ORIGINAL RESEARCH ARTICLE | UHD JOURNAL OF SCIENCE AND TECHNOLOGY

# A Hybrid Genetic Algorithm-Particle Swarm Optimization Approach for Enhanced Text Compression

**Tara Nawzad Ahmad Al Attar***

*Department of Computer Science, College of Science, University of Sulaimani, Iraq*

## ABSTRACT

Text compression is a necessity for efficient data storage and transmission. Especially in the digital era, volumes of digital text have increased incredibly. Traditional text compression methods, including Huffman coding and Lempel-Ziv-Welch, have certain limitations regarding their adaptability and efficiency in dealing with such complexity and diversity of data. In this paper, we propose a hybrid method that combines Genetic Algorithm (GA) with Particle Swarm Optimization (PSO) to optimize the compression of text using the broad exploration capabilities of GA and fast convergence properties of PSO. The experimental results reflect that the proposed hybrid approach of GA-PSO yields much better performance in compression ratio than the standalone methods by reducing the size to about 65% while retaining integrity in the original content. The proposed method is also highly adaptable to various text forms and outperformed other state-of-the-art methods such as the Grey Wolf Optimizer, the Whale Optimization Algorithm, and the African Vulture Optimization Algorithm. These results support that the hybrid method GA-PSO seems promising for modern text compression.

**Index Terms:** Text Compression, Genetic Algorithm, Particle Swarm Optimization, Hybrid Algorithm, Data Storage Efficiency

## 1. INTRODUCTION

Text compression is vital in data management and transmission since efficient storage and high-speed communication are critical concerns in the digital era [1,2]. It can be said that text compression aims to transform textual data into the most compact form possible with fewer bits to store or transfer information without compromising the authenticity of a message. While there is an exponential increase in the volume of digital text generated across web content, scientific data, and communications, among others, the need for effective compression also increases [3,4]. In addition to saving storage space, efficient methods of text compression enhance the rates of data transmission [4], becoming extremely important to a wide range of applications, starting from telecommunications and file storage to Internet communications [5,6].

Traditional text compression, such as Huffman coding [7], LZW [8], and RLE [9], has been extensively applied due to their high ratios. However, these perform static, pre-defined encoding schemes that are not optimal for all data types. They often cannot adapt to different features of the text; sometimes, they give abysmal performance in compressing highly variable or complex data [10,11]. Furthermore, with the ever-growing diversity of formats for information exchange and the growing complexity of modern digital content, the need is for far more adaptive and intelligent methods of text compression [12,13].

Access this article online

DOI: 10.21928/uhdjst.v8n2y2024.pp63-74

E-ISSN: 2521-4217

P-ISSN: 2521-4209



**Corresponding author's e-mail:** Tara Nawzad Ahmad Al Attar, Department of Computer Science, College of Science, University of Sulaimani, Iraq. E-mail: tara.ahmad@univsul.edu.iq

Received: 17-09-2024 Accepted: 27-10-2024 Published: 13-11-2024

Within the framework of evolutionary algorithms, robust methods for solving, such as Genetic Algorithms (GAs) [14] and Particle Swarm Optimization (PSO) [15], have emerged as quite useful [16]. GAs draw inspiration from natural evolution, with the primary vehicle of evolving solutions provided by mechanisms of selection, crossover, and mutation across successive generations. GAs were effective in text compression since encoding schemes can be generated dynamically to better adapt to the compressed text structure and characteristics. However, a common issue with GAs is the difficulty posed by the convergence problem, mainly when dealing with large search spaces or complex optimization problems [17]. Contrary to this, PSO draws inspiration from the social behavior of animals in a population, such as birds flocking or fish schooling. PSO works on the premise of candidate solutions, exploring the solution space in search of the best solution [11,18]. A good share of success has been realized where PSO has been applied toward a near-optimum convergence rate in many optimization tasks. However, like GAs, PSO also has one major drawback: the propensity of the algorithm to converge prematurely to a suboptimal solution due to the lack of diversity within the population [19,20].

In this paper, a hybrid approach is proposed for optimizing text compression using the strengths of both GA and PSO. Our approach will try to achieve a better compression ratio with the integrity of the original text by jointly utilizing the exploration capabilities of GAs and the fast convergence properties of PSO. In this hybrid approach, the algorithm based on GA-PSO dynamically adjusts the encoding scheme of each character in the text. Therefore, it can optimize the real-time compression process about the nature of the input data. The hybrid approach obviates the deficiencies of standalone algorithms by leveraging GA's broad exploration capability and the fast convergence capability of PSO to refine the best candidates. Its synergy provides a more adaptive and efficient solution for text compression, especially when dealing with diverse and complex datasets. In this context, the main contributions of this paper are as follows:

- A novel hybrid algorithm was introduced that combines GA and PSO for text compression, achieving superior compression ratios while preserving text integrity.
- A comprehensive analysis of the proposed GA-PSO algorithm is conducted, benchmarking its performance against established techniques such as Grey Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), African Vulture Optimization Algorithm (AVOA), PSO, and GA.
- The trade-off between compression efficiency and computational complexity across various datasets was analyzed, offering insights into the algorithm's resource usage and performance balance.
- The study includes an assessment of the GA-PSO algorithm's adaptability to different text characteristics, demonstrating its robustness and flexibility in handling diverse text formats.
- A dynamic encoding scheme that adjusts based on the characteristics of the input data was proposed, further optimizing the compression process and enhancing efficiency for real-time applications.

The rest of the paper is organized in the following fashion. Section 2 reviews the related text compression work and highlights the classical approaches' limitations. Section 3 presents the proposed hybrid GA-PSO algorithm with sufficient details about the designs, implementations, and optimizations. Section 4 presents the experimental results and performance evaluation of the proposed hybrid method compared to existing methods. Finally, the paper is concluded in Section 5 with future research directions.

## 2. RELATED WORKS

Text compression, the focal point of many research studies for enhancing data processing and storage efficiency, has received extensive theoretical and practical investigation. The continuous search for new means for even better compression efficiency, the need to balance between speed and accuracy requirements, and specific challenges related to particular languages and contexts have driven much of this exploration. This continuous development is driven by continuously increasing demands of applications for speedier transmission, affordable storage, and better integrity of data, among other requirements in telecommunications and artificial intelligence, to name a few.

It is observed that in a recent survey about adaptive modeling strategies in text compression, the authors, in Bell *et al.* [21], classify approaches into three types: Finite-context modeling, finite-state modeling, and dictionary-based modeling. Finite-context modeling estimates the probability of a character given the preceding ones. Finite-state modeling generalizes this by condition upon state. In dictionary modeling, strings of characters are replaced by references to an adaptive dictionary. Here, the paper discusses the adaptation of methods to text types. It starts with reviewing the performance of these algorithms on various samples then goes over some of the directions for future research that this paper has taken in adaptive text compression.

