Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Novel Hybrid Nature-Inspired Optimizer Based on Harris Hawks and Artificial Lemming Algorithm for Solving Engineering Problems

Krishnapada Mondal

krishnamondal151129@gmail.com

University of Information Technology and Sciences https://orcid.org/0009-0005-2512-1604

Md. Abu Salman

Northern University of Business and Technology Khulna

Md. Shariful Islam

University of Information Technology and Sciences

Md. Rakibul Hossain Mozumder

University of Information Technology and Sciences

Md. Khabirul Islam

University of Information Technology and Sciences

Shahed Ahmed

University of Information Technology and Sciences

Research Article

Keywords: Harris Hawks Optimizer, Artificial Lemming Algorithm, Metaheuristic Algorithms, Global Optimization, Manufacturing costs, Standard Test Functions

Posted Date: June 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9131166/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

A Novel Hybrid Nature-Inspired Optimizer Based on Harris Hawks and Artificial Lemming Algorithm for Solving Engineering Problems

Krishnapada Mondal[1*][0009-0005-2512-1604], Md. Abu Salman[2][0000-0003-4439-1317], Md. Shariful Islam[1] [0009-0004-0686-5942], Md. Rakibul Hossain Mozumder[1] [0009-0006-7328-1001], Md. Khabirul Islam[1] [0009-0004-1131-0861] , Shahed Ahmed[1] [0009-0003-8334-4203]

[1] Department of Electrical and Electronic Engineering, University of Information Technology & Sciences (UITS), Dhaka, Bangladesh
[2] Department of Electrical and Electronic Engineering, Northern University of Business and Technology Khulna (NUBTK), Khulna, Bangladesh
*Corresponding author's email: krishnamondal151129@gmail.com

Abstract. This study proposes a novel hybrid nature-inspired metaheuristic technique, the Hybrid Harris Hawks Optimizer with Artificial Lemming Algorithm (HHHOALA), for addressing challenging global optimization and constrained engineering design problems. HHHOALA integrates the tactical hunting strategies of the Harris Hawks Optimizer (HHO) with the versatile mobility patterns of the Artificial Lemming Algorithm (ALA); this makes the search process balanced and adaptable. The proposed algorithm seeks to make exploration stronger in the initial search phase and exploitation stronger in the later phase by using a coordinated transitional mechanism. This synergy lets the algorithm keep the population diverse while reliably moving toward high-quality solutions. This procedure is applied to solve on real-world constrained engineering problems, such as pressure vessel design and welded beam design, with the objective of minimizing the manufacturing cost. The suggested HHHOALA has been scrutinized and compared to other well-known algorithms, including WOA, ALA, GWO, and HHO. The hybrid and other optimizers optimized 22 standard benchmark functions from IEEE CEC 2020 and IEEE CEC 2022 test suites using MATLAB. Solution quality, convergence speed, and robustness are always better with the hybrid optimizer. Statistical Analysis, encompassing Wilcoxon Signed Rank Test and Friedman Rank Test evaluations, substantiates the relevance and reliability of HHHOALA's performance. In addition, convergence behavior, and boxplot analysis show that the proposed strategy can adapt and stay stable across a wide range of problem areas. The suggested method is a strong and effective way to solve a wide range of difficult globally and constrained engineering optimization challenges.

Keywords: Harris Hawks Optimizer, Artificial Lemming Algorithm, Metaheuristic Algorithms, Global Optimization, Manufacturing costs, Standard Test Functions.

1 Introduction

Engineering and scientific research are fields where optimization is highly valued because it attempts to find optimal solutions to a problem with a number of objectives and constraints. The reason why nature-inspired algorithms (NIAs) have received so much attention over the past few decades is due to their flexibility, robustness , and the ability

to solve optimization problems that are multi-dimensional, nonlinear, which conventional techniques often struggle to handle. These algorithms are guided to the solution space by concepts of biology, physics, swarm based to find and exploit the solution space efficiently.

The metaheuristic algorithms (MAs) have developed their importance in the past years as viable remedies to the intricate issues of optimization problems. There are many MAs, and various features and contributions characterize the algorithms developed in recent years. Even with a good performance, Seagull Optimization Algorithm (SOA) also has some shortcomings, which include premature convergence, loss of population diversity and inability to have an adequate balance between exploration and exploitation in complex optimization problems (Dhiman et al. 2019).The Arithmetic Optimization Algorithm (AOA) is a mathematics based optimization algorithm that is inspired by the distribution behavior of simple arithmetic operators. It has attracted much attention because it has proven effective in addressing the solution of different optimization problems in various fields (Dhal et al. 2023). A well-known swarm intelligence algorithm is the Grey Wolf Optimizer (GWO), which was based on the leadership structure and hunting behavior of the gray wolves. Because of its straightforward design, rapid convergence, and high optimization capacity, GWO has been extensively used in many areas and has given rise to many customized and hybrid versions to solve complex optimization problems (Liu et al. 2024). The Glider Snake Optimization algorithm (GSO) is a metaheuristic based on nature inspired algorithms, which simulates the gliding and serpentine movements of arboreal snakes to enhance both exploration and exploitation. Experiments with classical benchmarks, problems of high-dimensionality, and with constrained engineering problems show GSO is more accurate, converges more quickly, and is more robust than a number of state-of-the-art algorithms (El-Kenawy et al. 2026). However, it consists of various techniques that add computational complexity and underscores both the possibility and difficulty of single-solution optimization methods (Shami et al. 2024; Rodan 2025; Zolghadr et al. 2018).This algorithms might have weaknesses in premature convergence, and also computational complexity as well as lacks scalability (Anka 2025; Mohammadi et al. 2021). Tunicate Swarm Algorithm (TSO) has better performance, scaling and reduced computational time on benchmark functions and real world problems such as Economic Dispatch (Rizk et al. 2021). These algorithms are designed to increase the exploration and exploitation through adaptive parameters and growth functions and is able to perform better than the original CSA on benchmark functions and engineering design problems (Sheta et al. 2023).

A number of new hybrid and improved swarm-based optimization algorithms have been suggested to improve the exploration, convergence and quality of solution to benchmark and real-world engineering problems. It suggests a hybrid PSOGWO algorithm that incorporates the search power of the Particle Swarm Optimization (PSO) and the exploration power of the Grey Wolf Optimizer (GWO). Experimental tests on benchmark functions and real-world problems prove that the hybrid method has a superior convergence behavior and quality of solutions to a number of available optimization methods (Şenel et al. 2019). Binary Dwarf Mongoose Optimization algorithm (BDMO) is a swarm-based optimization method that is a relatively new algorithm. To

solve discrete problems a Binary DMO variant was suggested with varying transfer functions to transform continuous variables to binary values and experimentally it was established that it could perform better than a number of the available binary optimization algorithms (Bas 2024). The Modified Grey Wolf Optimizer (MGWO) is an improvement of the original GWO, which refines the position equations, simulates a four-level hierarchy of leadership, and demonstrates competitive performance on benchmark functions and real-world engineering design problems (Gopi et al. 2025).

The hybridization of optimization algorithms is an important process that makes significant contributions to enhancing their total performance and efficiency in the process of solving complex problems. Single algorithms strive to find a balance between exploration and exploitation and hybrid algorithms strive to find a higher balance between the two and convergence, solutions, and systems are more stable. Experiments conducted by several authors have proved that hybrid algorithms are better than single algorithm in several engineering optimization problems including scheduling jobs, structure design and power system optimization etc.

Among different optimization procedures, the Harris Hawks Optimizer (HHO) is a metaheuristic algorithm used to optimize the problem using a population-based approach whose name stems from the behavior observed by Harris hawks in nature, i.e. cooperative hunting(Alabool et al. 2021). It is efficient because it can balance exploration and exploitation dynamically thus being able to leave local optima and move to the global ones. HHO has gained success in a number of fields, such as machine learning model tuning, energy management, and engineering design etc. HHO however is sometimes too fast in very complex or multimode search spaces and as such, it is not applicable to particular engineering issues. However, Artificial Lemming Algorithm (ALA) is a nature inspired optimizing algorithm that is inspired by the lemming behavior in the wild (Xiao et al.2025). Lemmings are small rodents that are characterized by group activities and extensively searching new areas or food. It is also randomly behaving and hence it is an effective way of searching the search space as it will be able to search the search space in more ways and will not be attracted to local minima. ALA is attributable to the challenge of the exploitation phase that makes it take a longer time to reach the global optimum in the many-dimensional cases.

The research gap that this research paper intends to fill is that a novel type of hybrid optimizer, known as HHHOALA is developed. The total population is divided into two equal organizations in order to have a right balance in exploration and exploitation. To be more specific, half of the population is established to be in the HHO mechanism to be investigated, and the other half 50 percent is to be in the ALA mechanism to be exploited. The suggested approach combines the powerful exploration nature of HHO together with the efficient exploitation nature of the ALA. In this paradigm, HHO examines different parts of the search space to help escape local optima whereas ALA narrows down on good solutions by high degree of exploitation. By exploiting the complementary advantages of the two algorithms, HHHOALA is meant to offer a more viable and powerful remedy in addressing complex optimization problems, which eventually superior in improved performance results in real-world application.

The principal findings of this research are as follows:

1. To ensure an effective search, a hybrid approach combines the benefits of their strengths, balancing between exploration and exploitation.
2. The proposed optimizer evaluates using CEC 2020 and CEC 2022 benchmark suite and compared to various well-known optimization algorithms. The experimental findings prove that the offered method is observed to show high-quality performance of the solution and speed of convergence.
3. Wilcoxon Signed Rank Test and Friedman Rank Test prove the highly superiority of the suggested algorithm to the competing approaches. The convergence curves and boxplot analysis also demonstrate the soundness and stability of the proposed optimizer.
4. The proposed HHHOALA validates based on classical engineering optimization problems and real-world engineering optimization problems.

The rest of this paper is structured into the following way: Section 2 gives a detailed review of nature-inspired optimization methods. Section 3 describes the HHHOALA that is proposed and the conceptual information of algorithm design. Section 4 depicts the qualitative analysis of the proposed optimizer, while Section 5 presents the exploration and exploitation behavior of the algorithm. Section 6 displays experimental verification and performance analysis of the proposed method with the existing algorithms in terms of standard test functions and statistical tests. Section 7 is devoted to the real-life application, and Section 8 summarizes the paper and provides the possible way of future research.

2 Background

Conventional optimization techniques are weak in high dimensional search space. These challenges demand new and creative approaches. The algorithms are optimistic towards solutions by a well-planned search space exploration, and exploitation of areas of interest. The two approaches attempt to enhance convergence behavior and solution accuracy in difficult optimization problems through various approaches.

2.1 Harris Hawks Optimizer (HHO)

HHO is a famous optimization method that is based on the cooperative hunting mechanism of Harris hawks when hunting down their prey who have fled, and it can be predominantly divided into three phases, namely exploration phase, transition phase, and exploitation phase. The workflow of the HHO algorithm is contained in Algorithm 1.

2.1.1 Exploration Phase

The exploration phase is the initial step in mimicking how Harris' hawks hunt for food. In this phase, hawks are trying for seeking the optimal solution all across the globe. At this stage, it land in random places and seek around for their prey, which is the best solution. The approach stores a lot of distinct possible solutions so that it doesn't immediately find a local optimum. Mathematically, exploration happens when the prey's

escaping energy is $|E| \geq 1$. At this point, the algorithm is mostly concerned with its ability to search the entire space for promising locations. The mathematical framework for the initial stage is presented in equation 1.

$$L(t+1) = \begin{cases} L_{rand}(t) - p_1|L_{rand}(t) - 2p_1 L(t)| & m \geq 0.5 \\ L_{rand}(t) - L_m(t) - p_3(L_B + p_4(U_B - L_B) & m < 0.5 \end{cases} \quad (1)$$

In where, $L(t+1)$ is the position of the hawks in the successor, $L_{rand}(t)$ is the position of rabbit, $L(t)$ is the present position of the hawks, p_1, p_2, p_3, p_4, and m are random numbers within (0,1), updated in every iteration, L_B and U_B indicate upper and lower limits of the variables.

2.1.2 Transition phase

The transition phase determines whether the algorithm continues searching new things or begin utilizing the discovered things. This decision involves the escaping energy (E) of the prey. It varies with time to give the impression of the prey getting weary. The hawks keep looking for more when $|E| \geq 1$ and the method switches to exploitation when $|E| < 1$. Here, E is the escaping energy of the rabbit, n is the number of iterations and N is the maximum iterations.

$$E = 2E_i\left(1 - \frac{n}{N}\right) \quad (2)$$

2.1.3 Exploitation Phase

The exploitation phase is all about closely examining the best candidate solution that has been found so far. The hawks now work together to catch their prey by surrounding it. This phases uses different methods, like soft besiege, aggressive besiege, and quick dives, depending on how the prey tries to get away and how likely it is to do so. This stage makes it easier to search locally and narrows down the best response, which increases the chances of finding the global optimum. Adaptive switching strategies in exploitation speed up convergence and make solutions more accurate.

2.1.3.1 Soft besiege

In the case $r \geq 0.5$ and $|E| \geq 0.5$, the rabbit still possesses sufficient energy, and it attempts to escape by making some misleading jumps at random but eventually it cannot. In such attempts, the hawks go round it very lightly in order to tire it out, and then execute the surprise attack. This behavior is shown the following equation.

$$L(t+1) = \Delta L(t) - E|jL_{rabbit}(t) - L(t) \quad (3)$$

Where, $\Delta L(t)$ is the difference between the position vector of the rabbit and its current position in iteration t.

