompression process.

Huffman Coding Method

Developed by David Huffman, Huffman coding method is a familiar lossless compression technique using the principles of information theory and widely used to reduce the size of files. Sayood, K. (2012) (Table 7).

Steps involved in Huffman coding:

- All bases are arranged (descending order) based on frequencies of respective bases.
- Initiating with lowest frequency base, designate each base (leaf node).
- To devise a new node extract two minimum frequency nodes. Left node–base with minimum frequency. Right node–base with second minimum frequency. New node–sum of the frequencies of left node and right node.
- Add the new node

Table 6: Symbol Table Generations for Repeated patterns

Sequence identifier	*Repeated sequences*
RP0	AAAAA
RP1	TTTT
RP2	CCC
RP3	GGG

- Repeat (3) and (4) until it reaches the root.
- Assign "0" to left edge and "1" to non–leaf nodes.
- Determine codeword by traversing the tree.

Determine Minimum Bits for Positional Values

- Let S be a set of positional values S= $\{P_1, P_2, P_3, \ldots, P_n\}$ where n denotes total number of positional values within the coded table.
- Find the maximum positional value (*Max*): The maximum value among the positional values is *Max*. It calculates the upper limit for the individual bit.

$$Max = max\{P_1, P_2, P_3 \ldots Pn\}$$

Find the minimum bits needed for positional value: Determine the minimum number of bits needed to represent each positional value using binary logarithm function log_2.

$$\text{Minimum Bits for } P_i = \lceil \log_2(P_i) \rceil$$

For example, Consider the following positional values $P_1=5$, $P_2=12$, $P_3=21$ and $P_4=30$. Now find the maximum positional value *Max*=max {5, 12, 21, 30} = 30. Then determine the minimum number of bits needed for represent the positional values. For $P_1 = 5$, $P_1 = \lceil log_2(5) \rceil = 3$, For $P_2 = 12$, $P_2 = \lceil log_2(12) \rceil = 4$, For $P_3 = 21$, $P_3 = \lceil log_2(21) \rceil = 5$, For $P_4 = 30$, $P_4 = \lceil log_2(30) \rceil = 5$. In this example, the minimum number of bits needed for each positional value is 3, 4, 5 and 5 bits.

Illustration

Consider the DNA sequence
AAAAACCCTTAAAAAAAACCCCCCGGGTTTTTTTTTGGGGG
Step 1: Read the given DNA sequence.
Step 2: Find repeated patterns and generate symbol table.
Step 3: Find shortest codeword for repeated patterns using Huffman Coding.
Step 4: Form coded table for non–repetitive patterns (Table 8).
Step 5: Find number of bits needed for each positional value

Max=max {3, 5, 11, 13} = 13

hen determine the minimum number of bits needed to represent the positional values. For $P_0 = 3$, $P_0 = \lceil log_2(3) \rceil = 2$, For $P_1 = 5$, $P_1 = \lceil log_2(5) \rceil = 3$, For $P_2 = 11$, P2 $= \lceil log_2(11) \rceil = 4$, For $P_3 = 13$, $P_3 = \lceil log_2(13) \rceil = 4$. The

Table 7: Huffman coding analysis for DNA sequence repeated patterns

Sequence identifier	*Repeated sequences*	*Frequency*	*Code word*	*Length of codeword*	*Bits required*
RP0	AAAAA	2	00	2	4
RP1	TTTT	2	01	2	4
RP2	CCC	3	10	2	6
RP3	GGG	2	11	2	4
Required bits = 18 bits					

Table 8: Coded table for non–repetitive patterns in DNA sequence

Sequence identifier	*Non–repetitive pattern*	*Positional value*
NRP0	TT	3
NRP1	AAA	5
NRP2	T	11
NRP3	GG	13

minimum number of bits required for each positional value is 2, 3, 4 and 4 bits respectively as represented in Table 9.

Step 6: Write the following pattern into work file

00–10–00–10–10–11–01–01–11

Required bits = 2 x Frequency (RP0) + 2 x Frequency (RP1) + 3 x Frequency (RP2) + 2 x Frequency (RP3)

= 2 x 2 + 2 x 2 + 3 x 2 + 2 x 2

=4+4+6+4 = 18 bits

Size after Compression = Work file + Bits required for positional values

= 18 bits + 13 bits

= 31 bits = 3.8 bytes

Compression Ratio is given by

$$\text{Compression Ratio} = \left(\frac{\text{Size after Compression}}{\text{Size before Compression}} \right) \text{x8} \cdot$$

(3.8 / 41) x 8

= 0.74 bpb.

