Research Article

Hussam N. Fakhouri, Ahmad Sami Al-Shamayleh*, Abedelraouf Istiwi, Sharif Naser Makhadmeh, Faten Hamad, Sandi N. Fakhouri, and Zaid Abdi Alkareem Alyasseri

Hybrid white shark optimizer with differential evolution for training multi-layer perceptron neural network

https://doi.org/10.1515/jisys-2024-0362
received July 26, 2024; accepted May 26, 2025

Abstract: This study presents a novel hybrid optimization algorithm combining the white shark optimizer (WSO) with differential evolution (DE), referred to as WSODE, for training multi-layer perceptron (MLP) neural networks and solving systems design problems. The structure of WSO, while effective in exploitation due to its wavy motion-based search, suffers from limited exploration capability. This limitation arises from WSO's reliance on local search behaviors, where it tends to focus on a narrow region of the search space, reducing the diversity of solutions and increasing the likelihood of premature convergence. To address this, DE is integrated with WSO (WSODE) to enhance exploration by introducing mutation and crossover operations, which increase diversity and enable the algorithm to escape local optima. The performance of WSODE is evaluated on the CEC2022, CEC2021, and CEC2017 benchmark functions and compared against several state-of-the-art optimizers. The results demonstrate that WSODE consistently achieves superior or competitive performance, with faster convergence rates and higher solution quality across diverse benchmarks. Specifically, on the CEC2022 suite, WSODE ranked first or second across multiple functions, including high-dimensional, multi-modal, and deceptive landscapes and significantly outperforming other algorithms like whale optimization algorithm and spotted hyena optimizer. On the CEC2021 suite, WSODE ranked first in several complex functions, such as C6 and C10, with mean values of 3.32×10^{-1} and 4.90×10^{1}, respectively, showcasing its robustness in handling deceptive and rugged landscapes. Additionally, WSODE has been applied to the training of multi-layer perceptrons using various datasets with different levels of complexity and attribute counts. The results indicate that WSODE achieves lower mean squared error and higher classification accuracy compared to existing methods across most datasets, reinforcing its effectiveness and reliability in both benchmark and practical applications.

Keywords: optimization, multi-layer perceptron, machine learning, systems design problem, neural network

*** Corresponding author: Ahmad Sami Al-Shamayleh,** Department of Data Science and Artificial Intelligence, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, 19328, Jordan, e-mail: a.alshamayleh@ammanu.edu.jo
Hussam N. Fakhouri: Data Science and Artificial Intelligence Department, University of Petra, Amman 11196, Jordan, e-mail: hussam.fakhouri@uop.edu.jo
Abedelraouf Istiwi: School of Information Technology, University of Petra, Amman 11196, Jordan, e-mail: aishtaiwi@uop.edu.jo
Sharif Naser Makhadmeh: Department of Computer Science, University of Jordan, Amman 11962, Jordan, e-mail: s_makhadmeh@ju.edu.jo
Faten Hamad: Information Studies Department, Sultan Qaboos University, Muscat 123, Oman; Information Sciences and Educational Technology Department, School of Educational Sciences, The University of Jordan, Amman 11196, Jordan, e-mail: f.hamad@ju.edu.jo
Sandi N. Fakhouri: Department of Computer Science, University of Jordan, Amman 11962, Jordan, e-mail: s.fakhouri@ju.edu.jo
Zaid Abdi Alkareem Alyasseri: Information Technology Research and Development Center (ITRDC), University of Kufa, Najaf, 54001, Iraq; College of Engineering, University of Warith Al-Anbiyaa, Holy Karbala, 56001, Iraq, e-mail: zaid.alyasseri@uokufa.edu.iq

1 Introduction

Metaheuristics are advanced, high-level strategies designed to efficiently explore solution spaces in search of optimal or near-optimal solutions for complex optimization problems [1]. They are inherently problem-independent, which permits their application across a vast range of domains, from engineering design to machine learning and logistics [2]. Despite their robust capabilities, traditional metaheuristic algorithms face notable challenges that limit their effectiveness, particularly in high-dimensional, multi-modal landscapes. One such limitation is the tendency of many metaheuristics to converge prematurely, often becoming trapped in local optima, especially in cases where complex landscapes or multiple optima are present [3]. This weakness arises from the balance these algorithms attempt to strike between exploration (searching new areas of the solution space) and exploitation (refining existing solutions); traditional metaheuristics frequently struggle to maintain this balance, leading to suboptimal performance in scenarios with rugged search landscapes [4].

