JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 6 No. 1 (2025) 69-85
https://publisher.uthm.edu.my/ojs/index.php/jscdm

A Comparative Study of Metaheuristic Optimization Algorithms on Distinct Benchmark Functions

Muhamad Asyraf Anuar[1], Rosziati Ibrahim[1]*, Nurezayana Zainal[1], Mazidah Mat Rejab[1], Hana Hachimi[2]

[1] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400, Parit Raja, Batu Pahat, Johor, MALAYSIA*

[2] *National School of Applied Sciences, Ibn Tofail University of Kenitra, Kenitra, 14000, MOROCCO*

*Corresponding Author: rosziati@uthm.edu.my
DOI: https://doi.org/10.30880/jscdm.2025.06.01.005

Article Info

Received: 8 December 2024
Accepted: 20 June 2025
Available online: 30 June 2025

Keywords

Metaheuristic algorithm, metaheuristic optimization, benchmark functions, optimization techniques

Abstract

Metaheuristic optimization algorithms are widely applied to tackle optimization problems across various fields. Recently, these algorithms have gained prominence over traditional deterministic methods for solving complex optimization issues. However, no single technique is universally effective for all types of optimization challenges. As a result, researchers have focused on enhancing existing metaheuristic methods or creating new ones. Numerous nature-inspired meta-heuristic algorithms have emerged to address complex optimization problems. This study evaluates and compares the performance of four algorithms, Particle Swarm Optimization (PSO), Differential Evolution (DE), Grey Wolf Optimizer (GWO), and Salp Swarm Algorithm (SSA) on five benchmark functions, including both unimodal and multimodal types. Experimental results demonstrate that GWO consistently outperforms the other algorithms in terms of solution quality and convergence speed. GWO achieved the lowest average error across all five benchmark functions and demonstrated the fastest convergence overall, reaching near-optimal solutions in under 150 iterations for most cases and as few as 66 iterations on the simplest function. DE ranked second, while PSO and SSA trailed behind. Statistical analysis using the Wilcoxon signed-rank test confirmed that GWO's superiority is statistically significant ($p < 0.05$) when compared to all other algorithms, even after applying Holm-Bonferroni correction.

1. Introduction

Optimization problems permeate a variety of fields, including engineering, medical imaging, urban development, deep learning, and more. These domains form the backbone of modern technological and scientific advancement, and the presence of optimization problems within them highlights their complexity and importance. To tackle these challenges, numerous optimization algorithms have been developed, each tailored to specific problem types. In scenarios with multiple possible solutions, optimization algorithms help identify the most optimal outcome.

Among these, metaheuristic optimization algorithms, particularly those inspired by natural phenomena, have proven effective in addressing complex and nonlinear optimization problems. These algorithms blend random exploration with guided search mechanisms to efficiently evaluate and select the best solutions from vast solution spaces. Broadly, optimization algorithms fall into two main categories[1]. The first includes traditional optimization methods, such as gradient descent and Newton's method. These are generally easier to understand

but may be time-consuming and typically yield only a single solution per run. The second category encompasses metaheuristic optimization algorithms, which are designed to address the limitations of traditional methods.

1.1 Metaheuristics Optimization Algorithm

Metaheuristic optimization algorithms are a flexible and powerful class of nature-inspired, derivative-free solvers well-suited to complex, real-world problems where traditional analytical approaches fall short [2]. The four main categories of these algorithms are swarm-based, evolutionary, physics-based, and human-based [3]. Evolutionary algorithms (EAs), for instance, draw on the principles of Darwinian natural selection, with genetic algorithms as a well-known example that mimics biological evolution [4]. Over the past two decades, metaheuristic optimization algorithms have been extensively adapted, modified, or hybridized with other intelligent approaches across various engineering applications, from Proton Exchange Membrane Fuel Cell (PEMFC) design [5] to software testing such as its application in Test Case Prioritization [6, 7]. Maheuristics are particularly effective for problems with complex constraints or mixed-integer variables [8], where solutions are adjusted to meet predefined bounds and constraints are managed to transform constrained problems into unconstrained ones. Despite their widespread adoption, there remain questions about the convergence behavior and effectiveness of metaheuristic optimization algorithms on whether they can reliably achieve global optimality in diverse problem domains. The flexibility of these algorithms continues to be a focal point of research, as scholars work to improve their robustness and generalizability across applications.

1.2 Benchmark Function

Benchmark functions play a critical role in evaluating and comparing the performance of metaheuristic optimization algorithms. These functions present standardized problem instances with known optima, enabling researchers to assess how well an algorithm performs on artificial test problems that simulate real-world complexity[9]. By benchmarking algorithms on these test functions, researchers gain insights into their strengths, weaknesses, and convergence characteristics. Over the past twenty years, an array of benchmark problems has been introduced, each designed to test different algorithmic aspects under controlled conditions. The guiding assumption is that algorithms performing well on benchmark tests will exhibit similar reliability in real-world tasks. Thus, benchmarking serves not only as a means of comparing existing metaheuristic optimization algorithms but also as a foundation for developing, validating, and improving novel or hybrid approaches. By systematically running algorithms across diverse benchmark functions, researchers can identify the most effective tools for specific problem types and guide future optimization research and development[10]. As a result, benchmark test functions are indispensable for algorithm development, validation, and selection, helping optimize and enhance algorithmic performance across a range of practical applications.

2. Literature Review

In the current literature environment, various test function sets are used both to introduce new metaheuristic optimization algorithms and to propose modified variants of existing methods. Additionally, diverse dimensional settings are commonly employed within these studies. Nonetheless, most benchmark test functions tend to focus on similar types of problems, primarily numerical optimization or, more specifically, unconstrained numerical problems. Certain test functions are frequently chosen and repeated across multiple studies. This literature review will show us past attempts at using benchmark test functions to evaluate metaheuristic optimization algorithm performance, providing insights into the number of test functions and configurations used, which helps establish an overview of existing approaches.

2.1 Previous Benchmark Studies

Early benchmark testbeds tended to be small. Malini et al. [11] examined only three multimodal functions to obtain preliminary evidence for Differential Evolution (DE), Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO). Tithli [12] expanded to seven non-linear programming (NLP/MINLP) problems, allowing a broader appraisal of DE, Krill Herd (KH), Genetic Algorithm (GA), Simulated Annealing (SA) and PSO over 200 runs. Recent works have adopted medium-sized test suites. Jun Wang [13] validated the Black Kite Algorithm on 18 uni- and multimodal functions with 30 search agents over 500 iterations. Saleh [14] introduced the Carpet Weaving Algorithm and benchmarked it on 23 functions (unimodal, fixed- and high-dimensional multimodal) in ten runs of 500 iterations. Pavel's Walrus Optimization Algorithm (WaOA) [15] used the same 23-function set but repeated each run 20 times, highlighting robustness across 10 competing metaheuristic optimization algorithms.Large-scale studies now dominate. Benyamin's Mountain Gazelle Optimizer (MGO) [16] employed 52 benchmark functions, while Siamak [17] assessed Chaos Game Optimization (CGO) on 239 diverse mathematical functions grouped by modality, separability and dimensionality. These mega-testbeds illustrate today's emphasis on breadth and statistical reliability.

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 69-85

2.2 Algorithmic Comparisons

Benchmark studies also differ in the depth of head-to-head comparisons. Malini et al. contrasted three classical algorithms (DE, ABC, PSO) on a tiny testbed, averaging best, worst and mean values over 1 000 iterations. Tithli [12] compared five algorithms using standard-deviation and runtime statistics after 200 runs, demonstrating how function complexity amplifies algorithmic differences. Saleh [14] ranked 12 algorithms with six statistical indicators (best, mean, median, worst, σ, rank), while WaOA's study [15] added 20 runs times by 1000 iterations to strengthen significance testing. Benyamin [16] and Siamak [17] extended the comparison set to eight and six rival algorithms, respectively, emphasizing scalability and reliability under very large test suites. Collectively, these works show a progression from small, proof-of-concept testbeds to extensive, statistically rigorous evaluations that compare both newly proposed and well-established metaheuristic optimization algorithms across increasingly diverse benchmark landscapes. Table 1 summarizes previous studies that evaluated metaheuristic optimization algorithms using benchmark functions, highlighting the number of functions used, their types, and the algorithms compared in each study.