Results and Discussion

Datasets

To test the proposed algorithm, six GenBank benchmark datasets are employed. Table 10 illustrates the key characteristics of the datasets. The P2DNAComp algorithm is implemented using the Java. Analyzing and accounting the relation between original file size and compressed file size is vital. Three different scenarios contingent upon the DNA sequences containing the occurrence of both repetitive patterns and non–repetitive patterns are considered:

- Best case
- Average case
- Worst case

Table 9: Minimum bit representation for positional values

Identifier	*Positional value*	*Minimum bits required*	*Bit representation*
P0	3	2	10
P1	5	3	100
P2	11	4	1010
P3	13	4	1100
Bits Required			13

The above datasets are taken from different sources containing various range of sequences and each of them associated with particular information detailing the characteristics and source (Table 10). The sequences Chloroplast as “Chmpxx” (Length: 121024 Bytes and File size: 118.19 KB), Human sequence source as “Humdystrop” (Length: 38770 Bytes and File size: 37.86 KB), “Humhbb” (Length: 73308 Bytes and File size: 71.59 KB), “Humhprtb” (Length: 56737 Bytes and File size: 55.40 KB), Mitochondria as “Mpomtcg” (Length: 186609 Bytes and File size: 182.23 KB) and Virus as “Vaccg” (Length: 191737 Bytes and File size: 187.24 KB). The proposed algorithm is assessed and validated using these standard benchmark datasets which helps to contribute the development of compression algorithms and analyse the genetic data effectively.

Best Case

Consider the DNA sequence (40bytes)

AAAAAAAAAGGGGGGCCCCCCTTTTTCCCCCCCCCTTTTT

Step 1: Read the given DNA sequence.

Step 2: Find repeated patterns and generate symbol table (Table 11).

Step 3: Find short codeword for repeated patterns using Huffman Coding (Table 12).

Step 4: Form coded table for non–repetitive patterns (Table 13).

Step 5: Find number of bits needed for each positional value.

$$Max = max\{3, 9\} = 9$$

Then determine the minimum number of bits needed to represent the positional values. For $P_0 = 3$, $P_0 = \lceil log_2(3) \rceil = 2$, For $P_1 = 9$, P1 = $\lceil log_2(9) \rceil = 4$. The minimum

Table 10: Summary of Benchmark Datasets

Sequence source	*Sequence name*	*Length (bytes)*	*File size (kilobytes)*
Chloroplast	Chmpxx	121024	118.19
	Humdystrop	38770	37.86
Human	Humhbb	73308	71.59
	Humhprtb	56737	55.40
Mitochondria	Mpomtcg	186609	182.23
Virus	Vaccg	191737	187.24

Table 11: Symbol Table Generations for Repeated patterns

Sequence identifier	*Repeated sequences*
RP0	CCCCCC
RP1	TTTTT
RP2	AAAA
RP3	GGG

number of bits needed for each positional value is 2 and 4 bits respectively as given in Table 14.
Step 6: Write the following pattern into work file

10–10–11–11–00–01–00–01
Compression ratio for best case = 0.54 bpb

Average Case

Consider the DNA sequence (40 bytes)
AAAAAAATCCCCCGGTTTCCCCCCCCCCCCCCCAAA GGCC
Step 1: Read the given DNA sequence.
Step 2: Find repeated patterns and generate symbol table (Table 15).
Step 3: Find short codeword for repeated patterns using Huffman Coding (Table 16).
Step 4: Form coded table for non–repetitive pattern (Table 17).
Step 5: Find number of bits needed for each positional value.

$$Max=max\{3, 6, 12\} = 9$$

Then determine the minimum number of bits needed for represent the positional values. For $P_0 = 3$, $P_0 = \lceil log_2(3) \rceil = 2$, For $P_1 = 6$, $P_1 = \lceil log_2(6) \rceil = 3$, For $P_2 = 12$, $P_2 = \lceil log_2(12) \rceil = 4$. The minimum number of bits needed for each positional value is 2, 3 and 4 bits respectively as shown in Table 18.
Step 6: Write the following pattern into work file

10–10–0–11–0–0–0–10–11
Compression ratio for best case = 0.57 bpb.