2.1.3.2 Hard besiege

In the case $r \geq 0.5$ and $|E| < 0.5$, the prey is so fatigued and has low escaping energy. Here, the existing positions are updated involving the use of the current positions. Here, $L(t+1)$ is the updated positions of the hawks.

$$L(t+1) = L_{rabbit}(t) - E|\Delta L(t)| \quad (4)$$

2.1.3.3 Soft besiege with progressive rapid dives

When still $|E| \geq 0.5$ but $r < 0.5$, the energy of the rabbit is sufficient that it can escape successfully and yet a gentle siege is erected before the surprise pounce. This process

is smarter than the preceding case. Thus, the ultimate plan of revising the location of hawks during the soft besiege stage can be put in the following way. Where, $L(t+1)$ is the updated positions of the hawks.

$$L(t+1) = \begin{cases} P & if\ T(P) < T(L(t)) \\ Q & if\ T(Q) < T(L(t)) \end{cases} \quad (5)$$

2.1.3.4 Hard besiege with progressive rapid dives

When $|E| < 0.5$ and $r < 0.5$, in this case the rabbit does not have enough energy to escape, thus the hard besiege of the hawks followed by the surprise pounce and capture the prey. Thus, the following rule is carried out in hard besiege condition.

$$L(t+1) = \begin{cases} P & if\ T(P) < T(L(t)) \\ Q & if\ T(Q) < T(L(t)) \end{cases} \quad (6)$$

Algorithm 1 Pseudo-code for HHO optimizer [23]

Inputs: Initialize the P_{total} and i_{max}
Outputs: Determine $Prey_{Position}$ and fitness value
Initialize the random search agents $S_k(K = 1,2, \dots M)$
while (Not meet the stopping criteria) **do**
  Find out fitness value of hawks
  Set rabbit position, R_P
  for (each hawks(S_k)) **do**
    Update the initial energy and jump strength
    Update the equation (2)
    if $|E| \geq 1$ **then**
    Update the equation (1)
    end
    if $|E| < 1$ **then** % Exploitation phase
      if $|E| \geq 0.5$ and $r \geq 0.5$ **then** % Soft besiege
        Update the positioning vector by Eq.(3)
      else if $|E| < 0.5$ and $r \geq 0.5$ **then** % Hard besiege
        Update the positioning vector by Eq.(4)
      else if $|E| \geq 0.5$ and $r < 0.5$ **then** % Soft besiege with progressive rapid dives
        Update the positioning vector by Eq.(5)
      else if $|E| < 0.5$ and $r < 0.5$ **then** % Hard besiege with progressive fast dives
        Update the positioning vector by Eq.(6)
    end
  end
end
Return R_P

2.2 Artificial Lemming Algorithm (ALA)

ALA is a novel optimization technique inspired by the behavior and group migration of lemmings in the wild. The flow of the ALA algorithm is contained in Algorithm 2.This approach operates in the numerous phases. These are:

2.2.1 Long-distance migration (exploration)

One of the most important things lemmings do to stay alive is move large distances. When food is scarce or there are too many lemmings in one place, they relocate to different places to find new locations. This behavior is like global exploration in the ALA, where search agents explore at large areas of the search universe. When lemmings first start moving, they take enormous leaps. This mechanism makes it less likely that early convergence will happen and more likely that the global optimum will be found. Here, $\vec{L}_i(n+1)$ denotes the position of the i^{th} search agent in $(n+1)$ iteration. $\vec{L}_{best}(n)$ shows the optimum solution at that moment.

$$\vec{L}_i(n+1) = \vec{L}_{best}(n) + E \times \overrightarrow{PQ} \times (\vec{S} \times \left(\vec{L}_{best}(n) - \vec{L}_i(n)\right) + \left(1 - \vec{R}\right) \times \left(\vec{L}_i(n) - \vec{L}_a(n)\right) \quad (7)$$

2.2.2 Digging holes (exploration)

To find food and shelter, lemmings dig tunnels. This behavior explores more narrowly than long-distance migration. When search agents explore nearby places, digging holes is local exploration in this ALA. Lemmings do not drift over the search space. They shift based on adjacent search agents. This keeps the system interesting and finds good places. Digging tests neighboring solutions for improvement over time. Finding good exploitation sites before exploitation is key. In which L is a random number, $\vec{L}_b$ is a search individual chosen randomly in the population.

$$\vec{L}_i(n+1) = \vec{L}_i(n) + E \times L \times (\vec{L}_{best}(n) - \vec{L}_b(n)) \quad (8)$$

2.2.3 Searching for food (exploitation)

During this stage, lemmings have a developed sense of hearing and smell as they move around their burrows in a befuddled manner trying to find food. Lemmings normally mark out area of their habitat with food. This behavior is updated in the following equation.

$$\vec{L}_i(n+1) = \vec{L}_{best}(n) + E \times Spiral \times rand \times \vec{L}_i(t) \quad (9)$$

2.2.4 Avoiding predators in nature (exploitation)

Lemmings avoid danger and defend themselves. The burrow is safe for lemmings. It use their extreme speed to attack foes run back to burrow.

$$\vec{L}_i(n+1) = \vec{L}_{best}(n) + E \times H \times P(Dim) \times (\vec{L}_{best}(n) - \vec{L}_i(n)) \quad (10)$$

Where, H equivalence to escape co-efficient of lemmings, P the Levy flight function, used to simulate the misleading response of lemmings in escaping.

2.2.5 Transferring from exploring to exploiting

When lemmings reach locations with enough food in the nature, they slowly cease travelling. The ALA's search behavior grows steadily from exploration to exploitation, much like that. The most crucial aspect of the algorithm at the beginning is exploration,

which makes sure that the search space is extensively explored. As the number of iterations increases, exploitation becomes increasingly crucial for identifying the ideal solutions. There is a strong connection among the above steps in ALA and lemming energy levels. Where, n is the number of iterations and N_{max} is the maximum iterations.

$$E = 4 \times \arctan\left[1 - \frac{n}{N_{max}}\right] \times In(\frac{1}{rand}) \tag{11}$$

Algorithm 2 Pseudo-code for ALA optimizer [18]

Inputs: Initialize the P_{total} and i_{max}
Outputs: Determine $\vec{L}_{best}$ and fitness value f_{best}
Initialize the random search agents $\vec{L}_i(i = 1,2, \ldots M)$
    Record the optimal solution $\vec{L}_{best}$
    $n = 1$
    while $n \leq N_{max}$ **do**
        Calculate the value of E using equation (11)
        for each search individual $\vec{L}_i$ **do**
            if $|E| > 1$ **then**
                Exploration phase
                if $rand$ < 0.3 **then**
                    Update the current location using equation (7)
                else
                    Update the current location using equation (8)
                end
            else
                Exploitation phase
                if $rand$ < 0.5 **then**
                    Update the current location using equation (9)
                else
                    Update the current location using equation (10)
                end
    end
end

3 Proposed HHHOALA Algorithm

Metaheuristic optimizers are frequently employed to address intricate nonlinear optimization challenges, as conventional mathematical techniques frequently struggle with high-dimensional and multimodal search spaces. However, most single optimization algorithms have some problems, such converging too quickly, taking too long to converge, or not having the right balance between exploration and exploitation. The flow of the suggested HHHOALA algorithm is contained in Algorithm 3. Figure 1 shows the working procedure of the method.

Initially, the algorithm randomly starts out with a population of search agents in the defined search limits. The search agents are possible solutions to optimization problem. Once the initialization stage is complete, each agent is assessed based on the objective function on obtaining the best global solution.

Under the proposed framework, the overall population is split into two equal groups to have a right balance between exploration and exploitation. To be more precise, 50 percent of the population is determined to be in the HHO mechanism to be explored, and the rest 50 percent would be in the ALA mechanism to be exploited. This equal dispersion assists the algorithm in doing the global and the local search. Both teams use the global best solution for search. The elementary goal of this hybridization to merge the different search capability of these optimizers within unified frameworks. This hybrid structure boosts optimization algorithm convergence and exploration–exploitation balance. The two search processes work together in a way that makes the algorithm faster at finding solutions, more accurate, and more stable for addressing difficult optimization issues.

At this phase, the agents of the search examine various parts of the solution space by random search capabilities and global search procedures. The escaping energy parameter in the hawks governs the behavior movement of the hawks and assists the algorithm in exploring the areas in the search space that have not been explored. Such exploration capability enhances the chances of finding good prospects capable of holding its best solution. In the second phase of ALA algorithm, that has a strong exploitation capability through few strategies spiral movement, directed movement toward the optimal solution, and random walk based on Lévy flight. These tactics help the optimizer to improve the solutions and make it to reach the global optimum.

Once the positions of the agents are updated with the two mechanisms, there are also some boundary conditions that will ensure that all the solutions stay within the feasible search constraints. Then, the fitness of the new solutions is again calculated. In case a superior solution is received, the global best solution is updated. This is repeated until the maximum number of iterations is attained. The last step is that the algorithm provides the optimal solution and the optimal fitness. The primary benefit of 50 percent of exploration through HHO and 50 percent of exploitation through ALA is that it offers a good balance between the global exploration and the local exploitation. Exploration also avoids the algorithm to get stuck in local optima whereas exploitation enhances the accuracy of the end solution.

The rationale of this hybrid approach is that numerous traditional optimization algorithms for optimization are either highly exploratory yet weakly exploitative or highly

exploitative, but lack sufficient exploration properties. The proposed HHHOALA algorithm will ensure a high convergence, quality of solutions, and robustness to solving complex optimization problems by combining global search capability of HHO with local refinement capability of ALA.

Algorithm 3 Pseudo-code for HHOALA optimizer

Inputs: Initialize the P_{total} and i_{max}
Outputs: Determine Y_{best} and fitness value
Initialize the random search agents $S_k(K = 1,2, \dots M)$
while (Not meet the stopping criteria) **do**
    Divide population into 2 groups: HHO (50% for Exploration and ALA (50% for Exploitation)
        Update HHO group using exploration (random search, global search)
        Update ALA group using exploitation (spiral search , levy flight)
        Apply boundary check
    Evaluate new fitness and update Y_{best}
end
Return Y_{best} and best fitness

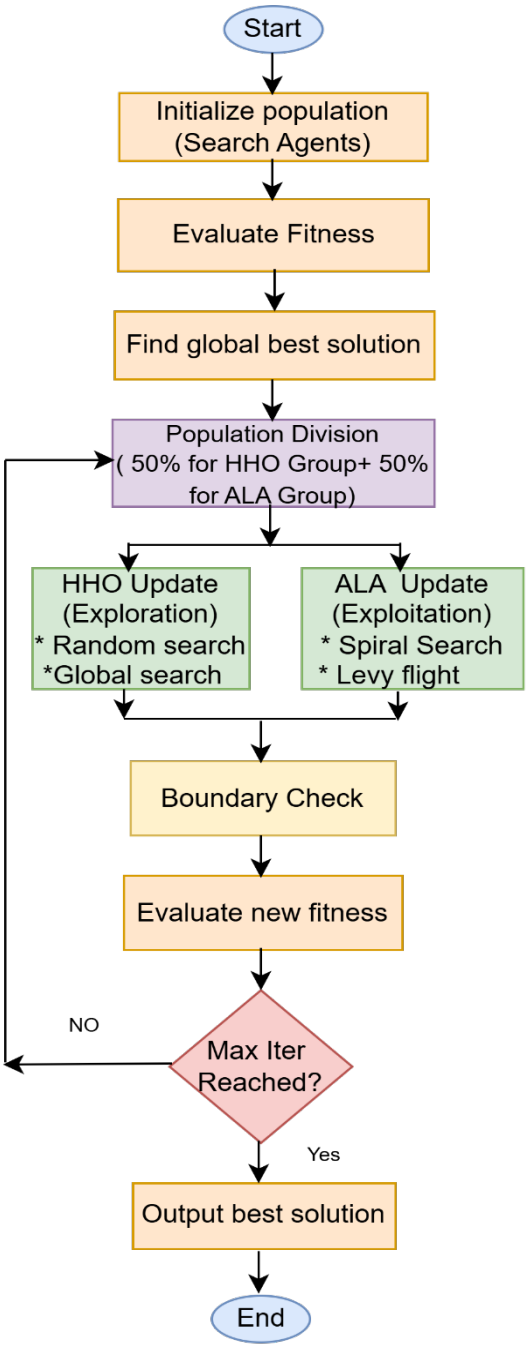
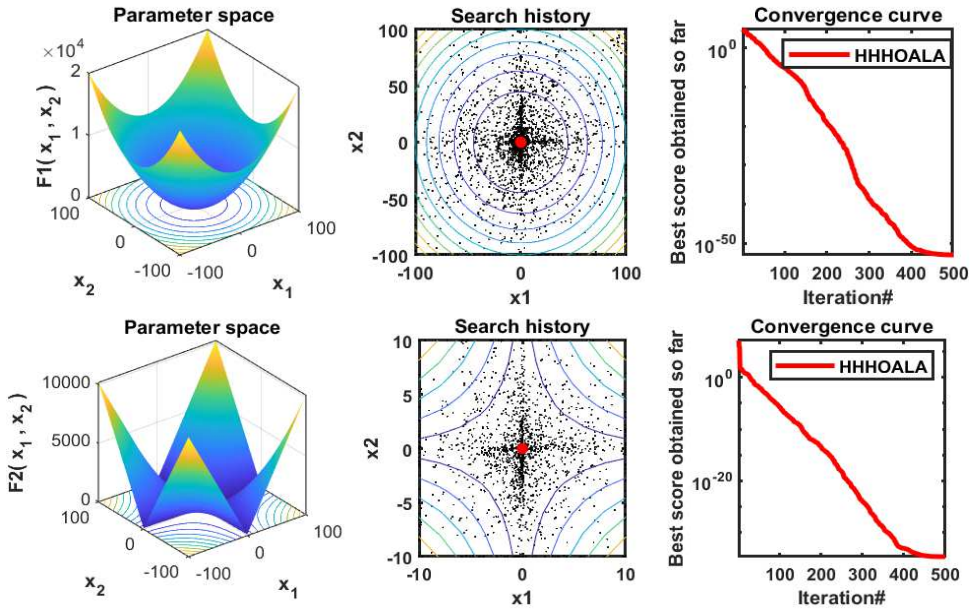


Fig.1 Flowchart of the HHHOALA algorithm

4 Qualitative Analysis of HHHOALA Optimizer

This section examines a set of qualitative metrics occupied to verify the outcome of hybrid optimizer. Figure 2 demonstrates the most important qualitative tests for HHHOALA's single objective optimization. The first column represents the three-dimensional parameter space of the chosen benchmark functions. This visualization shows how the objective function works, including where the global optimum is and how hard it is to search the space. The second column depicts the search history of the HHHOALA method. It is clear that the search agents are spread out across the whole search area at first, which makes sure that there is sufficient exploration. As the iterations go on, the agents get closer to the best area, which indicates that there is a good balance between exploration and exploitation. This behavior reveals that the suggested method keeps the population diverse and avoids converging too soon throughout the search. The third column displays the convergence curve of the proposed optimizer for the selected test function. The convergence curve shows how the best fitness value changes as the number of iterations changes. The fitness value consistently goes down as the number of iterations goes up, which shows that the convergence behavior is stable. The gradual reduce in the fitness value displays that the suggested algorithm quickly finds the global optimum. Overall, the qualitative study presents that the suggested HHHOALA algorithm is good at exploring, exploiting, and converging in a steady way. These results ensure that the suggested hybrid optimization algorithm works well and is strong enough to solve global optimization challenges.