Table 1 *Summary of benchmark studies for metaheuristic optimization algorithms*

Author	Algorithm(s) Evaluated	No. of Benchmark Functions	Function Types	Compared With
Malini et al. [11]	Comparative Analysis	3	Multimodal	DE, ABC, PSO
Tithli [12]	Comparative Analysis	7	NLP, MINLP	DE, KH, GA, SA, PSO
Wang [13]	Black Kite Algorithm	18	Unimodal, Multimodal	BKA, MVO, SCA, GWO, MPA, RIME, ALO, WOA, STOA, DO
Saleh [14]	Carpet Weaving Algorithm	23	Unimodal, Fixed-Dimensional Multimodal, High-Dimensional Multimodal	PSO, GA, GSA, BA, FFA, AOA, WOA, SMA, MPA, COA, TSA, RSA
Pavel [15]	Walrus Optimization Algorithm	23	Unimodal, Fixed-Dimensional Multimodal, High-Dimensional Multimodal	GA, PSO, GSA, GWO, TLBO, MVO, WSO, MPA, TSA, RSA
Benyamin [16]	Mountain Gazelle Optimizer	52	Unimodal, Multimodal	MVO, TSA, PSO, SCA, FFA, WOA, GSA, MFO
Siamak [17]	Chaos Game Optimization (CGO)	239	Continuous, Non-Continuous, Differentiable, Non-Differentiable, Separable, Non-Separable, Scalable, Non-Scalable, Unimodal, Multimodal,	ICA, FA, GWO, SOS, TLBO, WOA

3. Materials and Methods

This section outlines the experimental framework used to evaluate the performance of the proposed optimization algorithm. It includes a detailed description of the benchmark functions selected for testing, as well as the methodological approach taken to implement, execute, and assess each algorithm. The benchmark suite provides a diverse set of optimization challenges, enabling a comprehensive comparison of search efficiency, convergence behavior, and robustness. The goal is to ensure that the evaluation environment is rigorous and replicable, thereby providing meaningful insights into the strengths and limitations of each method under study.

3.1 Benchmark Test Functions

Benchmark test functions represent optimization problems as mathematical numerical functions, optimized using ideal parameter values to achieve the best solution across a defined problem dimension. The optimal solution is often hidden among numerous sub-optimal solutions dispersed across a complex landscape with various peaks and valleys. Metaheuristic optimization algorithms aim to locate the best solution as quickly as possible, although this is not always guaranteed. The effectiveness of these algorithms is measured by their global search capability and local convergence. Algorithms with robust global search abilities will have the ability to avoid getting stuck in local optima, while those with strong convergence are adept at finding the best solutions in nearby regions. Popular and used extensively in research and thesis, there even exist online resources, including MATLAB and R codes that provide additional information on these functions which make it easily accessible to other researchers.

Test function characteristics can be categorized into modality, basins, valleys, separability, and dimensionality group. Modality refers to the number of peaks in a problem landscape, which includes local and global minima which can be split into two types. First are unimodal functions which have a single global minimum, making them easier to solve, though transformations can increase complexity. These are useful for testing the local search capabilities of algorithms. The second type of modality is multimodal functions, which contain multiple local minima but only a single true global minimum [18], which are more challenging and ideal for assessing an algorithm's global search effectiveness. The second characteristic is basins, which represent a region of steep descent enclosed by high peaks. Multimodal functions typically possess a greater number of basins corresponding to local minima than to global minima. Inefficient algorithms often prone to becoming trapped in these basins of local minima, hindering their ability to locate the basin of the global ones. The third characteristic is the valley, which is a region enclosed by peaks, characterized by its narrow width and extended length, representing areas with limited variation. Metaheuristic optimization algorithms often require more time to explore these regions thoroughly, as local search processes tend to examine them in detail. Valleys are shaped differently across unimodal and multimodal functions, with varying frequencies depending on the landscape's complexity. The fourth characteristic is separability which describes how variables within a function can be optimized. Both unimodal and multimodal functions may exhibit separable or non-separable characteristics. In separable functions, each variable can be optimized independently, making these functions simpler to solve.

Conversely, in non-separable functions, variables are interdependent and cannot be optimized in isolation, which increases the complexity of solving these functions. The fifth and last characteristic is dimensionality. This property defines the scope of the search space. Greater dimensionality expands the landscape, introducing a higher number of sub-optimal locations. Functions with lower dimensions are generally easier to solve, with most metaheuristic optimization algorithms performing effectively on them. However, to thoroughly evaluate algorithmic performance, high-dimensional functions are almost mandatory in any optimization testing.

The test set includes both unimodal and multimodal functions to thoroughly assess the performance of the four algorithms. The unimodal functions consisting of F1 and F2 feature a single global optimum and serve to validate the optimization algorithm's effectiveness. In contrast, the multimodal functions of F3–F5 contain numerous local extrema, allowing assessment of the algorithm's exploratory capabilities. The unimodal functions are Sphere and Rosenbrock functions while the multimodal functions consist of Ackley, Rastrigin, and Griewank functions. All the functions are minimization functions with a global minimum value of zero.

3.1.1 Sphere Function (F1)

The first function, F1, is a sphere function which is a basic and unimodal optimization test function. It calculates the sum of the squares of the components of the input vector $x=[x_1,x_2,\ldots,x_n]$ with the global minimum located at $\mathbf{x} = \mathbf{0}$, where $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. This function is convex and smooth, making it relatively easy to optimize. The search boundary for this function is $x_i \in [-5.12,5.12]$, providing a compact domain for testing optimization algorithms. It is defined by the equation:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{1}$$

3.1.2 Rosenbrock Function (F2)

Marked as a F2 function for this paper is the Rosenbrock function, which is often referred to as the "banana function," It is a widely used test problem in optimization. It features a narrow, curved valley where the global minimum is located at x = [1,1,…,1]. The function is non-convex and has a single global minimum, but it is difficult for optimization algorithms to navigate due to the valley's narrowness and curvature. The search boundary is $x_i \in [-5,10]$, and the function is commonly tested in higher dimensions to evaluate optimization performance in

challenging landscapes. It is defined by the equation:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (2)$$

3.1.3 Rastrigin Function (F3)

The Rastrigin function will become the third function**,** the F3 function, that is tested in this paper. It is a multimodal function that is characterized by many local minima, caused by the cosine term. Its global minimum is located at x = 0, where the function value is f(x) = 0. The Rastrigin function has a well-known wavy shape, making it difficult for optimization algorithms to converge. The typical search boundary is $x_i \in [-5.12, 5.12]$, and it is commonly used to test optimization methods in problems with many local optima, as it can be challenging to escape from them. It is defined by the equation:

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \quad (3)$$

3.1.4 Ackley Function (F4)

The F4 function, which is the Ackley function**,** is highly multimodal, meaning it has many local minima. However, it has a single global minimum at x = 0, where f(x) = 0. This function features a complex landscape with oscillations caused by exponential and trigonometric terms, making it difficult for optimization algorithms to avoid local minima and reach the global minimum. The search boundary extends from $x_i \in [-32.768, 32.768]$, allowing a broader search space to test algorithms in a highly challenging environment. It is defined by the equation:

$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e \quad (4)$$

3.1.5 Griewank Function (F5)

F5 function is the Griewank function which combines quadratic and cosine terms, producing a multimodal landscape. It has a global minimum at x = 0, where f(x) = 0, but its oscillatory nature introduces multiple local minima. Though the function is smooth and continuous, its landscape remains challenging due to the oscillations. The typical search boundary is $x_i \in [-600, 600]$, which provides a wider search space, making it suitable for testing optimization algorithms in high-dimensional, complex problems. It is defined by the equation:

$$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (5)$$

The five benchmark functions mentioned above of Sphere, Rosenbrock, Rastrigin, Ackley, and Griewank will be used in this study and were selected to represent a diverse range of optimization challenges commonly found in numerical optimization. These functions collectively test different algorithmic capabilities, including handling unimodal versus multimodal landscapes, separable versus non-separable problems, and varying degrees of complexity. For example, the Sphere function is a simple unimodal and separable problem that evaluates convergence speed, while the Rosenbrock function is unimodal but non-separable and tests an algorithm's ability to navigate narrow valleys. The Rastrigin and Ackley functions are multimodal, with the former being separable and the latter non-separable, both designed to challenge exploration and the ability to escape local optima. The Griewank function, also multimodal and non-separable, further assesses convergence accuracy amidst multiple local minima. This selection offers a balanced and representative test set

3.2 Particle Swarm Optimization (PSO)

Inspired by the "Boid" model, created by Reynolds in 1987, this model was initially designed to simulate bird behavior and became a foundational idea behind PSO. In the simplest version, each bird is represented as a point in a Cartesian coordinate system with a random starting position and velocity. The model follows a rule called the

"nearest proximity velocity match," where each bird matches its speed to its nearest neighbor. Over time, this rule causes all birds to move at the same speed. However, because this model is overly simplified and doesn't closely reflect real-life behaviors, a random component was added to make it more realistic. With each iteration, a small random change is added to each bird's speed, allowing the simulation to behave more like a real flock of birds. Later, Heppner developed a model called the "cornfield model" to further explore how a flock of birds might search for food, inspired by a real-world scenario. In this model, a "cornfield" represents the location of food on a plane, and birds are initially scattered randomly across the plane. To locate the food, they move according to specific rules that guide their foraging behavior, gradually approaching the target location as they move across the plane. This more complex and realistic approach helped pave the way for the creation of the PSO algorithm, which uses similar principles to tackle optimization problems.

Particle Swarm Optimization (PSO) algorithm was inspired by the observed behavior of animals like bird flocking and fish schooling and has become a prominent metaheuristic optimization method in solving complex problems across various domains. PSO, developed by Eberhart and Kennedy [19], is a decentralized algorithm in which particles in a "swarm" share information to guide each particle toward optimal solutions in multidimensional spaces. Each particle updates its position and velocity based on its personal best and the global best, influenced by parameters such as inertia weight, cognitive coefficient, and social coefficient [20]. These parameters are key in balancing exploration and exploitation, which in turn impacts the algorithm's convergence and solution quality. Initially, particles are randomly positioned in the search space, each representing a potential solution. Each particle has a position and velocity, allowing it to "move" through space. As they explore, particles track their best-found position, or "personal best," and share information with others, learning from the best position found in the swarm, known as the "global best" [21]. In each iteration, particles update their velocity and position based on three factors which are current velocity, direction toward their personal best, and direction toward the global best. This approach enables the particles to balance exploration and exploitation. Over time, the particles converge to the optimal solution, guided by both individual experiences and collective knowledge of the swarm. Through these iterative updates, the particles gradually approach the best solution in the search space. For PSO, the following parameter settings were adopted. The swarm size was set to 25, the maximum velocity was defined as 15% of the longest axis-parallel interval in the search space, and the acceleration coefficients were set as $\varphi_1 = 1.8$ and $\varphi_2 = 1.8$, with an inertia weight $\omega = 0.6$. These values were chosen based on the configuration used by Vesterstrøm and Thomsen , who demonstrated their effectiveness across a suite of benchmark functions [22].

For each particle *i*
  For each dimension *j*
    Initialize position x_{ij} randomly with permissible rang
    Initialize velocity v_{ij} randomly with permissible rang
  End for
End for
Iteration k=1
Do
  For each particle *i*
    Calculate fitness value
    If the fitness value is better than $Pbest_{ij}$ in history
      Set current fitness value as the $Pbest_{ij}$
    End if
  End For
    Chose the particle having best fitness value as the $Gbest_j$
  For each particle *i*
   For each dimension *j*
     Calculate velocity according to the equation
     $v_{i,j}^{t+1} = \omega^t v_{i,j}^t + \phi_{1,j}^t \left(Pbest_{i,j}^t - x_{i,j}^t \right) + \phi_{2,j}^t \left(Gbest_i^t - x_{i,j}^t \right)$
     Update particle position according to the equation
     $x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}$
   End for
   End for
   K=k+1
  While maximum iteration or minimum error criteria are not attained.

Fig. 1 *Pseudocode for Particle Swarm Optimizer (PSO) Algorithm*

3.3 Grey Wolf Optimizer (GWO)

Canis lupus or Grey Wolf, is recognized as an apex predator by the nature where it lives, meaning it occupies the top position in the food chain. These wolves typically live in packs, with an average size of 5 to 12 members, and follow a strict social structure. Leading the pack are the alpha wolves, a male and a female, who make crucial

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 69-85

decisions, such as determining when and where to hunt or rest. Although the alphas are the leaders, there are instances of democratic behavior, where the alphas may follow the rest of the pack. The other wolves show respect for the alpha by lowering their tails in the alpha's presence. Next in the hierarchy are the beta wolves, who support the alphas in making decisions and maintaining order in the pack. The betas, who could be male or female, are typically seen as the most likely successors to the alphas if the latter becomes too old or die. The lowest-ranking member of the group is the omega wolf, often seen as the scapegoat and always required to submit to all the higher-ranked wolves. Omegas are the last to eat and are frequently picked on, but they play a crucial role in reducing tension within the pack. Wolves who do not occupy the alpha, beta, or omega positions are known as subordinates. This group includes wolves with specific roles, such as scouts, sentinels, elders, hunters, and caretakers. Scouts monitor the territory and warn the pack of any potential threats. Sentinels focus on the group's security, while elders are seasoned wolves, often former alphas or betas [23].

The Grey Wolf Optimizer (GWO) is a nature-inspired optimization algorithm modeled on the social hierarchy and hunting strategies of grey wolves. In GWO, wolves are categorized into four roles of alpha, beta, delta, and omega, representing the spectrum from the best to the least effective candidate solutions. The hunting process is directed by the top three wolves, designated as alpha, beta, and delta, with the remaining wolves updating their positions based on the leaders [24]. The wolves encircle the prey which is the optimal solution through a dynamic updating process, balancing exploration and exploitation with adaptive coefficients. Exploration occurs when wolves spread out, while exploitation happens as they converge around promising solutions. This iterative process continues until a stopping condition, such as reaching a maximum number of iterations, is met. The GWO algorithm has proven effective in solving complex optimization problems by simulating the social dynamics and cooperative behavior of grey wolves. For parameters, GWO does not rely on fixed control settings and instead it dynamically adjusts its behavior through a leadership hierarchy and adaptive encircling strategies, enabling a natural balance between exploration and exploitation throughout the optimization process.

