Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 13, No. 4, 2025, 515-540.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Enhanced Opposition-Based Coati Optimization Algorithm for Solving Global Optimization

Soodeh Shadravan* and Ali Karimi

1. *Department of Computer Engineering, Bardsir Branch, Islamic Azad University, Bardsir, Iran.*
2. *Department of Information Technology Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran.*

Article Info

Article History:
Received 09 December 2024
Revised 19 January 2025
Accepted 30 April 2025

DOI:10.22044/jadm.2025.15224.2627

Keywords:
Coati Optimization Algorithm (COA), Metaheuristic Algorithms, Opposition-Based Learning, Improved Opposition-Based Learning

**Corresponding author: so.shadravan@iau.ac.ir (S. Shadravan).*

Abstract

The Coati Optimization Algorithm (COA) is a newly developed metaheuristic algorithm, dr÷awing inspiration from the clever tactics Coatis use when attacking Iguanas as well as their strategies for dealing with and evading predators. This algorithm has shown a commendable level of effectiveness when compared to various other metaheuristic algorithms. Its performance metrics indicate that it outperforms many alternatives in terms of efficiency and results. To overcome challenges such as the imbalance between exploration and exploitation phases and become trapped in local optima for solving complex optimization problems, an innovative technique known as "Enhanced Opposition-Based Learning" (EOBL) has been integrated with the COA algorithm. This technique draws inspiration from Random Opposition-Based Learning methods and can effectively influence the balance between exploration and exploitation phases. The Enhanced of Coati Optimization Algorithm (EOBCOA) is a novel metaheuristic algorithm proposed to enhance the performance of the COA. This method has been applied on standard benchmark functions to improve the proposed optimization algorithm. To assess the effectiveness of the proposed EOBCOA method, it was tested on standard benchmark functions, including IEEE CEC2005, IEEE CEC2019, and seven engineering problems. The results show that the EOBCOA method outperforms other advanced algorithms in achieving global optimization.

1. Introduction

Before the era of exploratory optimization, analytical methods were a superior approach for solving mathematical problems that, in addition to initial information about the numerical value or objective function value, required information related to the derivatives of the objective functions. These methods, armed with supplementary information, were capable of efficiently uncovering the precise optimum for linear problems or convex nonlinear problems. However, they encountered challenges in more complex scenarios with multiple local optima, often becoming stuck in these local optima. The stochastic nature and unpredictable search space characteristic of real-world problems highlighted the need for the development of metaheuristic algorithms [1]. Optimization techniques are generally divided into two primary categories: deterministic methods and stochastic methods. Among the most prevalent deterministic approaches are linear and nonlinear programming. These methods are characterized by their dependence on gradient information, which aids in navigating the solution space to find optimal solutions. An alternative to these traditional methods is stochastic approaches, which are similar to metaheuristic algorithms in that they create and utilize random variables [36].

Metaheuristic techniques, owing to their inherently stochastic nature, operate independently of the specific characteristics of the problem at hand and do not necessitate the availability of derivative information related to the problem. This characteristic renders them particularly versatile, allowing for their application across a wide array of optimization challenges without being constrained by the underlying mathematical structure of the problem. This feature contrasts with mathematical programming, which usually requires extensive knowledge about mathematical issues. This lack of dependence on the specific problem has made metaheuristic methods a valuable and suitable tool for identifying the optimal solution for various optimization problems, regardless of the nonlinearity of the search space and its constraints.

Another advantage of this method is its flexibility, enabling it to address any type of optimization problem without requiring significant modifications to the algorithm's structure. This approach treats the problem as a black box with defined input and output states, making it a potential candidate for a user-friendly optimizer. In contrast to the algebraic methodologies commonly employed in mathematical approaches, this particular technique predominantly relies on random operators. This characteristic significantly reduces the likelihood of becoming ensnared in local optima, a common challenge faced by traditional algebraic methods. The ability of this technique to conduct a global exploration of the search space within a reasonable timeframe is a notable advantage [2].

A fundamental trait shared by metaheuristic algorithms is their structured approach to the optimization process, which is typically divided into two contrasting phases: exploration and exploitation. Exploration involves the search for new and diverse solutions across the solution space, enabling the algorithm to discover various potential areas for improvement. In contrast, the exploitation phase focuses on refining and enhancing the solutions that have already been identified, aiming to achieve optimal outcomes in specific regions of the search space. The interplay between these two phases is crucial, as effective optimization requires a balanced approach that maximizes both the breadth of the search and the depth of solution refinement. This duality not only enhances the algorithm's efficiency but also increases its ability to navigate complex optimization landscapes, ultimately leading to more robust and effective solutions [3].

Generally, metaheuristic algorithms represent an enhanced version of heuristic algorithms that merge random algorithms with local search techniques. These algorithms mainly tackle optimization problems by mimicking natural processes and human intelligence. A collection of modern metaheuristic algorithms, categorized into four groups—swarm intelligence, evolutionary, human behavior-based, and physics-based—can be found in Table 1.

In 2005, Tizhoosh introduced an innovative approach known as opposition-based learning (OBL), aimed at enhancing the convergence quality of meta-heuristic algorithms. This method leverages the concept of opposition to optimize the search process, thereby facilitating a more efficient exploration of the solution space. By incorporating strategies that consider not only the current solutions but also their oppositional counterparts, OBL seeks to accelerate the convergence towards optimal solutions. The significance of this method lies in its ability to improve the performance of various optimization algorithms, making them more robust and effective in solving complex problems across different domains [22]. This method is based on replacing the value of optimized solutions with its opposite values to improve the performance of optimization algorithms. In other words, if an optimization solution is denoted by x, its opposite value is denoted by $-x$. This idea is inspired by creating diversity in the optimization process and can result in significant improvements in the performance of optimization algorithms. A group of researchers led by Xiaobing Yu implemented opposition-based learning on the GWO algorithm in 2021 to increase the convergence accuracy of the GWO algorithm for solving multiple and complex functions [23]. In this research, the OBL algorithm has been integrated with GWO, resulting in the algorithm's superiority by preventing it from getting stuck in local optima without increasing computational complexity. However, this technique has faced the problem of early convergence and has not been able to converge in high-dimensional problems. In addition, this technique has a limitation that it can only search an opposite point in the search space. Additionally, In scenarios where the upper and lower bounds of specific functions indicate a negative relationship, the fixed distance between opposing points and the current position can significantly affect the diversit within the population.

This constraint arises because when the limits are negatively correlated, it restricts the potential for variation and exploration among the solutions. As a result, the algorithm may struggle to explore a wider solution space, potentially leading to a stagnation in finding optimal or diverse outcomes. Consequently, the ability to generate a rich and varied population is compromised, which is crucial for enhancing algorithmic performance and ensuring robust solutions. Thus, understanding the implications of these constraints is essential for improving the efficacy of the algorithm in diverse problem-solving contexts. [24].

In 2023, Jai and colleagues introduced an enhanced optimization algorithm known as ICOA, which builds on the principles of the Coati algorithm. This new algorithm leverages the coati's natural ability to communicate using auditory signals, improving its efficiency in locating prey through a method that combines searching and surrounding strategies. This innovative approach enables coatis to hunt and capture their prey with greater success. Moreover, when confronted with potential threats, these animals exhibit an instinctive behavior of fleeing to ensure their safety. This dual capability of effective hunting and instinctual evasion underscores the algorithm's inspiration from coati behavior, making ICOA a noteworthy advancement in optimization techniques. To simulate this behavior, ICOA adds a physical effort strategy to the algorithm and causes the ability of global exploration to be strengthened in the Coati algorithm [25]. Additionally, a theory known as the No Free Lunch (NFL) states that no metaheuristic algorithms can effectively solve all optimization problems [26]. A population-based optimization approach might yield favorable outcomes for one category of problems while underperforming in another category. The persistent challenges associated with complex optimization problems highlight the ongoing necessity for the advancement of novel metaheuristic algorithms.

Table 1. A collection of newly meta-heuristic algorithms.

The source of inspiration	Reference	Year of publication	Abbreviation	Algorithm Name	Category
Search behaviors, caching, and recovery of the nutcracker	[4]	2023	NO	Nutcracker Optimizer	SI
Static and dynamic swarm behaviors of bedbugs	[5]	2023	BMHA	Bedbug Meta-Heuristic Algorithm	
Seed dispersal process of willow trees	[6]	2023	WCO	Willow Catkin Optimization	
Hunting strategy of ospreys	[7]	2023	OOA	Osprey Optimization Algorithm OOA	
Behavior of porpoise for migration, feeding, reproduction and escape	[8]	2024	WA	Walrus optimizer	
Life and intelligent behavior of puma	[9]	2024	PO	Puma Optimizer	
Breeding and reproduction of deer herds	[10]	2024	EHO	Elk Herd Optimizer	
The concept of herd immunity to deal with the corona virus pandemic	[11]	2021	CHIO	Coronavirus Herd Immunity CHIO Optimization	EA
How the Ebola virus spreads	[12]	2022	EOSA	Ebola Optimization Search EOSA Algorithm	
Individuals with Substance Use Disorder	[35]	2025	ISUD	Individuals with Substance Use Disorder Algorithm	
Nuclear explosion of the Chernobyl nuclear reactor	[13]	2023	CDO	Chernobyl Disaster Optimizer	
Principles of physics about stability and different states of particle decomposition	[14]	2023	EVO	Energy Valley Optimizer	PHA
Newton-Raphson method	[15]	2024	NRBO	Newton-Raphson-based optimizer	
Triangular topology method in mathematics	[16]	2024	TTAO	Triangulation Topology Aggregation Optimizer	
Musical chairs game	[18]	2023	MCA	Musical Chairs Algorithm	Human based
The method that people in a group use to influence each other.	[19]	2023	GLA	Group Learning Algorithm	
The way humans cooperate	[20]	2023	MTBO	Mountaineering Team-Based Optimization	
Botox optimization algorithm	[21]	2024	BOA	Botox Optimization Algorithm	

This paper introduces an enhanced variant of the COA algorithm, referred to as EOBCOA, which incorporates principles from Opposition-Based Learning (EOBL). The proposed algorithm modifies the existing update mechanisms to circumvent the pitfalls of local optima and to facilitate a more rapid convergence towards optimal solutions. By integrating the EOBL approach into optimization processes, this method aims to improve the efficacy and robustness of

problem-solving strategies, ultimately contributing to more effective solutions in diverse application domains.

The structure of this paper is organized into several sections to facilitate a comprehensive understanding of the topics presented. In Section 2, we delve into the source of inspiration and the mathematical framework surrounding the opposition-based learning technique. This will lay the groundwork for understanding the theoretical underpinnings of the approach. Moving to Section 3, we introduce the proposed technique known as EOBCOA, providing detailed insights into its mechanisms and advantages. Section 4 is dedicated to the experimental setup, where we analyze the results obtained from various tests and engage in a discussion regarding their implications. Finally, Section 5 concludes the paper by summarizing the findings and offering recommendations for future research avenues, highlighting the potential for further exploration in this intriguing field.

2. Coati optimization algorithm (COA)

In 2023, Dehghani and his research team introduced the Coati Optimization Algorithm, which draws inspiration from the adaptive and intelligent behaviors exhibited by coatis in various survival scenarios, such as evading predators, launching attacks, and hunting for food. This innovative algorithm seeks to mimic the strategic decision-making processes of coatis, highlighting their ability to navigate complex environments and respond effectively to threats. By analyzing these natural behaviors, the Coati Optimization Algorithm aims to enhance problem-solving techniques in computational settings, providing a new approach to optimization challenges [27]. Coatis are a type of mammal characterized by their slender heads, flexible snouts, and distinct black claws. They possess small ears and long tails that are not only used for balance but also for communication with other members of their group. Adult coatis are comparable in size to a large domestic cat, with weights varying between 2 to 8 kilograms. Typically, they reach a height of approximately 30 centimeters. Notably, male coatis exhibit a significant size difference compared to females, often weighing nearly double and possessing large, sharp teeth that aid in their foraging and feeding behaviors. The green iguana is their favorite prey. Often found in trees, iguanas become targets when coatis climb to the treetops to hunt them. Some coatis will leap into the trees, startling the iguanas and causing them to fall to the ground, where other coatis are ready to attack. Despite their hunting prowess, coatis must be vigilant against predators such as dogs, foxes, pit vipers, maned wolves, and anacondas. They are also hunted by large birds of prey including harpy eagles, black eagles, and wedge-tailed eagles. In the following sections, we will explore the clever tactics coatis use when hunting iguanas and how they manage to confront and evade predators. We will then present an optimization algorithm rooted in a mathematical model.

2.1. Initialization of the COA Algorithm

The COA algorithm is a population-based meta-heuristic method that conceptualizes coatis as individual entities within its population framework. Each variable's location within the search space is representative of the corresponding values of the decision variables. The locations of the coatis are initialized randomly at the beginning of the COA algorithm by a specific equation:

$$x_{i,j} = lb_j + r \cdot (ub_j - lb_j), \quad i = 1, 2, \ldots, m \qquad (1)$$

So that x_i is the position of the i-th Coati in the search space and $x_{i,j}$ is the value of the jth decision variable, r is random number between [0,1]. Also N is the number of Coatis, m represents the number of decision variables and lb_j and ub_j are the lower limit and the upper limit of the jth decision variable. Mathematically, the population of the Coatis is defined by the matrix X according to the following equations.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N \times m} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,m} \end{bmatrix} \qquad (2)$$

To evaluate the merit of each Coati, the calculated value of the objective functions of each of them is displayed as an F matrix according to equation 3.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N \times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N \times 1} \qquad (3)$$

So that F_i indicates the degree of merit of each Coatis in this algorithm. In COA algorithm, the highest value of the objective function is regarded as the best candidate. As the algorithm undergoes

its iterative cycles, the candidate solutions are refined and update of the best individual during each iteration. Ultimately, the algorithm's efficiency heavily relies on this mechanism of iteratively identifying and promoting the best-performing candidates.

2.2. Mathematical modeling of the COA Algorithm

The process of adjusting the positions of coatis draws inspiration from two primary phases. These phases encompass the strategic methodologies employed by coatis during their pursuit of iguanas, as well as the adaptive tactics they implement when evading potential predators. This duality reflects the coatis' innate survival strategies, which inform the optimization process, enabling the algorithm to effectively navigate complex solution spaces. By mimicking these behaviors, the algorithm enhances its capability to identify optimal solutions while simultaneously adapting to dynamic challenges, thereby improving overall efficiency and effectiveness in problem-solving scenarios.

Phase 1: The strategy for attacking and hunting iguanas (exploration phase).