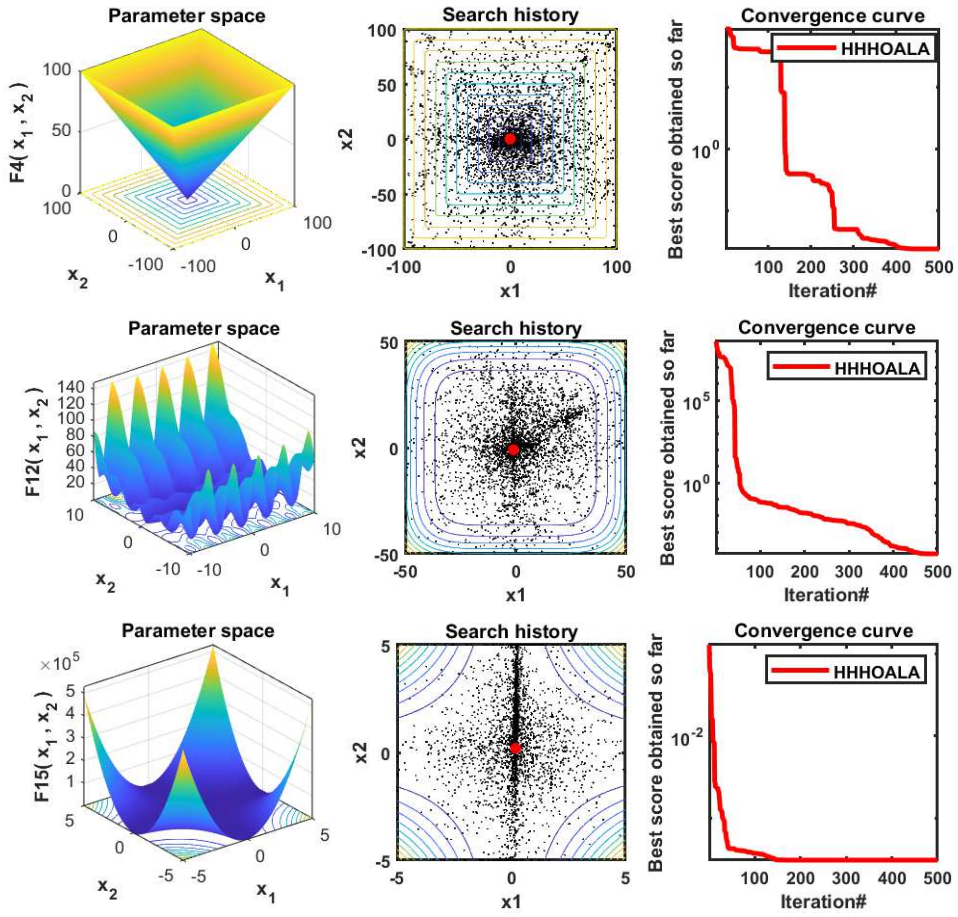


Fig. 2 Qualitative analysis of the HHHOALA algorithm

5 Exploration & Exploitation Analysis of HHHOALA Optimizer

Figure 3 shows how the suggested HHHOALA optimizer explores and exploits for the selected benchmark functions. After visualizing of the figure, it can be seen that the red line presents the percentage of exploration, while the green line shows the percentage of exploitation during the optimization process. At the commencing of the optimization process, the exploration percentage is high. This means that the algorithm searches a wide area of the solution space to find areas that look promising. This vast search ability keeps the algorithm from becoming stuck in local optima and makes it more likely that it will find the global optimum. The exploration percentage slowly goes down as the number of iterations goes up, while the exploitation percentage increases. For functions F1 and F5, the exploitation percentage goes up quickly, which means that the algorithm quickly finds the best areas and focuses on improving the solutions. On the other hand,

the exploration phase lasts longer for functions F3 and F7. This shows that these functions have more complicated search landscapes that need a lot of global search. In general, the findings demonstrate that this optimizer keeps a good balance between exploration and exploitation during the optimization process. This balanced behavior helps the solution converge faster and more accurately.

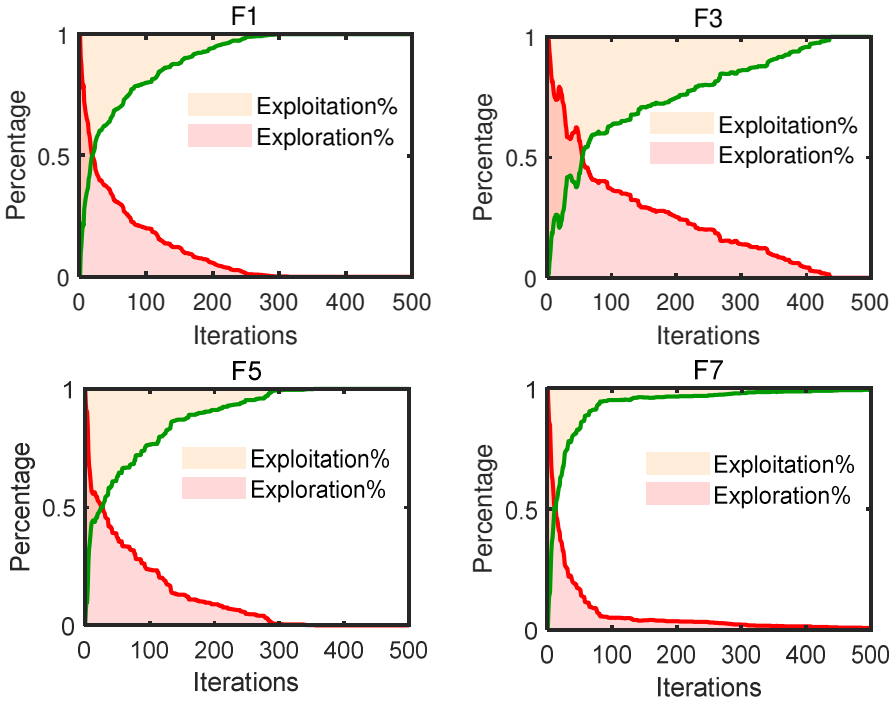


Fig. 3 Exploration-Exploitation analysis of the HHHOALA algorithm

6 Experimental Verification and Analysis

6.1 Standard Test Functions

This subsection gives a general idea of the benchmark functions used in the standard test functions for CEC 2020 and CEC 2022. The main purpose of these functions is to carefully compare the outcomes of our hybrid optimizer against a number of well-known techniques. We used a population size of 30, maximum number of iterations of 500 and 10 Dimensions for the both test groups. The Benchmark functions in Tables 1 and 2 have a wide range of features that let us fully evaluate how well the algorithms can solve hard optimization problems.

Table 1 Overview of the CEC 2020 test suite

Function	Function name	Name	Global
CEC-01	Unimodal	Shifted and rotated bent cigar function	100
CEC-02	Basic	Shifted and rotated Schwefel's function	1100
CEC-03		Shifted and rotated Lunacek bi-Rastrigin function	700
CEC-04		Expanded Rosenbrock's plus Griewank's function	1900
CEC-05	Hybrid	Hybrid function 1 (N = 3)	1700
CEC-06		Hybrid function 2 (N = 4)	1600
CEC-07		Hybrid function 3 (N = 5)	2100
CEC-08	Composition	Composition function 1 (N = 3)	2200
CEC-09		Composition function 2 (N = 4)	2400
CEC-10		Composition function 3 (N = 5)	2500

Table 2 Overview of the CEC 2022 test suite

Function	Function Name	Name	Global
CEC-01	Unimodal	Shifted and full rotated Zakharov	300
CEC-02	Basic	Shifted and full rotated Rosenbrock's	400
CEC-03		Shifted and full rotated expanded Schaffer's f_6	600
CEC-04		Shifted and full rotated Non-Continuous Rastrigin	800
CEC-05		Shifted and full rotated Levy	900
CEC-06	Hybrid	Hybrid 1 (N = 3)	1800
CEC-07		Hybrid function 2 (N = 6)	2000
CEC-08		Hybrid function 3 (N = 5)	2200
CEC-09	Composition	Composition 1 (N = 5)	2300
CEC-10		Composition (N = 4)	2400
CEC-11		Composition (N = 5)	2600
CEC-12		Composition (N = 6)	2700

6.2 Experimental Setup

This subsection talks about the experimental setup that was used to evaluate the suggested hybrid optimizer and see how it compared to other well-known optimization methods. The whole simulations were run using MATLAB R2024b and we compared our proposed technique to other method such as, HHO, ALA, WOA, and GWO here. All algorithms were performed under the identical conditions, with the identical population size and maximum number of iterations, to make sure that the comparisons were fair and consistent. Table 3 presents the parameters for the selected methods.

Table 3 Parameters for the optimizers

Algorithm	Parameters
HHO	$E_0 = 2, J = 0.5$
ALA	$\beta = 1.5, u \,\&\, v = [0,1]$
WOA	$b = 1,\ l = 1.5$
GWO	$a = [0,2]$

6.3 Results

In this portion, our proposed algorithm's performance evaluation across many benchmark test suites, including CEC-2020 and CEC-2022, are shown here. HHHOALA is compared to WOA, ALA, GWO, and HHO in each table. All statistical analysis are utilized to evaluate each function. These measures show how well HHHOALA solves complex optimization issues and compares to other methods.

6.3.1 CEC-2020 Results

Table 4 displays the outcomes of comparing the performance of the HHHOALA method to other optimization algorithms on ten different test functions (CEC-1 to CEC-10) from the CEC-2020 test suite. For each function, it shows five key metrics: best, average, standard deviation, worst, and the rank of the performance of each algorithm. The average value shows the mean fitness score that each method got over 30 runs. Lower mean values usually reflect higher optimization performance. The standard deviation shows how much performance varies; lower values mean that the findings are more consistent across runs. The rank value shows how well each algorithm did compared to the others. A rank of 1 means that the algorithm did the best perform for that function. For example, on CEC-1 function, HHHOALA had an average fitness score of 3.44960e+03, a standard deviation of 3.012279e+03, and a rank of 1, showing that it consistently delivered high-quality solutions. HHHOALA outperformed other algorithms, especially in terms of consistency (mean & standard deviation) compared to the best method for this function. Conversely, the HHO algorithm had a larger average value of 7.69474e+09, indicating difficulty in finding optimal solutions for this function. According to CEC-02, CEC-03, and CEC-04 function, HHHOALA do not secure the first position among all algorithms. In several functions, HHHOALA ranks first, including CEC-05 to CEC-10. It can be seen from these functions that our suggested strategy maintains lower mean and standard deviation values, which means that it consistently produced high-quality solutions across many runs. The suggested optimizer competed well for functions, demonstrating its robustness across problem landscapes. In optimization circumstances where reliability is key, HHHOALA's standard deviations indicate consistent performance.

Table 4 Performance evaluation of HHHOALA and other optimizers on CEC 2020 test suite (Dimension:10)

S.L	Algorithms	Best	Worst	Mean	Std.	Rank
CEC-1	HHHOALA	1.008437e+02	1.152975e+04	3.449601e+03	3.012279e+03	1
	WOA	4.232690e+06	3.599208e+08	7.397255e+07	8.612170e+07	3
	ALA	3.946559e+05	3.463821e+08	2.621939e+07	7.140899e+07	2
	GWO	2.953272e+04	1.559175e+09	1.150273e+08	3.082018e+08	4
	HHO	1.987399e+09	1.648504e+10	7.694747e+09	3.273048e+09	5
CEC-2	HHHOALA	1.456101e+03	2.420658e+03	1.909843e+03	2.464386e+02	3
	WOA	1.741571e+03	3.046879e+03	2.384721e+03	3.573494e+02	4
	ALA	1.243527e+03	2.116890e+03	1.772861e+03	2.000268e+02	2
	GWO	1.130461e+03	2.215236e+03	1.750816e+03	2.743438e+02	1
	HHO	2.010723e+03	2.950878e+03	2.476811e+03	2.513529e+02	5
CEC-3	HHHOALA	7.202665e+02	7.756921e+02	7.445785e+02	1.561745e+01	2