Initialize the grey wolf population X_i *(i = 1, 2, ..., n)*
Initialize a, A, and C
Calculate the fitness of each search agent
X_α*=the best search agent*
X_β*=the second best search agent*
X_δ*=the third best search agent*
while *(t < Max number of iterations)*
    for *each search agent*
        Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, *and* X_δ
    t=t+1
end while
return X_α

Fig. 2 *Pseudocode for Grey Wolf Optimizer (GWO) algorithm*

3.4 Differential Evolution (DE)

Differential Evolution (DE) is a population-based optimization algorithm inspired by natural evolutionary processes. Initially developed by Kenneth Price in 1994, DE draws inspiration from biological evolution, particularly mutation and selection mechanisms that drive adaptation to environmental challenges. The algorithm mimics the process of survival of the fittest, where solutions evolve over multiple generations, guided by mutation and natural selection to improve progressively [25]. DE is designed to adapt and optimize solutions in continuous search spaces, allowing it to explore and exploit the search space efficiently, making it highly effective for solving complex optimization problems. In practice, DE works by generating an initial population of candidate solutions, then applying a mutation process where differences between randomly selected candidates are scaled and used to produce new solutions. These new solutions are compared to the existing candidates, and if they represent a better solution, they replace the older ones. The performance of DE is controlled by key parameters like the population size, mutation factor, and crossover rate, which balance exploration that involves finding new solutions and exploitation that involves refining existing solutions. This algorithm's effectiveness and flexibility have made it successful across diverse fields, ranging from engineering optimization to machine learning, owing

to its simplicity and robustness in avoiding local optima[26]. For the parameters used, the parameter settings used were a crossover probability (CR) of 0.8 and a differential weight (F) of 0.3. These values were selected based on the study by Seng Poh Lim and Habibollah Haron where this configuration showed reliable performance in various optimization problems, balancing convergence stability with adequate exploration. [27].

```
1: begin
2: Set iteration t = 1.
3: Define problem dimension.
4: Generate initial population and evaluate the fitness.
5:   while (termination condition not reached).
6:     for each population in current generation do
7:         Generate three random integers.
8:         Create offspring using DE operators (mutation, crossover).
9:         Select the best offspring for the next generation.
10:    end for
11:  Set iteration t = t + 1.
12:  end while
13: end
```

Fig. 3 *Pseudocode for differential evolution algorithm*

3.5 Salp Swarm Algorithm (SSA)

The Salp Swarm Algorithm (SSA) is a nature-inspired optimization method that mimics the swarming behavior of marine organisms known as salps [28]. Salps are gelatinous, jellyfish-like creatures that move in chains through the ocean. In these chains, the first salp acts as a leader, guiding the movement of the rest. This organized movement enables them to efficiently navigate their environment. SSA leverages the organized behavior of salp chains to address optimization problems by steering a group of solutions toward the optimal solution within a search space. In SSA, each "salp" symbolizes a potential solution to the optimization problem, with its position determined by coordinates in a multi-dimensional space. This space represents the variables of the problem being solved. The algorithm divides the swarm into one leader and several followers. The responsible salp that is appointed as the leader is responsible for exploring the search space and moving towards the target solution based on an updating rule. The followers, meanwhile, adjust their positions based on the salp directly ahead of them, creating a chain-like movement that eventually leads the entire swarm closer to the optimal solution.

The leader's movement is governed by a control factor c, which balances exploration and exploitation. Initially, c is large, promoting wide exploration to avoid getting trapped in local minima, but it gradually decreases over time to encourage a more focused search around promising solutions. The leader's position is updated by moving toward the optimal solution using a probability-based rule that enables the swarm to explore various regions of the search space [29]. Followers adjust their positions by averaging with the salp in front of them, which naturally pulls the swarm toward convergence as iterations progress. Boundary checks are applied to ensure each salp remains within the designated search limits. If any salp moves beyond the boundaries of the search space, its position is reset to the closest limit, either the upper or lower boundary. This constraint keeps the swarm within the feasible solution area. The algorithm continues this movement of the leader and followers until a stopping criterion is reached, such as a maximum number of iterations or an acceptable solution quality.

SSA's design makes it a flexible and effective tool for various optimization tasks, including engineering design, machine learning feature selection, and neural network training. Its simplicity, ease of implementation, and ability to balance exploration and exploitation make it effective for finding near-global solutions to complex, high-dimensional problems [30]. Because of these qualities, SSA continues to be widely used and studied for its ability to solve real-world optimization problems efficiently. For parameters, SSA operates without predefined control values, as its search dynamics emerge from the natural leader–follower behavior of salps, allowing the algorithm to self-regulate its exploration and convergence during the optimization process.

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 69-85

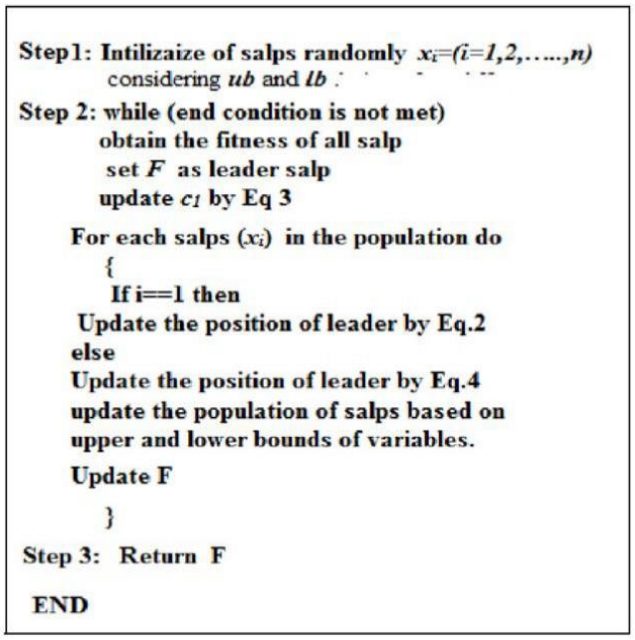

Fig. 4 *Pseudocode for Salp Swarm Algorithm*

4. Results and Discussion

The performance comparison of four metaheuristic optimization algorithms of PSO, DE, SSA, and GWO was conducted on a 3.30 GHz Ryzen 5 5600H CPU and 8 GB of RAM, using Python (v3.10) within the PyCharm 2023.3 environment. Due to the stochastic nature of these algorithms, each benchmark problem was executed 1000 times to ensure accuracy, and the population solutions were evaluated across 10 runs while the dimension of all the functions was taken by 10. For all algorithms evaluated in this study, parameter values were selected based on established settings from prior literature, which are commonly used in benchmark function evaluations. No additional parameter tuning was conducted. This decision was made to maintain consistency with standard practices and to ensure comparability across methods under typical conditions. While this approach may not yield optimal performance for each specific function, it reflects a fair and reproducible setup for benchmarking purposes. The comparative analysis considers parameters including the mean values and standard deviation, as summarized in Table 2, convergence behavior toward the global optimum depicted in Figures 6 to 10, and computational time detailed in Figure 5. A comparison of the optimal values obtained by the four selected metaheuristic optimization algorithms confirms that no single method dominates every benchmark.

Table 2 *Comparison of mean values and standard deviation of the objective functions*

Algorithms/ Function		PSO	DE	SSA	GWO
F1	Mean	0	0	0	0
	Std	8.03251E-09	2.07768E-13	1.57747E-09	1.79252E-65
F2	Mean	36.14	26.06	50.01	26.48
	Std	23.66094919	0.823856138	40.83873982	0.787944144
F3	Mean	`83.36	165.09	46.27	5.41
	Std	30.73018807	8.190202129	14.87080405	5.696890027
F4	Mean	0	0	1.56	0
	Std	0.002020198	1.13254E-07	0.700507743	1.23635E-14
F5	Mean	0.01	0	0.01	0
	Std	0.006647162	3.44623E-12	0.007383227	0.003350576

Table 2 shows that the Grey Wolf Optimizer (GWO) delivers the best solutions on three functions, whereas Differential Evolution (DE) leads on two. These outcomes can be better understood by examining how each algorithm's design and search dynamics interact with the specific landscape features of the benchmark functions. For unimodal functions such as F1 (Sphere) and F2 (Rosenbrock), the performance differences highlight the

balance between exploration and exploitation. F1 is smooth and convex, allowing all algorithms to eventually converge. However, GWO achieves the best result due to its encircling mechanism and the gradual reduction of its control parameter, which promotes intensification once a promising region is found. On the other hand, F2 includes a narrow, curved valley with a flat slope, making it challenging for algorithms that lack strong exploration. DE outperforms others here, as its mutation operator that is based on scaled vector differences, facilitates larger and more diverse jumps that can traverse the flat valley more effectively. The high crossover rate in DE also helps preserves these successful variations. In contrast, PSO, SSA, and even GWO struggle to maintain enough exploration pressure in such regions, resulting in slower or suboptimal convergence.