The high computational cost associated with some metaheuristics presents an additional challenge, particularly when dealing with large-scale, dynamic, or time-sensitive applications [5]. Many traditional algorithms suffer from a lack of efficiency when scaling to large problems, resulting in extended computation times and high resource consumption [6]. This inefficiency limits their practicality in real-world applications requiring quick decision-making or real-time adaptability. Furthermore, some metaheuristics lack the flexibility needed to address multiobjective optimization, where multiple conflicting objectives must be optimized simultaneously [7]. While multiobjective optimization is crucial in fields such as environmental management, financial portfolio optimization, and engineering design, traditional metaheuristics often fail to effectively balance these objectives and produce a comprehensive set of Pareto-optimal solutions [8].

These limitations underscore the need for advanced hybrid approaches that can enhance exploration–exploitation capabilities, reduce computational costs, and better address multiobjective optimization challenges [9]. Hybrid metaheuristics, which combine the strengths of multiple algorithms, have emerged as a promising solution to these challenges. By integrating complementary strategies, hybrid algorithms aim to leverage the exploration power of one approach with the exploitation efficiency of another, thus mitigating the weaknesses inherent in traditional algorithms [10]. This study contributes to this evolution by proposing a hybrid differential evolution-white shark optimizer (WSODE), a novel algorithm designed to address the weaknesses of traditional metaheuristics and extend their applicability across high-dimensional and multiobjective optimization problems.

In addition to traditional optimization applications, metaheuristics have found significant roles in complex, domain-specific tasks. In engineering, they are frequently employed for design optimization, system modeling, and process optimization, addressing challenges from structural design to energy management and manufacturing [11]. In computer science, metaheuristics support tasks such as feature selection, network design, and software testing, showcasing their versatility and domain adaptability [5]. The evolution of these algorithms, driven by continuous advancements in hybridization and adaptability, emphasizes their role in modern optimization landscapes, marking them as indispensable tools in solving increasingly complex real-world problems [12].

In the field of machine learning, optimization algorithms are fundamental to the training and tuning of models, particularly neural networks [13]. Machine learning has revolutionized numerous domains by enabling systems to learn patterns from data and make intelligent decisions. At the heart of many machine learning models are neural networks, which are inspired by the structure and function of the human brain. These networks, especially multi-layer perceptrons (MLPs), are widely used due to their ability to approximate complex functions and handle various types of data [14].

MLPs are a type of feed-forward neural network consisting of an input layer, one or more hidden layers, and an output layer [15]. Each neuron in a layer is connected to every neuron in the subsequent layer, with each connection having an associated weight. The training process of MLPs involves adjusting these weights to minimize the difference between the predicted output and the actual target values. This process requires efficient optimization techniques to navigate the high-dimensional and often nonconvex search space, making the choice of optimization algorithm critical for the network's performance [16].

Despite their potential, training neural networks effectively remains a challenging task. The optimization landscape of neural networks is complex, characterized by numerous local minima and saddle points.

Therefore, the optimizer's ability to explore the search space thoroughly and exploit promising regions is crucial. DE is known for its robustness in global optimization, utilizing population-based search strategies and mutation operations to explore the search space. On the other hand, the WSO mimics the hunting behavior of white sharks, offering adaptive strategies for exploration and exploitation.

This research introduces a novel hybrid optimization algorithm, the WSODE, which strategically integrates the strengths of WSO and DE. The WSODE algorithm leverages WSO's adaptive hunting strategy for extensive exploration and DE's dynamic parameter adjustment for robust exploitation, ensuring a balanced and effective search process. This hybrid approach is designed to enhance the algorithm's capability in handling high-dimensional and complex optimization problems, which are common in real-world applications. Additionally, this research explores the application of WSODE in training MLPs. The adaptive balance between exploration and exploitation in WSODE is particularly beneficial for optimizing the weights and biases of MLPs, leading to improved accuracy and generalization capabilities. The ability to effectively train MLPs using WSODE can significantly enhance the performance of machine learning models in various applications, from image recognition to natural language processing.