The initial stage of updating the coati population in our search space is inspired by how they hunt iguanas. In this strategy, part of the coati group climbs a tree to scare the iguana, while others patiently wait below for it to fall. When the iguana lands, the coatis quickly move in to catch it. This approach encourages significant movement among the coatis across various positions within the search area, showcasing how effective the COA is for performing a comprehensive search in problem-solving situations. Also in this algorithm assume that half of the coatis go up the tree, while the others stay on the ground, ready to spring into action when the iguana drops. This method not only reflects a clever hunting tactic but also emphasizes the dynamic nature of the search process, highlighting the importance of teamwork in achieving success. The mathematical representation of the positions for those climbing coatis is provided in equation (4).

$$x_{i,j}^{p1} = x_{i,j} + r \cdot (\text{Iguana}_j - I \cdot x_{i,j}), \quad i = 1, 2, \ldots, \left\lfloor \frac{N}{2} \right\rfloor, \quad j = 1, 2, \ldots, m \tag{4}$$

Once the iguana hits the ground, it gets moved to a new, random spot within the area being explored. This process is quite interesting because it allows the iguana to experience different environments and situations. This new position serves as its starting point for further exploration, the coatis move along the ground within the search space, and their positions are updated according to equations 5 and 6.

$$\text{Iguana}^G: \quad \text{Iguana}_j^G = lb_j + r \cdot (ub_j - lb_j), \quad j = 1, 2, \ldots, m \tag{5}$$

$$x_{i,j}^{p1} = \begin{cases} x_{i,j} + r \cdot (\text{Iguana}_j - I \cdot x_{i,j}), & \text{if } F(\text{Iguana}^G) < F_i \\ x_{i,j} + r \cdot (x_{i,j} - \text{Iguana}_j^G), & \text{else} \end{cases} \tag{6}$$

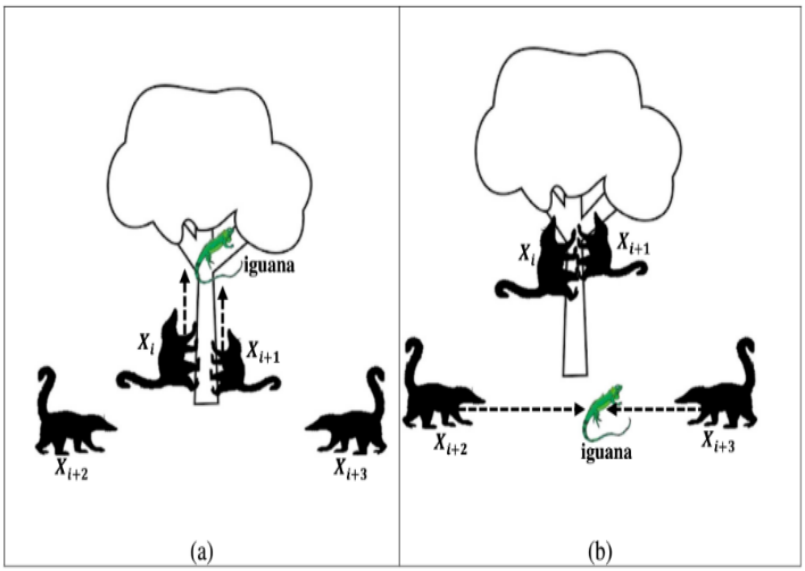

Figure 1. Model of the phase 1 (a) and phase 2 (b) in COA algorithm [27].

$$\text{Iguana}^{G}: \quad \text{Iguana}_{j}^{G} = lb_j + r \cdot (ub_j - lb_j), \quad j = 1, 2, \ldots, m \tag{6}$$

If the new location does not yield an improvement, the Coati will maintain its current position. This indicates that the Coatis are persistently seeking more favorable locations that can assist them in achieving their objectives with greater efficiency. If the new position doesn't offer any advantage, there's no reason for them to change their current place. This approach helps ensure that every move is a step towards better outcomes, making the overall process more efficient and productive. This update condition is modeled using an equation (7).

$$X_i = \begin{cases} X_i^{p1}, & \text{if } F_i^{p1} < F_i \\ X_i, & \text{else} \end{cases} \tag{7}$$

Here, X_i^{p1} represents the newly calculated position for each coati, $X_{i,j}^{p1}$ denotes its j-th dimension, and F_i^{p1} indicates its fitness value. Additionally, r is a random value between 0 and 1 in each iteration of the algorithm. The parameter Iguana represents the position of the iguana in the search space, which actually refers to the best position of the member. $Iguana_i$, represent the next j-th iguana, and I is an integer that is randomly select from the set {1, 2}. $Iguana^G$ is the location of the iguana on the ground, which is randomly generated, $Iguana_j^G$ represents the next j-th, F_{Iguana}^G its evaluation value is the objective function and $\lfloor \cdot \rfloor$ is the floor function.

Phase 2: The process of escaping from predators (exploitation phase)

In the subsequent phase of our study, we delve deeper into the positional dynamics of coatis within their habitats. This exploration is significantly influenced by their innate instincts, particularly when faced with potential threats from predators. In their natural environment, coatis remain perpetually vigilant, and their movement patterns are intricately linked to their survival needs. By employing mathematical modeling, we aim to quantitatively analyze these behaviors, thereby gaining insights into their reactive movements and adaptive strategies when confronted with danger. Phase 1 and phase 2 in COA algorithm are shown in Figure 3.

When a predator approaches, a coati instinctively retreats swiftly from its current location, illustrating its ability to identify and navigate towards a safer area nearby. This adaptive movement strategy highlights the coati's proficiency in local search tactics, which is crucial for its survival. The response mechanisms of coatis to predatory threats are visually represented in Figure 2.

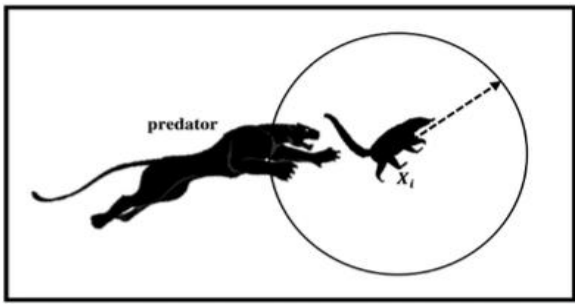

Figure 2. The pattern of the second phase. Coati's escape from the hunter in Algorithm COA [27].

To mimic this behavior, we create a random position close to where each coati is located. This is done using specific formulas that help us determine these nearby points. In essence, we're simulating how coatis might move around their environment by generating new potential spots for them to explore. This approach allows us to better understand their behavior and interactions within their habitat, giving us a clearer picture of their movement patterns and the factors influencing them.

$$lb_j^{\text{local}} = \frac{lb_j}{t}, \quad ub_j^{\text{local}} = \frac{ub_j}{t}, \quad t = 1, 2, \ldots, T \tag{8}$$

$$x_{i,j}^{p2} = x_{i,j} + (1 - 2r) \cdot \left(lb_j^{\text{local}} + r \cdot (ub_j^{\text{local}} - lb_j^{\text{local}}) \right), \quad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, m \tag{9}$$

The updated position is acceptable if it improves the objective function value while satisfying the condition of equation 10.

$$X_i = \begin{cases} X_i^{p2}, & \text{if } F_i^{p2} < F_i \\ X_i, & \text{else} \end{cases} \tag{10}$$

X_i^{p2} is new position calculated for the i-th Coati and $X_{i,j}^{p2}$ is the position of the j-th Coati following that, and F_i^{p2} represents its fitness value in the evaluation of the objective function.

1. Input the information related to the optimization problem.
2. Set the number of iterations (T) and the number of coats (N).
3. Initialize the positions of all coats using the initial population generation rule and evaluate the objective function.
4. For each iteration t=1 to T, perform the following steps:

5. Update the iguana's position based on the best individual in the population.

6. **Phase 1: Hunting and Attacking (Exploration Phase):**
   - For each coat i=1 to ⌊N/2⌋:
     - Calculate a new position for the iii-th coat using the corresponding exploration formula.
     - Update its position based on fitness comparison.
   - For each coat i=⌊N/2⌋+1 to N:
     - Generate a random position for the iguana.
     - Calculate a new position for the i-th coat using a second exploration formula.
     - Update the coat's position based on fitness evaluation.

7. **Phase 2: Escaping from Predators (Exploitation Phase):**
   - Determine local bounds for each variable.
   - For each coat i=1 to N:
     - Generate a new position using the local exploitation strategy.
     - Update the position based on objective value comparison.

8. Save the best solution found in the current iteration.

5. End loop.
6. Output the best solution found by COA for the given problem.

Figure 3. COA algorithm pseudo code.

Also, r is a random value between 0 and 1 in each iteration of the algorithm and t represents the number of iterations of the algorithm. lb_j^{local} and ub_j^{local} are local lower limit and upper limit respectively, and lb_j and ub_j show the lower limit and the upper limit of the j-th decision variable respectively. The various stages of COA execution are illustrated in the flowchart presented in Figure 3.

2.3. Enhanced Opposition-Based Learning Method (EOBL)

In this section, we will first explain the Enhanced Opposition-Based Learning (EOBL), which is inspired by the opposition strategies of OBL and ROBL. The EOBL strategy proposes solutions that are generated in opposition to previous solutions, thereby increasing diversity in the positions of search agents and helping the population escape from local optima with fast convergence. This technique is demonstrated using the following equation.

$$\hat{S}_{i,j} = \begin{cases} M + \dfrac{\text{rand}^2 \cdot S_{i,j}}{2}, & \text{if } \| S_{i,j} \| \leq \| M \| \\ M - \dfrac{\text{rand}^2 \cdot S_{i,j}}{2}, & \text{if } \| S_{i,j} \| > \| M \| \end{cases} \quad (11)$$

Here, χ and y represent the lower and upper bounds of the search space, respectively, while $rand^2$ is a small random number within the range [0, 1] that aids in exploring promising areas of the search space. The formulation expressed in equation 12 plays a crucial role in preventing the algorithm from becoming trapped in local optima, thereby facilitating convergence and mitigating undesirable fluctuations in the solution space. Furthermore, Figure 4 illustrates the positioning of point S and its corresponding antipodal point as determined by the EOBL mechanism within a three-dimensional framework.

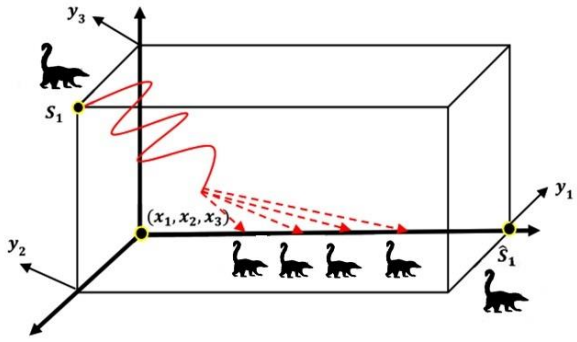

Figure 4. Implementation of EOBL mechanism in 3D space on COA algorithm.

3. Description of the proposed model

The present section delineates the framework of the proposed EOBCOA technique, designed to enhance the efficiency of the COA algorithm. By integrating the COA mechanism with the EOBL technique, this approach effectively utilizes time, thereby improving the capacity to explore the search space. This integration also mitigates several drawbacks, including slow convergence and the tendency to become trapped in local optima. The combination of EOBL with COA occurs through a two-step process. Initially, the population is generated using the EOBL method, followed by the development of new individuals based on existing data. A detailed explanation of these two procedural steps will be elaborated upon below. This structured approach not only streamlines the optimization process but also fosters a more robust exploration of potential solutions, contributing to a more effective algorithm overall.

3.1. Initialization phase

In this step, the initial population is randomly initialized according to the following equation.

$$S_i = \{S_{i1}, S_{i2}, \ldots, S_{ij}, \ldots, S_{iD}\}, \quad i = 1, 2, \ldots, NP; \quad j = 1, 2, \ldots, D \tag{12}$$

So that *N*P represents the population size in the *D* dimension. Through the application of the EOBL technique, In a population of solutions, we take a close look at the value each one holds. This process helps us figure out which solutions are performing the best and identify the optimal values among them. By evaluating their effectiveness, we can understand how each solution stacks up against the others, guiding us toward the best options available. This assessment is crucial for making informed decisions in any problem-solving scenario, allowing us to choose the most effective paths forward. Then the main population S_i and the opposite population $\widehat{S}_i$ merge into one group. Finally, the best ones from the merging group of size NP form the main population.

3.2. Updating phase

At this phase, every revised COA solution undergoes an evaluation process through the fitness function prior to being altered by equations 6 to 12. During this evaluation, the optimal solution along with its corresponding fitness value is recorded for further analysis. To bolster the exploration capabilities of the algorithm, the EOBL mechanism is utilized to produce new coatis at a specific probability rate, denoted as J_r. Some advantages of this method include comprehensive searching in the search space, prevention of premature convergence, and increased diversity of solutions. In this step, a random value between 0 and 1 is generated. If this random value is less than J_r, the EOBL mechanism is activated to create new coatis based on the existing population. Subsequently, the NP number of coatis with the highest fitness is selected from the combined pool of current coatis and the newly generated ones. Essentially, EOBL acts as a mutation factor, enabling the algorithm to balance its exploitation and exploration abilities with a probability of J_r =0.1. The pseudo-code for implementing the EOBL mechanism in the COA algorithm is presented in Figure 5.

Set the number of coatis *N* and the number of iterations *T*
Initialize the position of all coatis by Eq. (1) and compute the fitness of them.
Evaluate *M* by using $M = \frac{x+y}{2}$
For *I* =1 : *NP* for all population
  For *j* =1 : *D*
    If norm($S_{i,j}$) ≤ norm(M)
      $\hat{S}_{i,j} = M + \left(rand^2 \times \frac{S_{i,j}}{2}\right)$
    Else
      $\hat{S}_{i,j} = M - \left(rand^2 \times \frac{S_{i,j}}{2}\right)$
  End.
End.
Calculate $\{S_0, \hat{S}_{i,j}\}$ based on the fitness function.
Select *NP* fittest solution from $\{S_0, \hat{S}_{i,j}\}$ to form a coati's population.
For *t* =1 : *T*
  Update location of iguana based on the location of the best member of the population.
  Run **Phase 1** and **phase 2** of COA ()
  Save the best candidate solution found so far.
 If *rand* < J_r (jumping parameter).
    For *I* =1 : *NP* for all population
      For *j* =1 : *D*
        If norm($S_{i,j}$) ≤ norm(M)
          $\hat{S}_{i,j} = M + \left(rand^2 \times \frac{S_{i,j}}{2}\right)$
        Else
          $\hat{S}_{i,j} = M - \left(rand^2 \times \frac{S_{i,j}}{2}\right)$
      End.
    End.
 End.
 Calculate $\{S_0, \hat{S}_{i,j}\}$ based on the fitness function.
 Select NP fittest solution from $\{S_0, \hat{S}_{i,j}\}$ as the current coatis S_0

Figure 5. Pseudo code of EOBCOA algorithm.