Multimodal functions such as F3 (Ackley), F4 (Rastrigin), and F5 (Griewank) present a different challenge due to the presence of numerous local optima generated by periodic components. In F3, GWO stands out by effectively maintaining a balance between exploration and exploitation. Its use of three leader wolves (α, β, δ) encourages diverse search directions early on, while the gradual decrease in the parameter A allows the swarm to shift toward exploitation at a controlled rate. This mechanism helps GWO avoid premature convergence in the many local minima. In contrast, DE and SSA perform less consistently. While DE occasionally escapes local traps through large vector-based mutations, it lacks the adaptive pressure seen in GWO. SSA, in particular, suffers in F4 due to its rapidly declining control factor, which prematurely limits exploration and causes the algorithm to settle in broad, flat regions near, but not at the global optimum. PSO shows a similar pattern of stagnation once the global-best position becomes fixed in a local basin. Function F5, which is high-dimensional and involves subtle variable interactions, presents further difficulty. As dimensionality increases, the search space grows exponentially, and the number of local minima rises accordingly. Both PSO and SSA are seen to struggle in this setting, as they lose swarm diversity and become stuck in suboptimal regions. DE and GWO, however, are more robust: DE benefits from constant recombination, which enables it to sample the search space more thoroughly, while GWO retains directional diversity through its leader-based mechanism. These traits explain why GWO and DE produce the most competitive results in F5.

In summary, GWO's adaptive exploration-exploitation strategy, enabled by its social hierarchy and parameter tuning, makes it especially effective in complex, multimodal landscapes. DE performs strongly in scenarios requiring robust exploration, such as flat or narrow-valley functions. PSO and SSA, while efficient under certain conditions, are more sensitive to premature convergence in high-dimensional or rugged problem spaces. These characteristics collectively explain the observed performance patterns across the benchmark suite and underscore the importance of aligning algorithm design with problem landscape features.

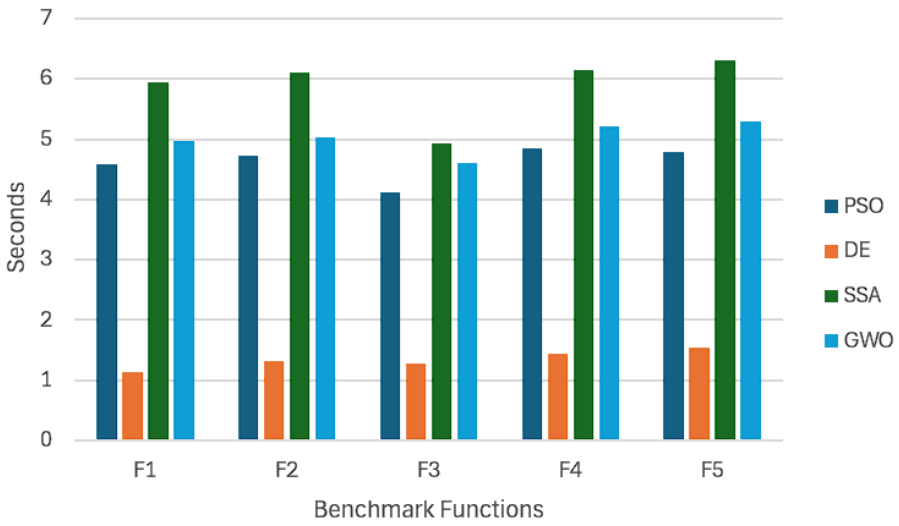

Fig. 5 *Group bar chart for optimization time*

The computation time for each optimization algorithm was recorded and is presented in Figure 5, based on 1000 iterations across four benchmark functions with 10 independent runs per algorithm. DE consistently exhibited the shortest computation time, which can be attributed to its relatively simple algorithmic structure involving straightforward mutation and crossover operations that incur low computational overhead. This efficiency aligns with findings from prior studies [27], where DE's balance of exploration and exploitation allowed it to converge quickly with minimal complexity.

Conversely, the Salp Swarm Algorithm (SSA) demonstrated the longest computation times in all tested scenarios. This higher computational cost primarily stems from SSA's large population size and the position update mechanisms based on salp chain dynamics, which require more function evaluations and complex position

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 69-85

calculations, particularly in higher-dimensional search spaces. This result is consistent with previous research highlighting SSA's increased runtime due to its swarm behavior and iterative updating rules. PSO and GWO fall between DE and SSA in terms of computation time, with PSO generally faster than GWO. Both PSO and GWO rely on population-based search strategies with position and velocity updates that are computationally more involved than DE but less so than SSA. The differences in their update equations and social hierarchy mechanisms impact their relative computational demands.

It is important to note that these timing results depend not only on algorithmic complexity but also on implementation details, including programming language, optimization of code, and hardware specifications. In this study, all algorithms were implemented in Python and run on the same hardware platform to ensure fairness. While DE showed the fastest runtimes, computation time alone does not fully capture algorithm effectiveness.