	WOA	7.479146e+02	8.365172e+02	7.918643e+02	2.254615e+01	4
	ALA	7.242196e+02	8.411211e+02	7.742920e+02	2.929695e+01	3
	GWO	7.183250e+02	7.570509e+02	7.357041e+02	1.261668e+01	1
	HHO	7.479652e+02	8.334805e+02	8.044118e+02	2.191011e+01	5
CEC-4	HHHOALA	1.900000e+03	1.900102e+03	1.900008e+03	2.445180e-02	2
	WOA	1.900102e+03	1.902595e+03	1.900101e+03	4.775943e-01	3
	ALA	1.901123e+03	1.906052e+03	1.903101e+03	1.373550e+00	5
	GWO	1.900000e+03	1.902396e+03	1.900915e+03	8.857838e-01	4
	HHO	1.900000e+03	1.900000e+03	1.900000e+03	0.000000e+00	1
CEC-5	HHHOALA	1.869647e+03	2.621823e+03	2.182919e+03	1.855232e+02	1
	WOA	6.442906e+03	9.133364e+05	1.798459e+05	2.228366e+05	4
	ALA	3.115108e+03	8.166836e+05	1.583793e+05	1.747392e+05	3
	GWO	2.706174e+03	5.657381e+05	6.695983e+04	1.630520e+05	2
	HHO	9.284780e+03	7.077949e+05	4.009492e+05	2.553108e+05	5
CEC-6	HHHOALA	1.601300e+03	1.913249e+03	1.691956e+03	8.150373e+01	1
	WOA	1.676523e+03	2.148007e+03	1.843535e+03	1.054320e+02	4
	ALA	1.763281e+03	1.956632e+03	1.830427e+03	4.731048e+01	3
	GWO	1.612257e+03	2.015864e+03	1.735940e+03	1.108786e+02	2
	HHO	1.734794e+03	2.367340e+03	1.996787e+03	1.576665e+02	5
CEC-7	HHHOALA	2.163249e+03	2.504223e+03	2.346765e+03	9.858520e+01	1
	WOA	4.489872e+03	1.138035e+07	9.194635e+05	2.229286e+06	4
	ALA	2.432004e+03	1.485491e+04	5.812688e+03	3.364555e+03	2
	GWO	2.688854e+03	2.899783e+04	1.154512e+04	7.426046e+03	3
	HHO	7.156384e+03	4.215021e+06	1.237217e+06	1.514781e+06	5
CEC-8	HHHOALA	2.219178e+03	2.309440e+03	2.298380e+03	1.939351e+01	1
	WOA	2.239818e+03	3.850571e+03	2.414383e+03	3.683351e+02	4
	ALA	2.238919e+03	2.317739e+03	2.303943e+03	1.805203e+01	2
	GWO	2.216194e+03	2.332753e+03	2.307719e+03	1.954021e+01	3
	HHO	2.279800e+03	3.719039e+03	2.789113e+03	3.806916e+02	5
CEC-9	HHHOALA	2.500195e+03	2.771946e+03	2.737570e+03	6.318145e+01	1
	WOA	2.742176e+03	2.872505e+03	2.795853e+03	2.974435e+01	4
	ALA	2.501351e+03	3.026563e+03	2.742037e+03	1.799818e+02	2
	GWO	2.700323e+03	2.775396e+03	2.747402e+03	1.584006e+01	3
	HHO	2.678495e+03	2.975446e+03	2.840295e+03	6.936011e+01	5
CEC-10	HHHOALA	2.897922e+03	2.959126e+03	2.922863e+03	2.403674e+01	1
	WOA	2.709058e+03	3.036026e+03	2.942419e+03	7.062917e+01	4
	ALA	2.898166e+03	3.038898e+03	2.936888e+03	2.832617e+01	2
	GWO	2.898299e+03	2.951802e+03	2.938034e+03	1.649117e+01	3
	HHO	2.991837e+03	3.867402e+03	3.288289e+03	1.987833e+02	5

6.3.2 CEC-2022 Results

Table 5 is an outcome of an in-depth performance analysis conducted on the HHHOALA compared to other popular optimization methods on twelve of the test functions (CEC-1 through CEC-12) in the CEC-2022 test set. In every function, it indicates the best, the average, the standard deviation, the worst as well as the rank of the performance of each algorithm. The mean value indicates the mean fitness value that each method received after 30 runs. The standard deviation indicates the variation in the performance; the lower the values, the higher the consistency in the performance across the runs. Rank illustrates the effectiveness of each method when compared to the other whereby rank value of 1 implies that it is the most efficient when it comes to that function. For instance, CEC-1 function, HHHOALA acquired an average value of 3.182458e+02, a standard value of 2.211492e+01, and a rank of 1. This analysis shows that our method worked well with other methods when it produced statistical consistency. For Function CEC-2, suggested method generated mean value of 4.142517e+02 and std. value of 2.129677e+01 with a rank 1, this shows that proposed method give the best solution. For function CEC-3, HHHOALA produced mean value 6.012706e+02 and std. value of 1.306728e+00 with a rank 2, this indicates that suggested method unable to find the optimal solutions for these functions. For function, CEC-4, HHHOALA gained rank 3, that means this method struggled to find the better outcomes. HHHOALA gives rank 1 for these test functions, such as CEC-05 to CEC-10. From these functions, we can see that our suggested technique keeps the mean and standard deviation values lower. This suggests that it consistently gave us high-quality solutions throughout many runs.

Table 5 Performance evaluation of HHHOALA and other optimizers on CEC 2022 test suite (Dimension: 10)

S.L	Algorithms	Best	Worst	Mean	Std.	Rank
CEC-1	HHHOALA	3.000306e+02	3.933085e+02	3.182458e+02	2.211492e+01	1
	WOA	5.171416e+03	4.586907e+04	2.566101e+04	1.182823e+04	5
	ALA	7.865209e+02	5.844393e+03	2.752560e+03	1.453847e+03	2
	GWO	4.354983e+02	7.764357e+03	3.713580e+03	2.466621e+03	3
	HHO	5.221160e+03	1.068629e+04	9.762380e+03	1.355516e+03	4
CEC-2	HHHOALA	4.001403e+02	4.759527e+02	4.142517e+02	2.129677e+01	1
	WOA	4.021690e+02	7.227209e+02	4.853589e+02	8.696800e+01	4
	ALA	4.000895e+02	6.458418e+02	4.727027e+02	6.617214e+01	3
	GWO	4.043157e+02	4.734983e+02	4.298941e+02	2.526120e+01	2
	HHO	4.805105e+02	1.997931e+03	9.921747e+02	3.975165e+02	5
CEC-3	HHHOALA	6.000345e+02	6.051342e+02	6.012706e+02	1.306728e+00	2
	WOA	6.138379e+02	6.578057e+02	6.380919e+02	1.159687e+01	4
	ALA	6.132909e+02	6.540327e+02	6.325952e+02	8.429708e+00	3
	GWO	6.000856e+02	6.067110e+02	6.011325e+02	1.497791e+00	1
	HHO	6.198372e+02	6.665864e+02	6.461774e+02	1.200180e+01	5
CEC-4	HHHOALA	8.129345e+02	8.498819e+02	8.270510e+02	8.461578e+00	3
	WOA	8.089898e+02	8.679409e+02	8.359165e+02	1.475587e+01	4
	ALA	8.082782e+02	8.281918e+02	8.190794e+02	5.380374e+00	2

	GWO	8.039857e+02	8.422888e+02	8.162278e+02	8.773407e+00	1
	HHO	8.185252e+02	8.524992e+02	8.377634e+02	8.597633e+00	5
CEC-5	HHHOALA	9.004628e+02	1.225671e+03	9.282777e+02	5.900781e+01	1
	WOA	9.242300e+02	3.411326e+03	1.609073e+03	4.948446e+02	5
	ALA	9.535731e+02	1.446124e+03	1.183584e+03	1.163086e+02	3
	GWO	9.001318e+02	1.181029e+03	9.304345e+02	6.781137e+01	2
	HHO	1.107209e+03	1.914849e+03	1.414035e+03	2.045128e+02	4
CEC-6	HHHOALA	1.832315e+03	8.202014e+03	2.831363e+03	1.375238e+03	1
	WOA	2.118406e+03	1.672518e+04	6.118087e+03	3.781958e+03	3
	ALA	2.201772e+03	1.261025e+04	6.065613e+03	2.729498e+03	2
	GWO	2.317055e+03	1.219048e+04	6.907807e+03	2.703574e+03	4
	HHO	2.226919e+03	7.883130e+06	5.461799e+05	1.572360e+06	5
CEC-7	HHHOALA	2.016296e+03	2.041282e+03	2.025550e+03	6.184427e+00	1
	WOA	2.020623e+03	2.126146e+03	2.078492e+03	2.689858e+01	4
	ALA	2.027666e+03	2.108151e+03	2.065001e+03	2.199061e+01	3
	GWO	2.002319e+03	2.097694e+03	2.033860e+03	1.637557e+01	2
	HHO	2.048434e+03	2.168248e+03	2.110511e+03	3.272727e+01	5
CEC-8	HHHOALA	2.204792e+03	2.230789e+03	2.222358e+03	5.547795e+00	1
	WOA	2.215085e+03	2.287303e+03	2.236103e+03	1.253232e+01	4
	ALA	2.208803e+03	2.233592e+03	2.224904e+03	5.428039e+00	2
	GWO	2.212502e+03	2.231133e+03	2.225814e+03	3.903324e+00	3
	HHO	2.225721e+03	2.289249e+03	2.245998e+03	1.931348e+01	5
CEC-9	HHHOALA	2.529284e+03	2.529840e+03	2.529340e+03	1.696527e-01	1
	WOA	2.530935e+03	2.707596e+03	2.607692e+03	5.393258e+01	3
	ALA	2.596157e+03	2.703633e+03	2.650002e+03	3.182268e+01	4
	GWO	2.529323e+03	2.665489e+03	2.569924e+03	3.810927e+01	2
	HHO	2.588711e+03	2.793753e+03	2.709444e+03	5.634892e+01	5
CEC-10	HHHOALA	2.500358e+03	2.627152e+03	2.524531e+03	4.892057e+01	1
	WOA	2.500418e+03	3.033720e+03	2.583077e+03	1.131115e+02	3
	ALA	2.500862e+03	2.714202e+03	2.608328e+03	7.410995e+01	4
	GWO	2.500298e+03	2.617945e+03	2.567883e+03	5.571949e+01	2
	HHO	2.508253e+03	4.020361e+03	2.745101e+03	3.392397e+02	5
CEC-11	HHHOALA	2.600011e+03	3.000006e+03	2.844731e+03	1.193948e+02	1
	WOA	2.711199e+03	5.747364e+03	3.098138e+03	5.148648e+02	4
	ALA	2.648354e+03	3.188374e+03	2.935826e+03	1.399008e+02	2
	GWO	2.775358e+03	3.375267e+03	3.032810e+03	1.477897e+02	3
	HHO	2.873129e+03	4.663710e+03	3.651298e+03	5.347221e+02	5
CEC-12	HHHOALA	2.858619e+03	2.866174e+03	2.862982e+03	1.916664e+00	1
	WOA	2.865035e+03	3.183143e+03	2.910254e+03	6.160764e+01	3
	ALA	2.898667e+03	3.154385e+03	3.024581e+03	7.811600e+01	5
	GWO	2.862605e+03	2.906044e+03	2.868497e+03	9.694910e+00	2
	HHO	2.881314e+03	3.325014e+03	3.011867e+03	1.013088e+02	4

6.3.3 Wilcoxon Signed-Rank Test

The findings of the Wilcoxon signed-rank test rely on statistical evidence on the superiority of the proposed HHHOALA algorithm over all of the contrasted algorithms on both CEC 2020 and CEC 2022 test suites. HHHOALA has a win/draw/loss record of 9/1/0 against WOA, 7/3/0 against ALA, 7/2/1 against GWO and 9/1/0 against HHO in CEC 2020 which shows a consistent dominance. The outcomes were even stronger

in CEC 2022, where HHHOALA score was 10/0/0 in opposition to WOA, 9/0/1 in opposition to ALA, 8/3/1 to GWO and 10/0/0 to HHO. The p-values on most of the benchmark functions are extremely tiny with a majority of the values in the range of 1.73e-06 that is significantly lower than the standard significance level of 0.05 and hence the performance differences are statistically significant. Against WOA and HHH, proposed optimizer gained a zero loss percentage in CEC 2022, which demonstrates its strong optimization capability. In both test suites, HHHOALA was statistically significant in comparison in the overwhelming majority of cases, which supports its validity and competence as a metaheuristic optimizer. The combination of these Wilcoxon test results with the Friedman rank test results give a good statistical evidence that HHHOALA has the best results when compared to all the benchmark algorithms in solving complex optimization problems. Tables 6 & 7 represent the statistical analysis of the optimizers.

Table 6 Wilcoxon Signed-Rank Test for CEC 2020 test suite

HHHOALA vs	WOA		ALA		GWO		HHO	
Function	p-value	winner	p-value	winner	p-value	winner	p-value	winner
CEC-1	1.73e-06	+	1.73e-06	+	1.73e-06	+	1.73e-06	+
CEC-2	6.34e-06	+	5.19e-02	=	7.52e-02	=	1.92e-06	+
CEC-3	1.92e-06	+	1.36e-04	+	4.95e-02	-	1.92e-06	+
CEC-4	1.00e+00	=	1.73e-06	+	8.30e-06	+	1.25e-01	=
CEC-5	1.73e-06	+	1.73e-06	+	1.73e-06	+	1.73e-06	+
CEC-6	6.34e-06	+	8.47e-06	+	3.00e-02	+	2.60e-06	+
CEC-7	1.73e-06	+	1.73e-06	+	1.73e-06	+	1.73e-06	+
CEC-8	1.89e-04	+	4.39e-03	+	1.71e-03	+	1.92e-06	+
CEC-9	4.29e-06	+	8.94e-01	=	3.39e-01	=	1.13e-05	+
CEC-10	7.27e-03	+	5.71e-02	=	1.29e-03	+	1.73e-06	+
WRST(+/=/-)		9/1/0		7/3/0		7/2/1		9/1/0

Table 7 Wilcoxon Signed-Rank Test for CEC 2022 test suite

HHHOALA vs	WOA		ALA		GWO		HHO	
Function	p-value	winner	p-value	winner	p-value	winner	p-value	winner
CEC-1	1.73e-06	+	1.73e-06	+	1.73e-06	+	1.73e-06	+
CEC-2	3.41e-05	+	3.06e-04	+	1.96e-03	+	1.73e-06	+
CEC-3	1.73e-06	+	1.73e-06	+	3.93e-01	=	1.73e-06	+
CEC-4	6.84e-03	+	2.83e-04	-	5.71e-04	-	1.25e-04	+
CEC-5	2.60e-06	+	2.13e-06	+	4.91e-01	=	1.73e-06	+
CEC-6	4.07e-05	+	9.32e-06	+	1.64e-05	+	2.16e-05	+
CEC-7	1.92e-06	+	1.73e-06	+	1.38e-03	+	1.73e-06	+
CEC-8	1.13e-05	+	1.96e-02	+	5.32e-03	+	1.92e-06	+
CEC-9	1.73e-06	+	1.73e-06	+	1.73e-06	+	1.73e-06	+
CEC-10	8.31e-04	+	6.98e-06	+	7.19e-02	=	2.60e-05	+
CEC-11	1.36e-05	+	4.68e-03	+	4.73e-06	+	1.73e-06	+
CEC-12	1.73e-06	+	1.73e-06	+	1.36e-04	+	1.73e-06	+
WRST(+/=/-)		10/0/0		9/0/1		8/3/1		10/0/0

6.3.4 Friedman Rank Test

The Friedman Rank Test is a statistical procedure that is used to compare the performance of various optimization techniques. The results of the chosen optimizers for the CEC 2020 and CEC 2022 test functions are shown in Tables 8 and 9, respectively. For each function, scores are provided based on their working capabilities, with a lower mean rank meaning stronger optimization ability. For the CEC 2020, HHHOALA wins the best mean rank of 1.53 and HHO exhibits worst response value with a rank of 4.47. Again, for CEC 2022 test suite, HHHOALA keeps the better response that was rank value of 1.53 and HHO gives worst performance here that was 4.42. After analyzing the tables, it can be summary that, the ranking order — HHHOALA > GWO>ALA>WOA>HHO- stills unchanged for the both IEEE standard test functions. These results strongly show that HHHOALA is the best and most reliable algorithm among the ones that were examined. This test confirms that HHHOALA is better at optimizing problems in a variety of benchmark environment.