3.3 Complexity Analysis

The time complexity and memory space for this proposed algorithm are computed. The time complexity is related to the number of coati (N_c), the problem dimension (D) and the calculation of the cost function (F) features, consequently, the

time complexity of EOBCOA is computed as follows:

$$O(Maximum_{Iteration} * (N_c * D + N * F))$$

3.4. Advantages of the EOBCOA algorithm

The advantages of the EOBCOA algorithm for solving complex optimization problems are outlined below:

- Utilizing EOBL in the proposed algorithm, demonstrates a high capability to escape local optima. This feature enables the algorithm to perform better in complex and multi-modal problems, allowing it to find the global optimum with greater accuracy.
- By integrating EOBL, EOBCOA achieves an optimal balance between exploration and exploitation phases. This balance ensures that the algorithm can broadly explore the search space while also conducting detailed searches in promising regions.
- With enhanced update rules and the use of EOBL, the EOBCOA algorithm exhibits a higher convergence speed compared to the original COA. This allows the algorithm to reach optimal solutions in less time.
- EOBCOA, through the use of EOBL and improved mechanisms, achieves higher accuracy in solving complex and non-linear optimization problems. It demonstrates a strong ability to find precise solutions, even in the presence of multiple local optima.
- EOBCOA ensures continuous progress toward better solutions by preventing premature stagnation. This feature provides stable and reliable performance, even when tackling challenging problems.

4. Performance evaluation of the proposed algorithm

In this section, we evaluate the performance of the newly proposed EOBCOA method by comparing it with the original COA algorithm as well as several contemporary algorithms, including SWO [28], KOA [29], RSA [30], SHO [31], and NOA [32]. The configurations of the parameters utilized for these algorithms are comprehensively presented in Table 2. The experimental results indicate that the EOBCOA method demonstrates superior performance compared to most existing meta-heuristic algorithms across a variety of benchmark functions. A detailed analysis of the outcomes generated by the EOBCOA method will be conducted in the following sections, providing deeper insights into its effectiveness and applicability in solving optimization problems.

4.1. Benchmark functions and adjustable parameters

Initially, the effectiveness of the EOBCOA algorithm is evaluated in comparison to the original COA algorithm. Following this, its performance is further assessed against the classical benchmark functions established in CEC2005 [33] and CEC2019 [34]. The CEC2005 functions consist of

Table 2. Parameter settings of comparative meta-heuristic algorithms.

Algorithm	Parameter	Value
SHO	-	-
RSA	Alpha, Beta	0.1, 0.005
KOA	TC, M0, Lambda	3, 0.1, 15
SWO	TR, Cr, N_min	0.3, 0.2, 20
NOA	Alpha, Pa2, Prb	0.05, 0.2, 0.2
COA	-	-
EOBCOA	J_f	0.1

There exists a total of seven unimodal functions and six multimodal functions, in addition to ten multimodal functions characterized by fixed dimensions. The unimodal test functions, designated as F1 through F7, each possess a unique global optimum and they evaluate the performance of algorithms during the exploitation phase, where the focus is on refining solutions. Conversely, the multimodal test functions, labeled F8 to F13, feature several local optima. Their primary purpose is to assess the capabilities of algorithms in the exploration phase, where discovering diverse and potentially optimal solutions is critical. This distinction between unimodal and multimodal functions highlights the different challenges that algorithms face depending on the nature of the optimization landscape they are navigating. Compared to other multimodal functions, the remaining fixed-dimension multimodal functions (F14 to F23) exhibit a reduced number of local optima and operate within lower dimensional spaces. These functions are suitable for evaluating both the exploration and exploitation phases, which are essential for conducting global and local searches effectively in metaheuristic algorithms. Furthermore, to rigorously assess the performance and efficacy of the proposed method, ten challenging benchmark functions from the CEC2019 reference suite are employed. These functions vary in dimensions and can be adjusted. The ranges for functions CF01 to CF03 differ, while the range for the other functions is set between [-100, 100]. The CEC2019 reference functions used in this study have significantly greater complexity compared to the CEC2005

benchmark functions. The optimization results of the EOBCOA method are compared with those of other contemporary metaheuristic algorithms, such as SWO, KOA, RSA, SHO, and NOA.

This comparative analysis highlights the advancements and challenges in optimization techniques, particularly in light of the evolving complexity of benchmark functions, which necessitates more sophisticated approaches for effective problem-solving in various applications. To assess each algorithm, 30 search agents were utilized to explore the search space. Additionally, each function was executed 30 times, with a maximum limit of 500 iterations and 15,000 function evaluations (FEs). The average outcomes from these runs were subsequently used for comparison. All tests were conducted on a Windows 10 system with a 1.00 GHz processor, 8.00 GB of RAM, and MATLAB R2022a. The comparisons were based on two metrics: mean and standard deviation. The average refers to the mean value of the top results achieved by an algorithm over several runs. This can be calculated in the following way:

$$\text{avg} = \frac{1}{R}\sum_{i=1}^{R}\text{Best}_i \qquad (13)$$

Where $Best_i$ denotes the best solution obtained in the i-th execution, and R represents the number of times the algorithm is executed. The standard deviation serves as a crucial metric in assessing the reliability of an algorithm's performance across multiple executions. It provides insights into the algorithm's ability to yield consistent and optimal outcomes, thereby allowing researchers and practitioners to gauge the stability and repeatability of its results. The calculation of standard deviation involves a specific mathematical formula, which quantifies the degree of variation or dispersion in a set of data points. By analyzing this statistical measure, one can better understand the algorithm's efficiency and predict its behavior under varying conditions. This understanding is essential for both theoretical explorations and practical applications in algorithm development and evaluation.

$$\text{std} = \sqrt{\frac{1}{R-1}\sum_{i=1}^{R}\left(\text{Best}_i - \text{avg}\right)^2} \qquad (14)$$

4.2. Comparison of EOBCOA algorithm with new meta-heuristic algorithms

In this section, the EOBCOA algorithm is initially compared with the original COA algorithm. According to the results presented in Tables 3 and 4, the enhanced EOBCOA algorithm has demonstrated its superiority over the original COA algorithm in 28 out of 33 functions.

Table 3. The results of COA algorithm and the improved version of EOBCOA on 23 classical test functions (CEC2005).

Function	EOBCOA	COA
F1	Avg	Avg
F2	0	0
F3	0	1.809E-180
F4	0	0
F5	0	8.212E-184
F6	0	0
F7	9.433E-05	1.735E-04
F8	-1.256E+04	-1.256E+04
F9	0	0
F10	8.881E-16	8.881E-16
F11	0	0
F12	1.570E-32	1.570E-32
F13	1.349E-32	1.349E-32
F14	9.980E-01	9.980E-01
F15	4.405E-04	4.120E-04
F16	-1.031E+00	-1.031E+00
F17	3.986E-01	3.988E-01
F18	3.267E+00	3.219E+00
F19	-3.821E+00	-3.801E+00
F20	-2.807E+00	-2.541E+00
F21	-1.015E+00	-1.015E+00
F22	-1.040E+00	-1.040E+00
F23	-1.053E+00	-1.053E+00

Table 4. The results of the implementation of the COA algorithm and the improved version of EOBCOA on 10 CEC2019 test functions.

Function	EOBCOA	COA
F1	1.000E+00	1.000E+00
F2	5.000E+00	5.000E+00
F3	6.417E+00	6.757E+00
F4	8.622E+01	9.064E+01
F5	6.665E+01	8.711E+01
F6	1.030E+01	1.044E+01
F7	1.780E+03	1.872E+03
F8	4.834E+00	4.898E+00
F9	3.613E+00	3.681E+00
F10	2.140E+01	2.145E+01

Table 5. Comparison of optimization results for the CEC benchmark functions 2005.

Functions			EOBCOA	COA	KOA	NOA	RSA	SWO	SHO
Unimodal Functions	F1	Mean	**0.00E+00**	**0.00E+00**	1.49E+04	2.74E-08	**0.00E+00**	1.18E+02	**0.00E+00**
		Std	0.00E+00	0.00E+00	3.65E+03	1.50E-07	0.00E+00	3.20E+02	0.00E+00
		CPU	9.192	3.840	0.095	0.158	1.133	0.082	1.845
	F2	Mean	**0.00E+00**	4.51E-183	1.00E+03	4.99E-02	**0.00E+00**	4.44E+00	**0.00E+00**
		Std	0.00E+00	0.00E+00	2.70E+03	2.73E-01	0.00E+00	8.41E+00	0.00E+00
		CPU	8.654	3.487	0.027	0.024	0.842	0.016	1.817
	F3	Mean	**0.00E+00**	**0.00E+00**	4.56E+04	1.99E+03	**0.00E+00**	8.90E+02	**0.00E+00**
		Std	0.00E+00	0.00E+00	1.19E+04	1.09E+04	0.00E+00	1.67E+03	0.00E+00
		CPU	22.097	13.095	0.031	0.049	1.066	0.031	1.659
	F4	Mean	**0.00E+00**	5.34E-184	5.57E+01	1.26E-05	**0.00E+00**	3.60E+00	**0.00E+00**
		Std	0.00E+00	0.00E+00	7.79E+00	6.59E-05	0.00E+00	4.71E+00	0.00E+00
		CPU	8.215	3.274	0.005	0.027	0.756	0.014	1.621
	F5	Mean	**0.00E+00**	**0.00E+00**	1.61E+07	2.91E+01	1.54E+01	5.20E+04	2.88E+01
		Std	0.00E+00	0.00E+00	6.25E+06	5.40E-01	1.47E+01	1.97E+05	8.99E-02
		CPU	9.912	4.416	0.014	0.024	0.887	0.011	1.501
	F6	Mean	**0.00E+00**	**0.00E+00**	1.43E+04	6.70E+00	6.72E+00	2.03E+02	2.68E+00
		Std	0.00E+00	0.00E+00	3.42E+03	7.42E-01	9.81E-01	5.25E+02	2.35E+00
		CPU	8.154	3.193	0.007	0.024	0.758	0.010	1.196
	F7	Mean	**3.35E-05**	5.22E-05	8.43E+00	2.80E-02	1.39E-04	9.35E-02	9.03E-05
		Std	2.68E-05	5.03E-05	4.15E+00	2.93E-02	1.30E-04	1.27E-01	9.87E-05
		CPU	15.999	8.753	0.009	0.017	0.779	0.007	1.542
Multimodal Functions	F8	Mean	-1.26E+04	**-1.26E+04**	-3.90E+03	-4.62E+03	-5.44E+03	-4.13E+03	-2.53E+03
		Std	1.15E-01	3.43E-02	4.03E+02	7.78E+02	2.33E+02	4.66E+02	4.97E+02
		CPU	10.038	4.544	0.006	0.034	0.845	0.010	0.497
	F9	Mean	**0.00E+00**	**0.00E+00**	2.91E+02	7.79E-08	**0.00E+00**	6.38E+01	**0.00E+00**
		Std	0.00E+00	0.00E+00	2.26E+01	4.27E-07	0.00E+00	6.69E+01	0.00E+00
		CPU	8.805	3.772	0.006	0.019	0.729	0.011	1.463
	F10	Mean	**4.44E-16**	**4.44E-16**	1.90E+01	5.96E-01	**4.44E-16**	3.01E+00	**4.44E-16**
		Std	0.00E+00	0.00E+00	9.55E-01	3.26E+00	0.00E+00	3.40E+00	0.00E+00
		CPU	9.129	3.942	0.002	0.018	0.886	0.006	1.477
	F11	Mean	**0.00E+00**	**0.00E+00**	1.23E+02	1.45E-01	**0.00E+00**	7.88E+00	**0.00E+00**
		Std	0.00E+00	0.00E+00	2.88E+01	6.20E-01	0.00E+00	1.84E+01	0.00E+00
		CPU	10.503	5.003	0.003	0.018	0.790	0.017	1.528
	F12	Mean	**1.57E-32**	**1.57E-32**	9.86E+06	5.24E+04	1.33E+00	1.76E+00	2.11E-04
		Std	5.57E-48	5.57E-48	6.92E+06	2.87E+05	2.93E-01	1.46E+00	2.23E-05
		CPU	29.665	18.420	0.016	0.026	0.955	0.019	1.673
	F13	Mean	**1.35E-32**	**1.35E-32**	5.63E+07	2.82E+00	1.99E-01	1.96E+05	2.96E+00
		Std	5.57E-48	5.57E-48	2.99E+07	2.62E-01	6.94E-01	1.04E+06	2.45E-02
		CPU	29.386	18.605	0.020	0.032	0.935	0.026	1.706
Fixed-Dim Multimodal Functions	F14	Mean	9.98E-01	**9.98E-01**	5.77E+00	5.31E+00	3.75E+00	5.08E+00	1.11E+01
		Std	1.09E-10	8.56E-11	4.25E+00	3.25E+00	2.84E+00	2.82E+00	2.30E+00
		CPU	42.922	30.518	0.029	0.033	0.502	0.030	0.835
	F15	Mean	4.19E-04	4.67E-04	8.05E-03	5.99E-03	1.54E-03	6.80E-03	**3.17E-04**
		Std	1.39E-04	2.18E-04	5.08E-03	5.94E-03	7.09E-04	6.73E-03	4.45E-06
		CPU	4.828	3.256	0.005	0.017	0.195	0.007	0.489
	F16	Mean	-1.03E+00	**-1.03E+00**	-9.96E-01	-1.01E+00	-1.03E+00	-1.02E+00	-9.76E-01
		Std	9.04E-05	5.54E-05	3.24E-02	4.25E-02	5.75E-03	1.61E-02	1.14E-01
		CPU	4.655	3.278	0.007	0.011	0.111	0.010	0.552
	F17	Mean	3.98E-01	**3.98E-01**	4.19E-01	4.33E-01	4.11E-01	4.06E-01	5.71E-01
		Std	1.17E-03	7.27E-04	2.46E-02	4.52E-02	1.72E-02	1.81E-02	7.32E-01
		CPU	3.904	2.754	0.007	0.007	0.145	0.006	0.378
	F18	Mean	3.78E+00	**3.17E+00**	4.54E+00	5.75E+00	6.91E+00	4.57E+00	1.62E+01
		Std	3.17E+00	4.20E-01	2.22E+00	6.60E+00	1.69E+01	4.07E+00	1.42E+01
		CPU	4.151	2.976	0.003	0.012	0.123	0.005	0.390
	F19	Mean	-3.84E+00	-3.81E+00	-3.84E+00	-3.84E+00	-3.81E+00	**-3.85E+00**	-3.76E+00
		Std	4.15E-02	6.68E-02	1.83E-02	2.72E-02	4.59E-02	1.65E-02	1.50E-01
		CPU	5.297	3.537	0.002	0.005	0.157	0.016	0.406
	F20	Mean	-2.74E+00	-2.60E+00	-2.96E+00	-2.92E+00	-2.66E+00	**-3.10E+00**	-2.76E+00
		Std	2.17E-01	3.04E-01	1.19E-01	1.71E-01	4.85E-01	1.15E-01	2.59E-01
		CPU	5.656	3.632	0.019	0.012	0.241	0.007	0.430
	F21	Mean	**-1.02E+01**	-1.02E+01	-2.00E+00	-3.63E+00	-5.06E+00	-3.76E+00	-3.51E+00
		Std	1.56E-03	2.41E-03	1.07E+00	1.11E+00	3.20E-07	2.19E+00	1.25E+00
		CPU	6.171	4.115	0.003	0.008	0.198	0.010	0.359
	F22	Mean	**-1.04E+01**	-1.04E+01	-2.14E+00	-3.73E+00	-5.09E+00	-3.61E+00	-4.22E+00
		Std	1.36E-04	9.07E-04	8.29E-01	1.24E+00	8.30E-07	1.84E+00	1.15E+00
		CPU	6.692	4.531	0.002	0.009	0.178	0.003	0.357
	F23	Mean	-1.05E+01	**-1.05E+01**	-2.61E+00	-3.77E+00	-5.13E+00	-3.68E+00	-4.45E+00
		Std	1.56E-04	5.06E-05	1.62E+00	1.51E+00	1.99E-06	2.41E+00	1.28E+00
		CPU	7.682	5.107	0.004	0.008	0.199	0.008	0.366