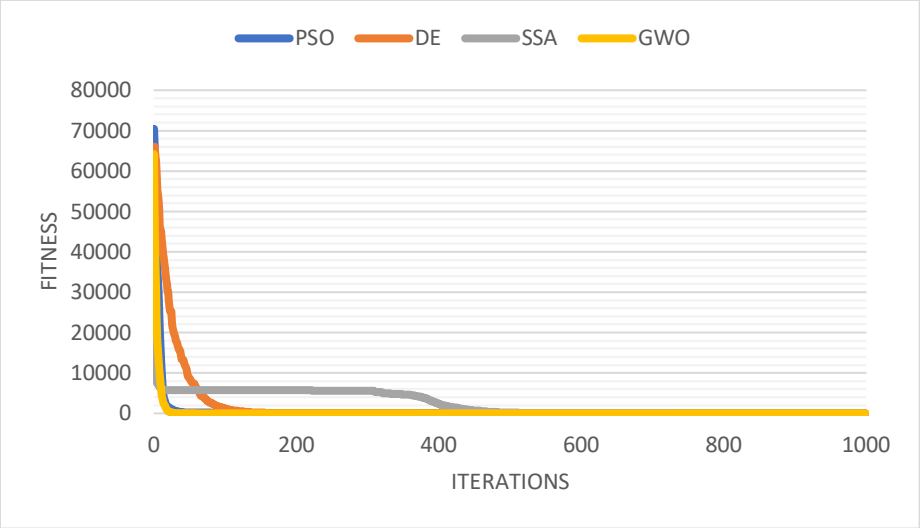

Fig. 6 *Convergence rate of F1*

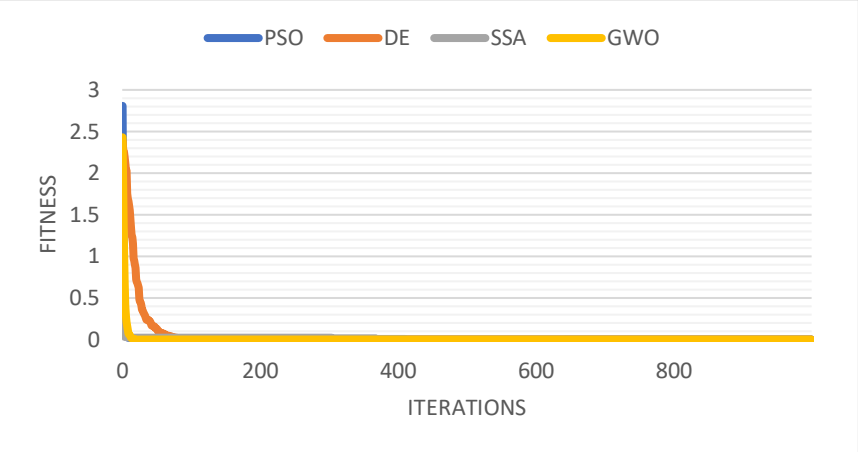

Fig. 7 *Convergence rate of F2*

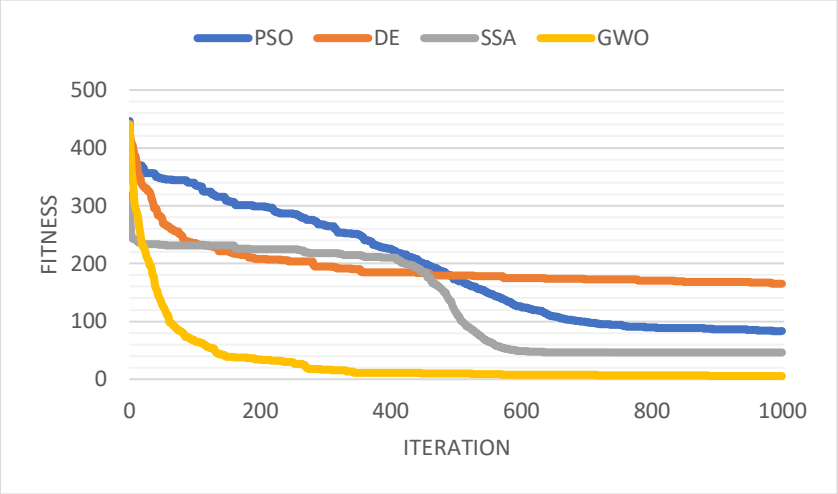

Fig. 8 *Convergence rate of F3*

Fig. 9 *Convergence rate of F4*

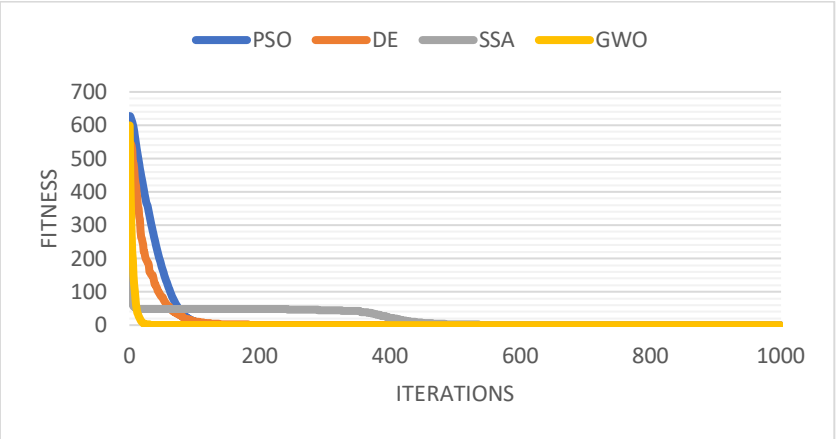

Fig. 10 *Convergence rate of F5*

Figures 6 through 10 illustrate the average convergence rates of the benchmark functions F1 through F5 when optimized by PSO, DE, SSA, and GWO. Across all test functions, the Grey Wolf Optimizer (GWO) consistently demonstrates the fastest convergence rate. For instance, GWO achieves convergence as early as iteration 66 for

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 69-85

F1, 407 for F2, 355 for F3, 74 for F4, and 143 for F5. This performance reflects GWO's efficient exploration-exploitation balance and adaptive leadership hierarchy. Differential Evolution (DE) ranks second in convergence speed, completing convergence by iteration 410 (F1), 580 (F2), 645 (F3), 503 (F4), and 439 (F5), benefiting from its aggressive mutation and crossover strategies. PSO typically converges in third place, requiring more iterations than DE and GWO, specifically 660 (F1), 765 (F2), 680 (F3), 805 (F4), and 567 (F5). Its slower convergence is partly due to susceptibility to local optima, especially in multimodal landscapes. The Salp Swarm Algorithm (SSA) consistently shows the slowest convergence across all five functions, with iterations reaching 764 (F1), 775 (F2), 720 (F3), 710 (F4), and 778 (F5). SSA's weak exploitation capability in the latter phases and rapid decay in exploration pressure appear to hinder its convergence efficiency.

In summary, GWO outperforms the other algorithms in terms of convergence rate across all benchmark functions (Figures 6–10), followed by DE, PSO, and finally SSA. These trends underline GWO's superior adaptability and ability to strike an effective balance between exploration and exploitation during the optimization process.

Table 3 *Wilcoxon Signed-Rank test results for algorithm performance comparison on benchmark functions*

Function	Comparison	Asymp. Sig. (2-tailed)	α Holm	Conclusion
Sphere	GWO vs. DE	0.00000191	0.0167	Reject the null hypothesis
	GWO vs. PSO	0.00000191	0.0250	Reject the null hypothesis
	GWO vs. SSA	0.00000191	0.0500	Reject the null hypothesis
Rosenbrock	GWO vs. DE	0.00000191	0.0167	Reject the null hypothesis
	GWO vs. PSO	0.00000191	0.0250	Reject the null hypothesis
	GWO vs. SSA	0.00000191	0.0500	Reject the null hypothesis
Rastrigin	GWO vs. DE	0.00000191	0.0167	Reject the null hypothesis
	GWO vs. PSO	0.00000191	0.0250	Reject the null hypothesis
	GWO vs. SSA	0.00000191	0.0500	Reject the null hypothesis
Ackley	GWO vs. DE	0.00000191	0.0167	Reject the null hypothesis
	GWO vs. PSO	0.00000191	0.0250	Reject the null hypothesis
	GWO vs. SSA	0.00000191	0.0500	Reject the null hypothesis
Griewank	GWO vs. DE	0.00000191	0.0167	Reject the null hypothesis
	GWO vs. SSA	0.00000191	0.0250	Reject the null hypothesis
	GWO vs. PSO	0.00000381	0.0500	Reject the null hypothesis

To further evaluate the performance of the proposed Grey Wolf Optimizer (GWO) and determine whether the differences in optimization results between GWO and other metaheuristic algorithms were statistically meaningful, a rigorous statistical analysis was conducted using the Wilcoxon signed-rank test. This non-parametric statistical test is specifically designed to compare two related samples and was conducted at a 95% confidence level ($\alpha = 0.05$), making it appropriate for determining whether the observed performance differences are likely to be due to chance. The Wilcoxon signed-rank test was selected as the primary statistical method for this comparative analysis because of its non-parametric nature makes it robust for optimization algorithm comparison studies where normality cannot be guaranteed, as metaheuristic optimization results often exhibit non-normal distributions. The paired sample design of the experimental setup, where each algorithm was run on identical benchmark functions with the same initial conditions across multiple independent runs of 20 times, creates naturally paired data suitable for this test.

Because several pairwise comparisons were performed across multiple benchmark functions, the Holm-Bonferroni correction was applied to control the family-wise error rate and correct for the increased risk of Type I errors (false positives) when conducting multiple pairwise comparisons, ensuring the reliability of statistical conclusions across all benchmark functions. The Holm-Bonferroni procedure works by ranking the p-values from smallest to largest and adjusting the significance threshold for each test accordingly, helping ensure that conclusions remain statistically valid even when multiple comparisons are made. From Table 3, the Wilcoxon signed-rank test results reveal remarkably consistent and statistically significant performance differences across all benchmark functions, with most comparisons producing p-values of 0.00000191 which is minimum value after the calculation, indicating GWO consistently outperformed other algorithms in all 20 independent runs. For the Sphere, Rosenbrock, Rastrigin, and Ackley functions, all pairwise comparisons yielded this minimum p-value, while the Griewank function showed one comparison with $p = 0.00000381$ where PSO outperformed GWO in only one of 20 runs. These extremely low p-values, maintained after Holm-Bonferroni correction, demonstrate both statistical and practical significance with substantial effect sizes. The consistency of results across five distinct benchmark functions with different characteristics of unimodal vs. multimodal suggests that GWO's performance advantages are not limited to specific problem types but represent a general algorithmic superiority, enhancing the generalizability of the findings to other optimization problems and establishing a strong empirical foundation for recommending GWO as a preferred metaheuristic approach for continuous optimization problems.