Similarly, there have been developed practical algorithms for arithmetic coding in Moffat [22]. The authors further show that advanced models, such as move-to-the-front and variable-order Markov, apply to text compression. These word-matching and word-recording algorithms can exhibit text compression to an optimized rate of <2.2 bits per character for English text, with added fast-encoding and fast-decoding procedures to make them practical for real-life applications.

One of the critical challenges in the field of text compression is what is known as the zero-frequency problem, where novel events and, hence, new, never-previously-encountered events are challenging to encode efficiently. Witten and Bell [23] tries to solve the problem using a Poisson process model for the adaptive text compression system to predict new, unseen tokens. This theoretically sound model outperforms traditional empirical techniques and improves compression efficiency through predictive modeling that reduces the zero-frequency problem. In Brisaboa *et al.* [24], the authors review the recent word-based text compression techniques by proposing two new variants of Huffman encoding: End-Tagged Dense Code and (s, c)-Dense Code, which attempts to compress the natural language text effectively. There is a significantly improved compression ratio and speed when words instead of single characters are used as symbols. This study offers near-optimal compression, with a minor overhead due to the dense codes, and gains in search and access efficiency, so it is suitable for large-scale data-processing applications.

The next explores another approach toward text compression within a natural language processing (NLP) context, proposed by Li *et al.* [25]. The authors integrate explicit and implicit text compression into the Transformer encoder, a fundamental building block of modern NLP models. By centering around core or "backbone" information in the text, this research illustrates how compression can improve language representation, especially in deep understanding tasks. They show that compressor-enhanced models outperform the class of traditional transformer-based models in several NLP benchmarks, while compression is critical to improving efficiency and performance.

Apart from that, Sarker and Rahman [26] propose a new method of compressing Bengali text transliterated into English. The authors combine Huffman coding and an adjacent distance array using the transliterated text to minimize symbol counts, thus ensuring proper compression efficiency. The proposed technique can boast very promising compression ratios, especially in processing transliterated Bengali text; therefore, it may be a promising way of applying adaptive compression techniques in multilingual data processing environments. In Priyono and Mustafidah [27], one can compare popular data compression algorithms: Huffman, Shannon-Fano, and Half-Byte. The text data focused on Indonesian texts. Based on the results of applying the algorithm to the abstract of scientific research articles, Huffman is still better than other algorithms in terms of its compression ratio, making it suitable for Indonesian text compression. This work points out the importance of algorithm choice when compressing text, mainly based on language characteristics.

Gilbert *et al.* [28] discussed how large language models, such as GPT-4, are used in approximate text compression. It modifies the relevant treatment of the exact recovery to result in a new metric toward the evaluation of the goodness of the compressed text that allows it to maintain the intended meaning of the original. Results indicate the promising use of LLMs for compressing texts, especially in tasks that advocate content sense rather than recovery. Adeniji *et al.* [29] incorporate security into text compression by integrating Huffman coding with cryptographic algorithms that will counter the vulnerabilities in the data transmission, using RFID technology. This approach not only improves data compression efficiency but also strengths the protection of compressed data through encryption. Hence, this approach is relevant in those scenarios where data security and compression go hand in glove.

It finally presents the dynamic algorithm [30] for choosing the best compression technique suitable for steganography, whereby compressing texts is vital in placing secret information inside cover images. An adaptive algorithm increases performance in steganographic encoding by selecting the compression techniques that result in the smallest embedding space relative to the hidden message and characteristics of the cover image. These works together testify to the broad range of applications that involve text compression techniques for the benefit of machine learning models, protection of sensitive data, reduction of large data sets in storage, and optimization of big data. The domain further evolves with new models, algorithms, and integration technique proposals for fending off specific challenges regarding linguistic diversity, real-time data processing, and security in data transmission.

In recent years, numerous studies have explored hybrid techniques combining PSO and GA for various optimization challenges. For instance, in Garg [31], a PSO-GA hybrid approach addresses constrained optimization problems, effectively balancing exploration and exploitation through genetic operators while achieving superior solutions compared to traditional methods. Similarly, in Zhang *et al*. [32], the authors demonstrate the efficacy of a hybrid PSO-GA method in optimizing engine parameters, showcasing improved performance and emissions outcomes over conventional GA approaches. In addition, in Li *et al*. [33], a hybrid PSO-GA is utilized for optimizing heliostat fields, significantly enhancing daily energy collection during seasonal benchmarks. Furthermore, Sheikhalishahi *et al*. [34] present a hybrid GA-PSO method for Reliability Redundancy Allocation Problems, which enhance computational efficiency and reliability across various system architectures.

In this work, a novel approach was adopted by partitioning the population: half of the candidates are processed using GA for decision vector modifications, while the other half employs PSO to refine the solution vector. This strategy not only maintains computational efficiency but also prevents an increase in complexity, ensuring optimal performance. Notably, this study is the first to apply this hybrid algorithm specifically to text compression, contributing a unique perspective to the existing body of literature on PSO-GA techniques.

## 3. MATERIALS AND METHODS

In the following, we describe the materials and methods adopted in implementing and testing our study's hybrid text compression approach, which involves the integration of GA and PSO. We outline herein the basic principles of GA and PSO, including how these methods could be hybridized to improve performance. Further, this section will describe the datasets used for experiments, the evaluation metrics, and the computational setup. This section should, therefore, be utterly informative on what techniques and resources are available for the proposed solution.

### 3.1. Problem Formulation

This problem uses a GA-PSO to design an almost optimal encoding scheme for text compression. In the critical aspects of data storage and transmission, text compression makes one of the significant objectives to reduce the size of the text without losing information in it. Each text character will be encoded in this formulation using a binary string of variable length. The problem is formulated as an optimization problem that aims to minimize the total length of the compressed text by optimizing the encoding for each character.

That is, let $T$ be a source text formed by a set of characters $C = \{c_1, c_2, \ldots, c_n\}$ with frequencies $f(c_i)$ respectively. The task is to assign a unique binary string $e(c_i)$ to each character $c_i$ in such a way, the total length of encoded text would be the least. In mathematical terms, the length L(E) of encoded text using encoding E can be written as:

$$L(E) = \sum_{i=1}^{n} f(c_i) \cdot |e(c_i)| \tag{1}$$

where $|e(c_i)|$ is the length of the binary string assigned to character $c_i$, and $f(c_i)$ is the frequency of $c_i$ in the text. The objective is to minimize $L(E)$.