Furthermore, the practical utility of WSODE is validated through its application in various system design problems, including robot gripper optimization, welded beam design, pressure vessel design, spring design, and speed reducer design. These applications highlight the versatility and robustness of WSODE in finding high-quality solutions for complex engineering problems.

The use of optimization algorithms for training MLP neural networks has gained significant attention, with thousands of algorithms available for optimizing MLPs. Despite the vast number of options, the WSO offers several unique characteristics that justify its selection in this study. WSO is inspired by the hunting behavior of white sharks, where the wavy motion of sharks provides an efficient local search mechanism. This characteristic is beneficial in balancing the exploitation process during optimization, making WSO particularly suitable for problems where refining solutions within promising regions of the search space is critical.

However, while WSO is good at exploitation, its structure lacks effective exploration mechanisms. This limitation can lead to premature convergence in highly complex and multimodal problems, such as training MLPs, where finding the global optimum is crucial. By combining WSO with DE, which excels in exploration through mutation and crossover, the resulting hybrid algorithm, WSODE, is designed to overcome WSO's exploration limitations while leveraging its exploitation strengths.

The primary motivation for using WSO lies in its ability to fine-tune solutions efficiently, especially in scenarios where local search is paramount. Moreover, WSO's simplicity and computational efficiency make it an attractive choice compared to more computationally expensive algorithms. The hybridization with DE further enhances WSO's performance, providing a more balanced approach that can compete with other state-of-the-art algorithms for optimizing MLPs.

The primary contributions of this article are as follows:

- Development of a novel hybrid optimization algorithm: We introduce a novel WSODE, which addresses the structural limitations of the original WSO. The key innovation lies in the hybridization of WSO with DE, where WSO's exploitation capabilities are augmented with DE's strong exploration mechanisms. This combination enables the proposed algorithm to maintain a more balanced search, overcoming the tendency of WSO to focus too narrowly on local regions of the search space.
- Enhanced exploration and mitigation of premature convergence: The structural limitation of WSO, which results in reduced exploration and a higher likelihood of premature convergence, is effectively mitigated through the integration of DE. The hybrid algorithm significantly improves exploration, ensuring that the search space is more thoroughly explored, thus avoiding local optima and ensuring convergence to high-quality solutions.
- Effective training of MLPs: WSODE is specifically tailored for optimizing the training process of MLPs. The algorithm's balanced exploration and exploitation phases enable the identification of optimal weights and biases, resulting in superior training performance. This leads to higher classification accuracy and lower mean squared error (MSE), showcasing the hybrid algorithm's effectiveness in neural network training tasks.
- Demonstrated robust performance across benchmark datasets: The performance of WSODE has been rigorously evaluated across multiple datasets, including wine, abalone, hepatitis, breast cancer, housing, and

banknote authentication. In each case, WSODE achieves superior results in terms of classification accuracy and MSE compared to other optimization methods, demonstrating its robustness and versatility in a variety of machine learning applications.
- Applications beyond machine learning: While the primary application of WSODE in this article is the optimization of MLPs, the algorithm's effectiveness extends to broader engineering optimization problems. WSODE's enhanced exploration and exploitation capabilities are demonstrated in system design optimization, including optimizing the design of a robot gripper. This demonstrates WSODE's potential in solving high-dimensional, complex optimization problems beyond the domain of machine learning.

This article is organized into several key sections that comprehensively cover the proposed WSODE algorithm and its applications. Following the introduction, Section 2 discusses the literature review, followed by detailed overviews of the DE and WSO algorithms. An overview of feed-forward neural networks and MLPs sets the stage for the hybrid algorithm. Section 3 presents the WSODE's mathematical model, pseudocode, exploration and exploitation features, and its application in single objective optimization problems. Section 4 introduces the CEC2022, CEC2021, and CEC2017 benchmarks. Section 5 discusses parameter settings and results across these benchmarks, including convergence curves, search history, and heatmap analysis. Section 6 details the experimental setup and results for various datasets, such as wine, abalone, hepatitis, breast cancer, housing, and banknote authentication. Finally, Section 7 demonstrates the effectiveness of the application of WSODE in system design problems in optimizing robot gripper design, welded beam design, pressure vessel design, spring design, and speed reducer design. A summary of findings and implications are given in Section 8.