In this section, the performance of the EOBCOA algorithm, in addition to being compared with the original COA algorithm, has been tested against five other well-known metaheuristic algorithms, including SHO [28], RSA [29], KOA [30], SWO [31], and NOA [32]. The evaluation results of the

mean and standard deviation are shown in Tables 5 and 6, respectively. In terms of mean values, Table 5 shows that the EOBCOA algorithm performs better than all other algorithms on the IEEE CEC2005 benchmark functions. For example, in terms of mean values for unimodal functions, the proposed EOBCOA has achieved the exact optimal solution for six functions except for F7. For multimodal functions (F8-F13), the exact optimal solution has been achieved for most functions and it has provided superior results compared to other algorithms. In terms of standard deviation, the proposed algorithm has shown better performance for most functions. Table 6 demonstrates that the proposed EOBCOA algorithm performs better than other algorithms in solving the IEEE CEC2019 benchmark functions. For instance, in terms of mean values, Table 6 shows that the proposed technique has superior performance for functions F1 to F3 and F10. Moreover, in terms of standard deviation, it has achieved better results for most functions except F10 compared to other algorithms.

4.3. Analysis of the state of convergence of the **EOBCOA algorithm**

The relationship between the value of the objective function and the number of iterations performed by an optimization algorithm is represented by the convergence curve. These curves offer valuable insights into the pathways taken by search agents as they navigate the optimization landscape. Notably, in the initial phases of the optimization process, these agents often exhibit a tendency to rapidly approach the optimal solution.

Table 6. Comparison of optimization results for the CEC benchmark functions 2019.

Functions			EOBCOA	COA	KOA	NOA	RSA	SWO	SHO
Multimodal Functions	F1	Mean	**1.00E+00**	**1.00E+00**	1.68E+08	1.03E+00	**1.00E+00**	8.83E+06	**1.00E+00**
		Std	0.00E+00	0.00E+00	7.21E+07	1.50E-01	0.00E+00	2.30E+07	0.00E+00
		CPU	15.736	10.630	0.083	0.119	0.644	0.061	1.005
	F2	Mean	**5.00E+00**	5.00E+00	1.24E+04	5.02E+00	**5.00E+00**	1.74E+03	5.00E+00
		Std	1.52E-03	5.86E-03	2.55E+03	9.73E-02	2.11E-02	2.18E+03	7.54E-04
		CPU	10.032	5.894	0.017	0.028	0.595	0.022	1.257
	F3	Mean	**6.38E+00**	6.84E+00	1.14E+01	1.08E+01	8.23E+00	1.01E+01	8.74E+00
		Std	1.05E+00	1.30E+00	5.75E-01	7.11E-01	1.06E+00	8.64E-01	1.02E+00
		CPU	9.810	5.631	0.029	0.031	0.654	0.020	0.570
	F4	Mean	8.93E+01	9.50E+01	8.41E+01	8.66E+01	8.27E+01	**7.30E+01**	1.27E+02
		Std	1.78E+01	1.44E+01	1.43E+01	1.51E+01	1.40E+01	1.31E+01	1.52E+01
		CPU	9.696	6.264	0.007	0.015	0.356	0.014	0.654
	F5	Mean	9.80E+01	8.84E+01	**3.01E+01**	4.24E+01	7.72E+01	3.67E+01	1.41E+02
		Std	3.73E+01	3.44E+01	1.26E+01	2.06E+01	2.40E+01	1.34E+01	3.86E+01
		CPU	9.838	6.318	0.017	0.012	0.438	0.010	0.379
	F6	Mean	1.09E+01	1.05E+01	1.15E+01	1.13E+01	1.04E+01	**9.74E+00**	1.24E+01
		Std	1.06E+00	1.03E+00	1.06E+00	8.19E-01	1.05E+00	1.31E+00	1.37E+00
		CPU	81.456	56.982	0.040	0.062	1.023	0.040	1.046
	F7	Mean	1.81E+03	1.81E+03	2.19E+03	2.18E+03	**1.79E+03**	2.03E+03	2.64E+03
		Std	1.53E+02	2.01E+02	2.13E+02	1.71E+02	1.94E+02	2.09E+02	2.86E+02
		CPU	10.399	6.734	0.004	0.022	0.378	0.005	0.376
	F8	Mean	4.85E+00	**4.80E+00**	5.16E+00	5.15E+00	5.05E+00	5.06E+00	4.96E+00
		Std	2.10E-01	2.16E-01	1.87E-01	2.12E-01	1.45E-01	2.02E-01	2.14E-01
		CPU	9.910	6.437	0.011	0.005	0.380	0.013	0.731
	F9	Mean	3.79E+00	3.65E+00	2.46E+00	3.03E+00	2.92E+00	**2.40E+00**	3.85E+00
		Std	6.59E-01	5.87E-01	4.33E-01	5.93E-01	6.53E-01	5.37E-01	8.37E-01
		CPU	9.217	5.818	0.020	0.013	0.341	0.017	0.446
	F10	Mean	**2.14E+01**	2.14E+01	2.18E+01	2.18E+01	2.14E+01	2.17E+01	2.15E+01
		Std	1.08E-01	1.18E-01	1.59E-01	1.27E-01	9.82E-02	1.33E-01	8.48E-02
		CPU	9.803	6.345	0.009	0.014	0.491	0.019	0.721

The central goal of conducting a convergence analysis is to assess the efficacy of the optimization process while visually depicting the algorithm's performance through these graphical representations. Through the examination of these curves, researchers can obtain a more profound understanding of the underlying dynamics involved in the optimization process. This understanding can ultimately facilitate the design and enhancement of algorithms. Additionally, such analyses can uncover significant details regarding the speed and stability of convergence, which are crucial for evaluating the effectiveness of various algorithms across different situations. In this context, Figures 6 to 10 illustrate the convergence curves related to the CEC2005 and CEC2019 benchmark functions, further enriching the analysis of optimization performance. The graphical representations in Figures 6 and 7 clearly illustrate that the EOBCOA algorithm demonstrates a significantly quicker convergence rate when compared to the traditional COA algorithm. This finding implies that the EOBCOA algorithm is more adept at efficiently arriving at optimal solutions, highlighting its potential benefits across various practical applications. The analysis of the

convergence patterns indicates that the EOBCOA not only expedites the optimization process but also improves overall performance metrics. The observed advancements suggest that integrating an adversarial learning strategy within the COA framework could lead to substantial improvements in performance and effectiveness.

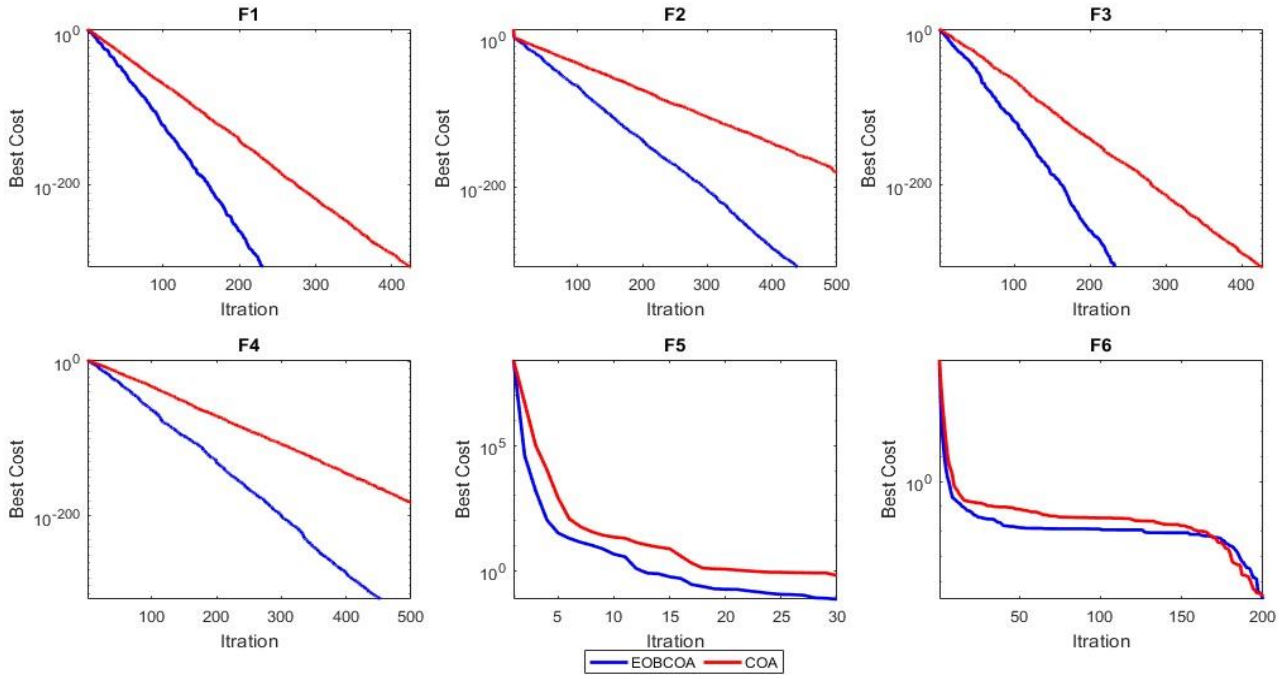

Figure 6. Comparison of convergence speed of COA and EOB COA algorithm for function F1 to F6 from CEC2005 classic benchmark functions.

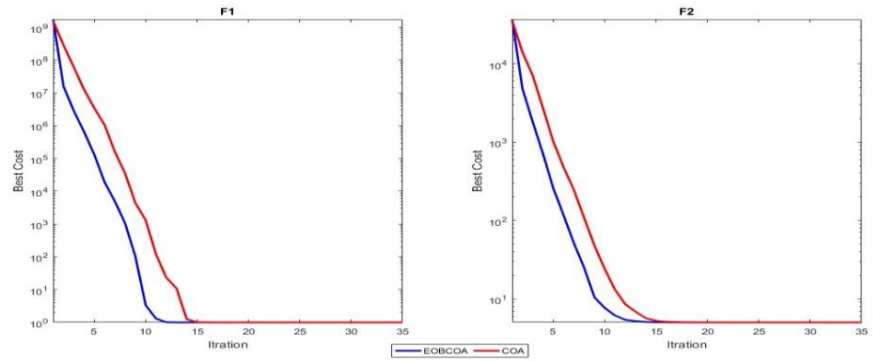

Figure 7. Comparison of convergence speed of COA and EOBCOA algorithm for function F1 and F2 from CEC2019 benchmark functions.

Figures 8 to 10 present a comparative analysis of the proposed algorithm against various other metaheuristic algorithms, with a particular emphasis on their rates of convergence. This comparison highlights the efficiency and effectiveness of the proposed method, showcasing its ability to reach optimal solutions more swiftly than its counterparts. The graphical representations provide clear insights into the performance differences, allowing for a thorough understanding of how the proposed algorithm enhances convergence speed. As seen in Figure 8, the EOBCOA algorithm quickly converges across all unimodal functions in the CEC2005 benchmark set. In the case of multimodal functions, the proposed algorithm demonstrates the best performance with functions F9, F10, and F11. Additionally, Figure 9 shows that for functions F14, F17, F19, and F23, the proposed technique exhibits significant convergence in multimodal functions with fixed dimensions. Furthermore, Figure 10 illustrates that the EOBCOA technique achieves better convergence accuracy than other algorithms for functions F1 to F3, as well as F7, F8,

and F10 in the CEC2019 benchmark set.The analysis of the results indicates that the proposed enhancements in this research have successfully optimized the equilibrium between exploration and exploitation within the EOBCOA algorithm. These improvements in both convergence and search efficiency empower the EOBCOA method to outperform the competing algorithms in the comparative analysis. The findings underscore the significance of refining algorithmic strategies to achieve superior performance metrics, highlighting the potential of the EOBCOA framework in various applications.

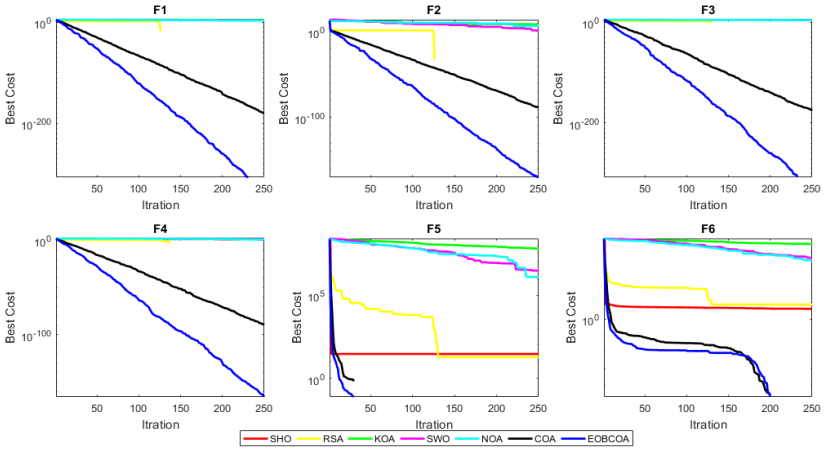

Figure 8: Comparison of convergence speed of algorithms for single-peak functions from CEC2005 benchmark functions

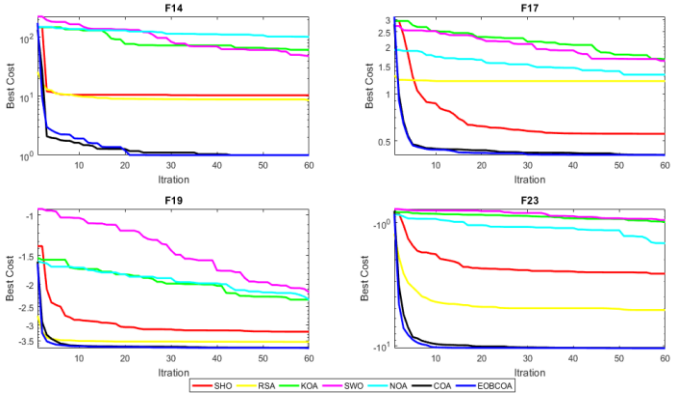

Figure9. Comparison of convergence speed of algorithms for multi-peak benchmark functions from CEC2005 benchmark functions.