The proposed method searches for the encoding scheme $E^*$ that minimizes $L(E)$ by evolving a population of candidate encoding over successive generations. Each character's encoding is represented as a random binary string of length from 2 to 5 bits. The population's encoding has undergone evolution through various genetic operations such as selection, crossover, mutation, and PSO operators.

### 3.2. Fitness Function

The fitness function is designed to minimize the total length of the encoded text. For a given encoding $E$, the fitness function $F(E)$ is defined as:

$$F(E) = L(E) \tag{2}$$

Thus, a lower F(E) value corresponds to a better encoding scheme.

### 3.3. GA

The GA [35] is a robust search heuristic inspired by natural selection and genetic principles. This algorithm represents potential solutions to a problem as individuals within a population. These individuals evolve over generations to find the optimal solution to a given problem. In this study, the GA is applied to text compression by optimizing binary encodings for characters, aiming to reduce the overall size of the compressed text.

The GA begins by initializing a population of random encodings, where each encoding represents a binary string of variable length for each character in the text. This population represents potential solutions to the problem, and each individual (or solution) is evaluated using a fitness function.

The fitness function is a critical component of the GA, guiding the selection of the best individuals. In this case, the fitness function *f(E)* is defined as the total length of the compressed text:

$$f(E)=\sum_{i=1}^{n} f(c_i)\cdot\left|E(c_i)\right| \tag{3}$$

where $E(c_i)$ is the binary encoding for character $c_i$, $E(c_i)$ is the length of the binary string, and $f(c_i)$ is the frequency of occurrence of the character $c_i$ in the original text. In this way, the goal of the GA will be the minimization of this fitness function, which directly reduces the size of the encoded text.

The algorithm only proceeds to execute selection after evaluating the fitness of every individual within the population. Often, roulette wheel selection is used to apply selection within GAs, where selection happens with probabilities proportional to their fitness. Thus, better solutions are more likely to be selected to contribute to the next generation. In contrast, less optimal solutions also get a chance, and the diversity in the population is preserved.

After selection, the crossover can generate new individuals, called offspring, resulting from the combination of the encoding of two-parent individuals. In this paper, the single-point crossover is adopted. The crossover point is randomly chosen, and the segments of the binary strings are exchanged between two parents. This approach thus generates new individuals with mixed features from both parents, attempting to explore new areas in the solution space. The second step in GA is provided by mutation, where random changes within the binary encoding of the offspring are performed. This is done by flipping a bit in the encoding string with some probability, which is said to be a mutation rate. Mutation prevents a population from becoming too homogeneous and allows the algorithm to avoid local optima because it maintains genetic variation. Elitism is incorporated in the algorithm, ensuring that the fittest members of each generation are passed on to the next generation without any modification. This will maintain reasonable solutions and accelerate convergence toward the optimal or near-optimal solution.

The GA iterates for a fixed number of generations. It applies to all operators in every generation, namely selection, crossover, mutation, and elitism. Eventually, the population evolves while the algorithm converges to the best binary encoding that minimizes the size of the compressed text. This evolutionary process lets the GA discover efficient encodings, balancing exploring the solution space with exploiting the best solutions.

### 3.4. PSO

PSO [15] is an optimization in which inspiration for the algorithm was obtained from the collective behavior of swarms. Swarms refer generally to flocks of birds or schools of fish. Each particle of the swarm represents a solution to the problem; therefore, all the particles move within the solution space due to their own best-known position and the entire swarm best known. PSO can be powerful in solving optimization problems, such as text compression because it can efficiently explore large search spaces.

Every particle in the swarm computes the new position and the velocity using the formulas:

$$v_i(t+1) = wv_i(t)+c_1r_1(p_i-x_i(\mathrm{t}))+c_2r_2\,(g-x_i\,(t)) \tag{4}$$

$$x_i(t+1) = x_i(t)+v_i(t+1) \tag{5}$$

In the formulas below, $v_i(t)$ denotes the velocity of particle $i$ at step $t$, $x_i(t)$ is the position of particle $i$, $p_i$ does that particle find the personal best position so far, and g represents the global best position the swarm has found. Parameters $w$, $c_1$, and $c_2$ are the weights for inertia, personal influence, and social influence correspondingly, and $r_1$ and $r_2$ are random variates introducing diversity.

Due to the variation in their velocities and positions, particles move toward the optimal solution through every iteration. PSO can efficiently search for the best compressive settings in text compression, obtaining a minimum file size while retaining text quality. The algorithm is said to terminate at the swarm's convergence point to an optimal or near-optimal solution.

### 3.5. Proposed Method

This work presents a hybrid algorithm incorporating a GA with PSO for optimal text compression. The proposed hybrid algorithm splits the population into two groups at every iteration. Half of the population undergoes the evolutionary process of GA, while the other half is optimized using PSO. This approach represents an effort to combine the strengths of two optimization techniques: One is GA, which effectively explores the diverse solutions space through mutation and crossover operators, and the other is PSO, which converges well in refining solutions with its velocity and position updates. The hybrid method starts with creating random encoding schemes using the GA component. Each scheme here will map characters into a unique bitstring of variable lengths. These encoding schemes then evolve generation by generation through selection, cross-over, and mutation operations to reduce the size of the compressed text. The fitness function ranks each encoding scheme according to the compactness of a given text that it manages to achieve. The best encoding schemes are carried over to the next generation; genetic operations generate newer candidates.

In contrast, the PSO component views the encoding scheme as a particle in a search space. Every particle's position represents a solution, while its velocity describes the amount of movement in the search space. PSO updates each particle's position through its personal and global best positions found by the swarm. This, in turn, allows PSO to converge very fast onto promising solutions using efficient exploitation of the search space. It initializes a population of encoding schemes and splits them into one for the GA algorithm to process and the other sub-population for PSO to optimize. After every iteration, the best solutions from both groups are pooled into a new population for the next iteration. This hybrid ensures that, through GA, the algorithm explores many possible solutions and then refines the best ones with PSO. It checks the balance between exploration and exploitation by dividing the population to find the optimal text compression scheme. This approach allows the algorithm to balance exploration and exploitation in finding the optimal text compression scheme.

The flow of the algorithm can be briefed as follows:

1. Initialize the population with random encoding schemes.
2. Divide the population into two groups, one for GA and the other for PSO
3. Apply GA operations selection, crossover, and mutation among the individuals in one group.
4. Perform PSO updates on the other group.
5. Combine the best individuals of both groups.
6. Perform the above processes for a fixed number of generations or till convergence.
7. Return the best encoding scheme using the smallest size compressed text.