2 Literature review

Metaheuristic algorithms have been widely studied and developed over the years. They can be broadly classified into four main categories: Evolutionary algorithms, swarm intelligence, local search algorithms, and other nature-inspired algorithms.

First, in the category of evolutionary algorithms, the memetic algorithm was introduced by Moscato in 1989 [17], laying the foundation for hybrid algorithms that combine global and local search strategies. Evolutionary programming was introduced by Fogel et al. in 1966 [18], and the evolution strategy was developed by Rechenberg in 1973 [19]. Genetic algorithms, proposed by Holland in 1975 [20], are highly influential and widely used. Furthermore, the co-evolving algorithm, created by Hillis in 1990 [21], and the cultural algorithm, developed by Reynolds in 1994 [22], have expanded the scope of evolutionary computation. Genetic programming, introduced by Koza in 1994 [23], and the estimation of distribution algorithm by Mühlenbein and PaaB in 1996 [24], have also made significant contributions. DE, proposed by Storn and Price in 1997 [25], is another widely cited algorithm. Additionally, grammatical evolution, developed by Ryan et al. in 1998 [26], and the gene expression algorithm, introduced by Ferreira in 2001 [27], have proven effective in various applications. The quantum evolutionary algorithm by Han and Kim in 2002 [28], the imperialist competitive algorithm proposed by Gargari and Lucas in 2007 [29], and the differential search algorithm developed by Civicioglu in 2011 [30] further showcase the diversity and innovation within this category. Moreover, the backtracking optimization algorithm, also by Civicioglu in 2013 [31], the stochastic fractal search introduced by Salimi in 2014 [32], and the synergistic fibroblast optimization developed by Dhivyaparbha et al. in 2018 [33], illustrate the continuous evolution and adaptation of these algorithms to new challenges.

Moving on to swarm intelligence, this category includes several influential algorithms. Ant colony optimization, developed by Dorigo in 1992 [34], simulates the foraging behavior of ants and has been widely applied in combinatorial optimization problems. Furthermore, particle swarm optimization (PSO), proposed by Eberhart and Kennedy in 1995 [35], mimics the social behavior of birds and fish. The binary PSO variant was later introduced by Kennedy and Eberhart in 1997 [36], extending the algorithm to discrete spaces. Moreover, numerous bee-inspired algorithms have been developed, including the artificial bee colony algorithm by

Karaboga and Basturk in 2007 [37] and the virtual bee algorithm by Yang in 2005 [38], demonstrating the versatility and effectiveness of swarm-based optimization techniques.

Moreover, the category of local search algorithms includes several notable contributions. The self-organizing migrating algorithm, proposed by Zelinka in 2000 [39], and the shuffled frog leaping algorithm, developed by Eusuff and Lansey in 2003 [40], combine local search heuristics with global search strategies to enhance solution quality. Additionally, the termite swarm algorithm, introduced by Roth and Wicker in 2006 [41], employs termite behavior for optimization tasks, showcasing the potential of local search mechanisms inspired by nature.

Finally, other nature-inspired algorithms encompass a diverse range of approaches. The artificial fish swarm algorithm, developed by Li et al. in 2002 [42], mimics fish schooling behavior to perform search and optimization. The bat algorithm, introduced by Yang in 2010 [43], is inspired by the echolocation behavior of bats, while the cuckoo search, developed by Yang and Deb in 2009 [44], utilizes the brood parasitism strategy of cuckoos. Moreover, the biogeography-based optimization proposed by Simon in 2008 [45], and the invasive weed optimization by Mehrabian and Lucas in 2006 [46] illustrate the innovative application of ecological and biological principles in solving complex optimization problems.

Recent research highlights the effectiveness of hybrid metaheuristic algorithms for complex optimization challenges across various fields. Chandrashekar et al. introduced a hybrid weighted ant colony optimization algorithm to optimize task scheduling in cloud computing, outperforming traditional methods in efficiency and cost-effectiveness. In dynamic system identification, an augmented sine cosine algorithm-game theoretic approach improves accuracy and robustness, particularly for nonlinear systems like the twin-rotor system and electro-mechanical positioning system. Additionally, Rao's arithmetic optimization algorithm (AOA) leverages fundamental arithmetic operations for broad optimization tasks, showing superior performance on engineering benchmarks. These studies affirm the advantages of hybrid and novel algorithms in solving complex, real-world optimization problems.