Table 13: Coded table for non–repetitive patterns in DNA sequence

Sequence identifier	*Non–repetitive pattern*	*Positional value*
NRP0	A	3
NRP1	CCC	9

Table 14: Minimum bit representation for positional values

Identifier	*Positional value*	*Minimum bits required*	*Bit representation*
P0	3	2	10
P1	9	3	1000
Bits Required			6

Table 15: Symbol Table Generations for Repeated patterns

Sequence identifier	*Repeated sequences*
RP0	CCCCC
RP1	AAA
RP2	GG

Worst Case

Suppose DNA sequence (40 bytes)
TAACGGGTCTCGGGGTTTTTCCCCACGTCCCGGCTAAAGT
Step 1: Read the given DNA sequence.
Step 2: Find repeated patterns and generate symbol Table 19.
Step 3: Find short codeword for repeated patterns using Huffman Coding (Table 20).
Step 4: Form coded table for non–repetitive patterns (Table 21).
Step 5: Find number of bits needed for each positional value.

$$Max=max\{3, 5, 9, 10, 12, 14\} = 14$$

Then determine the minimum number of bits needed for represent the positional values. For $P_0 = 3$, $P_0 = \lceil log_2(3) \rceil = 2$, For $P_1 = 5$, $P_1 = \lceil log_2(5) \rceil = 3$, For $P_2 =$

Table 12: Huffman coding analysis for DNA sequence repeated patterns

Sequence identifier	*Repeated sequences*	*Frequency*	*Code word*	*Length of codeword*	*Bits required*
RP0	CCCCCC	2	00	2	4
RP1	TTTTT	2	01	2	4
RP2	AAAA	2	10	2	4
RP3	GGG	2	11	2	4
Required bits = 16 bits					

Table 16: Huffman coding analysis for DNA sequence repeated patterns

Sequence identifier	*Repeated sequences*	*Frequency*	*Code word*	*Length of codeword*	*Bits required*
RP0	CCCCC	4	0	1	4
RP1	AAA	3	10	2	6
RP2	GG	2	11	2	4
Required bits = 14 bits					

9, $P_2 = \lceil log_2(9) \rceil = 4$, For $P_3 = 10$, $P_3 = \lceil log_2(10) \rceil = 4$, For $P_4 = 12$, $P_4 = \lceil log_2(12) \rceil = 4$, For $P_5 = 14$, $P_5 = \lceil log_2(14) \rceil = 4$. The minimum number of bits needed for each positional value is 2, 3, 4, 4, 4 and 4 bits respectively (Table 22).

Step 6: Write the following pattern into work file

10–00–00–11–11–01–01–10

Compression ratio for best case = 0.92 bpb.

Results of P2DNAComp for standard datasets

The performance of P2DNAComp across different DNA datasets is shown in Table 23. The original size of Chmpxx 121,024 is compressed to 15,649 bytes with compression ratio of 1.03 and compression gain of 87.07 percent. The original size of Humdystrop 38770 is compressed to 5451 bytes with compression ratio of 1.12 and compression gain of 85.94 percent. Similarly, for Humhbb (compression ratio: 1.08, compression gain: 86.47), Humhprtb (compression ratio: 1.09, compression gain: 86.37), Mpomtcg (compression ratio: 1.13, compression gain: 85.85) and Vaccg (compression ratio: 1.12, compression gain: 85.96). The average compression ratio and compression gain of P2DNAComp are 1.09 bpb and 86.28 percent for the standard datasets. In table 23, compression time and decompression time is indicated to emphasize the efficiency of the algorithm. Table 23 underscores the robust performance of P2DNAComp which helps reduce the file size of datasets and achieve high compression gain.

Statistical analysis of P2DNAComp over DNAC–SBE, CBCA, ECT, HOARDNA Comp, Bit Reduction, IBDNASCA and EIBDNASCA is shown in Table 24. The proposed algorithm achieves compression ratio of 1.03 for the dataset "Chmpxx" than other algorithms. It highlights the percentage improvement of 35% over DNAC–SBE, 35% over ECT and 22% over HOARDNA. The results establish the efficacy of P2DNAComp in reducing the various genetic datasets (DNA sequences).