This hybrid method is quite suitable for text compression problems, as it efficiently investigates the large and complicated solution space of possible encoding schemes. A combination of GA with PSO may provide solutions to the algorithm such that the obtained solution reduces the length of the compressed text and does so efficiently over multiple iterations. Fig. 1 illustrates the overall process of the proposed method.

## 4. RESULTS AND DISCUSSION

Testing the proposed hybrid GA-PSO algorithm showed good performance with a different number of text lengths and even impressively performed well in the metrics derived from compression and processing, showing its strength and adaptability for various text data.

### 4.1. System Specification

The methods are executed on Google Colab Pro, a cloud environment offering powerful computational facilities to implement the proposed method. It entertains Colab Pro with GPU and TPU resources, among other benefits, with enhanced RAM and higher execution speed than basic Colab. The advantages derived from this environment are rather attractive for optimization algorithms such as GA-PSO, which deal with massive population sizes and multiple generations.

The virtual machine applied in Colab Pro uses an Intel Xeon processor running at 2.20GHz, with 25GB of RAM and an NVIDIA Tesla P100 or V100 GPU, whichever is available. With this configuration, the hybrid algorithm will have enough processing power to handle large datasets and iterate over many generations within reasonable time limits. Besides, the available disk space is 166 GB; thus, there is ample room to store intermediate results or log experimental data.

Besides that, Google Colab Pro supports Python libraries such as Numpy and Scipy, which provide support for effectively handling data structures and the mathematical operations at the heart of both components of the hybrid algorithm.

### 4.2. Dataset

GPT-4 has prepared a dataset with four different contexts to assess the hybrid GA-PSO algorithm's performance. These contexts targeted testing the algorithm on each text length, which could range from very short to very long texts. Each context represents another challenge for the compression algorithm, enabling us to establish how well it can adapt to different compression scenarios.

- Context 1 (medium-length sentence)
  - Text: "The golden rays of the setting sun bathed the city in a warm, peaceful glow."
  - Character Count: 77 characters (including spaces and punctuation)
  - Bit Size: 616 bits (assuming 8 bits per character).

- Context 2 (paragraph)
  - Text: "In a small village at the edge of a great forest, people lived simple lives, working the land and

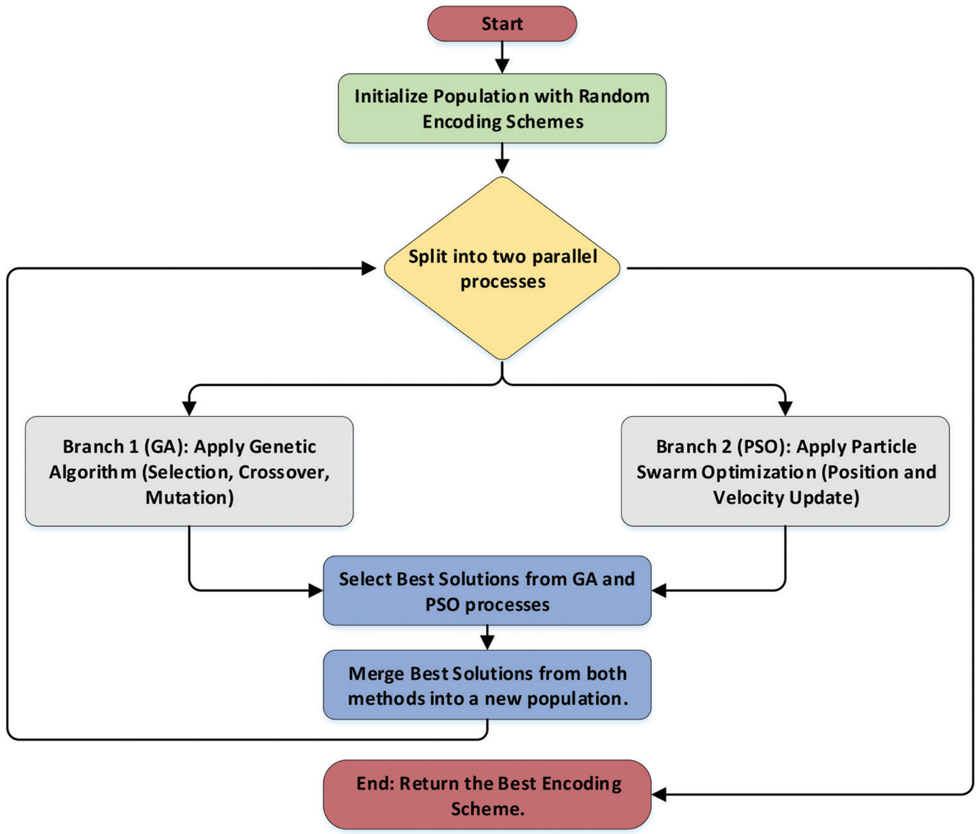


**Fig. 1.** General process of the proposed method.

raising their families. The sun would rise daily over the mountains, casting a golden light over the fields. Life was peaceful and predictable, with the rhythms of nature guiding the villagers' every action."
- Character Count: 343 characters
- Bit Size: 2,744 bits.

- Context 3 (more extended passage)
  - Text: "The world has changed dramatically over the past century. What was once a planet dominated by sprawling cities and bustling industries had now returned to quiet solitude. The forests had regrown, reclaiming much of the land cleared for agriculture. Rivers ran clear again, and animals roamed freely without fear of human interference. Those who remained lived in harmony with the Earth, understanding that balance was key to survival."
  - Character Count: 453 characters
  - Bit Size: 3624 bits.

- Context 4 (full-text sample)
  - Text: "In the early days of the new era, when humanity first began to understand the true cost of its actions, many doubted that change was possible. But over time, as ecosystems began to collapse and resources grew scarcer, people were forced to confront the reality of their situation. A global movement arose, driven to preserve the planet for future generations. It wasn't easy, and many sacrifices were made, but eventually, a new equilibrium was reached, one in which nature and human society coexisted in harmony."
  - Character Count: 451 characters
  - Bit Size: 3608 bits.

These contexts provide a comprehensive range of text lengths, from short phrases to entire passages, allowing us to evaluate the hybrid algorithm's compression ratio, encoding time, and decoding accuracy.

### 4.3. Evaluated Algorithms

Besides the proposed hybrid GA-PSO, we test five more algorithms to compare with. These include nature-inspired and conventional optimization algorithms, providing a broad set of algorithms for text compression. The WOA [36] is a metaheuristic inspired by the hunting strategy of humpback whales using bubble nets. It is a practical algorithm for searching for global optima in various optimization problems, including text compression. GWO [37] has been inspired by the leadership hierarchy and hunting strategy of grey wolves. This algorithm fulfilled the requirement for the proper balance between exploration and exploitation in the search space.