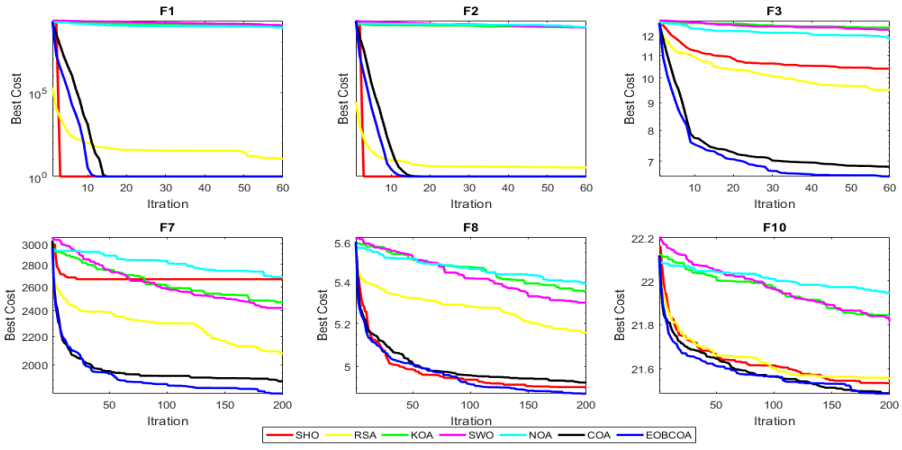

Figure 10. Comparison of convergence speed of algorithms on functions F1 to F3 and F8, F7 and F10 from CEC2019 benchmark functions.

4.4. Analysis of the EOBCOA algorithm

In this section, the convergence behavior of SFO and the performance of the proposed algorithm in terms of exploration and exploitation is discussed. To confirm these items, five parameters are employed as follows:

- The appearance of the function.
- Search history.
- Route of the first coati in its first dimension.
- Fitness history.
- Convergence curve.

The experiments are re-done with 2 variables and 4 coati over 500 iterations. The results are shown in Figure 11. As it can be seen in this figure, the second column shows the history of search agent's positions over the course of iterations. These spots show that the coati explore promising areas of the search space and they also exploit around the global optima very accurately. The approximating of global optima can effectively confirm with these observations. The third column indicates the changes of the position of the first coati in the first dimension. This parameter tracks the position of coati and it assists to observe the moving of candidate solutions. These observations demonstrate the abrupt changes for the implementation of exploration and gradual changes for the implementation of exploitation.

The fourth column shows the average fitness of all coatis. The average fitness of coati shows the decrement of the fluctuations over the course of iteration on all of the test functions. The fifth column of this figure displays the convergence plot. The convergence plot is used to evaluate the speed and quality of convergence in optimization algorithms. It helps us determine whether the algorithm is approaching the optimal solution and whether parameter adjustments or method changes are needed.

4.5. Optimization of large-scale problems using EOBCOA

To assess the scalability of EOBCOA, we tested the algorithm for solving the 100-dimensional and 500-dimensional versions of the unimodal and multimodal test functions. The results are presented in Tables 7 and 8 which clearly demonstrate the algorithm's ability to maintain performance as the problem size increases.

As demonstrated in Tables 7 and 8, the proposed algorithm not only maintains its superiority in achieving the global optimum compared to other state-of-the-art optimization algorithms but also delivers a reasonable and acceptable computational time. The algorithm's efficient design, incorporating advanced mechanisms such as enhanced opposition-based learning and adaptive parameter tuning, ensures robust performance even in large-scale and high-dimensional problems. While scalability challenges are inherent in optimization tasks with increasing complexity, the proposed algorithm effectively balances solution quality and computational efficiency, making it a practical and competitive choice for a wide range of optimization problems.

4.6. Statistical analysis

In optimization, the objective is to address a specific problem by employing various methods and identifying the most effective approach for its resolution. It is essential to establish whether the variations in average performance among these methods are statistically significant. The t-test [30] can be employed for this analysis. This statistical test compares the averages of two groups to assess the significance of their differences. When the p-value obtained from a statistical test is less than the significance level (usually 0.05), it indicates that the difference in means between the two evaluated groups is statistically significant. Such a finding provides a strong foundation for selecting the most effective method for optimization. In the following sections, we present the results of the evaluation of the effectiveness of the EOBCOA technique across 23 benchmark functions from the CEC2005 dataset, as well as 10 benchmark functions from CEC2019. These results are systematically summarized in Tables 5 and 6. The p-values included in these tables clearly illustrate the statistical advantages of the EOBCOA algorithm when compared to the other algorithms evaluated in this study. This statistical evidence underscores the superiority of the EOBCOA method, reinforcing its potential as a preferred choice for optimization tasks in various applications.

4.7. Solving engineering problems using the EOBCOA algorithm

Researchers have demonstrated that no single optimization algorithm consistently outperforms others across all problems with varying structures. While some algorithms may effectively and efficiently address certain problems, they may not yield satisfactory results for others. In this section, we will tackle seven well-known engineering challenges that have been assessed by a variety of optimization algorithms.

Table 7. Comparison of optimization results for the CEC benchmark functions 2005 (100 Dim).

Functions			EOBCOA	COA	KOA	NOA	RSA	SWO	SHO
Unimodal	F1	Mean	**0.00E+00**	**0.00E+00**	7.59E+04	2.50E-03	**0.00E+00**	2.49E+02	**0.00E+00**
Functions		Std	0.00E+00	0.00E+00	1.15E+04	1.37E-02	0.00E+00	5.85E+02	0.00E+00
		CPU	22.425	5.197	0.149	0.286	3.306	0.112	4.742
	F2	Mean	**0.00E+00**	5.76E-181	2.05E+22	7.73E+00	**0.00E+00**	1.41E+01	**0.00E+00**
		Std	0.00E+00	0.00E+00	1.12E+23	4.23E+01	0.00E+00	1.98E+01	0.00E+00
		CPU	19.824	4.400	0.042	0.065	2.772	0.036	4.810
	F3	Mean	**0.00E+00**	**0.00E+00**	5.29E+05	1.79E+04	**0.00E+00**	1.83E+04	**0.00E+00**
		Std	0.00E+00	0.00E+00	1.33E+05	9.80E+04	0.00E+00	3.42E+04	0.00E+00
		CPU	76.430	44.395	0.086	0.120	3.204	0.054	5.099
	F4	Mean	**0.00E+00**	1.43E-183	7.32E+01	3.17E+00	**0.00E+00**	5.19E+00	**0.00E+00**
		Std	0.00E+00	0.00E+00	6.87E+00	1.44E+01	0.00E+00	6.21E+00	0.00E+00
		CPU	19.391	4.259	0.014	0.057	2.714	0.027	4.020
	F5	Mean	**0.00E+00**	**0.00E+00**	1.33E+08	9.89E+01	9.90E+01	3.16E+06	9.89E+01
		Std	0.00E+00	0.00E+00	4.35E+07	1.10E-01	5.74E-03	1.03E+07	1.44E-01
		CPU	23.126	5.921	0.019	0.063	2.951	0.023	4.699
	F6	Mean	**0.00E+00**	**0.00E+00**	7.09E+04	2.88E+01	2.45E+01	8.48E+02	2.17E+01
		Std	0.00E+00	0.00E+00	8.60E+03	2.94E+01	1.16E+00	1.60E+03	3.53E+00
		CPU	19.735	4.124	0.013	0.049	3.029	0.015	2.612
	F7	Mean	**3.98E-05**	6.39E-05	1.84E+02	7.78E+00	1.98E-04	7.95E-01	7.61E-05
		Std	3.52E-05	6.81E-05	6.66E+01	4.24E+01	1.61E-04	2.46E+00	8.20E-05
		CPU	44.245	21.935	0.031	0.066	2.824	0.036	4.618
Multimodal	F8	Mean	-4.19E+04	**-4.19E+04**	-8.22E+03	-1.33E+04	-1.71E+04	-7.68E+03	-4.50E+03
Functions		Std	4.45E-01	1.00E-01	1.81E+03	1.78E+03	1.14E+03	7.71E+02	1.08E+03
		CPU	23.257	6.856	0.006	0.055	2.755	0.009	0.587
	F9	Mean	**0.00E+00**	**0.00E+00**	1.13E+03	1.71E-12	**0.00E+00**	2.48E+02	**0.00E+00**
		Std	0.00E+00	0.00E+00	4.74E+01	9.34E-12	0.00E+00	2.73E+02	0.00E+00
		CPU	21.432	5.348	0.018	0.040	2.577	0.013	4.276
	F10	Mean	**4.44E-16**	**4.44E-16**	1.96E+01	5.00E-04	**4.44E-16**	2.45E+00	1.85E+00
		Std	0.00E+00	0.00E+00	5.29E-01	2.28E-03	0.00E+00	2.53E+00	4.34E+00
		CPU	21.094	5.768	0.017	0.042	2.623	0.016	3.801
	F11	Mean	**0.00E+00**	**0.00E+00**	6.89E+02	1.22E-02	**0.00E+00**	1.32E+01	**0.00E+00**
		Std	0.00E+00	0.00E+00	8.75E+01	6.66E-02	0.00E+00	3.13E+01	0.00E+00
		CPU	22.804	7.046	0.016	0.037	2.579	0.016	4.240
	F12	Mean	**4.71E-33**	**4.71E-33**	1.43E+08	9.72E-01	1.24E+00	3.73E+03	2.95E-04
		Std	1.39E-48	1.39E-48	8.05E+07	2.28E-01	1.53E-01	1.41E+04	2.04E-05
		CPU	75.092	45.136	0.043	0.077	3.140	0.041	4.765
	F13	Mean	**1.35E-32**	**1.35E-32**	3.77E+08	9.52E+00	9.85E+00	5.23E+06	9.96E+00
		Std	5.57E-48	5.57E-48	1.64E+08	1.08E+00	5.62E-02	2.37E+07	2.80E-02
		CPU	76.055	44.871	0.041	0.064	3.093	0.046	4.837

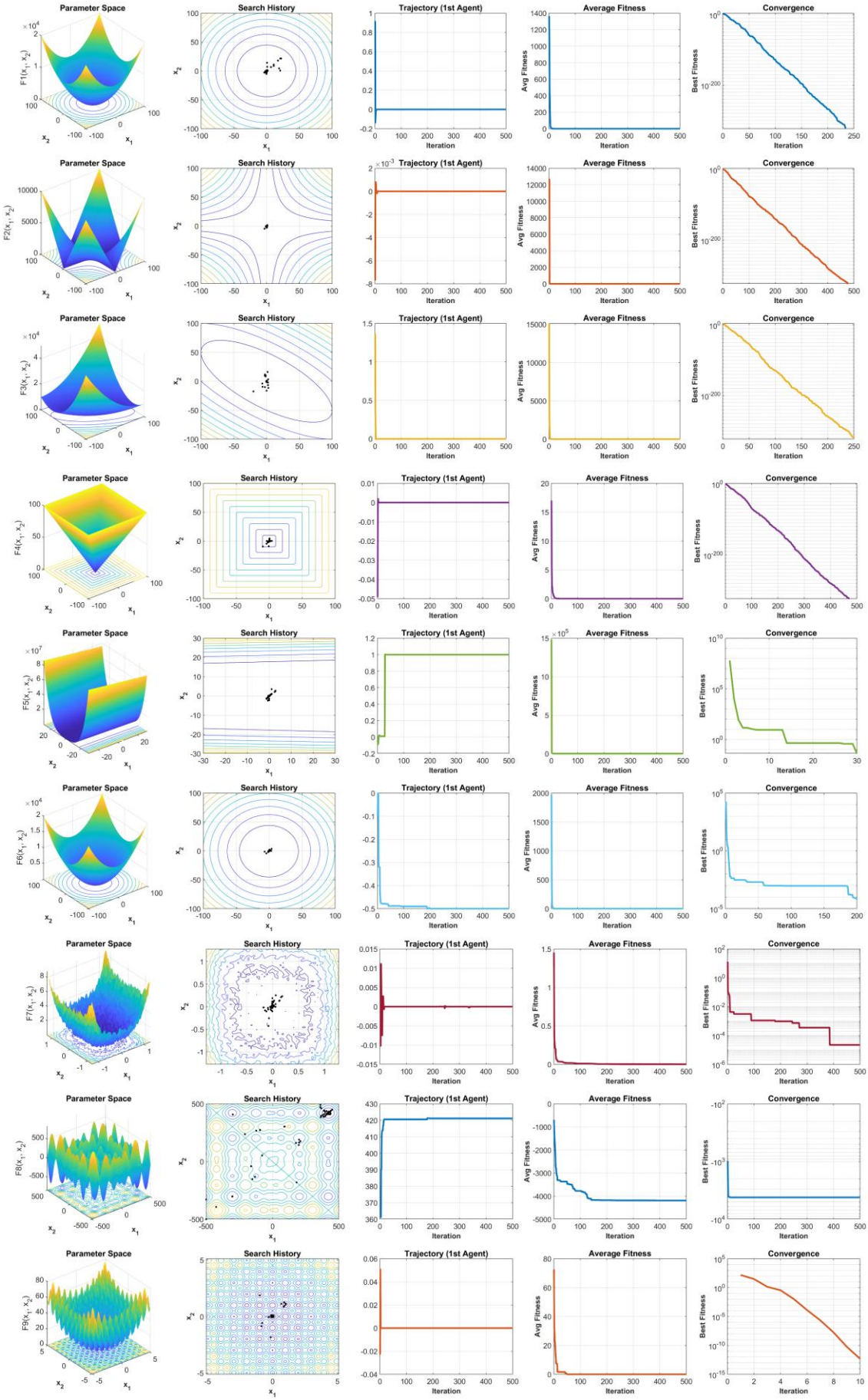

Figure 11. Qualitative results for the studied problems.

Table 8. Comparison of optimization results for the CEC benchmark functions 2005 (500 Dim).