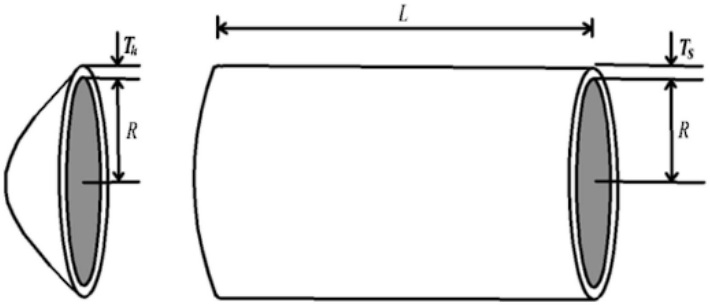

Fig. 11 *Pressure vessel design problem*

To complement the benchmarking experiments conducted on standard test functions, this paper will also incorporate a real-world engineering case, the Pressure Vessel Design Optimization Problem. This problem has been widely used in engineering optimization research due to its practical relevance, nonlinear formulation, and mix of discrete and continuous variables. It involves minimizing the total cost of constructing a cylindrical pressure vessel with hemispherical heads. It is known to be challenging due to the discrete-continuous mix in variable types, the nonlinear and non-convex search space, and the presence of multiple constraints. Moreover, its widespread use in the optimization literature makes it an excellent candidate for comparative analysis of optimization algorithms. Figure 11 illustrates that the goal of the pressure vessel design optimization problem is to minimize the total cost, which includes material, forming, and welding expenses, while adhering to the design variables of shell thickness $Ts(x_1)$, head thickness $Th(x_2)$, inner radius $R(x_3)$, and cylindrical length. The mathematical formulation is provided in Equation (6).

$$\text{Min } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1x_3 \quad (6)$$

Subjected to:

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 69-85

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0$$
$$g_2(x) = -x_2 + 0.0954x_3 \leq 0$$
$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}x_3^2 + 1{,}296{,}000 \leq 0$$
$$g_4(x) = x_4 - 240 \leq 0$$
$$x_1, x_2 \in [0.0625, 99 * 0.0625]$$
$$x_3, x_4 \in [10, 200]$$

Table 4 *Optimization results for the pressure vessel design problem*

Algorithms	Ts	Th	R	L	Optimum Cost
PSO	0.8125	0.4375	42.0843	176.8123	6062.1152
DE	0.8125	0.4375	42.1029	176.5904	6060.2045
GWO	0.78125	0.390625	40.6458	195.7071	5915.3672
SSA	0.890625	0.4375	47.4982	119.3826	6049.2386

The best solutions found by each algorithm are presented in Table 4. All four algorithms produced feasible solutions to the Pressure Vessel Design Problem, but the GWO delivered the lowest cost at 5915.37, clearly outperforming PSO (6062.12), DE (6060.20), and SSA (6049.24). GWO's solution featured a smaller radius and extended length, indicating a more efficient material distribution that satisfied all constraints. In contrast, PSO and DE converged to nearly identical configurations with higher costs, reflecting limited exploration around known optima. SSA found a distinct solution but still fell short of GWO's performance. GWO's ability to maintain diversity while converging efficiently allowed it to discover a more cost-effective design. These results underscore GWO's effectiveness in handling nonlinear, constrained optimization with mixed variable types. Its adaptive leadership and encircling mechanisms guided the search toward better trade-offs, making it the most robust and efficient algorithm in this comparative study.

5. Conclusion and Future Works

This study benchmarked four widely used metaheuristic optimization algorithms, PSO, DE, GWO, and SSA on five standard benchmark functions of Sphere, Rastrigin, Griewank, Ackley, and Rosenbrock, representing a spectrum of optimization challenges. The experimental results show that GWO consistently delivered the best overall performance, achieving optimal or near-optimal solutions on all the benchmark functions and exhibiting the fastest average convergence across all tests as it reach convergence as few as 66 iterations on F1. GWO's leader-based, adaptive search mechanism enabled strong resilience against local optima in complex multimodal landscapes. DE, while slightly behind in solution accuracy, achieved the lowest computation times across all test functions, indicating high efficiency in runtime performance. PSO and SSA generally trailed behind in both convergence speed and solution quality, with SSA in particular demonstrating the slowest convergence and highest computational cost.

The computation time for each algorithm, averaged over 10 independent runs and 1000 iterations across four benchmark functions, showed that DE recorded the shortest runtimes, reflecting its low computational overhead and algorithmic simplicity. In contrast, SSA exhibited the longest times due to its large population size and complex salp chain dynamics. PSO and GWO fell between, with PSO being slightly faster than GWO. To assess the statistical significance of GWO's superior performance, Wilcoxon signed-rank tests were performed at a 95% confidence level ($\alpha = 0.05$), with Holm-Bonferroni correction applied to control for multiple comparisons. The analysis revealed p-values as low as 0.00000191 for most functions, confirming GWO's consistent outperformance. Only one instance on the Griewank function had $p = 0.00000381$, where PSO outperformed GWO in a single run, highlighting GWO's overall reliability and effectiveness across diverse optimization landscapes.

Nonetheless, each algorithm displayed certain limitations. GWO, despite its strong performance, showed signs of premature convergence in high-conditioning problems like Rosenbrock, suggesting that its exploitation phase may benefit from enhancement. DE tended to stagnate in multimodal landscapes due to limited diversity in later search stages, and PSO was prone to early convergence, especially in rugged fitness landscapes. SSA's performance suffered due to its random-walk nature, which hindered both convergence speed and accuracy.

To address these limitations, future work will explore hybrid approaches that combine the strengths of different algorithms such as combining GWO's leadership hierarchy with DE's differential mutation for better balance between exploration and exploitation. Additionally, adaptive parameter control and problem-specific tuning may help tailor each algorithm's performance to the characteristics of the optimization problem. Finally, the study will extend GWO to the domain of combinatorial testing, a key phase in software validation, to evaluate its applicability in discrete search spaces and enhance its utility in real-world optimization scenarios.

Acknowledgement

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2024/ICT03/UTHM/01/1).

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Muhamad Asyraf Anuar, Rosziati Ibrahim;* ***analysis and interpretation of results:*** *Muhamad Asyraf Anuar, Rosziati Ibrahim, Nurezayana Zainal, Mazidah Mat Rejab;* ***draft manuscript preparation:*** *Muhamad Asyraf Anuar, Rosziati Ibrahim, Hana Hachimi. All authors reviewed the results and approved the final version of the manuscript.*

Reference

[1] Diab, N., & Smaili, A. (2008). Optimum exact/approximate point synthesis of planar mechanisms. Mechanism and Machine Theory, 43(12), 1610–1624. https://doi.org/10.1016/j.mechmachtheory.2007.12.006

[2] Ghosh, S., Singh, A., & Kumar, S. (2024). HPB3C-3PG algorithm: A new hybrid global optimization algorithm and its application to plant classification. Ecological Informatics, 81, 102581. https://doi.org/10.1016/j.ecoinf.2024.102581

[3] Gopi, S., & Mohapatra, P. (2024). Fast random opposition-based learning Aquila optimization algorithm. Heliyon, 10(4), e26187. https://doi.org/10.1016/j.heliyon.2024.e26187

[4] Petrowski, A., Ben Hamida, S. (2016). Evolutionary Algorithms. In: Siarry, P. (eds) Metaheuristics. Springer, Cham. https://doi-org.ezproxy.uthm.edu.my/10.1007/978-3-319-45403-0_6