Table 8 Friedman Rank Test for CEC 2020 test suite

Function	HHHOALA	WOA	ALA	GWO	HHO
CEC-1	1.00	3.63	2.67	2.70	5.00
CEC-2	2.37	4.17	1.97	2.00	4.50
CEC-3	1.90	4.00	3.20	1.47	4.43
CEC-4	1.27	2.03	4.90	3.87	2.93
CEC-5	1.00	3.57	3.53	2.43	4.47
CEC-6	1.47	3.33	3.37	2.30	4.53
CEC-7	1.00	4.37	2.30	2.77	4.57
CEC-8	1.47	3.69	2.47	2.60	4.83
CEC-9	2.10	3.40	3.30	1.90	4.30
CEC-10	1.80	3.27	2.33	2.60	5.00
Sum ranks	15.38	35.46	30.04	24.64	44.56
Mean ranks	**1.53**	**3.54**	**3.00**	**2.46**	**4.47**

Table 9 Friedman Rank Test for CEC 2022 test suite

Function	HHHOALA	WOA	ALA	GWO	HHO
CEC-1	1.00	4.90	2.37	2.63	4.10
CEC-2	1.50	3.07	3.00	2.47	4.97
CEC-3	1.57	3.97	3.50	1.43	4.53
CEC-4	3.10	3.97	2.10	1.63	4.20
CEC-5	1.67	4.37	3.27	1.50	4.20
CEC-6	1.40	3.10	3.13	3.50	3.87
CEC-7	1.23	3.67	3.40	1.93	4.77
CEC-8	1.70	4.10	2.27	2.50	4.43
CEC-9	1.00	3.13	3.83	2.37	4.67
CEC-10	1.67	3.10	3.70	2.27	4.27
CEC-11	1.27	3.33	2.60	3.10	4.70
CEC-12	1.20	3.07	4.50	1.90	4.33
Sum ranks	18.31	43.78	37.67	27.23	53.04
Mean ranks	**1.53**	**3.64**	**3.14**	**2.27**	**4.42**

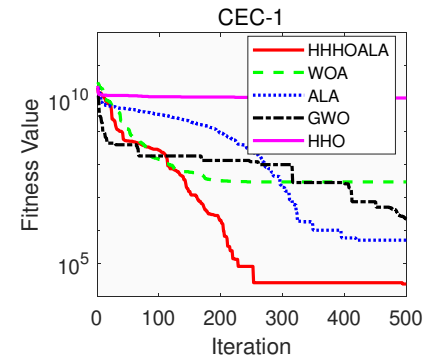
6.3.5 Convergence and Boxplot Analysis

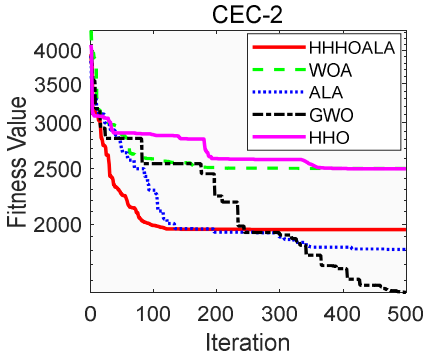
The suggested HHHOALA algorithm is tested on CEC benchmark functions, and compared to four popular optimization algorithms, i.e. WOA, ALA, GWO, and HHO. Comparison of performance is made in terms of convergence curves and boxplot representation of the CEC-2020 and CEC-2022 benchmark suites. Based on the intersection points of the CEC-2020 benchmark functions, it is possible to note that the hybrid optimizer has a faster convergence phenomenon than the other algorithms are shown in figure 4. In a number of functions, including CEC-1, and CEC-5 through CEC-10, HHHOALA quickly drops the fitness value of the initial iterations and keeps on improving the solution of the subsequent iterations. This means that the balance between the capabilities of exploration and exploitation of the proposed algorithm are good. Conversely the other algorithms like WOA and GWO tend to take longer before converging whereas, HHO tends to converge at higher values of the fitness because of premature convergence.

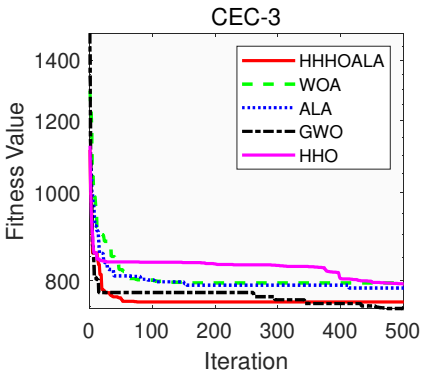
The stability and strength of the proposed method is further demonstrated in the boxplot analysis of the CEC-2020 functions. It is clear that the boxplots indicate that HHHOALA tends to have smaller median fitness values and small interquartile ranges relative to the competing algorithms. It means that the given algorithm is less prone to inconsistency in the results of independent executions. Further, the non-existence of outliers in HHHOALA shows that it is a good optimizer, and algorithm, like WOA and HHO, has a greater variation in various benchmark functions.

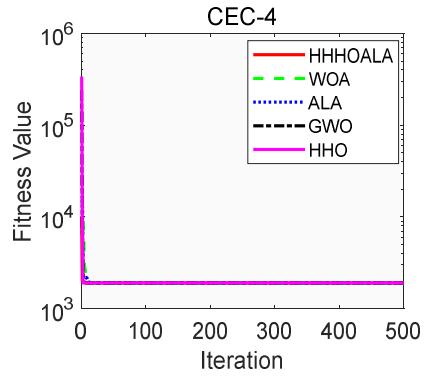
In the same way, the effectiveness of the proposed HHHOALA algorithm is also proven by the convergence curves of the CEC-2022 benchmark functions. HHHOALA is more likely to converge quickly and attain less fit values in most cases than the other algorithms. The proposed algorithm exhibits a significant acceleration in convergence speed and solution quality especially in its use in CEC-5, CEC-7, CEC-8, CEC-9, and CEC-10 are shown in figure 6. The sharp decline in the fitness value in the initial iterations of the work shows a great global search capacity, whereas the gradual work refinement in the later iterations demonstrates efficient local exploitation. Boxplot analysis is used to statistically test and visually assess how well the suggested swarm-driven computational intelligence strategy works compared to other techniques on the CEC 2020 and CEC 2022 test functions shown in figure 5 & 7 respectively. The boxplots show how the fitness values from several independent runs are spread out. This findings show that the HHHOALA method consistently has superior convergence behavior with a decrease across most evaluation functions than competing optimizers. This illustrates that the suggested technique works well to keep the quality of the solution high while avoiding early convergence, especially in landscapes with intricate functions.

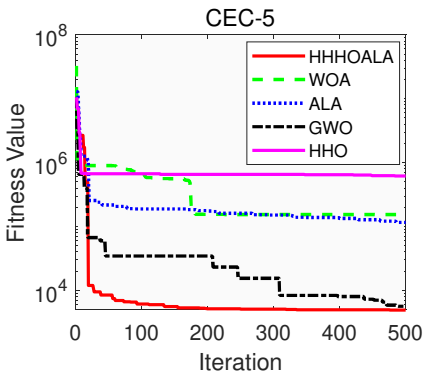
On balance, the obtained experimental results according to both CEC-2020 and CEC-2022 benchmark functions indicate that the offered HHHOALA algorithm has better optimization behavior in terms of convergence rate, accuracy of the solutions, and stability, as compared to WOA, ALA, GWO, and HHO. It can be explained by the fact that the hybrid mechanism of search of HHHOALA works to improve the balance between exploration and exploitation in the optimization process.

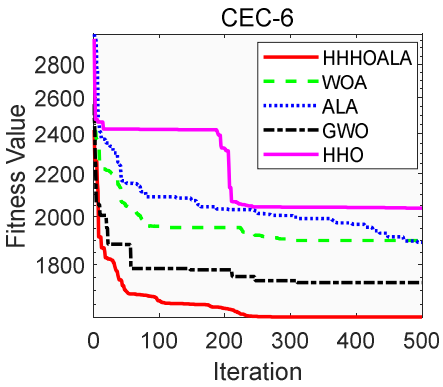

CEC-1
HHHOALA
WOA
ALA
GWO
HHO
Fitness Value
10^{10}
10^{5}
0
100
200
300
400
500
Iteration


CEC-2
HHHOALA
WOA
ALA
GWO
HHO
Fitness Value
4000
3500
3000
2500
2000
0
100
200
300
400
500
Iteration


CEC-3
HHHOALA
WOA
ALA
GWO
HHO
Fitness Value
1400
1200
1000
800
0
100
200
300
400
500
Iteration


CEC-4
HHHOALA
WOA
ALA
GWO
HHO
Fitness Value
10^{6}
10^{5}
10^{4}
10^{3}
0
100
200
300
400
500
Iteration


CEC-5
HHHOALA
WOA
ALA
GWO
HHO
Fitness Value
10^{8}
10^{6}
10^{4}
0
100
200
300
400
500
Iteration


CEC-6
HHHOALA
WOA
ALA
GWO
HHO
Fitness Value
2800
2600
2400
2200
2000
1800
0
100
200
300
400
500
Iteration