Functions			EOBCOA	COA	KOA	NOA	RSA	SWO	SHO
Unimodal Functions	F1	Mean	**0.00E+00**	**0.00E+00**	4.41E+05	2.92E-32	**0.00E+00**	2.76E+04	**0.00E+00**
		Std	0.00E+00	0.00E+00	9.62E+04	6.53E-32	0.00E+00	5.10E+04	0.00E+00
		CPU	71.403	6.281	0.019	0.122	16.188	0.031	19.003
	F2	Mean	**0.00E+00**	1.46E-181	5.40E+107	2.65E-05	**0.00E+00**	9.27E+00	**0.00E+00**
		Std	0.00E+00	0.00E+00	1.21E+108	5.93E-05	0.00E+00	6.15E+00	0.00E+00
		CPU	73.141	7.150	0.016	0.103	16.319	0.038	19.394
	F3	Mean	**0.00E+00**	**0.00E+00**	1.40E+07	3.52E-14	**0.00E+00**	6.35E+05	**0.00E+00**
		Std	0.00E+00	0.00E+00	2.23E+06	7.87E-14	0.00E+00	8.47E+05	0.00E+00
		CPU	447.931	272.916	0.216	0.300	19.791	0.166	22.941
	F4	Mean	**0.00E+00**	9.23E-188	9.00E+01	3.34E-14	**0.00E+00**	6.84E+00	**0.00E+00**
		Std	0.00E+00	0.00E+00	4.12E+00	7.04E-14	0.00E+00	5.35E+00	0.00E+00
		CPU	71.900	6.538	0.028	0.122	16.381	0.050	19.225
	F5	Mean	**0.00E+00**	**0.00E+00**	8.92E+08	4.99E+02	4.99E+02	2.46E+06	4.99E+02
		Std	0.00E+00	0.00E+00	3.30E+08	3.35E-01	5.34E-03	4.37E+06	1.87E-02
		CPU	74.678	8.416	0.038	0.113	16.263	0.069	19.350
	F6	Mean	**0.00E+00**	**0.00E+00**	4.03E+05	2.81E+03	1.25E+02	1.54E+04	1.24E+02
		Std	0.00E+00	0.00E+00	3.30E+04	6.03E+03	1.86E-04	1.90E+04	7.57E-01
		CPU	73.319	6.306	0.028	0.097	16.678	0.022	12.316
	F7	Mean	**1.71E-05**	7.09E-05	5.48E+03	4.90E-02	1.24E-04	3.57E+01	4.42E-05
		Std	1.44E-05	3.99E-05	1.56E+03	5.86E-02	1.15E-04	6.29E+01	7.13E-05
		CPU	188.147	90.666	0.113	0.138	17.056	0.081	20.416
Multimodal Functions	F8	Mean	**-2.09E+05**	-2.09E+05	-4.38E+04	-5.78E+04	-6.22E+04	-1.93E+04	-1.26E+04
		Std	6.53E-01	6.46E+00	1.45E+04	8.32E+03	3.71E+03	3.53E+03	6.45E+03
		CPU	89.259	18.178	0.091	0.213	16.456	0.088	0.944
	F9	Mean	**0.00E+00**	**0.00E+00**	6.14E+03	**0.00E+00**	**0.00E+00**	9.88E+02	**0.00E+00**
		Std	0.00E+00	0.00E+00	2.57E+02	0.00E+00	0.00E+00	9.23E+02	0.00E+00
		CPU	79.763	14.125	0.094	0.063	16.294	0.044	19.234
	F10	Mean	**4.44E-16**	**4.44E-16**	1.96E+01	1.76E-09	**4.44E-16**	2.28E+00	2.16E+00
		Std	0.00E+00	0.00E+00	3.60E-01	3.93E-09	0.00E+00	1.95E+00	4.23E+00
		CPU	80.841	15.047	0.034	0.113	16.650	0.050	19.666
	F11	Mean	**0.00E+00**	**0.00E+00**	3.60E+03	5.12E-05	**0.00E+00**	1.59E+01	**0.00E+00**
		Std	0.00E+00	0.00E+00	4.75E+02	1.14E-04	0.00E+00	1.42E+01	0.00E+00
		CPU	88.263	18.397	0.044	0.119	17.138	0.041	19.338
	F12	Mean	**9.42E-34**	**9.42E-34**	1.19E+09	9.51E-01	1.20E+00	1.97E+00	1.07E+00
		Std	0.00E+00	0.00E+00	7.78E+08	1.46E-01	1.96E-03	9.27E-01	2.62E-01
		CPU	332.247	190.994	0.113	0.253	18.675	0.113	21.719
	F13	Mean	**1.35E-32**	**1.35E-32**	3.83E+09	4.90E+01	4.99E+01	2.78E+04	5.00E+01
		Std	0.00E+00	0.00E+00	1.69E+09	2.10E+00	4.31E-02	6.21E+04	3.76E-02
		CPU	333.253	189.975	0.138	0.291	18.469	0.116	21.372

Table 9. p-value for 23 CEC 2005 benchmark functions.

Function	EOBCOA versus COA	EOBCOA versus SHO	EOBCOA versus RSA	EOBCOA versus KOA	EOBCOA versus SWO	EOBCOA versus NOA
F1	0	0	0	2.68E-19	2.28E-02	3.23E-01
F2	0	0	0	5.27E-02	1.04E-03	3.25E-01
F3	0	0	0	1.76E-17	2.25E-02	3.25E-01
F4	0	0	0	1.39E-27	2.50E-04	2.81E-01
F5	0	6.68E-72	8.72E-08	1.48E-10	1.24E-01	2.27E-03
F6	0	1.13E-10	7.54E-42	1.73E-19	3.85E-02	4.18E-25
F7	3.77E-02	2.68E-01	7.00E-01	4.33E-11	2.73E-02	1.11E-03
F8	5.25E-01	1.16E-41	7.11E-36	5.21E-42	1.92E-37	2.17E-30
F9	0	0	0	2.61E-31	7.45E-04	3.25E-01
F10	0	0	0	2.40E-69	9.73E-05	1.31E-01
F11	0	0	0	3.90E-18	4.27E-04	3.25E-01
F12	1	1.22E-28	6.69E-21	4.69E-09	2.43E-05	1.08E-19
F13	1	2.80E-58	6.25E-03	1.24E-12	1.73E-01	9.19E-34
F14	2.29E-01	2.44E-16	1.28E-06	2.18E-08	3.55E-07	1.99E-07
F15	3.99E-01	4.53E-05	1.83E-07	2.56E-08	1.17E-04	1.47E-05
F16	4.76E-01	2.45E-02	9.16E-03	1.42E-07	3.48E-05	1.98E-04
F17	7.13E-01	5.96E-02	4.29E-07	2.60E-05	1.67E-02	5.70E-05
F18	7.46E-01	2.44E-04	5.58E-02	9.67E-05	8.19E-01	1.32E-01
F19	1.83E-01	4.32E-05	2.24E-02	5.72E-07	5.80E-01	3.74E-05
F20	3.13E-03	3.77E-01	2.13E-02	8.71E-01	3.79E-06	8.38E-02
F21	6.94E-01	3.11E-25	2.53E-134	8.34E-35	5.13E-17	1.97E-17
F22	5.59E-01	8.39E-24	9.57E-128	1.95E-22	1.28E-17	2.28E-22
F23	3.76E-01	2.24E-22	2.64E-135	7.71E-32	8.88E-16	1.58E-17

Table 10. p-value for 10 CEC 2019 benchmark functions.

Function	EOBCOA versus COA	EOBCOA versus SHO	EOBCOA versus RSA	EOBCOA versus KOA	EOBCOA versus SWO	EOBCOA versus NOA
F1	0	0	0	5.55E-13	9.60E-02	3.25E-01
F2	3.13E-01	1.62E-01	3.13E-01	1.79E-21	2.17E-04	3.26E-01
F3	3.75E-01	4.89E-11	9.53E-08	1.23E-24	2.51E-19	1.66E-23
F4	2.70E-01	1.74E-14	7.62E-01	7.25E-01	2.97E-03	4.90E-04
F5	1.55E-03	5.37E-15	9.42E-04	5.67E-03	3.94E-12	1.94E-01
F6	5.93E-01	3.37E-11	1.21E-07	2.52E-12	1.95E-01	4.53E-07
F7	6.78E-02	7.87E-20	2.31E-03	6.55E-13	8.46E-05	6.08E-15
F8	2.00E-01	2.64E-01	1.29E-03	7.24E-12	3.53E-05	9.13E-12
F9	6.88E-01	8.26E-03	9.58E-01	8.68E-03	2.25E-14	9.60E-01
F10	8.37E-02	6.74E-03	7.53E-03	7.01E-12	4.04E-10	5.77E-20

4.7.1 Speed Reducer

This problem involves designing a speed reducer gearbox that connects the propeller to the airplane engine, allowing both components to rotate at optimal speeds. The main objective of any optimization method applied to the speed reducer problem is to minimize the weight of the gear design. To achieve this, it is necessary to determine the optimal values for seven design variables, which are illustrated in Figure 11. Additionally, the mathematical formulation of this problem is provided in Equation 15.

The comparative results of the performance metrics for the competitive algorithms are presented in Table 11. These results indicate that the EOBCOA algorithm has achieved superior outcomes compared to the other seven algorithms. Figure 12 depicts the convergence behavior and speed of these algorithms in addressing the speed reducer problem, which is a significant optimization challenge in civil engineering, often used to assess various optimization algorithms due to its stringent search space for strength estimation.

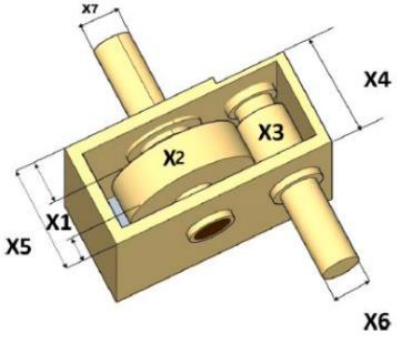

Figure 11. The Speed Reducer Problem.

Minimize.

$$f(\vec{x}) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) \quad (15)$$

Subject to

$$g_1(\vec{x}) = \frac{27}{x_1 x_2^2 x_3^2} - 1 \le 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0$$

$$g_3(\vec{x}) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \le 0$$

$$g_4(\vec{x}) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0$$

$$g_5(\vec{x}) = \frac{\sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6}}{110.0 x_6^3} - 1 \le 0$$

$$g_6(\vec{x}) = \frac{\sqrt{\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6}}{x_2 x_3} - 1 \le 0$$

$$g_7(\vec{x}) = \frac{x_2 x_3}{40} - 1 \le 0$$

$$g_8(\vec{x}) = \frac{5 x_2}{x_1} - 1 \le 0$$

$$g_9(\vec{x}) = \frac{x_1}{12 x_2} - 1 \le 0$$

$$g_{10}(\vec{x}) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(\vec{x}) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0$$

Where

$$2.6 \le x_1 \le 3.6, \quad 0.7 \le x_2 \le 0.8, \quad 17 \le x_3 \le 28,$$
$$7.3 \le x_4 \le 8.3, \quad 7.8 \le x_5 \le 8.3, \quad 2.9 \le x_6 \le 3.9,$$
$$5.0 \le x_7 \le 5.5$$

Table11. Comparative Results for the Speed Reducer Problem.

Algorithm	Best	Mean	Worst	Std.
EOBCOA	3.03E+03	3.099E+03	3.192E+03	5.103E+01
COA	3.00E+03	3.173E+03	3.587E+03	1.520E+02
SFO	3.02E+03	3.128E+03	3.340E+03	9.824E+01
RSA	3.12E+03	3.248E+03	3.348E+03	5.286E+01
KOA	3.04E+03	3.117E+03	3.212E+03	4.142E+01
SWO	3.04E+03	3.145E+03	3.280E+03	6.731E+01
SHO	3.18E+03	3.388E+03	3.853E+03	1.759E+02
NOA	3.02E+03	3.102E+03	3.301E+03	6.355E+01

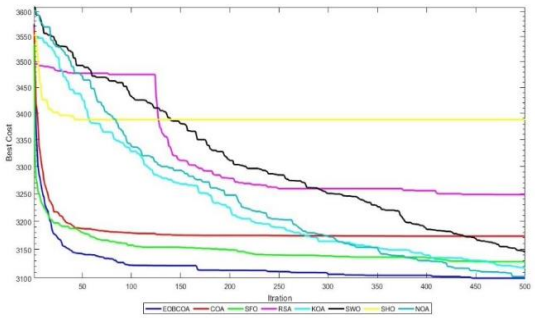

Figure 12. Comparison of Convergence Rate of the EOBCOA Algorithm with Other Competing Algorithms for the Speed Reducer Problem.

4.7.2 Three-bar truss design problem

Figure 13 illustrates a schematic diagram related to a design problem involving a three-member truss. The primary aim of this design challenge is to ascertain the optimal cross-sectional areas of the three truss components. This optimization process is geared towards reducing the total weight of the structure while simultaneously enhancing the stress capacity of each member. The interplay between minimizing weight and maximizing stress is crucial, as it directly influences the structural efficiency and performance under various loading conditions. By addressing these objectives, engineers can create a more effective and resilient truss design that meets the necessary safety and functional requirements. This problem is formulated as shown in relation 16. The results comparing the performance of the competitive algorithms are detailed in Table 12, which demonstrates that the EOBCOA algorithm has yielded more optimal results than the seven other algorithms. Figure 14 illustrates the convergence behavior and speed of the competitive algorithms in solving the design problem of a three-bar truss.

Consider: $\vec{x} = [x_1 \; x_2] = [A_1 \; A_2]$

Minimize: $f(\vec{x}) = \left(2\sqrt{2x_1} + x_2\right) \cdot l$

Subject to: $g_1(\vec{x}) = \frac{\sqrt{2x_1} + x_2}{\sqrt{2x_1^2 + 2x_1 x_2}} \cdot P - \sigma \le 0$

$$g_2(\vec{x}) = \frac{x_2}{\sqrt{2x_1^2 + 2x_1 x_2}} \cdot P - \sigma \le 0 \qquad (16)$$

$$g_3(\vec{x}) = \frac{1}{\sqrt{2x_2} + x_1} \cdot P - \sigma \le 0$$

Variable bounds: $0 \le x_1, x_2 \le 1$

Where: $l = 100$ cm, $P = 2$ kN/cm^2, $\sigma = 2$ kN/cm^2

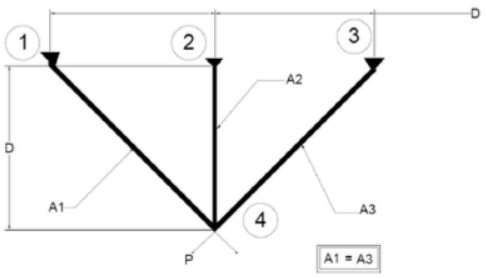

Figure 13. Triangular truss design problem.

Table 12. Comparative results for the three-bar truss problem.