[5] Raji, S., Dehnamaki, A., Somee, B., & Mahdiani, M. R. (2022). A new approach in well placement optimization using metaheuristic algorithms. Journal of Petroleum Science and Engineering, 215, 110640. https://doi.org/10.1016/j.petrol.2022.110640

[6] Asyraf , M. A., Sahid, M. Z. ., & Zainal, N. . (2023). Comparative Analysis of Test Case Prioritization Using Ant Colony Optimization Algorithm and Genetic Algorithm. *Journal of Soft Computing and Data Mining*, *4*(2), 52-58. https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/13585

[7] Yue, L. Z. ., & Ibrahim, R. . (2023). Comparative Analysis for Test Case Prioritization Using Particle Swarm Optimization and Firefly Algorithm. *Journal of Soft Computing and Data Mining*, *4*(2), 67-77. https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/13579

[8] Tomar, V., Bansal, M., & Singh, P. (2024). Metaheuristic Algorithms for Optimization: A Brief Review. Engineering Proceedings, 59(1), 238. https://doi.org/10.3390/engproc2023059238

[9] Talpur, K., Salleh, M., Cheng, S., & Naseem, R. (2017). Common Benchmark Functions for Metaheuristic Evaluation: A Review. International Journal of Information Visualization, 1(4), 218–223. https://doi.org/10.30630/joiv.1.4-2.65

[10] Van Stein, N., Vermetten, D., Kononova, A. V., & Bäck, T. (2024). Explainable Benchmarking for Iterative Optimization Heuristics. arXiv preprint arXiv:2401.17842. https://doi.org/10.48550/arXiv.2401.17842

[11] Mohan, M., & Joseph, M. V. (2019, August 8). A comparative study of different metaheuristic optimization algorithms using standard test functions [Conference paper]. AIP Conference Proceedings. AIP Publishing. https://doi.org/10.1063/1.5120226.

[12] Sadhu, T., Chowdhury, S., Mondal, S., Roy, J., Chakrabarty, J., & Lahiri, S. K. (2022). A comparative study of metaheuristics algorithms based on their performance of complex benchmark problems. Decision Making: Applications in Management and Engineering, 5(2), 109–128. https://doi.org/10.31181/dmame0306102022r

[13] Wang, J., Wang, W., Hu, X., Qiu, L., & Zang, H. (2024). Black-winged kite algorithm: a nature-inspired meta-heuristic for solving benchmark functions and engineering problems. Artificial Intelligence Review, 57(4), 98. https://doi.org/10.1007/s10462-024-10723-4

[14] Alomari, S., Kaabneh, K., AbuFalahah, I., Gochhait, S., Leonova, I., Montazeri, Z., Dehghani, M., & Eguchi, K. (2024). Carpet Weaver Optimization: A novel simple and effective human-inspired metaheuristic algorithm. International Journal of Intelligent Engineering and Systems, 17(4), 230–242. https://doi.org/10.22266/ijies2024.0831.18

[15] Trojovský, P., Dehghani, M. A new bio-inspired metaheuristic algorithm for solving optimization problems based on walruses behavior. Sci Rep 13, 8775 (2023). https://doi.org/10.1038/s41598-023-35863-5

[16] Abdollahzadeh, B., Gharehchopogh, F. S., Khodadadi, N., & Mirjalili, S. (2022). Mountain Gazelle Optimizer: A new Nature-inspired Metaheuristic Algorithm for Global Optimization Problems. Advances in Engineering Software, 174, 103282. https://doi.org/10.1016/j.advengsoft.2022.103282

[17] Talatahari, S., & Azizi, M. (2021). Chaos Game Optimization: A novel metaheuristic algorithm. Artificial Intelligence Review, 54(2), 917–1004. https://doi.org/10.1007/s10462-020-09867-w

[18] Kusuma, P., & Hasibuan, F. (2024). Swarm flip-crossover algorithm: a new swarm-based metaheuristic enriched with a crossover strategy. International Journal of Electrical and Computer Engineering (IJECE), 14(2), 2145-2155. doi: https://doi.org/10.11591/ijece.v14i2.pp2145-2155

[19] Mohanty, C.P., Singh, M.R., Mahapatra, S.S., Chatterjee, S. (2015). A Particle Swarm Approach Embedded with Numerical Analysis for Multi-response Optimization in Electrical Discharge Machining. In: Panigrahi, B., Suganthan, P., Das, S. (eds) Swarm, Evolutionary, and Memetic Computing. SEMCCO 2014. Lecture Notes in Computer Science (), vol 8947. Springer, Cham. https://doi-org.ezproxy.uthm.edu.my/10.1007/978-3-319-20294-5_7.

[20] Panigrahi, P. N. Suganthan, & S. Das (Eds.), Swarm, Evolutionary, and Memetic Computing (pp. 74–87). Springer International Publishing. https://doi.org/10.1007/978-3-319-20294-5_7

[21] De Santana Junior, J. C. (2023). A network science approach to analysing, comparing, and evaluating population-based metaheuristics (Doctoral thesis). University of Exeter. http://hdl.handle.net/10871/134480

[22] Mulumba, D. M., Liu, J., Hao, J., Zheng, Y., & Liu, H. (2023). Application of an Optimized PSO-BP Neural Network to the Assessment and Prediction of Underground Coal Mine Safety Risk Factors. Applied Sciences, 13(9), 5317. https://doi.org/10.3390/app13095317

[23] Vesterstrom, J., & Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Proceedings of the 2004 Congress on Evolutionary Computation (pp. 1980-1987). IEEE. https://doi.org/10.1109/CEC.2004.1331139

[24] Guha, D., Roy, P. K., & Banerjee, S. (2016). Load frequency control of interconnected power system using grey wolf optimization. Swarm and Evolutionary Computation, 27, 97–115. https://doi.org/10.1016/j.swevo.2015.10.004

[25] Mustaffa, Z., Sulaiman, M. H., & Yusof, Y. (2015). An Application of Grey Wolf Optimizer for Commodity Price Forecasting. Applied Mechanics and Materials, 785, 473–478. https://doi.org/10.4028/www.scientific.net/AMM.785.473

[26] Eltaeib, T., & Mahmood, A. (2018). Differential Evolution: A Survey and Analysis. Applied Sciences, 8(10), 1945. https://doi.org/10.3390/app8101945

[27] Ahmad, M. F., Isa, N. A. M., Lim, W. H., & Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works. Alexandria Engineering Journal, 61(5), 3831–3872. https://doi.org/10.1016/j.aej.2021.09.013

[28] Lim, S. P., & Haron, H. (2013). Performance comparison of Genetic Algorithm, Differential Evolution and Particle Swarm Optimization towards benchmark functions. In 2013 IEEE Conference on Open Systems (pp. 41-46). IEEE. https://doi.org/10.1109/ICOS.2013.6735045

[29] Yaseen, Z. M., Faris, H., & Al-Ansari, N. (2020). Hybridized Extreme Learning Machine Model with Salp Swarm Algorithm: A Novel Predictive Model for Hydrological Application. Complexity, 2020, 8206245. https://doi.org/10.1155/2020/8206245

[30] Faris, H., Mafarja, M. M., Heidari, A. A., Aljarah, I., Ala'M, A. Z., Mirjalili, S., & Fujita, H. (2018). An efficient binary Salp Swarm Algorithm with crossover scheme for feature selection problems. Knowledge-Based Systems, 154, 43–67. https://doi.org/10.1016/j.knosys.2018.05.009

[31] Musa, Z., Ibrahim, Z., & Shapiai, M. I. (2024). Multi-Agent cubature Kalman optimizer: A novel metaheuristic algorithm for solving numerical optimization problems. International Journal of Cognitive Computing in Engineering, 5, 140–152. https://doi.org/10.1016/j.ijcce.2024.03.003