Table 17: Coded table for non–repetitive patterns in DNA sequence

Sequence identifier	*Non–repetitive pattern*	*Positional value*
NRP0	AT	3
NRP1	TTT	6
NRP2	CC	12

Table 18: Minimum bit representation for positional values

Identifier	*Positional value*	*Minimum bits required*	*Bit representation*
P0	3	2	10
P1	6	3	101
P2	12	4	1011
Bits Required			9

Table 19: Symbol Table Generations for Repeated patterns

Sequence identifier	*Repeated sequences*
RP0	CGGG
RP1	TCCC
RP2	TAA
RP3	TT

The proposed algorithm achieves good compression ratio of 1.12 for the dataset "Humdystrop" than other algorithms. It highlights the percentage improvement of 43% over CBCA, 19% over ECT and 32% over IBDNASCA. The results establish the ability of P2DNAComp in reducing the various genetic datasets (DNA sequences) and optimizing the capacity of storage medium. The dataset "Humhbb" achieves 1.08 compression ratio and an improvement in

Table 20: Huffman coding analysis for DNA sequence repeated patterns

Sequence identifier	*Repeated sequences*	*Frequency*	*Code word*	*Length of codeword*	*Bits required*
RP0	CGGG	2	00	2	4
RP1	TCCC	2	01	2	4
RP2	TAA	2	10	2	4
RP3	TT	2	11	2	4
Required bits = 16 bits					

Table 21: Coded table for non–repetitive patterns in DNA sequence

Sequence identifier	*Non–repetitive pattern*	*Positional value*
NRP0	TCT	3
NRP1	G	5
NRP2	C	9
NRP3	ACG	10
NRP4	GGC	12
NRP5	AGT	14

Table 22: Minimum bit representation for positional values

Identifier	*Positional value*	*Minimum bits required*	*Bit representation*
P0	3	2	10
P1	5	3	101
P2	9	4	1000
P3	10	4	1001
P4	12	4	1011
P5	14	4	1101
Bits Required			21

percentage of 37% over CBCA, 41% over ECT and 25% over IBDNASCA. The average compression ratio of P2DNAComp is 1.09 bpb which exhibits the improvement from 6% to 38% compared with other compression algorithms.

The results highlight the performance of the proposed algorithm and emphasize its need for efficient sequence compression. It helps to analyse the sequence, optimize storage device capacity and transmission of genetic dataset.

Table 25 presents the comparison of the proposed algorithm over standard DNA sequence compression algorithms such as WinRAR, Bio–Compres2, Gen Compress, DNA Compress, Ge–NML and DNASC. The analysis includes compression ratios for different DNA sequence datasets which unveils the superiority of P2DNAComp. The dataset "Chmpxx", achieves better compression ratio of 1.03 compared to WinRAR (2.25 bpb), Bio–Compres2 (1.68 bpb), Gen Compress (1.67 bpb), DNA Compress (1.67 bpb), Ge–NML (1.66 bpb) and DNASC (1.50 bpb). In particular, the P2DNAComp algorithm does 54% better compared with others.

The dataset "Humdystrop", achieves better compression ratio of 1.12 compared to WinRAR (2.37 bpb), Bio–Compres2 (1.93 bpb), Gen Compress (1.92 bpb), DNA Compress (1.91 bpb), Ge–NML (1.91 bpb) and DNASC (1.89 bpb). It shows the improvement of P2DNAComp algorithm (54%) over other algorithms. The result obtained by P2DNAComp for the dataset "Humhbb" is 1.08 bpb. This surpasses WinRAR (2.22 bpb), Bio–Compres2 (1.88 bpb), Gen Compress (1.82 bpb) and DNA Compress (1.79 bpb).