Function	Best	Mean	Worst	Std
EOBCOA	263.8965	263.9086	263.9434	1.2046E-02
COA	263.8969	263.9160	263.9728	1.8029E-02
AO	263.9007	263.9409	263.9945	2.7100E-02
RSA	263.8985	263.9834	264.3777	1.0260E-01
KOA	263.9096	264.1005	264.4647	1.4249E-01
SWO	263.8982	263.9511	264.1495	5.8391E-02
SHO	263.9036	265.0137	269.0898	1.5019E+00
NOA	263.9067	264.0527	264.3465	1.3399E-01

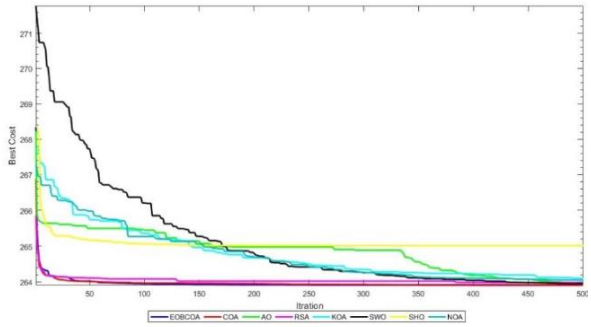

Figure 14. Comparison of the convergence rate of the EOBCOA algorithm with other competitive algorithms for the three-bar truss problem.

4.7.3 Cantilever beam

This study addresses a design challenge that involves five key variables, which must be identified and optimized throughout the design process. As depicted in Figure 15, the optimization task focuses specifically on cantilever beams. To tackle this design issue effectively, the proposed EOBCOA is utilized, which has been formally structured as outlined in Relation 17. This algorithm aims to enhance the efficiency and accuracy of the optimization, ensuring that the design variables are finely tuned to achieve the best performance of the cantilever beams. By applying this method, the research contributes to a deeper understanding of the optimization dynamics in structural engineering contexts.

Consider: $\vec{x} = [x_1\ x_2\ x_3\ x_4\ x_5]$

Minimize: $f(\vec{x}) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5)$

Subject to: $g(\vec{x}) = \frac{60}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0$ (17)

Variable bounds: $0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100$

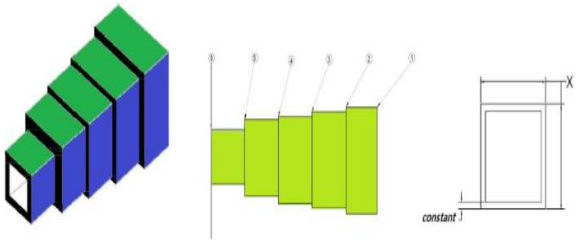

Figure 15. Cantilever beam problem.

Table 13 presents the comparative results between the EOBCOA algorithm and other competitive algorithms for solving the cantilever beam problem. The results indicate that the proposed algorithm has performed satisfactorily in comparison to six other algorithms. Figure 16 depicts the convergence behavior and speed of the competitive algorithms in tackling the cantilever beam problem.

Table 13. Comparative results for the cantilever beam problem.

Function	Best	Mean	Worst	Std
EOBCOA	1.3842	1.4294	1.5600	3.9774E-02
COA	1.3544	1.4564	1.5600	5.8900E-02
SFO	1.3588	1.4329	1.5421	4.6100E-02
KOA	3.1343	4.5861	5.8699	8.3130E-01
SWO	1.6563	2.6986	4.5220	6.6540E-01
SHO	1.3973	3.7584	7.3919	2.1366E+00
NOA	1.5146	3.2556	5.8440	9.9070E-01

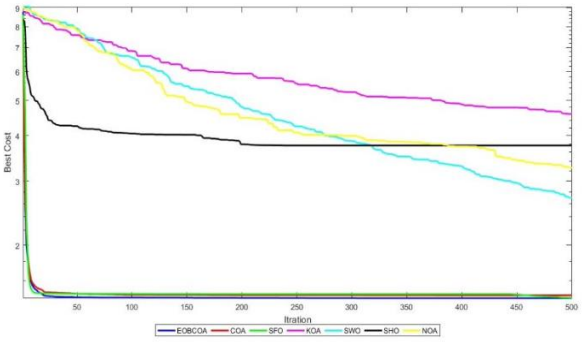

Figure 16. Comparison of the convergence rate of the EOBCOA algorithm with other competing algorithms for the cantilever beam problem.

4.7.4 I-beam design

This problem represents an optimization challenge focused on minimizing the vertical rise of a structure. I-shaped sections, commonly constructed from structural steel, are widely utilized in construction and civil engineering. In this context, several structural parameters come into play, including length, height, and the two thicknesses of the I-shaped sections (flange and web thicknesses), as shown in Figure 17. The mathematical formulation for this problem is provided in Equation 18.

$$\begin{aligned}
&\text{Consider:} \quad x=\left[x_1, x_2, x_3, x_4\right]=\left[b, h, t_w, t_f\right] \\
&\text{Minimize:} \quad f(x)=\frac{5000}{\frac{t_w(h-2t_f)^3}{12}+\frac{bt_f^3}{6}+2bt_f\left(\frac{h-t_f}{2}\right)^2} \\
&\text{Subject to:} \quad g(x)=2bt_w+t_w(h-2t_f)\le 0 \\
&\text{Variable bounds:} \quad 10\le x_1\le 50, \quad 10\le x_2\le 80, \\
&\quad 0.9\le x_3\le 5, \quad 0.9\le x_4\le 5
\end{aligned} \tag{18}$$

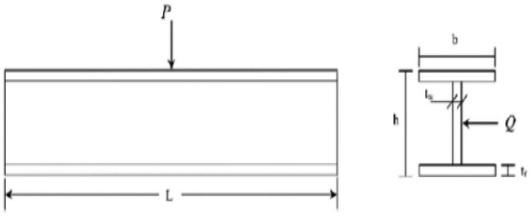

Figure 17. I-beam Design Problem.

The comparative analysis of performance metrics for various competitive algorithms is presented in Table 14. The findings indicate that the EOBCOA algorithm has achieved superior solutions when contrasted with seven other algorithms. Additionally, Figure 18 illustrates the dynamics and convergence rates of these algorithms as they tackle the I-beam design challenge. This evidence highlights the effectiveness of the EOBCOA algorithm in optimizing design outcomes, suggesting its potential as a leading approach in this domain. The convergence behavior depicted in the figure further emphasizes the algorithms' varying efficiencies and adaptability in solving complex engineering problems, particularly in structural design applications.

Table 14. Comparative Results for the I-beam Design Problem.

Algorithm	Best	Mean	Worst	Std
EOBCOA	0.0131	0.0131	0.0135	1.1783E-04
COA	0.0131	0.0300	0.2510	5.4831E-02
AO	0.0131	0.0132	0.0134	1.1193E-04
SFO	0.0131	0.0133	0.0159	8.1189E-04
KOA	0.0131	0.0152	0.0384	4.5251E-03
SWO	0.0140	0.0222	0.1373	2.5158E-02
SHO	0.0135	0.0634	0.4320	8.0108E-01
NOA	0.0131	0.0175	0.0471	6.7536E-03

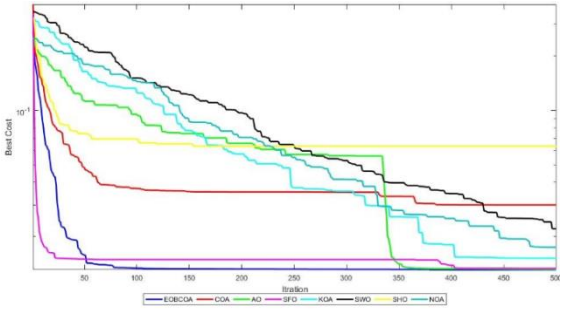

Figure 18. Convergence Curve Comparing the EOBCOA Algorithm with Other Algorithms for the I-beam Design Problem.

4.7.5 Tubular column design

This problem is centered on designing a uniform column with a tubular section to support a compressive load while minimizing costs. The design involves two variables: the average diameter of the column (d = x_1) and the thickness of the tube (t = x_2), as depicted in Figure 19. The mathematical formulation of this problem is presented in Equation 19.

Table 15 presents the comparative results of the proposed algorithm in relation to other competing algorithms, focusing on performance metrics. The results indicate that the EOBCOA algorithm achieved more favorable outcomes compared to seven other algorithms. Figure 20 illustrates the convergence behavior and speed of the competing algorithms in solving the tubular column design problem.

$$\text{Minimize:} \quad f(\vec{x}) = 9.8x_1x_2 + 2x_1$$

$$\text{Subject to:} \quad g_1(\vec{x}) = \frac{P}{\pi x_1 x_2 \sigma_y} - 1 \le 0$$

$$g_2(\vec{x}) = \frac{8PL^2}{\pi^3 E x_1 x_2 (x_1^2 + x_2^2)} - 1 \le 0$$

$$g_3(\vec{x}) = \frac{2.0}{x_2 x_3 x_6^4} - 1 \le 0$$

$$g_4(\vec{x}) = \frac{x_1}{14} - 1 \le 0 \qquad (19)$$

$$g_5(\vec{x}) = \frac{0.2}{x_2} - 1 \le 0$$

$$g_6(\vec{x}) = \frac{x_2}{8} - 1 \le 0$$

$$\text{Variable bounds:} \quad 2 \le x_1 \le 14, \quad 0.2 \le x_2 \le 0.8$$

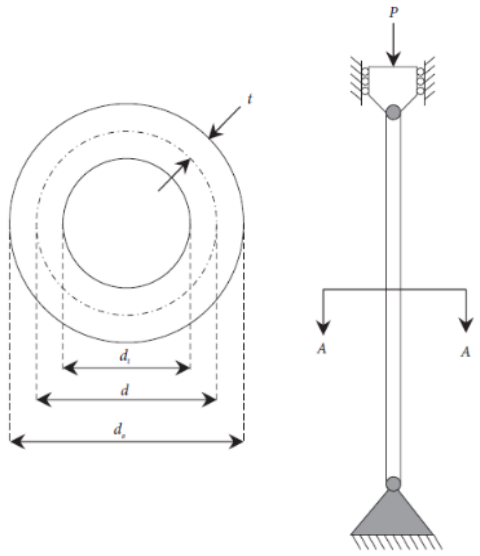

Figure 19. Tubular column design problem.

Table 15. Comparative Results for the Tubular Column Design Problem.

Algorithm	Best	Mean	Worst	Std
EOBCOA	26.4937	26.5307	26.6257	2.8264E-02
COA	26.4892	26.5504	26.7021	4.9400E-02
AO	26.5360	26.7008	26.9041	1.0400E-01
RSA	26.6708	28.0454	30.3859	7.7480E-01
KOA	26.5757	27.6660	29.2436	7.5380E-01
SWO	26.4906	26.8581	29.1481	5.3180E-01
SHO	26.9878	29.2000	32.9677	1.7961E+00
NOA	26.5186	27.1730	28.3251	4.0810E-01

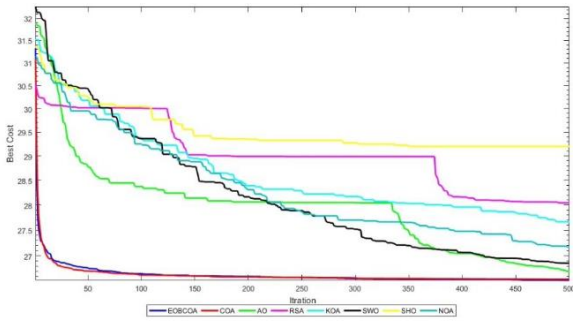

Figure 20. Convergence Comparison Chart of EOBCOA with Other Algorithms for the Tubular Column Design Problem.

4.7.6 Corrugated bulkhead design

The aim of this problem is to reduce the weight of a corrugated wall within a chemical tank. The design variables include width (x_1), depth (x_2), length (x_3), and plate thickness (x_4). This problem is represented in Figure 21 and is mathematically formulated in Equation 20.

$$\text{Minimize:} \quad f(\vec{x}) = \frac{5.885x_4(x_1 + x_3)}{x_1 + \sqrt{\left|x_3^2 - x_2^2\right|}}$$

Subject to:

$$g_1(\vec{x}) = -x_4x_2\left(0.4x_1 + \frac{x_3}{6}\right) + 8.94\left(x_1 + \sqrt{\left|x_3^2 - x_2^2\right|}\right) \le 0$$

$$g_2(\vec{x}) = -x_4x_2^2\left(0.2x_1 + \frac{x_3}{12}\right) + 2.2\left(8.94\left(x_1 + \sqrt{\left|x_3^2 - x_2^2\right|}\right)\right)^{4/3} \le 0 \qquad (20)$$

$$g_3(\vec{x}) = -x_4 + 0.0156x_1 + 0.15 \le 0$$

$$g_4(\vec{x}) = -x_4 + 0.0156x_3 + 0.15 \le 0$$

$$g_5(\vec{x}) = -x_4 + 1.05 \le 0$$

$$g_6(\vec{x}) = -x_3 + x_2 \le 0$$

Variable bounds:

$$0 \le x_1, x_2, x_3 \le 100, \quad 0 \le x_4 \le 5$$

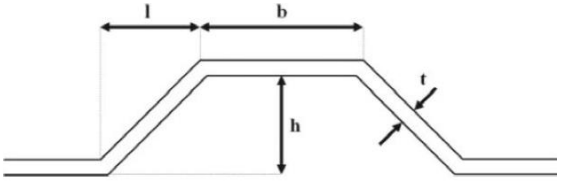

Figure 21. Corrugated wall design problem.

Table 16 provides a comparative analysis of the performance metrics for various competitive algorithms, demonstrating that the EOBCOA algorithm has surpassed six other algorithms in terms of effectiveness. Additionally, Figure 22 depicts the dynamics and convergence rates of these algorithms while tackling the design challenges associated with corrugated walls. The results underscore the superiority of the EOBCOA algorithm, suggesting its potential for more efficient solutions in complex design scenarios. The visual representation in Figure 22 further enhances our understanding of how these algorithms operate and their relative speeds in reaching optimal solutions.

Table 16. Comparative results for the corrugated wall design problem.

Algorithm	Best	Mean	Worst	Std
EOBCOA	6.9744	7.5488	9.0102	4.6736E-01
COA	6.9476	7.9645	9.8387	8.1390E-01
RSA	7.2502	8.0769	9.4206	5.5340E-01
KOA	7.2687	8.0829	9.7388	5.7961E-01
SWO	7.0293	7.6714	8.4802	4.0590E-01
SHO	0.8208	8.5644	14.4607	3.5771E+00
NOA	7.1009	7.8272	8.7478	3.7989E+00

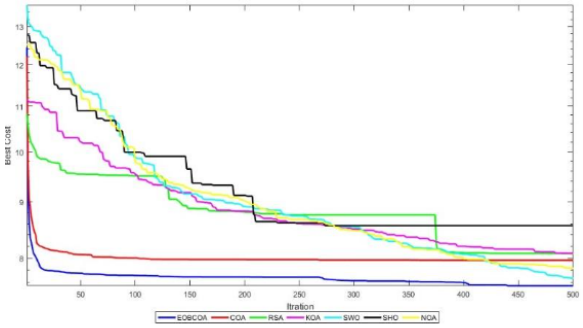

Figure 22. Convergence diagram comparing the EOBCOA algorithm with other algorithms for the corrugated wall design problem.