The AVOA [38] is one of the newer nature-inspired algorithms that model the movement characteristics of vultures when foraging for food. AVOA performed better than some state-of-the-art algorithms in significant, multi-dimensional optimization problems. Along with the nature-driven approaches, two classical algorithms are introduced: a standard GA, which is very popular in optimization problems because it searches a vast solution space using crossover and mutation, and PSO, known as an efficient fine-tuning optimizer in the search space. These two algorithms form the basis for comparing with the hybrid approach.

### 4.4. Computational Complexity

The time complexity of a proposed algorithm can be expressed in terms of key components such as population size, number of iteration, and the complexity of individual operations such as fitness evaluation, selection, crossover, and mutation. Initially, population generation incurs a complexity of $O(P)$, where $P$ is the population size. Fitness evaluation, applied to each individual in the population, has a complexity of $O(G \times P \times F)$, where $G$ is the number of generations and $F$ is the time required for the fitness function. The selection process, involving sorting based on fitness, adds $O(G \times P \log P)$ to the overall complexity. Crossover and mutation operations, with their respective complexities $O(C)$ and $O(M)$, further contribute $O(G \times P \times [C+M])$. Combining these factors, the total time complexity $T$ is expressed as $O(G \times [P \times (F+\log P+C+M)])$. This formula encapsulates the computational costs across all generations and operations involved in the optimization process.

### 4.5. Numerical Results

This section discusses how we applied the hybrid GA-PSO algorithm to our developed dataset, which includes all text types, from few-word phrases to longer texts. Our evaluation is directed at critical metrics such as compression ratio, encoding time, and decoding accuracy. We assess the algorithm's effectiveness in diverse text handling with these metrics and deduce its performance levels across different contexts. This work investigates the performance of various algorithms on several datasets, all of which provided different results concerning the size at which data were finally compressed when using a particular method. Further, the subsequent sections compare these methods, clearly defining the best and worst performers regarding compression efficiency. The baseline size of every dataset represents the original size against which the reduction attained by an algorithm will be measured. These results are presented as tables and figures to achieve numeric and visual insights into algorithm performance. The key results are summarized for each table and figure, outlining the difference between the original and compressed size.

Table 1 presents the performance results of different algorithms concerning their performance for compressing an original size of 616 bits. Indeed, the results indicate significant gaps in the compression efficiency of the method proposed here compared to the others. The current paper will present the best compression performance that can get as low as 184 bits; thus, it is the most effective in this context. Other approaches such as GWO and WOA resulted in sizes of 204 bits and 195 bits, respectively, where the poorest performance was given by PSO and GA to sizes of 208 and 209 bits, respectively. For this case, the proposed method performed better; it achieved a minimum compressed size compared to other methods. The compressed sizes of the two differ because of the variation in algorithmic strategies to recognize the patterns and optimize accordingly. Overall, the proposed method performs better than all of these. In contrast, others have very significant size reduction compared to the original, GA and PSO being the worst in this particular context.

Results across various algorithms for the original size of 2744 bits are tested and are shown in Table 2 again; the proposed method shows the best compression performance, with the

**TABLE 1: Algorithm Performance Outcomes for Context 1**

| Original size: 616 bits | Proposed method | Grey Wolf optimization | Whale optimization algorithm | African vulture optimization algorithm | Particle Swarm optimization | Genetic algorithm |
|---|---|---|---|---|---|---|
| Compressed size | 184 bits | 204 bits | 195 bits | 199 bits | 208 bits | 209 bits |
| Best Encoding | {'T': '10', 'h': '01', 'e': '00', ' ': '11', 'g': '0011', 'o': '100', 'l': '10', 'd': '00', 'n': '11', 'r': '00', 'a': '0101', 'y': '11', 's': '10', 'f': '101', 't': '11', 'i': '0000', 'u': '011', 'b': '000', 'c': '00', 'w': '10', 'm': '01', ',': '10', 'p': '011', '.': '000'} | | | | | |
| Compressed Text | 1001001100111001000001111000101111011100101111101001110001111000011001111100111111000010111010000111 10100110000000111111000011110101111001010001101101100010100001010111011001110100100000 | | | | | |

data size being compressed to 744 bits, while GWO and WOA managed to compress it only up to sizes of 847 and 808 bits, respectively. The most exciting thing is that while AVOA and PSO could compress data to 811 and 847 bits, GA could achieve a compressed size of 825 bits. These compressed data size variations can increase these algorithms' capacity to optimize the minimized data size. The proposed method does the most prominent size reduction, which could do with about a tenfold decrease compared to the original, while the most minor reduction was by PSO and GWO. Therefore, the proposed method has more power in dealing with complicated patterns and redundancy inside big datasets.

Furthermore, in Table 3, with an original size of 3624 bits, the best result using the proposed method compressed the data into 1167 bits. Other methods using GWO and WOA compressed the data into 1249 and 1291 bits, respectively, while AVOA and PSO compressed the data into 1337 and 1304, respectively. In this regard, GA was relatively better, compressing the size to 1240 bits. The ranges of the compressed sizes indicate that the proposed method outperforms the other optimization algorithms in terms of efficiency. The performance differences among the methods increase when the dataset size increases, showing that the proposed method is particularly well suited for dealing with significant or complicated data structures and reducing redundancy. Although performing satisfactorily, the GA and PSO are moderately efficient, while GWO and WOA performed a little better but still lag significantly behind the proposed method.

In Table 4, the original size of 3608 bits was compressed using different methods; again, the proposed method outperformed with a size reduction of 1404 bits. Other compressed sizes by various algorithms are GWO-1562 bits, WOA-1497 bits, PSO-1449 bits, and GA-1443 bits. Thus, GA reduced the size by 6 bits, a minor enhancement given PSO's result. It showed that the proposed method had a significant

**TABLE 2: Algorithm performance outcomes for context 2**

| Original size: 2,744 bits | Proposed method | Grey Wolf optimization | Whale optimization algorithm | African vulture optimization algorithm | Particle Swarm optimization | Genetic algorithm |
|---|---|---|---|---|---|---|
| Compressed size | 744 Bits | 847 Bits | 808 Bits | 811 Bits | 847 Bits | 825 Bits |
| Best Encoding | {'I': '001', 'n': '10', ' ': '00', 'a': '00', 's': '10', 'm': '100', 'l': '10', 'v': '101', 'i': '1100', 'g': '0111', 'e': '11', 't': '00', 'h': '01', 'd': '10', 'o': '011', 'f': '000', 'r': '00', ',': '00', 'p': '101', 'w': '0111', 'k': '1001', '.': '10', 'E': '00', 'y': '10', 'u': '101', 'c': '01', 'L': '100', 'b': '1110', '"': '1101'} | | | | | |
| Compressed Text | 00110000000101000010100010111001010000111110000000000001110011100111110001100000000000111001100000000000011<br>001110000000101110111011011001011001011110001011001001011011001011001011110000001110110010011100100111 0<br>0000111001000101000001010000000110010110010011100000111110000000000001001100101100111010000010111001000 1<br>00010000000001110010101100001110111011010000011001011000111011100000000111001000111011000001100101000000 1<br>00100011001001110000000111011101011100010110001110100000111011100000000111000001100111010101000100110000 0<br>01100011100100010111000111000101100000101000101001110110001000011101011000001111100000100000111000001 10<br>000110010000110000010000010100110001111011100101100100111000001110010111001010000111110010110100111011 1<br>0010000000100110001110 10 | | | | | |