Several of these algorithms, despite their diverse inspirations, encounter similar fundamental challenges of getting trapped in local optima and maintaining a proper balance between exploration and exploitation. For instance, the Marine Predator Algorithm (MPA) [50] uses various stages of foraging strategies inspired by marine life to enhance search efficiency, yet it can sometimes suffer from premature convergence if the exploration phase is not sufficiently adaptive. Likewise, Sine Cosine Algorithm (SCA) [51] employs sinusoidal functions to control the movement of candidate solutions, but its performance heavily depends on parameter settings that govern the algorithm's ability to escape local minima. Nonlinear variations of SCA aim to address this by incorporating adaptive or chaotic factors, although systematic challenges persist in tuning these parameters across different problem landscapes.

Another issue cutting across many metaheuristics is the imbalance between exploration and exploitation. When excessive emphasis is placed on exploration, the algorithm may wander around the search space without converging efficiently; conversely, too much exploitation can cause stagnation in suboptimal regions. Existing strategies such as adaptive parameter control, chaotic maps, and hybridizing multiple metaheuristics have shown promise in alleviating these issues, but gaps remain in creating unified frameworks that robustly handle diverse optimization scenarios. The research gap thus lies in developing metaheuristic algorithms with self-tuning or context-aware mechanisms that dynamically modulate their search behavior to avoid local optima and maintain a balanced exploration-exploitation process. By examining these gaps more closely, future work can focus on integrating insights from successful hybrid approaches and novel adaptation strategies to further advance the state of the art in metaheuristic optimization.

2.1 Overview of DE

DE, introduced by Rainer Storn and Kenneth Price in 1995, is a popular population-based optimization algorithm for solving multi-dimensional real-valued functions. DE operates using three main operators: mutation, crossover, and selection. The structure of DE is defined as follows:

Population initialization: DE begins by initializing a population of N individuals randomly within the search space. Each individual's initial position $\mathbf{X}_i^{(0)}$ is generated within the lower and upper bounds of the search space, as shown in the following equation:

$$\mathbf{X}_i^{(0)} \in [\mathbf{lb}, \mathbf{ub}], \quad i = 1, 2, \ldots, N \tag{1}$$

Mutation: DE creates a mutant vector $\mathbf{V}_i^{(t)}$ for each individual by adding the weighted difference between two randomly selected population vectors to a third vector, as shown in the following equation:

$$\mathbf{V}_i^{(t)} = \mathbf{X}_{r1}^{(t)} + F \cdot (\mathbf{X}_{r2}^{(t)} - \mathbf{X}_{r3}^{(t)}), \tag{2}$$

where F is the scaling factor controlling the mutation strength, and $r1$, $r2$, $r3$ are distinct random indices from the population.

Crossover: The next step is to combine the mutant vector with the target vector to form a trial vector $\mathbf{U}_i^{(t)}$. This is done through a crossover operation, as shown in the following equation:

$$U_{i,j}^{(t)} = \begin{cases} V_{i,j}^{(t)} & \text{if } \text{rand}_j \le CR \text{ or } j = j_{\text{rand}}, \\ X_{i,j}^{(t)} & \text{otherwise}, \end{cases} \tag{3}$$

where CR is the crossover probability, rand_j is a random number between 0 and 1, and j_{rand} is a randomly chosen index.

Selection: In the selection phase, the trial vector is compared to the target vector based on their fitness values. The vector with the better fitness value is selected for the next generation, as shown in the following equation:

$$\mathbf{X}_i^{(t+1)} = \begin{cases} \mathbf{U}_i^{(t)} & \text{if } f(\mathbf{U}_i^{(t)}) < f(\mathbf{X}_i^{(t)}) \\ \mathbf{X}_i^{(t)} & \text{otherwise}. \end{cases} \tag{4}$$

Despite its robustness, DE has some limitations in terms of balancing exploration and exploitation. Excessive exploration caused by large mutation factors can slow down convergence, while too much exploitation, caused by smaller factors, can lead to premature convergence and getting stuck in local optima. The mutation step in equation (2) contributes significantly to exploration, but the overall diversity of the population may decline after several generations, making it prone to stagnation.