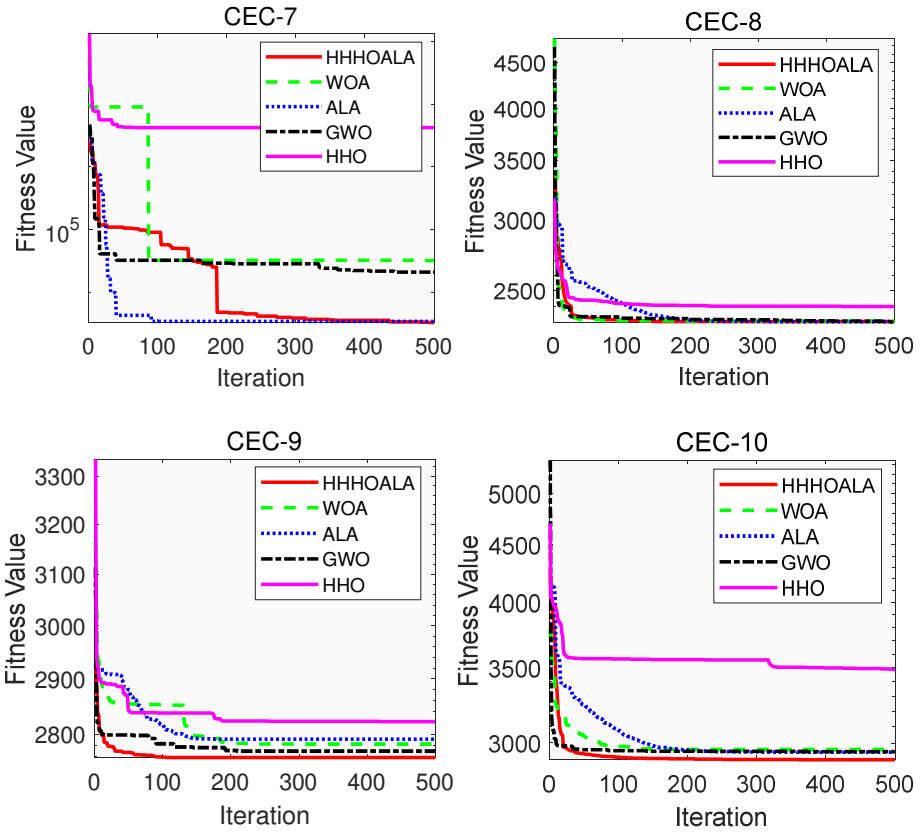
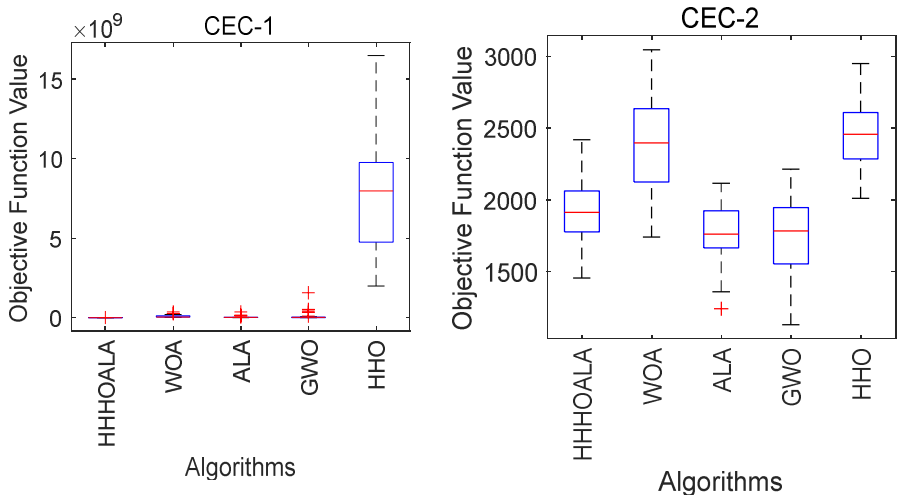
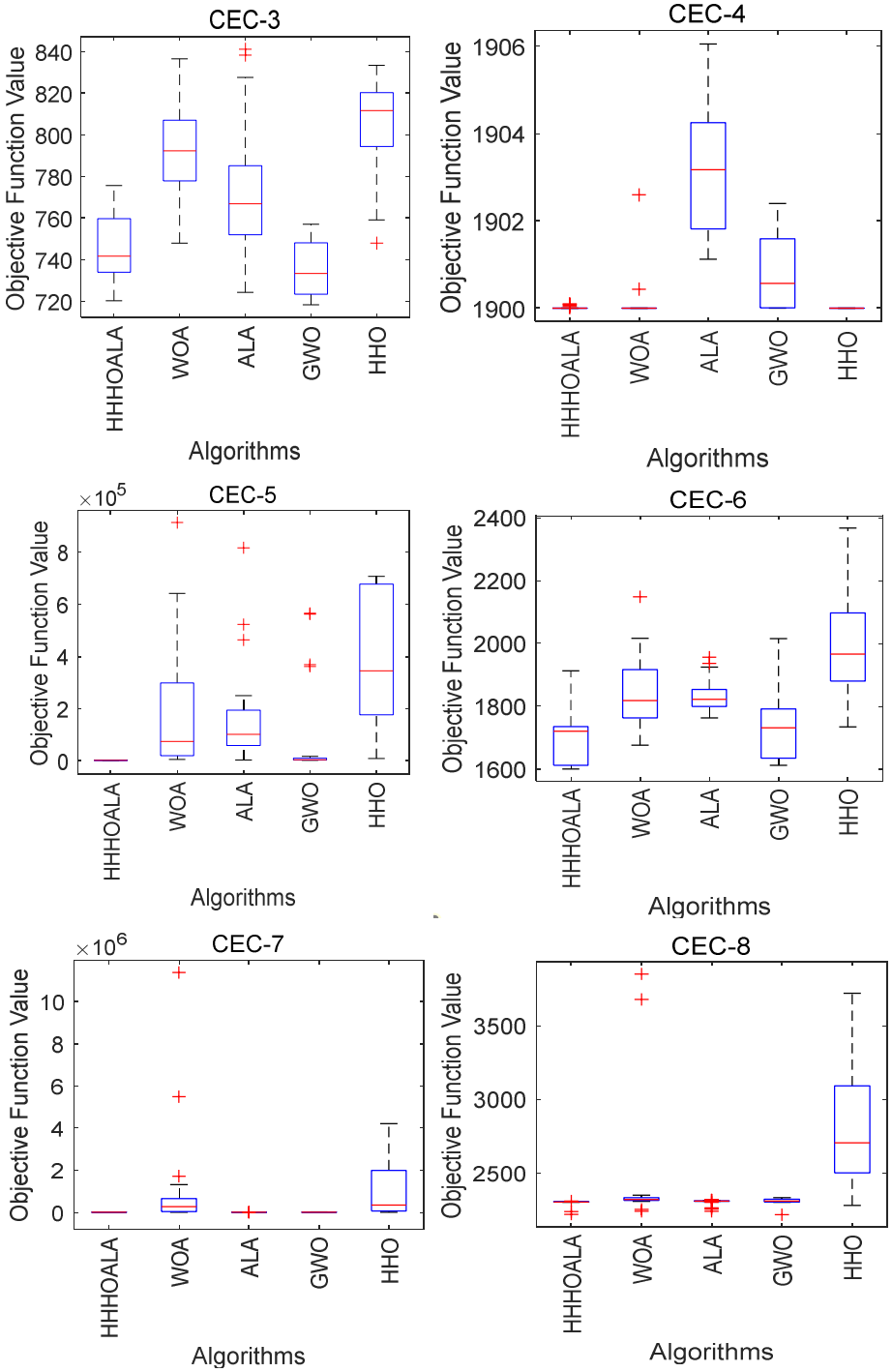


Fig. 4 Convergence curve of different optimizers for CEC 2020 test functions




CEC-3
Objective Function Value
840
820
800
780
760
740
720
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-4
Objective Function Value
1906
1904
1902
1900
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-5
×10^5
Objective Function Value
8
6
4
2
0
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-6
Objective Function Value
2400
2200
2000
1800
1600
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-7
×10^6
Objective Function Value
10
8
6
4
2
0
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-8
Objective Function Value
3500
3000
2500
HHHOALA
WOA
ALA
GWO
HHO
Algorithms

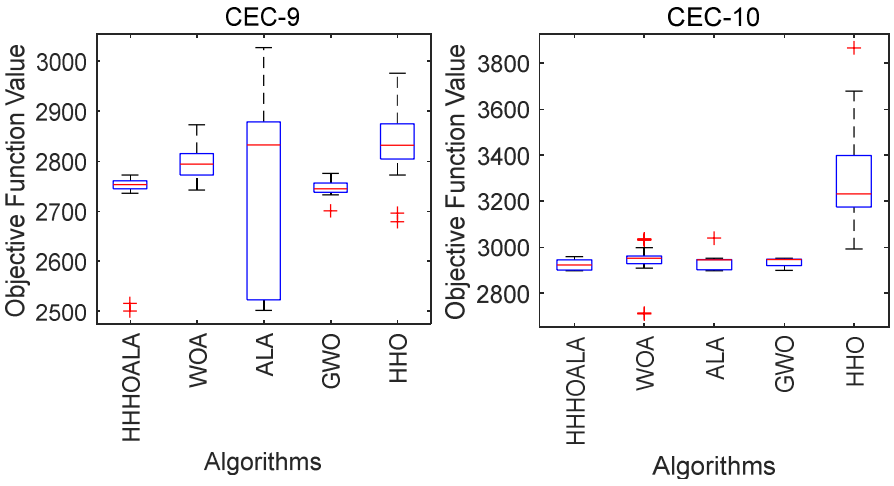
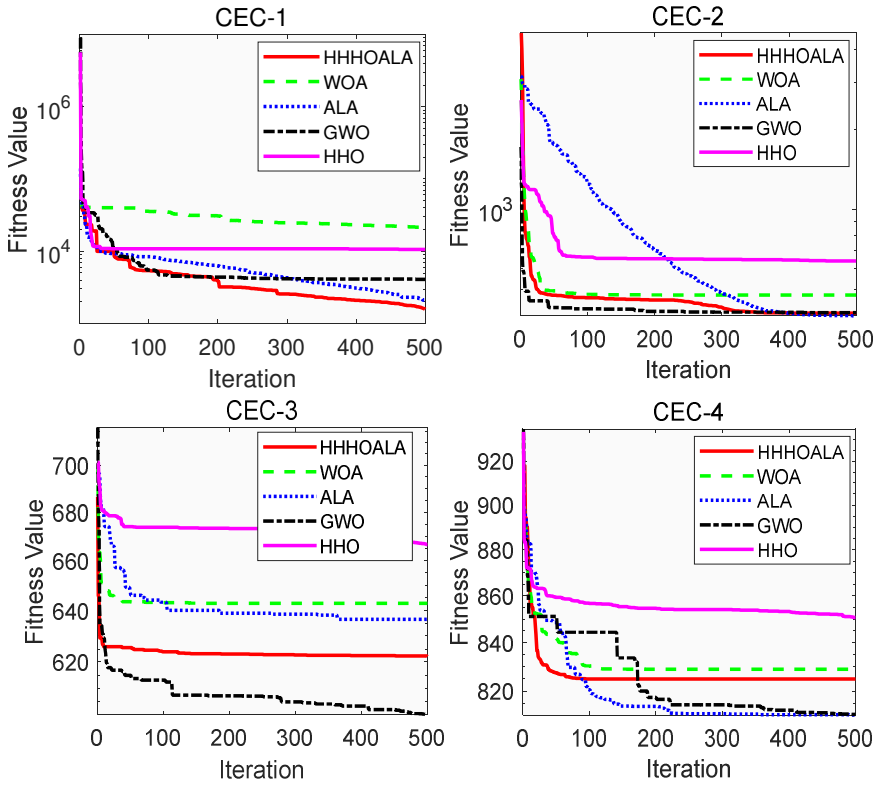
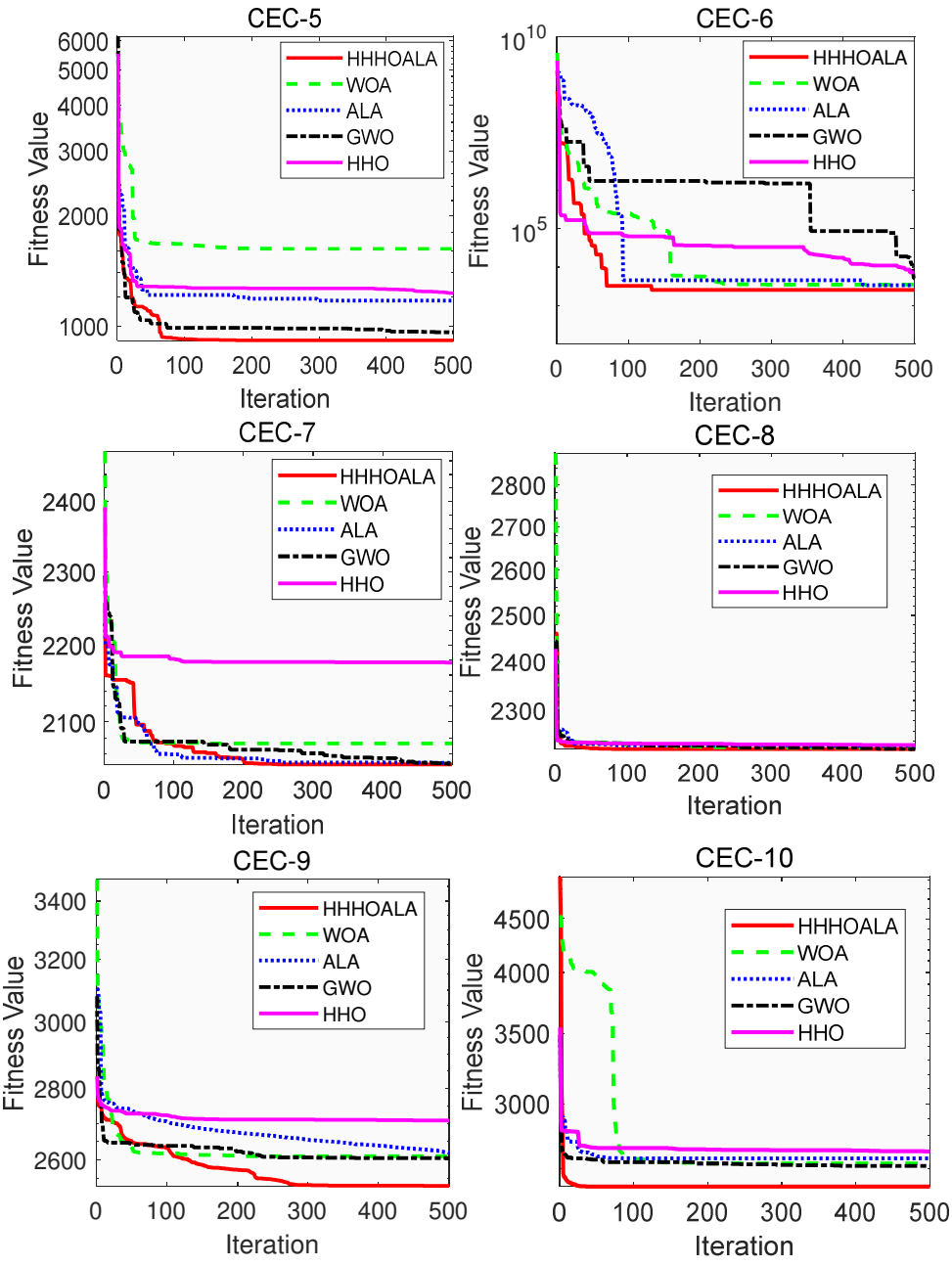


Fig.5 Boxplot analysis of different optimizers for CEC 2020 test functions




CEC-5
CEC-6
CEC-7
CEC-8
CEC-9
CEC-10
Fitness Value
Iteration
HHHOALA
WOA
ALA
GWO
HHO