**TABLE 3: Algorithm performance outcomes for context 3**

| Original size: 3,624 bits | Proposed method | Grey Wolf optimization | Whale optimization algorithm | African vulture optimization algorithm | Particle Swarm optimization | Genetic algorithm |
|---|---|---|---|---|---|---|
| Compressed size | 1167 Bits | 1249 | 1291 Bits | 1337 Bits | 1304 Bits | 1240 Bits |
| Best Encoding | {'T': '10', 'h': '011', 'e': '10', ' ': '11', 'w': '0001', 'o': '10', 'r': '01', 'l': '00', 'd': '011', 'a': '010', 'c': '10', 'n': '11', 'g': '01011', 'm': '11', 't': '010', 'i': '111', 'y': '01', 'v': '00', 'p': '00101', 's': '0110', 'u': '00', '.': '00010', 'W': '0101', 'b': '01', 'f': '10', 'q': '11010', ',': '11', 'R': '01', 'E': '000', 'k': '0000'} | | | | | |
| Compressed Text | 1001110110001100100011110110100111110011010110101110011110110101011010010111100100000011110001001110 1<br>0011101100101010011001011101011010000101000101101010110100101100010100110111011101011010110010100010 1<br>110010110111011111111010010100111101011101100010101010000100111110101111101110101111001101101011011110<br>10001100100011111010111111111011000110010011111001101101101001111111000011101100100001111001111010101<br>1010110110010010010101110101111010001111001011011010001110100001110000101110011101110100110011001001 1<br>0110110100111101100101101100001111111011010000101111111111010111111001001111101011010011101100010110 1<br>1110100110100101101101001111011010111110001001001100111110100111010010110111110000001000011000010110 1<br>1110010010110110101011111000100100111010010110101111111111010110111101011111110100001101101100101110 01<br>11110011010000111110001111010011100001011101001001111010110110011010111111111010100110100110111010000<br>10111001110011010110001011101101101101011111100111100111001001111111111101101001111011011100011110100<br>11110100111011000010010100111111001101110010110010010110111111101011110100110100101101010000101110101<br>10001010011011000010011101010110110000100111000100000010 | | | | | |

**TABLE 4: Algorithm performance outcomes for context 4**

| Original size: 3,608 bits | Proposed method | Grey Wolf optimization | Whale optimization algorithm | African vulture optimization algorithm | Particle Swarm optimization | Genetic algorithm |
|---|---|---|---|---|---|---|
| Compressed size: | 1404 | 1562 | 1497 | 1448 | 1449 | 1443 |
| Best Encoding: | {'I': '11', 'n': '10', ' ': '001', 't': '00', 'h': '11', 'e': '00', 'a': '1101', 'r': '01', 'l': '11', 'y': '10', 'd': '01101', 's': '01', 'o': '011', 'f': '0100', 'w': '100', ',': '001', 'u': '101', 'm': '110', 'i': '000', 'b': '111', 'g': '100', 'c': '010', 'p': '00', '.': '11', 'B': '0101', 'v': '10', 'A': '010', '"': '01', 'q': '001', 'x': '11'} | | | | | |
| Compressed Text: | 1110001001100001001101011110001011011101100100101101000010011000011000100001000111010010011001100100<br>1111011101101100000010001010000001010000111100100110110001000110011011001101000101001101100110100100<br>1100000100011010000101001101000010110100001000000100111010100000001110010010011101101101000101101011101<br>1110000001101001001111010000101011110110100000011001101010010001101010001111100110010101101000010111 00<br>0010010000001100000100111010100100010011011001000011001001111001001101100010001100101001111111101000 10<br>0001110110011010010100010111010101000010011000100100001010101101010100001001001000001100110000110 0000<br>1000010100011010100001101001000110010100111001000101110000001001100001010011011100000010001011010000100<br>1100000001001010000010111010000000111011001010001100110111111101110011100111000110001000001110101011010<br>0001001011010100010001000111110001001100001011010001000010000100011001000100010001100000010011000 01001<br>1110110000000101000110100101001010010101000011000010000111010000000111001110011100001100110101100100 00<br>1001101011000100111011001101001110110110100010111010100100001000000010000100110000001000011101101011010<br>0001001111101000010010001000101110111111000100111010011000100001000011010001100011101000101110001100 1<br>10101001010011010101100011010010010111000000100010001100110000101100110110100101010000111011001101001 1<br>110111011011000101011010000000001000101001100110000100000110100100010001111101011100111 01011 | | | | | |