2.2 Overview of WSO

The WSO is a bio-inspired metaheuristic algorithm that simulates the hunting behavior of white sharks, particularly their ability to detect, pursue, and capture prey. The WSO algorithm operates by balancing exploration and exploitation through position and velocity updates, driven by both random and deterministic factors. Below, we outline the key steps and equations used in the WSO.

Population initialization: WSO starts by randomly initializing the positions of a population of white sharks (*whiteSharks*) within the search space. The positions are initialized as in the following equation:

$$\mathbf{X}_i^{(0)} \in [\mathbf{lb}, \mathbf{ub}], \quad i = 1, 2, \ldots, \text{whiteSharks}, \tag{5}$$

where $\mathbf{X}_i^{(0)}$ is the initial position of the ith white shark, and **lb** and **ub** are the lower and upper bounds of the search space, respectively.

Velocity initialization: The initial velocity of each white shark is set to zero, as shown below:

$$\mathbf{V}_i^{(0)} = 0 \cdot \mathbf{X}_i^{(0)}. \tag{6}$$

Fitness evaluation: The fitness of each white shark is computed using the objective function *fobj*. The initial fitness of the population is calculated and stored as follows:

$$\text{fitness}_i = fobj(\mathbf{X}_i^{(0)}), \quad i = 1, 2, \ldots, \text{whiteSharks}. \tag{7}$$

Velocity update: During each iteration, the velocity of each white shark is updated based on both the best-known global position (*gbest*) and the best position of a randomly chosen individual (*wbest*). This update is influenced by an inertia parameter μ and a weighting factor wr, as shown in the following equation:

$$\mathbf{V}_i^{(t)} = \mu \mathbf{V}_i^{(t-1)} + \text{wr} \cdot (\mathbf{wbest}_{v(i)}^{(t-1)} - \mathbf{X}_i^{(t-1)}), \tag{8}$$

where $v(i)$ is the index of a randomly selected white shark, and wr is a random weighting factor that ensures the stochastic nature of the velocity updates.

Position update: The position of each white shark is updated based on its velocity and the wavy motion of the shark. The position update is governed by the frequency f, which controls the wavy motion and is calculated based on the minimum and maximum frequencies ($f_{\min}$ and $f_{\max}$), as shown in the following equation:

$$f = \frac{f_{\max} - f_{\min}}{f_{\max} + f_{\min}}. \tag{9}$$

The new position of the white shark is then updated as follows (as shown in equation (10)):

$$\mathbf{X}_i^{(t)} = \mathbf{X}_i^{(t-1)} + \frac{\mathbf{V}_i^{(t)}}{f}. \tag{10}$$

The position update ensures that white sharks follow a wavy motion, characteristic of their natural hunting behavior.

Boundary check: After updating positions, WSO ensures that all individuals remain within the boundaries of the search space. This boundary correction is applied as the following equation:

$$\mathbf{X}_i^{(t)} = \max(\mathbf{lb}, \min(\mathbf{X}_i^{(t)}, \mathbf{ub})). \tag{11}$$

Sensing mechanism and prey pursuit: WSO incorporates a prey-sensing mechanism where sharks adjust their positions based on their proximity to the global best position (*gbest*). If the shark senses prey, it uses a "sensing" parameter *mv* to either randomly relocate within the bounds or continue moving based on its velocity. The probability of pursuing prey is adjusted during iterations, and the sensing parameter *mv* is updated as shown below:

$$mv = \frac{1}{a_0 + \exp\left(\frac{\text{itemax}/2 - t}{a_1}\right)}, \tag{12}$$

where a_0 and a_1 are constants controlling the decay of the sensing mechanism over time.

Fishing school mechanism: Another key feature of WSO is the "fishing school" mechanism, where white sharks follow the best global position while accounting for their distance from *gbest*. If a randomly generated number is smaller than the school size parameter s_s, the shark updates its position according to the following rule (as shown in equation (13)):

$$\mathbf{X}_i^{(t)} = \frac{\mathbf{X}_i^{(t-1)} + \mathbf{X}_{i-1}^{(t)}}{2} \cdot \text{rand}, \tag{13}$$

where rand is a random number between 0 and 1, and $\mathbf{X}_{i-1}^{(t)}$ is the previous position of the neighboring shark.