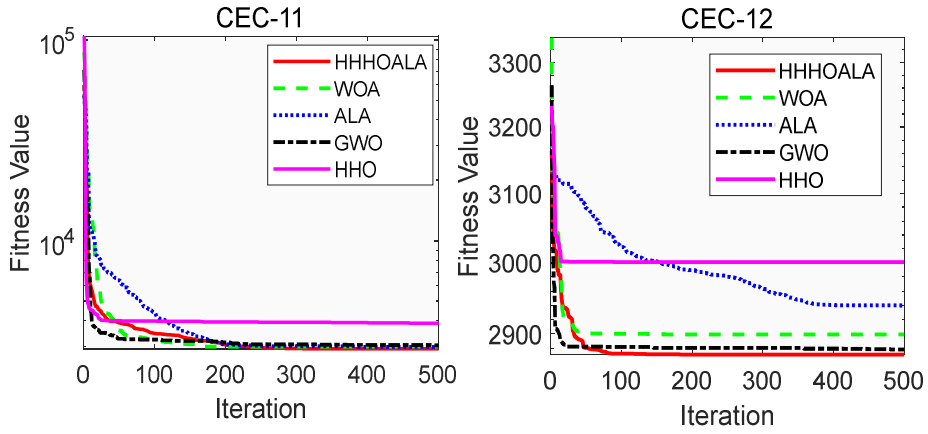
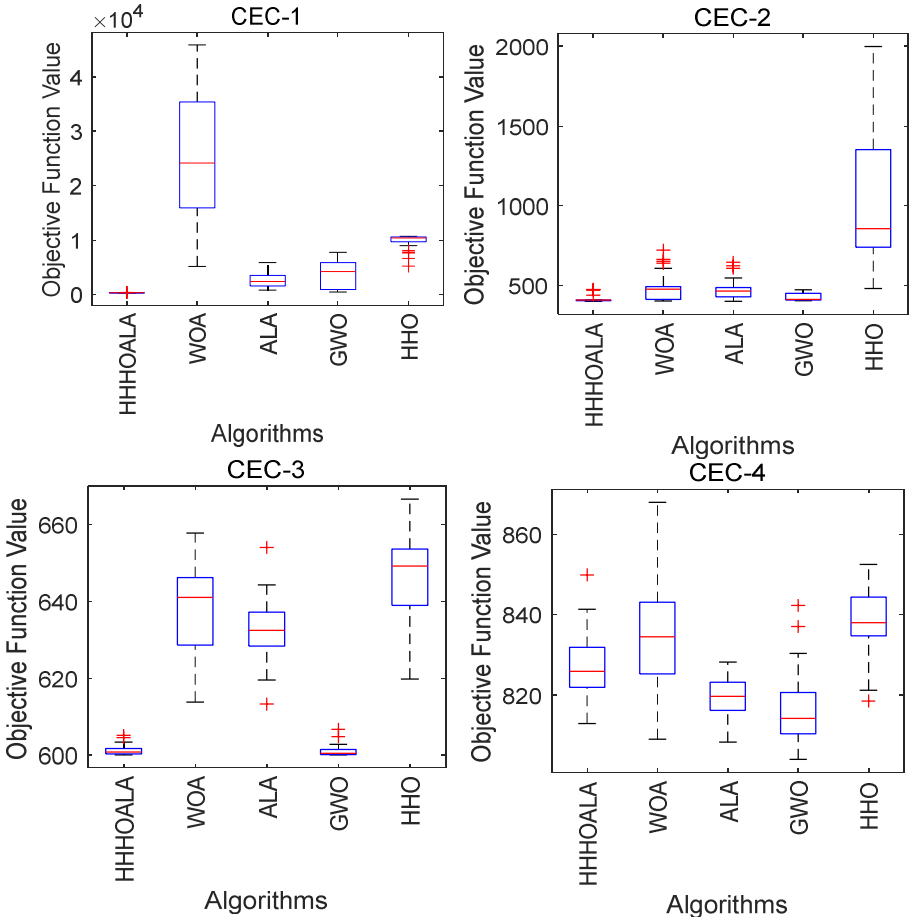
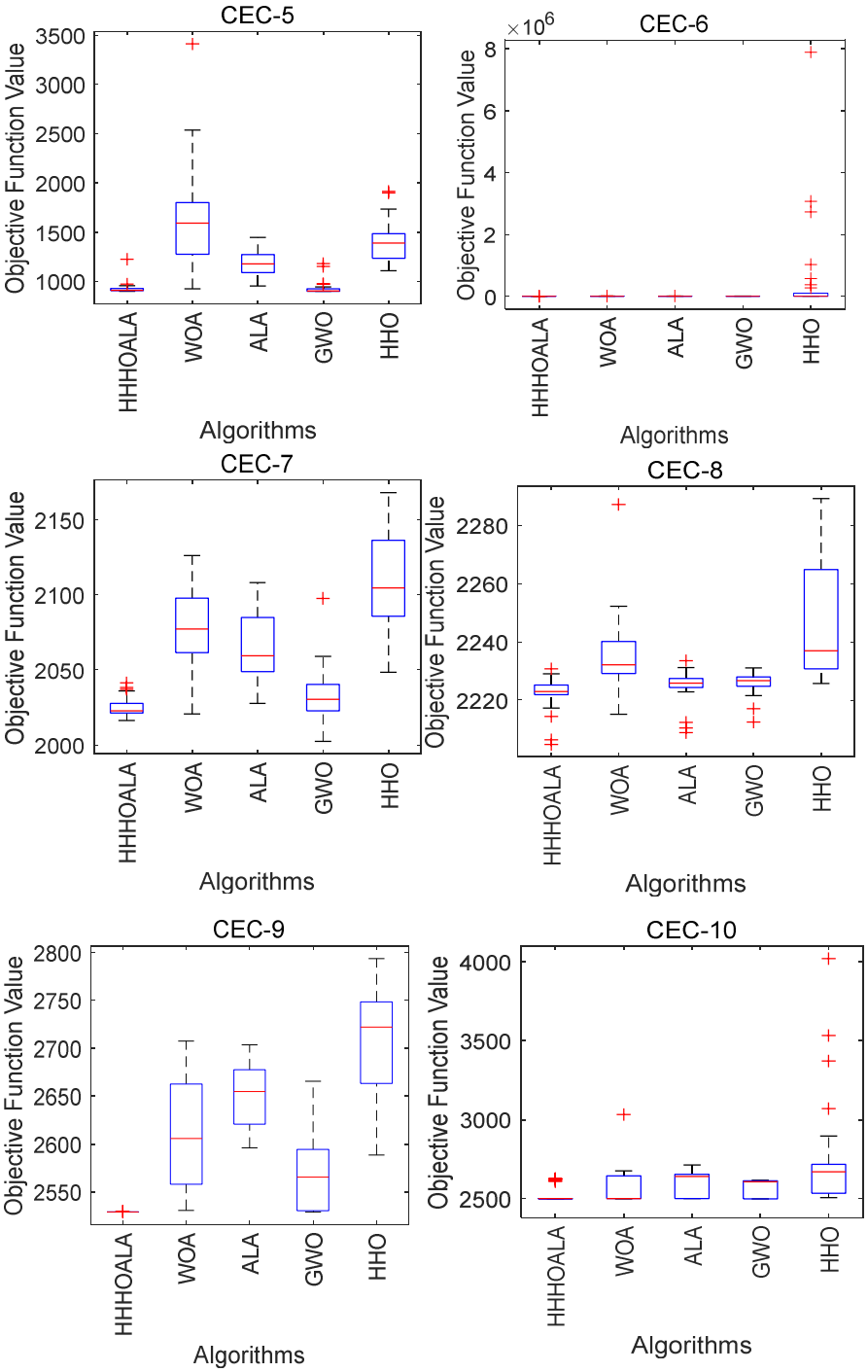


Fig.6 Convergence curve of different optimizers for CEC 2022 test functions




CEC-5
Objective Function Value
3500
3000
2500
2000
1500
1000
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-6
×10^6
Objective Function Value
8
6
4
2
0
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-7
Objective Function Value
2150
2100
2050
2000
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-8
Objective Function Value
2280
2260
2240
2220
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-9
Objective Function Value
2800
2750
2700
2650
2600
2550
HHHOALA
WOA
ALA
GWO
HHO
Algorithms
CEC-10
Objective Function Value
4000
3500
3000
2500
HHHOALA
WOA
ALA
GWO
HHO
Algorithms

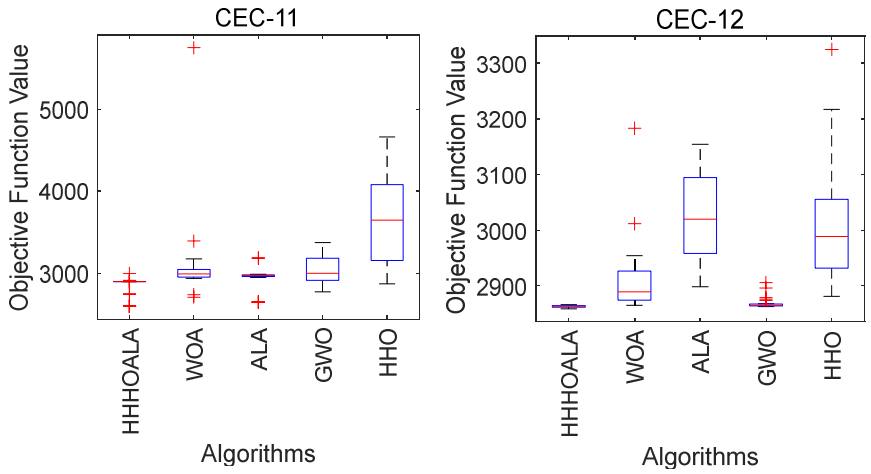


Fig.7 Boxplot analysis of different optimizers for CEC 2022 test functions

7 Solving Engineering Design Problems Using HHHOALA

The suggested HHHOALA technique is utilized to solve engineering design issues in this section demonstrates that it can be employed in various ways and has the ability to improve working systems which are considered complex. This section considers some of the case studies including designing a pressure vessel, and a welded beam design. In both instances, our suggested optimizer reveals that it is able to reduce costs while at the same time adhering to critical design principles. This demonstrates that it may prove to be a highly valuable tool to optimizing field.

7.1 Pressure Vessel Design Problem

Pressure vessels is a container that store liquids or gasses and endure internal pressure to avoid leaks. Figure 8 depicts the main diagram of the pressure vessel. The target of this problem is to minimize the manufacturing costs of the pressure vessel while meeting strength and safety constraints. Tables 10 and 11 give the results that were achieved in solving this problem using HHHOALA and other strategies. This problem has four decision variables: thickness of the shell, shell diameter, head thickness, and head diameter. The objective and constraints functions for this problem are as follows.

Consider: $\varepsilon = [\varepsilon_{1,}\varepsilon_{2,}\varepsilon_{3,}\varepsilon_{4,}] = [T_s, T_h, R, L]$
Minimize: $f(\varepsilon) = 0.6224\varepsilon_1\varepsilon_3\varepsilon_4 + 1.778\varepsilon_1\varepsilon_3^2 + 3.1661\varepsilon_1^2\varepsilon_4 + 19.84\varepsilon_1^2\varepsilon_3$
Subject to :

$$g_1(y) = -\varepsilon_1 + 0.0193\varepsilon_3 \leq 0$$
$$g_2(y) = -\varepsilon_2 + 0.00954\varepsilon_3 \leq 0$$
$$g_3(y) = -\pi\varepsilon_3^2\varepsilon_4 - \frac{4}{3}\pi\varepsilon_3^3 + 1296000 \leq 0$$
$$g_4(y) = \varepsilon_4 - 240 \leq 0$$
$$0 \leq \varepsilon_1, \varepsilon_2 \leq 100 \quad and \; 10 \leq \varepsilon_3, \varepsilon_4 \leq 200$$

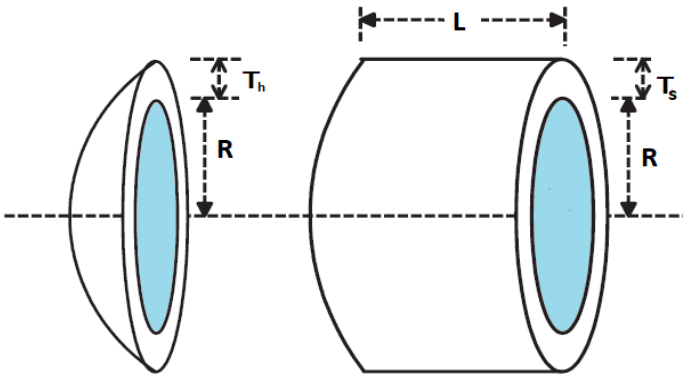


Fig.8 Pressure vessel design problem

Table 10 Statistical analysis for the optimizers

Algorithm	Best	Mean	Worst	SD	Median
HHHOALA	**5902.2904750340**	6036.1961198331	7243.8407380125	262.5786814302	5931.9302982593
WOA	6174.9605832870	15976.627926254	53212.941177070	11613.45593416	11857.945849910
ALA	6158.6018270574	131019.14331860	387106.06087793	120199.3110305	116325.36855222
GWO	6089.4045106726	96339.323126278	285601.93264257	81604.93047916	76664.157764610
HHO	7073.6936830593	12907.192894599	54884.663521647	8985.920769848	9790.8804020339

Table 11 Outcomes for the pressure vessel design

Algorithm	Optimal Variables				Optimum Cost
	T_s	T_h	R	L	
HHHOALA	0.7823	0.3887	40.5222	197.3018	**5902.2904750340**
WOA	0.8973	0.4373	45.8127	135.4706	6174.9605832870
ALA	0.8248	0.4219	41.8304	181.2002	6158.6018270574
GWO	0.7906	0.4211	40.8075	196.5723	6089.4045106726
HHO	1.0526	0.4466	46.7820	126.1177	7073.6936830593

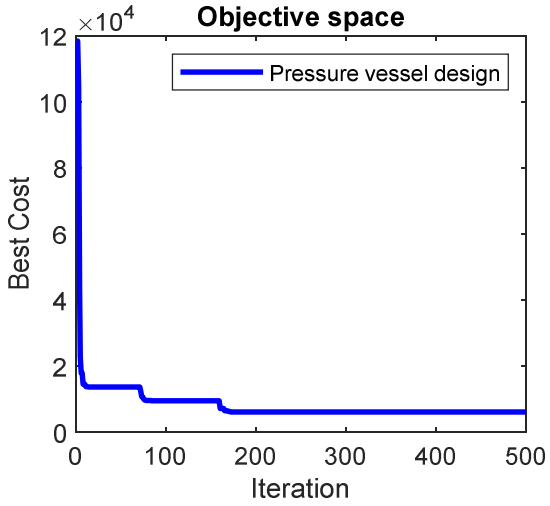


Fig. 9 Optimal cost value for HHHOALA optimizer

7.2 Welded Beam Design

The goal of this problem is to find the best values for the weld thickness (h), bar length (l), height (t), and thickness (b) such that the overall cost of making a welded beam is as cheap as possible. The frameworks of the design that is being analyzed is depicted in figure 10, and the mathematical formulation that corresponds to it is indicated in equation. The outcomes that were obtained by addressing this problem utilizing HHHOALA and other strategies are presented in Table 12 and 13 respectively. From analyzing the table, it is clear that suggested method has lower optimal cost values**.**