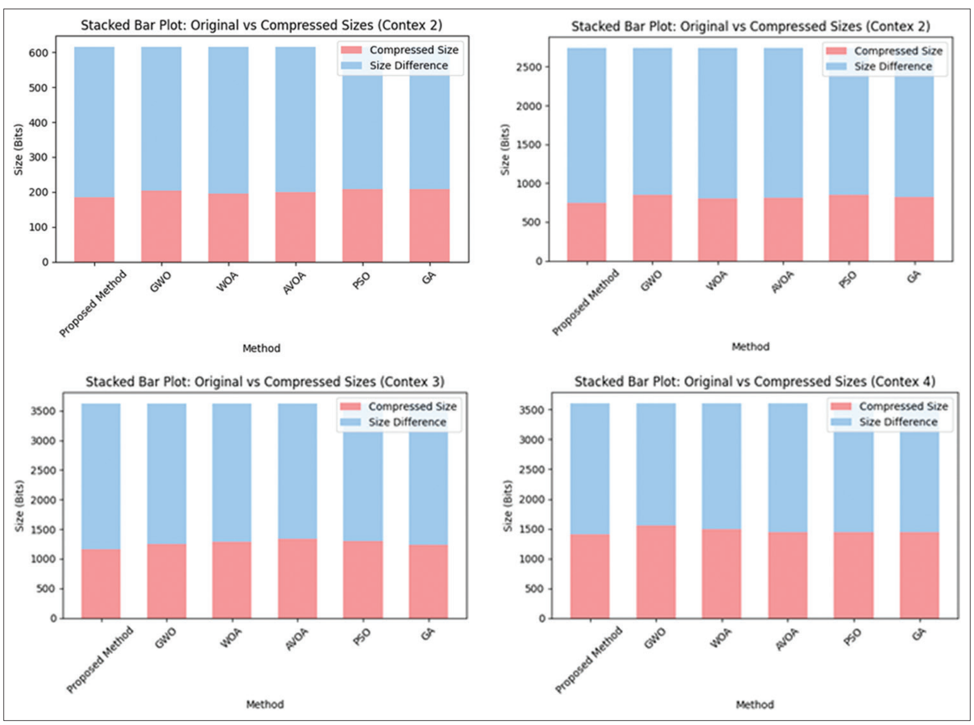


**Fig. 2.** Stacked bar plot comparing original and compressed sizes across different methods.

advantage in size reduction compared to the other algorithms; hence, it was the top-performing method for this dataset. The methods vary more subtly in this table compared to the previous ones. The proposed one is outstanding due to its ability to reduce size and retain key data elements. While the other algorithms also show different compression levels, the proposed method remains the most efficient.

Fig. 2 compares the original size with that compressed by various methods in a stacked bar plot. In all data sets, the original size is the same; this naturally serves as the apparent reference value against which the various compressed sizes achieved through different algorithms are compared. The stacked bars give evidence about the degree of compression achieved effectively by the techniques, each showing the

**TABLE 5: Algorithm Performance Outcomes based on time consumption criteria (s) for context 1–4**

| Dataset | Proposed method | Grey Wolf optimization | Whale optimization algorithm | African vulture optimization algorithm | Particle Swarm optimization | Genetic algorithm |
|---|---|---|---|---|---|---|
| Context 1 | 2.45 | 2.68 | 2.54 | 2.25 | 2.39 | 2.51 |
| Context 2 | 3.17 | 4.52 | 4.10 | 2.94 | 3.09 | 4.27 |
| Context 3 | 4.27 | 5.36 | 4.46 | 4.00 | 4.15 | 4.33 |
| Context 4 | 5.95 | 6.78 | 6.08 | 5.47 | 5.85 | 6.08 |

smallest compressed size across the datasets obtained by the proposed method. This visually evidences just how effective the method proposed herein was in outperforming the others such as GWO, WOA, AVOA, PSO, and GA, which show larger sizes. This is summarized in the relative performances of each method in terms of compression, as shown in Figure below, with the proposed method yielding almost always the best size reduction. The dispersion in the plot also makes clear the variability in the other methods' performances, with GWO, WOA, and PSO falling behind the proposed method most of the time. GA sometimes yields competitive results but often ends up worse than the proposed method. The figure confirms the numerical data presented in the tables, showing the size difference between the original and each of the methods' compressed sizes, thereby reconfirming that the proposed method is the most efficient for any of the datasets considered.

Table 5 shows the performance results of six optimization algorithms, namely Proposed Method, GWO, WOA, AVOA, PSO, and GA, in terms of time consumption in seconds for four different contexts. For Context 1, AVOA has the best performance with a time of 2.25 s, but GWO is at 2.68 s, the worst. The performance of the Proposed Method was faster than GWO, WOA, and GA, taking 2.45 s but slower compared to AVOA and PSO.

Context 2: The Proposed Method consumed 3.17 s, which defeated GWO, WOA, and GA, with performances over 4 s, whereas AVOA outperformed it at 2.94 s. In Context 3, the Proposed Method again was competitive at 4.27 s, behind only AVOA at 4.00 s yet faster than all other algorithms. Finally, for Context 4, the Proposed Method completed its process in 5.95 s, ranking above all except AVOA, which finished in 5.47 s. GWO was generally the slowest algorithm in most contexts. In general, the Proposed Method did a great job in various contexts and generally came up among the best algorithms when taking into consideration the consumption of time.

## 5. CONCLUSION AND FUTURE WORKS

Better text compression methods will be required due to the ever-growing rate at which the digital world produces text data. This work considers a hybrid Genetic-PSO technique for solving common issues related to text compression, primarily when highly variable or complex data is handled. It can be inferred that the proposed method has successfully implemented a combination that provided a robust, adaptive, and efficient text compression algorithm by considering the exploratory advantages of GA with the rapid convergence attributes of PSO. Experimental results indicate that the hybrid GA-PSO outperforms the traditional algorithms by a large margin in terms of achieving better compression ratios without losing the integrity of the original text. Moreover, the hybrid has proven its strength on quite good performance on various datasets, proving its ability to adapt to multiple natures and formats of text.

The hybrid GA-PSO algorithm will be further invested with advanced adaptive techniques to maintain the balance between exploration and exploitation for better performance. Using machine learning models to predict the optimum parameters of compression based on the features of the text is likely to enhance compression efficiency further. More evaluation of the proposed approach with real-time applications and larger datasets is needed for deeper insight into its scalability and effectiveness in diverse contexts. Further research on the integrated approach of the GA-PSO with the encryption algorithms may open new avenues in those scenarios where integrity and security both become critical issues in front of compressed data. Combining adaptive optimization techniques can set new standards for future text compression methods and show a way for more intelligent and responsive data management methods.

## REFERENCES

[1] M. V. Mahoney. "*Fast Text Compression with Neural Networks. In: Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference* (*FLAIRS 2000*)". FLAIRS, 2000.

[2] B. K. Kim, H. K. Song, T. Castells and S. Choi. "On Architectural Compression of Text-to-Image Diffusion Models. *In Proceedings of the Neural Information Processing Systems (NeurIPS 2023)*. New Orleans, Louisiana, United States of America.

[3] U. Manber. "A text compression scheme that allows fast searching directly in the compressed file". *ACM Transactions on Information Systems* (*TOIS*), vol. 15, no. 2, pp. 124-136, 1997.

[4] Z. Jiang, M. Yang and M. Tsirlin. “Low-resource” text classification: A parameter-free classification method with compressors”. In: *Findings of the Association for Computational Linguistics*: ACL, Stroudsburg, PA, 2023.

[5] Y. Marton, N. Wu and L. Hellerstein. “*On Compression-based Text Classification. In Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005*”. Springer, Santiago de Compostela, Spain, 2005.

[6] P. Lewan and C. Khancome. “*Bit-Level Affixation Text Compression Algorithms. In: 2024 21st International Joint Conference on Computer Science and Software Engineering* (*JCSSE*)”. IEEE, 2024.