Fitness update and global best update: After updating the positions, the fitness of each white shark is recalculated. If the new fitness of an individual is better than its previous fitness, the individual's best position is updated. The global best position (*gbest*) is updated if the best individual's fitness improves further, as shown in the following equation:

$$gbest = \mathbf{wbest}_{\text{index}}, \quad \text{where index} = \arg\min(\text{fitness}). \tag{14}$$

The White Shark Optimizer (WSO) presents several limitations that impact its performance in optimization problems. One significant drawback is its susceptibility to local optima trapping, particularly in high-dimensional and complex multimodal search spaces. While WSO incorporates mechanisms to balance exploration and exploitation, it may still converge prematurely to suboptimal solutions. Additionally, WSO's performance is highly sensitive to certain key parameters, such as the frequency of the wavy motion (f), acceleration coefficient (τ), and movement forces (p_1, p_2, mv, ss). Improper tuning of these parameters can result in poor convergence behavior or inefficient searches.

Another limitation of WSO is the curse of dimensionality, as its effectiveness tends to degrade with increasing problem dimensions, making it less suitable for large-scale optimization tasks. Finally, although WSO attempts to balance exploration and exploitation dynamically, improper balance can lead to excessive wandering in the search space or premature convergence, limiting its ability to find optimal solutions efficiently.

2.3 Overview of FNNs and MLP

FNNs are a type of neural network where the connections between neurons flow in one direction: from the input layer, through any hidden layers, and finally to the output layer. In FNNs, information only moves forward, with no loops or feedback connections. One specific type of FNN is the MLP, which contains one or more hidden layers. MLPs are widely used for supervised learning tasks, including classification and regression, due to their ability to learn complex patterns in data.

The structure of an MLP consists of three main components: the input layer, hidden layer(s), and output layer. The input layer consists of n neurons, each corresponding to one of the n features in the input data. The hidden layer (or layers) contains h neurons, with each neuron in the hidden layer receiving inputs from all neurons in the previous layer. The output layer has m neurons, and the number of neurons in this layer depends on the type of problem being solved (e.g., one neuron for binary classification, multiple neurons for multi-class classification or regression).

Each connection between neurons has an associated weight, which is adjusted during training to minimize the error between the predicted outputs and the actual values. Additionally, each neuron has a bias term that helps the network fit the data better by shifting the activation function.

The computation in an MLP proceeds in several steps. First, each neuron in the hidden layer computes the weighted sum of its inputs from the input layer, adding a bias term. This can be expressed mathematically as follows:

$$s_j = \sum_{i=1}^{n} (W_{ij} \cdot X_i) - \theta_j, \quad j = 1, 2, \ldots, h, \tag{15}$$

where s_j represents the weighted sum for the jth neuron in the hidden layer, where W_{ij} is the weight connecting the ith input neuron to the jth hidden neuron, X_i is the input value, and θ_j is the bias for the jth hidden neuron.

After calculating the weighted sum, each hidden neuron applies a nonlinear activation function to introduce nonlinearity into the network, which allows the network to model complex relationships in the data. A commonly used activation function is the sigmoid function, which is defined as follows:

$$S_j = \text{sigmoid}(s_j) = \frac{1}{1 + \exp(-s_j)}, \quad j = 1, 2, \ldots, h, \tag{16}$$

where S_j represents the output of the jth hidden neuron after applying the sigmoid activation function to the weighted sum s_j.

The outputs of the hidden layer neurons are then passed to the output layer. Each neuron in the output layer computes a weighted sum of the inputs from the hidden layer, similar to the process in the hidden layer:

$$o_k = \sum_{j=1}^{h} (w_{jk} \cdot S_j) - \theta_k, \quad k = 1, 2, \ldots, m, \tag{17}$$

where o_k is the weighted sum for the kth output neuron, where w_{jk} is the weight connecting the jth hidden neuron to the kth output neuron, and θ_k is the bias for the kth output neuron.

Finally, the output layer applies an activation function to the weighted sum to produce the final output. For classification tasks, the sigmoid function is commonly used in the output layer as well:

$$O_k = \text{sigmoid}(o_k) = \frac{1}{1 + \exp(-o_k)}, \quad k = 1, 2, \ldots, m, \tag{18}$$

where O_k represents the final output of the kth output neuron after applying the sigmoid function to the weighted sum o_k.