4.7.7 Reinforced concrete beam design

The problem outlined in Figure 23 presents a simplified optimization challenge in the design of a reinforced concrete beam. This beam is considered to be simply supported, spans 30 feet, and is subjected to an applied load of 2000 pounds. To minimize the overall cost of the structure, it is necessary to determine the reinforcement area (As = x1), beam width (b = x2), and beam depth (h = x3). This problem is formulated as expressed in Relation 21.

$$M_u = 0.9A_s\sigma_y(0.8h)\left(1.0-0.59\frac{A_s\sigma_y}{0.8bh\sigma_c}\right) \geq 1.4M_d + 1.7M_p$$

$$\text{Minimize:} \quad f(\vec{x}) = 2.9x_1 + 0.6x_2x_3$$

$$\text{Subject to:} \quad g_1(x) = \frac{x_2}{x_3} - 4 \leq 0 \tag{21}$$

$$g_2(x) = 180 + 7.375\frac{x_1^2}{x_3} - x_1x_2 \leq 0$$

Variable bounds:

$$x_1 \in \{6, 6.16, 6.32, 6.6, 7, 7.2, 7.8, 7.9, 8, 8.4\}$$

$$x_2 \in \{28, 29, 30, \ldots, 40\}, \quad 5 \leq x_3 \leq 10$$

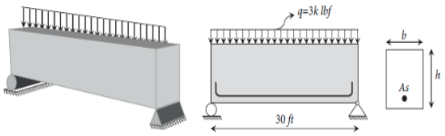

Figure 23. Reinforced concrete beam design problem.

Table 17 presents a comparative analysis of multiple algorithms evaluated against specific performance metrics. The findings indicate that the EOBCOA algorithm exhibits significant efficacy when compared to nine other algorithms in the study. Additionally, Figure 24 provides a visual representation of the dynamics and convergence rates of the various algorithms as they tackle the design challenges associated with reinforced concrete beams. This analysis highlights the strengths and potential advantages of the EOBCOA algorithm, suggesting its effectiveness in optimizing design solutions within this specific engineering context.

Table 17. Comparative results for the reinforced concrete beam design plan.

Algorithm	Best	Mean	Worst	Std
EOBCOA	263.8965	263.9086	263.9434	1.2046×10^{-2}
COA	263.8969	263.916	263.9728	1.8029×10^{-2}
AO	263.9007	263.9409	263.9945	2.7100×10^{-2}
RSA	263.8985	263.9834	264.3777	1.0260×10^{-1}
KOA	263.9096	264.1005	264.4647	1.4249×10^{-1}
SWO	263.8982	263.9511	264.1495	5.8391×10^{-2}
SHO	263.9036	265.0137	269.0898	1.5019×10^{0}
NOA	263.9067	264.0527	264.3465	1.3399×10^{-1}

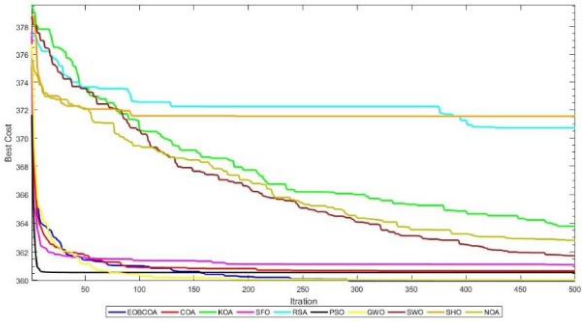

Figure 24. Convergence diagram comparing the EOBCOA algorithm with other algorithms for the reinforced concrete beam problem.

5. Conclusion

In this research, we introduce an enhanced variant of the Coati algorithm, termed EOBCOA, which aims to address global optimization problems. This algorithm draws inspiration from the intriguing strategies employed by coatis during their iguana hunts, as well as their adaptive behaviors when faced with predatory threats. The primary objective of this novel approach is to improve convergence accuracy and to bolster the global search capabilities inherent in the original Coati algorithm.

To evaluate the effectiveness of the EOBCOA method, we conducted tests against 23 benchmark functions derived from IEEE CEC2005 and 10 reference functions from IEEE CEC2019. We compared its performance not only with the original Coati algorithm but also with several established algorithms such as SHO, RSA, KOA, SWO, and NOA. Additionally, we examined the algorithm's performance on seven real-world engineering optimization problems, thereby illustrating its practical applications and benefits. Through these assessments, we aim to illustrate that EOBCOA not only produces superior results but also signifies a noteworthy advancement in the domain of optimization algorithms. The findings from these comparisons suggest that the proposed EOBCOA method achieves better performance on the IEEE CEC2005 standard reference functions and the complex reference functions from IEEE CEC2019. Nevertheless, further investigations into EOBCOA are warranted using additional functions and engineering challenges, which we intend to pursue in our future research endeavors.

References

[1] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13187-13257, 2023.

[2] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering*, vol. 137, p. 106040, 2019.

[3] P. Sharma and S. Raju, "Metaheuristic optimization algorithms: A comprehensive overview and classification of benchmark test functions," *Soft Computing*, vol. 28, no. 4, pp. 3123-3186, 2024.

[4] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash, "Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems," *Knowledge-Based Systems*, vol. 262, Art. no. 110248, 2023.

[5] K. Rezvani, A. Gaffari, and M. R. E. Dishabi, "The Bedbug Meta-heuristic Algorithm to Solve Optimization Problems," *Journal of Bionic Engineering*, pp. 1–21, 2023.

[6] J.-S. Pan, S. Zhang, S. Chu, H. Yang, and B. Yan, "Willow Catkin Optimization Algorithm Applied in the TDOA-FDOA Joint Location Problem," *Entropy*, vol. 25, no. 1, Art. no. 1, 2023.

[7] M. Dehghani and P. Trojovský, "Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems," *Frontiers in Mechanical Engineering*, vol. 8, Art. no. 1126450, 2023.

[8] M. Han et al., "Walrus optimizer: A novel nature-inspired metaheuristic algorithm," *Expert Systems with Applications*, vol. 239, Art. no. 122413, 2024.

[9] B. Abdollahzadeh et al., "Puma optimizer (PO): A novel metaheuristic optimization algorithm and its application in machine learning," *Cluster Computing*, pp. 1–49, 2024.

[10] M. A. Al-Betar, M. A. Awadallah, M. S. Braik, S. Makhadmeh, and I. A. Doush, "Elk herd optimizer: a novel nature-inspired metaheuristic algorithm," *Artificial Intelligence Review*, vol. 57, no. 3, p. 48, 2024.

[11] M. A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah, and I. Abu Doush, "Coronavirus herd immunity optimizer (CHIO)," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5011–5042, 2021.

[12] O. Olaide, E. S. Ezugwu, T. Mohamed, and L. Abualigah, "Ebola Optimization Search Algorithm: A new nature-inspired metaheuristic optimization algorithm," *IEEE Access*, vol. 10, pp. 1–38, 2022.

[13] H. A. Shehadeh, "Chernobyl disaster optimizer (CDO): A novel meta-heuristic method for global optimization," *Neural Computing and Applications*, vol. 35, no. 15, pp. 10733–10749, 2023.

[14] M. Azizi, U. Aickelin, A. Khorshidi, H. Baghalzadeh, and M. Shishehgarkhaneh, "Energy valley optimizer: A novel metaheuristic algorithm for global and engineering optimization," *Scientific Reports*, vol. 13, no. 1, Art. no. 1, 2023.

[15] R. Sowmya, M. Premkumar, and P. Jangir, "Newton-Raphson-based optimizer: A new population-based metaheuristic algorithm for continuous optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 128, Art. no. 107532, 2024.

[16] S. Zhao, T. Zhang, L. Cai, and R. Yang, "Triangulation topology aggregation optimizer: A novel mathematics-based meta-heuristic algorithm for continuous optimization and engineering applications," *Expert Systems with Applications*, vol. 238, Art. no. 121744, 2024.

[17] A. M. Eltamaly and A. H. Rabie, "A Novel Musical Chairs Optimization Algorithm," *Arabian Journal for Science and Engineering*, pp. 1–33, 2023.

[18] C. M. Rahman, "Group learning algorithm: A new metaheuristic algorithm," *Neural Computing and Applications*, pp. 1–16, 2023.

[19] C. M. Rahman, "Group learning algorithm: A new metaheuristic algorithm," *Neural Computing and Applications*, pp. 1–16, 2023.

[20] I. Faridmehr, M. L. Nehdi, I. F. Davoudkhani, and A. Poolad, "Mountaineering Team-Based Optimization: A Novel Human-Based Metaheuristic Algorithm," *Mathematics*, vol. 11, no. 5, Art. no. 5, 2023.

[21] M. Hubálovská, Š. Hubálovský, and P. Trojovský, "Botox Optimization Algorithm: A New Human-Based Metaheuristic Algorithm for Solving Optimization Problems," *Biomimetics*, vol. 9, no. 3, p. 137, 2024.

[22] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in Proc. Int. Conf. on Computational Intelligence for Modelling, Control and Automation and Int. Conf. on Intelligent Agents, Web Technologies and Internet Commerce *(CIMCA-IAWTIC'06),* vol. 1, 2005.

[23] X. Yu, W. Y. Xu, and C. L. Li, "Opposition-based learning grey wolf optimizer for global optimization," *Knowledge-Based Systems*, vol. 226, p. 107139, 2021.

[24] M. Ma et al., "Chaotic random opposition-based learning and Cauchy mutation improved moth-flame optimization algorithm for intelligent route planning of multiple UAVs," *IEEE Access*, vol. 10, pp. 49385–49397, 2022.

[25] H. Jia et al., "Improve coati optimization algorithm for solving constrained engineering optimization problems," *Journal of Computational Design and Engineering*, vol. 10, no. 6, pp. 2223-2250, 2023.

[26] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[27] M. Dehghani, Z. Montazeri, E. Trojovská, and P. Trojovský, "Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 259, p. 110011, 2023.

[28] G. Dhiman and V. Kumar, "Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications," *Advances in Engineering Software*, vol. 114, pp. 48-70, 2017.

[29] L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer," *Expert Systems with Applications*, vol. 191, p. 116158, 2022.

[30] M. Abdel-Basset, R. Mohamed, S. A. A. Azeem, M. Jameel, and M. Abouhawwash, "Kepler optimization algorithm: A new metaheuristic algorithm inspired by Kepler's laws of planetary motion," *Knowledge-Based Systems*, vol. 268, p. 110454, 2023.

[31] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash, "Spider wasp optimizer: A novel meta-heuristic optimization algorithm," *Artificial Intelligence Review*, vol. 56, no. 10, pp. 11675-11738, 2023.

[32] P. N. Suganthan et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *KanGAL report 2005005, 2005.*

[33] J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2019 special session on multimodal multiobjective optimization," *Zhengzhou University, 2019.*

[34] E. Nikolic-aoric, K. Cobanovic, and Z. Lozanov-Crvenkovic, "*Statistical curve ics and experimental data," 2006.*

[35] Zandi, Farzad, Parvaneh Mansouri, and Reza Sheibani. "ISUD (Individuals with Substance Use Disorder): A Novel Metaheuristic Algorithm for Solving Optimization Problems." *Journal of AI and Data Mining*, Vol. 13, no. 2, pp. 207-226, 2025.

[36] Shadravan, Soodeh, H. Naji, and Vahid Khatibi. "A distributed sailfish optimizer based on multi-agent systems for solving non-convex and scalable optimization problems implemented on GPU." *Journal of AI and Data Mining*, Vol. 9, no. 1 pp. 59-71, 2021.

الوریم بهینه سازی کواتی مبتنی بر مخالفت بهبودیافته برای حل بهینه سازی سراسری

سوده شادروان[۱،*] و علی کریمی[۲]

[۱] دانشده مهندسی کامپیوتر، واحد بردسیر، دانشگاه آزاد اسلامی، بردسیر، ایران.

[۲] دانشکده مهندسی فناوری اطلاعات، واحد کرمان، دانشگاه آزاد اسلامی، کرمان، ایران.

ارسال ۲۰۲۴/۱۲/۰۹؛ بازنگری ۲۰۲۵/۰۱/۱۹؛ پذیرش ۲۰۲۵/۰۴/۳۰

چکیده:

الگوریتم بهینه‌سازی کوآتی (COA) یکی از الگوریتم‌های فراابتکاری نوین است که اخیرا ابداع شده و از روش هوشمندانه کواتی‌ها هنگام حمله به ایگوانا و رفتار آن‌ها در مواجهه و فرار از شکارچیان الهام گرفته است. این الگوریتم عملکرد مطلوبی را نسبت به الگوریتم‌های فرا ابتکاری نشان داده است. با این حال، مانند سایر الگوریتم‌های بهینه‌سازی محدودیتهایی در برقراری توازن بین فازهای کاوش و بهره برداری دارد و امکان دارد در هنگام حل مسائل بهینه‌سازی پیچیده در بهینه محلی گرفتار آید. برای رفع این محدودیت‌ها، یک تکنیک نوآورانه به نام "یادگیری مبتنی بر مخالفت بهبود یافته" (EOBL) پیشنهاد شده و با الگوریتم COA ادغام گردیده است. تکنیک EOBL الهام گرفته شده از تکنیک‌های یادگیری مبتنی بر مخالفت (OBL) و یادگیری تصادفی مبتنی بر مخالفت (ROBL) است و می‌تواند به طور کارآمد بر میزان توازن بین فازهای کاوش و بهره‌برداری تاثیرگذار باشد. الگوریتم بهینه‌سازی کوآتی بهبود یافته (EOBCOA)، یک الگوریتم فرا ابتکاری نوین است که برای افزایش کارایی الگوریتم COA پیشنهاد شده است. برای ارزیابی کارآیی روش EOBCOA پیشنهادی، این روش بر روی توابع محک استاندارد IEEE CEC2005 ، IEEE CEC2019 و هفت مسئله مهندسی آزمایش شده و نتایج بیانگر آن است که روش EOBCOA پیشنهادی نسبت به سایر الگوریتم‌های پیشرفته در یافتن بهینه‌سازی سراسری عملکرد بهتری از خود نشان داده است.

کلمات کلیدی: بهینه‌ساز کواتی (COA)، الگوریتم‌های فرا ابتکاری، یادگیری مبتنی بر مخالفت، یادگیری مبتنی بر مخالفت بهبود یافته.