[7] A. Moffat. “Huffman coding”. *ACM Computing Surveys*, vol. 52, no. 4, pp. 1-35, 2019.

[8] H. N. Dheemanth. “LZW data compression”. *American Journal of Engineering Research*, vol. 3, no. 2, pp. 22-26, 2014.

[9] E. P. Capo-Chichi, H. Guyennet and J. M. Friedt. “*K-RLE: A New Data Compression Algorithm for Wireless Sensor Network. In 2009 Third International Conference on Sensor Technologies and Applications*”. IEEE, 2009.

[10] T. Li., T. Zhao, M. Nho and X. Zhou. “*A Novel RLE & LZW for Bit-stream compression. In: 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology* (ICSICT)”. IEEE, 2016.

[11] Y. Zhou, F. Zhang, T. Lin, Y. Huang, S. Long, J. Zhai and X. Du. “*F-TADOC: FPGA-Based Text Analytics Directly on Compression with HLS. In: 2024 IEEE 40th International Conference on Data Engineering* (*ICDE*)”. IEEE, 2024.

[12] A. Moronfolu and D. Oluwade. “An enhanced LZW text compression algorithm”. *The African Journal of Computing and ICT*, vol. 2, no. 2, pp. 13-20, 2009.

[13] F. Zhou, X. Huang, P. Zhang, M. Wang, Z. Wang, Y. Zhou and Y. I. N. Haibing. “*Enhanced Screen Content Image Compression: A Synergistic Approach for Structural Fidelity and Text Integrity Preservation*”. ACM Multimedia, New York, 2024.

[14] S. Mirjalili. “Genetic algorithm”. In: *Evolutionary Algorithms And Neural Networks: Theory and Applications*. Springer, Cham, pp. 43-55, 2019.

[15] J. Kennedy and R. Eberhart. “*Particle Swarm Optimization. In: Proceedings of ICNN’95-International Conference on Neural Networks*”. IEEE, 1995.

[16] V. Kachitvichyanukul. “Comparison of three evolutionary algorithms: GA, PSO, and DE”. *Industrial Engineering and Management Systems*, vol. 11, no. 3, pp. 215-223, 2012.

[17] L. Haldurai, T. Madhubala and R. Rajalakshmi. “A study on genetic algorithm and its applications”. *International Journal of Computer Sciences and Engineering*, vol. 4, no. 10. pp. 139-143, 2016.

[18] M. P. Song and G. C. Gu. “*Research on Particle Swarm Optimization: A Review. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics* (*IEEE Cat. No. 04EX826*)”. IEEE, 2004.

[19] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh and S. Mirjalili. “Particle swarm optimization: A comprehensive survey”. *IEEE Access*, vol. 10, pp. 10031-10061, 2022.

[20] J. Zhou, R. Du, T. Yushan, J. Yang, Z. Yang, W. Luo, Z. Luo, X. Zhou and W. Hu. “*Context Compression and Extraction: Efficiency Inference of Large Language Models. In: International Conference on Intelligent Computing”*. Springer, 2024.

[21] T. Bell, I. H. Witten and J. G. Cleary. “Modeling for text compression”. *ACM Computing Surveys*, vol. 21, no. 4, pp. 557-591, 1989.

[22] A. Moffat. “Word-based text compression”. *Software: Practice and Experience*, vol. 19, no. 2, pp. 185-198, 1989.

[23] I. H. Witten and T. C. Bell. “The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression”. *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1085-1094, 1991.

[24] N. R. Brisaboa, A. Fariña, G. Navarro and J. R. Paramá. “Lightweight natural language text compression”. *Information Retrieval Journal*, vol. 10, pp. 1-33, 2007.

[25] Z. Li, Z. Zhang, H. Zhao, R. Wang, K. Chen, M. Utiyama and E. Sumita. “Text compression-aided transformer encoding”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3840-3857, 2021.

[26] P. Sarker and M. L. Rahman. “*Introduction to Adjacent Distance Array with Huffman Principle: A New Encoding and Decoding Technique for Transliteration Based Bengali Text Compression. In Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2020*”. Springer, 2021.

[27] E. Priyono and H. Mustafidah. “Text compression using the Shannon-fano, Huffman, and half-byte algorithms”. *International Journal of Scientific Research and Management*, vol. 12, pp. 1422-1427, 2024.

[28] H. Gilbert, M. Sandborn, D. C. Schmidt, J. Spencer-Smith and J. White. “*Semantic Compression with Large Language Models. In: 2023 Tenth International Conference on Social Networks Analysis, Management and Security* (*SNAMS*)”. IEEE, 2023.

[29] O. D. Adeniji, O. E. Akinola, A. O. Adesina and O. Afolabi. “*Text Encryption with Advanced Encryption Standard* (*AES*) *for Near Field Communication* (*NFC*) *Using Huffman Compression. In: International Conference on Applied Informatics*”. Springer, 2022.

[30] J. R. Jayapandiyan, C. Kavitha and K. Sakthivel. “Optimal secret text compression technique for steganographic encoding by dynamic ranking algorithm”. *Journal of Physics: Conference Series*, vol. 1427, p. 012005.

[31] H. Garg. “A hybrid PSO-GA algorithm for constrained optimization problems”. *Applied Mathematics and Computation*, vol 274, pp. 292-305, 2016.

[32] Q. Zhang, R. M. Ogren and S. C. Kong. “A comparative study of biodiesel engine performance optimization using enhanced hybrid PSO-GA and basic GA”. *Applied Energy*, vol. 165, pp. 676-684, 2016.

[33] C. Li, R. Zhai. H. Liu, Y. Yang and H. Wu. “Optimization of a heliostat field layout using hybrid PSO-GA algorithm”. *Applied Thermal Engineering*, vol. 128. pp. 33-41, 2018.

[34] M. Sheikhalishahi, V. Ebrahimipour, H. Shiri, H. Zaman and M. Jeihoonian. “A hybrid GA-PSO approach for reliability optimization in redundancy allocation problem”. *The International Journal of Advanced Manufacturing Technology*, vol. 68, pp. 317-338, 2013.

[35] J. H. Holland. “*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence”*. MIT Press, United States, 1992.

[36] S. Mirjalili and A. Lewis. “The whale optimization algorithm”. *Advances in Engineering Software*, vol. 95, pp. 51-67, 2016.

[37] S. Mirjalili, S. M. Mirjalili and A. Lewis. “Grey wolf optimizer”. *Advances in Engineering Software*, vol. 69, pp. 46-61, 2014.

[38] B. Abdollahzadeh, F. S. Gharehchopogh and S. Mirjalili. “African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems”. *Computers & Industrial Engineering*, vol. 158, p. 107408, 2021.