For the dataset "Humhprtb", the proposed algorithm maintains good performance with a compression ratio of 1.09 when compared with WinRAR (2.23 bpb), Bio–Compres2 (1.91 bpb), Gen Compress (1.85 bpb), DNA Compress (1.82 bpb), Ge–NML (1.76 bpb) and DNASC (1.71

Table 23: Performance evaluation of P2DNAComp on Standard datasets

DNA sequence	*Actual size (Bytes)*	*Reduced size (Bytes)*	*Compression ratio (bps)*	*Compression Gain %*	*Time Taken(Seconds)*	
					Compression	*Decompression*
Chmpxx	121024	15649	1.03	87.0695069	0.510	0.546
Humdystrop	38770	5451	1.12	85.9401599	0.498	0.501
Humhbb	73308	9913	1.08	86.4776014	0.508	0.526
Humhprtb	56737	7732	1.09	86.3722086	0.504	0.531
Mpomtcg	186609	26395	1.13	85.8554518	0.583	0.597
Vaccg	191737	26912	1.12	85.9641071	0.679	0.681
Average			1.09	86.279	0.547	0.563

Table 24: Comparison analysis of P2DNAComp over existing algorithms

DNA sequence	*DNAC–SBE*	*CBCA*	*ECT*	*HOARDNA comp*	*Bit reduction*	*IBDN-ASCA*	*EIBDN-ASCA*	*P2DNA comp*
Chmpxx	1.60	–	1.58	1.33	–	1.40	1.14	1.03
Humdystrop	1.72	1.55	–	1.39	1.64	1.53	1.27	1.12
Humhbb	1.71	1.55	1.83	1.44	1.65	1.50	1.21	1.08
Humhprtb	1.72	1.54	1.85	1.45	–	1.51	1.25	1.09
Mpomtcg	1.72	1.55	–	1.40	1.62	1.57	1.28	1.13
Vaccg	1.67	1.57	1.78	1.32	1.66	1.52	1.23	1.12
Average Ratio	1.69	1.55	1.76	1.38	1.64	1.51	1.23	1.09

Table 25: Comparison analysis of P2DNAComp over standard algorithms

DNA sequence	*WinRAR*	*Bio–compres2*	*Gen compress*	*DNA compress*	*Ge–NML*	*DNASC*	*P2DNAComp*
Chmpxx	2.25	1.68	1.67	1.67	1.66	1.50	1.03
Humdystrop	2.37	1.93	1.92	1.91	1.91	1.89	1.12
Humhbb	2.22	1.88	1.82	1.79	–	–	1.08
Humhprtb	2.23	1.91	1.85	1.82	1.76	1.71	1.09
Mpomtcg	2.30	1.94	1.91	1.89	1.88	1.88	1.13
Vaccg	2.23	1.76	1.76	1.76	1.76	1.70	1.12
Average Ratio	2.27	1.85	1.82	1.80	1.79	1.74	1.09

bpb). P2DNAComp shows an improvement of 52% over other algorithms. For "Mpomtcg", the proposed algorithm maintains good performance with compression ratio of 1.13 when compared with WinRAR (2.23 bpb), Bio–Compres2 (1.91 bpb), Gen Compress (1.85 bpb), DNA Compress (1.82 bpb), Ge–NML (1.76 bpb) and DNASC (1.71 bpb). It shows the improvement of P2DNAComp algorithm about 51% over other algorithms. The result obtained by P2DNAComp for the dataset "Vaccg" is 1.12 bpb. This surpasses WinRAR (2.23 bpb), Bio–Compres2 (1.91 bpb), Gen Compress (1.85 bpb), DNA Compress (1.82 bpb), Ge–NML (1.76 bpb)and DNASC (1.71 bpb). It shows the improvement of P2DNAComp algorithm about 51% over other algorithms.

Figure 1 displays compression ratio of P2DNAComp algorithm of various DNA sequences (datasets). P2DNAComp algorithm achieves compression ratio of 1.03 bpb for "Chmpxx", 1.12 bpb for "Humdystrop", 1.08 bpb for "Humhbb", 1.09 bpb for "Humhprtb", 1.13 bpb for "Mpomtcg" and 1.12 bpb for "Vaccg" respectively.

Figure 2 shows the comparative analysis of P2DNAComp algorithm over various existing compression algorithms for DNA sequences. The x–axis indicates existing algorithms and y–axis identifies average compression ratio achieved by each algorithms such as DNAC–SBE, CBCA, ECT, HOARDNAComp, Bit Reduction, IBDNASCA, EIBDNASCA and P2DNAComp. These algorithms are evaluated using various standard DNA sequence datasets ("Chmpxx", "Humdystrop", "Humhbb", "Humhprtb", "Mpomtcg" and "Vaccg").