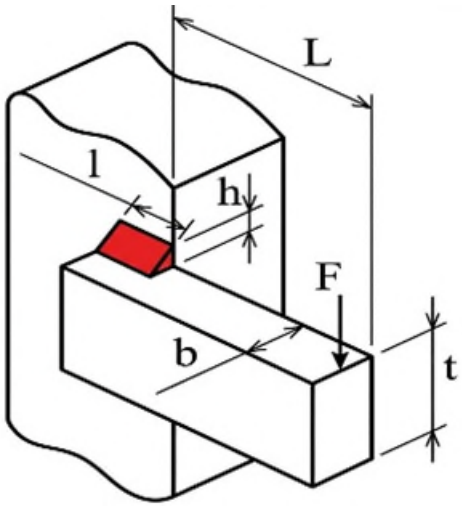


Fig. 10 Welded beam design problem

Consider: $\varepsilon = \left[\varepsilon_{1,}\varepsilon_{2,}\varepsilon_{3,}\varepsilon_{4,}\right] = [h\,, l, t, b]$

Minimize: $f(\varepsilon) = 1.1047\varepsilon_1^2\varepsilon_2 + 0.04811\varepsilon_3\,\varepsilon_4(14 + \varepsilon_2)$

Subject to:

$$g_1(\varepsilon) = \tau(\varepsilon) - 13600 \le 0,$$
$$g_2(\varepsilon) = \sigma(\varepsilon) - 30000 \le 0,$$
$$g_3(\varepsilon) = \varepsilon_1 - \varepsilon_4 \le 0,$$
$$g_4(\varepsilon) = 0.1047\varepsilon_1^2 + 0.04811\varepsilon_3\varepsilon_4(14 + \varepsilon_2) - 5.0 \le 0,$$
$$g_5(\varepsilon) = 0.125 - \varepsilon_1 \le 0,$$
$$g_6(\varepsilon) = \delta(\varepsilon) - 0.25 \le 0,$$
$$g_7(\varepsilon) = 6000 - p_c(\varepsilon) \le 0,$$

$$\tau(\varepsilon) = \sqrt{(\tau')^2 + (2\tau\tau')\ \frac{\varepsilon_2}{2Q} + (\tau'')^2}\ , \tau' = \frac{6000}{\sqrt{2}\varepsilon_1\varepsilon_2}\ , \tau'' = \frac{PQ}{R},$$

$$R = 2\{\varepsilon_1\varepsilon_2\sqrt{2}\left[\frac{\varepsilon_2^2}{4} + \left(\frac{\varepsilon_1+\varepsilon_3}{2}\right)^2\right]\}, \sigma(\varepsilon) = \frac{504000}{\varepsilon_4\varepsilon_3^2},$$

$$P = 6000\left(14 + \frac{\varepsilon_2}{2}\right), Q = \sqrt{\frac{\varepsilon_2^2}{4} + \left(\frac{\varepsilon_1+\varepsilon_3}{2}\right)^2},$$

$$\delta(\varepsilon) = \frac{65856000}{(30.10^6)\varepsilon_4\varepsilon_3^3}\ , p_c(\varepsilon) = \frac{4.013(30.10^6)\sqrt{\frac{\varepsilon_3^2\varepsilon_4^6}{36}}}{196}\ (1 - \frac{\varepsilon_3}{28}\sqrt{\frac{30.10^6}{4(12.10^6)}}),$$

$$0.1 \le \varepsilon_1, \varepsilon_4 \le 2\ and\ 0.1 \le \varepsilon_2, \varepsilon_3 \le 10$$

Table 12 Statistical analysis for the optimizers

Algorithm	Best	Mean	Worst	SD	Median
HHHOALA	**1.7248875120**	1.7329777623	1.7946047832	0.0184540499	1.7261389433
WOA	1.8475076855	2.7069902450	5.3137006656	0.8037217700	2.5095923519
ALA	1.7703593256	2.3051361869	3.2212220226	0.3661790159	2.2003420694
GWO	1.7352172430	1.7492410194	1.7538757961	0.0017460498	1.7392134536
HHO	1.7769903196	2.8353574658	4.7509884205	0.7289024435	2.6121890787

Table 13 Outcomes for the welded beam design

Algorithm	Optimal Variables				Optimum Cost
	h	*l*	*t*	*b*	
HHHOALA	0.2057	3.4710	9.0367	0.2057	**1.7248875120**
WOA	0.1604	4.7721	9.0973	0.2084	1.8475076855
ALA	0.2018	3.7267	8.9595	0.2098	1.7703593256
GWO	0.2054	3.4791	9.0396	0.2058	1.7352172430
HHO	0.1785	4.2051	9.0327	0.2059	1.7769903196

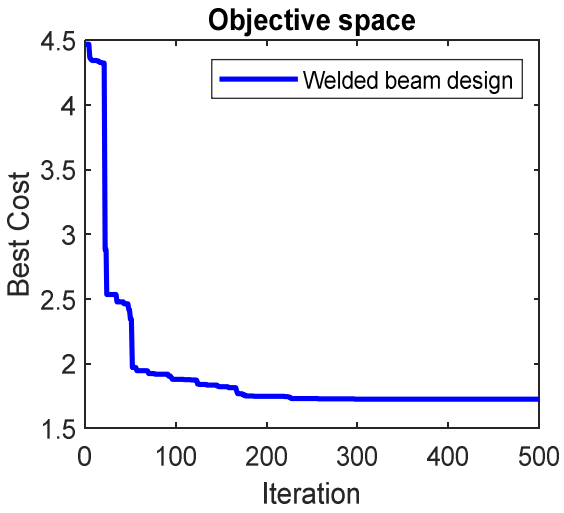


Fig. 11 Optimal cost value for HHHOALA optimizer

8 Conclusion

This study presents an innovative hybrid metaheuristic, HHHOALA, formulated to address a diverse range of global optimization and limited engineering challenges. Many tests were done on standard benchmark suites including IEEE CEC 2020, IEEE CEC 2022 to show that HHHOALA is better in terms of solution quality, convergence speed, and statistical robustness. HHHOALA usually gets the lowest mean objective values, which means it does better than cutting-edge algorithms like HHO, ALA, WOA, and GWO. The Wilcoxon Signed Rank Test and the Friedman Rank Test demonstrate the consistency and dependability of the proposed optimizer by showing that these results are statistically significant. Furthermore, the effectiveness of the proposed HHHOALA algorithm is validated on two widely recognized constrained engineering optimization problems: the pressure vessel design and the welded beam design. In both cases, HHHOALA either finds the best solutions that have been reported in the literature or gives results that are very close to the best ones but with much lower standard deviation values. This shows how strong and consistent it is. The convergence behavior seen in all studies shows that HHHOALA has a quick convergence speed and steady, reliable performance during the whole optimization process. The study encompasses comprehensive performance including Qualitative analysis, statistical analysis, and testing of exploration-exploitation behavior to rigorously confirm the algorithm's efficacy. In subsequent study, the suggested HHHOALA can be expanded and utilized for many emergent real-world optimization challenges. Some possible uses are tweaking hyper parameters in machine learning models, finding the best way to use resources in massive systems, optimizing smart grids and energy management, and economic load dispatch under uncertainty, demonstrating its practical value across diverse fields. In addition, adding adaptive control methods or hybrid deep learning frameworks to HHHOALA could make it work better in contexts that change and have a lot of dimensions. These extensions would show even more how useful and scalable the approach is in a wide range of fields.

References

1 Dhiman, G, Kumar, V (2019) Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowledge-Based Systems 165, 169–196. https://doi.org/10.1016/j.knosys.2018.11.024

2 Dhal, K.G, Sasmal, B, Das, A (2023) A Comprehensive Survey on Arithmetic Optimization Algorithm. Arch Computat Methods Eng 30:3379–3404.https://doi.org/10.1007/s11831-023-09902-3

3 BAS, E (2024) BinDMO: a new Binary Dwarf Mongoose Optimization algorithm on based Z-shaped, U-shaped, and taper-shaped transfer functions for CEC-2017 benchmarks. Neural Comput & Applic 36:6903–6935. https://doi.org/10.1007/s00521-024-09436-0

4 Liu, Y, As'arry, A, Hassan, M.K (2024) Review of the grey wolf optimization algorithm: variants and applications. Neural Comput & Applic 36: 2713–2735. https://doi.org/10.1007/s00521-023-09202-8

5 Rana, N, Latiff, M.S.A, Abdulhamid, S.M (2020) Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments. Neural Comput & Applic 32:16245–16277. https://doi.org/10.1007/s00521-020-04849-z

6 Şenel, F.A, Gökçe, F, Yüksel, A.S (2019) A novel hybrid PSO–GWO algorithm for optimization problems. Engineering with Computers 35:1359–1373. https://doi.org/10.1007/s00366-018-0668-5

7 Abdalla, R, Hollstein, W, Carvajal, C.P (2023) Actor-critic reinforcement learning leads decision-making in energy systems optimization—steam injection optimization. Neural Comput & Applic 35:16633–16647. https://doi.org/10.1007/s00521-023-08537-6

8 El-kenawy, ES.M, Khodadadi, N, Mirjalili, S (2026) Glider snake optimizer (GSO): a nature-inspired metaheuristic algorithm for global and engineering optimization problems. Artif Intell Rev 59:91. https://doi.org/10.1007/s10462-026-11504-x

9 Shami, T.M, Grace, D, Burr, A (2024) Single candidate optimizer: a novel optimization algorithm. Evol. Intel. 17:863–887. https://doi.org/10.1007/s12065-022-00762-7

10 Gopi, S, Mohapatra, P (2025) A modified grey wolf optimization algorithm to solve global optimization problems. OPSEARCH 62:337–367. https://doi.org/10.1007/s12597-024-00785-x

10 Rodan, A. (2025) Enhanced Frilled Lizard Optimizer for Global Optimization and Engineering Design Problems. Int J Comput Intell Syst 18:257. https://doi.org/10.1007/s44196-025-00969-3

11 Anka, F, Gharehchopogh, F.S, Tejani, G.G (2025) Advances in Mountain Gazelle Optimizer: A Comprehensive Study on its Classification and Applications. Int J Comput Intell Syst 18:247. https://doi.org/10.1007/s44196-025-00968-4

12 Zolghadr-Asli, B, Bozorg-Haddad, O, Chu, X. (2018) Crow Search Algorithm (CSA). In: Bozorg-Haddad, O. Advanced Optimization by Nature-Inspired Algorithms. Studies in Computational Intelligence, vol 720. Springer, Singapore. https://doi.org/10.1007/978-981-10-5221-7_14

13 Wang, W, Lu, Y. & Wang, S (2025) A multi-strategy enhanced circle search algorithm based on adaptive evolutionary framework and its application in image segmentation. Cluster Comput 28:329. https://doi.org/10.1007/s10586-024-04982-7

14 Rizk-Allah, R.M, Saleh, O, Hagag, E.A (2021) Enhanced Tunicate Swarm Algorithm for Solving Large-Scale Nonlinear Optimization Problems. Int J Comput Intell Syst 14:189. https://doi.org/10.1007/s44196-021-00039-4

15 Ehteram, M, Seifi, A, Banadkooki, F.B (2023) Rat Swarm Optimization Algorithm. In: Application of Machine Learning Models in Agricultural and Meteorological Sciences. Springer, Singapore. https://doi.org/10.1007/978-981-19-9733-4_9

16 Mohammadi-Balani, A, Dehghan Nayeri, M, Azar, A, Taghizadeh-Yazdi, M (2021) Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Computers & Industrial Engineering*, **152,** 107050. https://doi.org/10.1016/j.cie.2020.107050

17 Sheta, A, Braik, M, Al-Hiary, H (2023) Improved versions of crow search algorithm for solving global numerical optimization problems. Appl Intell 53:26840–26884. https://doi.org/10.1007/s10489-023-04732-z

18 Xiao, Y, Cui, H, Khurma, R.A (2025) Artificial lemming algorithm: a novel bionic meta-heuristic technique for solving real-world engineering optimization problems. Artif Intell Rev 58:84.https://doi.org/10.1007/s10462-024-11023-7

18 Gehad I.S (2022) A Novel Multi-Objective Rat Swarm Optimizer-Based Convolutional Neural Networks for the Diagnosis of COVID-19 Disease. Aut. Control Comp. Sci. 56:198–208. https://doi.org/10.3103/S0146411622030075

19 Medjahed, S.A , Boukhatem, F (2025) A new optimization-based framework for enhanced feature selection with the Narwal optimizer. J Supercomput 81:1274. https://doi.org/10.1007/s11227-025-07741-6

20 Si-Ma, S, Liu, HM, Zhan, HX (2025) Efficient maximum iterations for swarm intelligence algorithms: a comparative study. Artif Intell Rev 58:87.https://doi.org/10.1007/s10462-024-11104-7

21 Mohammed, B.O, Aghdasi, H.S. & Salehpour, P (2025) Dhole optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Cluster Comput 28:430. https://doi.org/10.1007/s10586-024-05005-1

22 Brabazon, A, O'Neill, M, McGarraghy, S (2015) Genetic Algorithm. In: Natural Computing Algorithms. Natural Computing Series. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-43631-83

23 Alabool, H.M, Alarabiat, D, Abualigah, L. et al (2021) Harris hawks optimization: a comprehensive review of recent variants and applications. *Neural Comput & Applic* 33:8939–8980. https://doi.org/10.1007/s00521-021-05720-5

24 Gharehchopogh, F.S, Maleki, I. & Dizaji, Z.A (2022) Chaotic vortex search algorithm: metaheuristic algorithm for feature selection. Evol. Intel. 15, 1777–1808. https://doi.org/10.1007/s12065-021-00590-1