The results show that P2DNAComp algorithm outperforms existing algorithms across various datasets. It highlights the efficiency of P2DNAComp algorithm in terms of compression ratio over other competitive algorithms.

Figure 2 gives the identification of approaches to reduce the size of DNA sequences and gives the valuable insights of various DNA sequence compression algorithms.

Figure 3 shows the average compression ratio of different standard lossless DNA sequence compression algorithms and P2DNAComp across the datasets "Chmpxx", "Humdystrop", "Humhbb", "Humhprtb", "Mpomtcg" and "Vaccg".

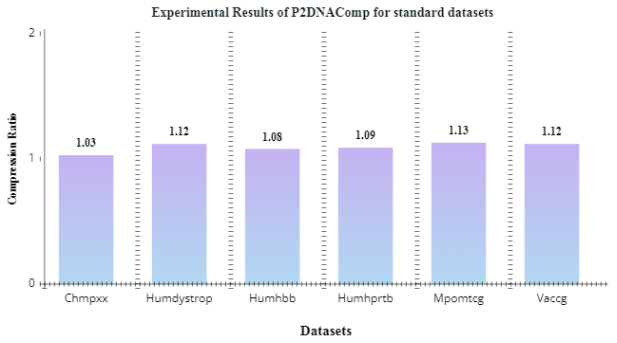

Figure 1: Experimental results of P2DNAComp for standard datasets

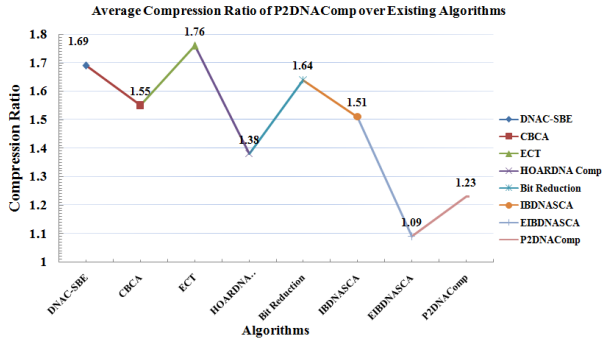

Figure 2: Average compression ratio of P2DNAComp over existing algorithms

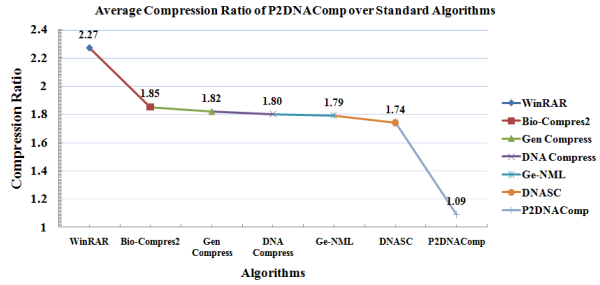

Figure 3: Average compression ratio of P2DNAComp over standard algorithms

WinRAR achieved a compression ratio of 2.27 bpb, Bio–Compres2 1.85 bpb, Gen Compress 1.82 bpb, DNA Compress 1.80 bpb, Ge–NML 1.79 bpb, DNASC 1.74 bpb whereas P2DNAComp algorithm achieves a compression ratio of

1.09 bpb. This highlights the effectiveness of P2DNAComp algorithm in reducing the size of the DNA sequence datasets and also development in compression ratio when compared with other algorithms.

Conclusion and Future work

This study addresses the critical challenges associated with the analysis and storage of DNA sequences, particularly the difficulty in handling massive datasets originating from diverse genomic sources. The proposed P2DNAComp algorithm effectively compresses both repetitive and non-repetitive pattern bases within DNA sequences, offering adaptability across various pattern types. This positions it as a versatile and robust tool in the domain of lossless DNA compression. The algorithm adopts a systematic approach, beginning with the construction of a symbol table and the application of Huffman coding to optimize storage capacity. For non-repetitive patterns, a coded table is created, followed by the use of positional encoding to minimize the number of bits required for efficient representation. The final compressed sequence—comprising Huffman codes and positional encoding—significantly reduces storage requirements while preserving the integrity of the genetic information. Performance evaluation was conducted using standard datasets from the GenBank database. The compression ratio, compression gain, compression time, and decompression time were used as key metrics. The results demonstrate the efficiency of P2DNAComp, with an average compression ratio of 1.09 bits per base (bpb), compression gain of 86.279%, compression time of 0.547 seconds, and decompression time of 0.563 seconds. Overall, P2DNAComp stands out as a promising advancement in the field of DNA sequence compression. It offers a comprehensive and efficient solution to the growing challenges of large-scale genomic data storage and transmission. In the future, this research can be extended by integrating machine learning techniques to predict optimal encoding strategies based on DNA sequence characteristics. Additionally, enhancing real-time compression speed and evaluating performance across more heterogeneous genomic datasets will further establish the algorithm's utility in clinical genomics, personalized medicine, and cloud-based bioinformatics platforms.

Acknowledgment

None.

References

Chen X., Kwong, S., & Li, M. (1999). A compression algorithm for DNA sequences and its application in genome comparison. *The Tenth Workshop on Genome and Informatics (GIW9)*, pp. 340–350.

Chen X., Li, M., Ma, B., & Tromp, J. (2002). DNACompress: Fast and effective DNA sequence compression. *Bioinformatics*, https://doi.org/10.1093/bioinformatics/18.12.1696

Deloula, M., & Yuan, X. (2018). One-bit DNA compression algorithm. *Proceedings of the International Conference on Neural Information Processing*, Cambodia, https://doi.org/10.1007/978-3-030-04239-4_34

Govind Prasad Arya, & Bharti, R. (2017). DNA compression using particle swarm optimization (DCPSO). *Journal of Advanced Research in Dynamical and Control Systems*, 5(Special Issue), 295–302.

Grumbach, S., & Tahi, F. (1993). Compression of DNA sequences. In *Proceedings of the Conference on Data Compression (DCC)*, pp. 340–350. https://doi.org/10.1109/DCC.1993.253115

Grumbach, S., & Tahi, F. (1994). A new challenge for compression algorithms: Genetic sequences. *Information Processing and Management*, pp. 875–886. https://doi.org/10.1016/0306-4573(94)90014-0

Hui Chen. (2020). Application of genome sequence based on entropy coding. In *International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pp. 156–159.

Khalid, S. (2012). *Introduction to Data Compression* (4th ed.). Morgan Kaufmann Series, Elsevier.

Korodi, G., & Tabus, I. (2005). An efficient normalized maximum likelihood for DNA sequence compression. *ACM Transactions on Information Systems*, https://doi.org/10.1145/1055709.1055711

Krishnamoorthy, M., & Karthikeyan, R. (2022). Classification techniques for medicinal databases using auto-regression and firefly algorithm. *Journal of Algebraic Statistics*, 13(3), 1130–1136.

Mishra, K. N., Aaggarwal, A., & Abdelhadi, E. (2010). An efficient horizontal and vertical method for online DNA sequence compression. *International Journal of Computer Applications*, https://doi.org/10.5120/757-954

Murugan, A., & Punitha, K. (2021). An efficient DNA sequence compression using small sequence pattern matching. *International Journal of Computer Science and Network Security (IJCSNS)*, https://doi.org/10.22937/IJCSNS.2021.21.8.37

Murugesan, G. (2020). Codon-based compression algorithm for DNA sequences with two-bit encoding. *European Journal of Molecular and Clinical Medicine*, 7(10), 33–41.

National Library of Medicine. (2024). *GenBank and WGS Statistics*. https://www.ncbi.nlm.nih.gov/genbank/statistics/. Accessed 29 March 2024.

Rosario Gilmary, & Murugesan, G. (2021). Bit reduction based compression algorithm for DNA sequences. *International Journal of Scientific Research in Science, Engineering and Technology*, https://doi.org/10.32628/IJSRSET218529

Syed Mahamud Hossein, De, D., Mohapatra, P. K. D., Mondal, S. P., Ahmadian, A., Ghaemi, F., & Senu, N. (2020). DNA sequences compression by GP2R and selective encryption using modified RSA technique. *IEEE Access*, https://doi.org/10.1109/ACCESS.2020.2985733

Tabus, I., Korodi, G., & Rissanen, J. (2003). DNA sequence compression using the normalized maximum likelihood model for discrete regression. In *Proceedings of the Data Compression Conference (DCC)*, https://doi.org/10.1109/DCC.2003.1194016

Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, https://doi.org/10.1109/TIT.1977.1055714