The key variables in the MLP structure are essential for understanding how the network processes inputs and generates outputs. The weights W_{ij} and w_{jk} determine how strongly each neuron in one layer is connected to the neurons in the next layer. The biases θ_j and θ_k allow each neuron to shift its activation function, improving the network's ability to fit the data. The inputs X_i represent the raw data fed into the network, while the outputs S_j and O_k are the intermediate and final results produced by the network, respectively.

As illustrated by equations (15)–(18), the relationship between the inputs and outputs in an MLP is determined by the network's weights and biases. These parameters are adjusted during the training process, which seeks to find the optimal set of weights and biases that minimize the error between the predicted outputs and the actual values. In Section 3, the hybrid WSODE will be applied to train the MLP by optimizing the weights and biases.

3 Hybrid WSODE algorithm

The hybrid WSODE algorithm is proposed as a novel optimization technique that synergistically combines the strengths of DE and the WSO. The primary objective of this hybrid algorithm is to leverage the explorative capabilities of DE and the exploitative efficiency of WSO to solve complex global optimization problems more effectively.

DE is well-known for its robustness in exploring the search space and its ability to avoid local optima by maintaining population diversity through mutation and crossover operations. DE generates new candidate solutions by perturbing existing ones with the scaled differences of randomly selected individuals. This process helps in exploring various regions of the search space and ensures a wide coverage.

On the other hand, the WSO excels in exploiting the search space by fine-tuning the solutions found during the exploration phase. WSO simulates the hunting behavior of white sharks, where individuals update their positions based on the best solutions found so far, adjusting their velocities and positions to converge toward the global optimum. WSO's exploitation mechanism makes it highly efficient in refining solutions and enhancing convergence speed.

By integrating DE and WSO, the hybrid algorithm benefits from the initial global search capability of DE and the subsequent local search efficiency of WSO. The DE phase focuses on broad exploration, ensuring diverse solutions and avoiding premature convergence. Once the DE phase concludes, the best solutions are passed to the WSO phase, which refines these solutions to achieve a higher precision in locating the global optimum.

This hybrid approach is particularly important for solving high-dimensional and multimodal optimization problems where the search space is vast and complex. The combined strategy helps in balancing exploration and exploitation, thereby improving the overall optimization performance. The hybrid WSODE algorithm is expected to outperform traditional single-method approaches, providing a more robust and reliable solution for various real-world optimization tasks.

3.1 WSODE mathematical model

In this section, we introduce the WSODE algorithm, which is a hybrid approach combining the strengths of DE and the WSO. The goal of WSODE is to leverage DE for exploration and WSO for exploitation, aiming to

enhance optimization performance. The procedure for WSODE can be broken down into two main phases: the DE phase and the WSO phase.

Population initialization: Initialize the population randomly within the search space (as shown in equation (19)).

$$\mathbf{X}_i^{(0)} \in [\mathbf{lb}, \mathbf{ub}], \quad i = 1, 2, \ldots, N, \tag{19}$$

where $\mathbf{X}_i^{(0)}$ is the initial position of the ith individual, N is the population size, and **lb** and **ub** are the lower and upper bounds of the search space, respectively.

DE phase: Run DE for a predefined number of iterations to explore the search space and find a good initial solution.

Mutation: Create a mutant vector by adding the weighted difference between two population vectors to a third vector (as shown in equation (20)).

$$\mathbf{V}_i^{(t)} = \mathbf{X}_{r1}^{(t)} + F \cdot (\mathbf{X}_{r2}^{(t)} - \mathbf{X}_{r3}^{(t)}), \tag{20}$$

where F is the mutation factor, $r1, r2, r3$ are distinct random indices, and t is the current iteration .

Crossover: Create a trial vector by combining elements from the target vector and the mutant vector (as shown in equation (21)).

$$U_{i,j}^{(t)} = \begin{cases} V_{i,j}^{(t)} & \text{if } \text{rand}_j \le CR \text{ or } j = j_{\text{rand}} \\ X_{i,j}^{(t)} & \text{otherwise}, \end{cases} \tag{21}$$

where CR is the crossover probability, rand_j is a random number between 0 and 1, and j_{rand} is a randomly chosen index.

Selection: Select the better vector between the trial vector and the target vector (as shown in equation (22)).

$$\mathbf{X}_i^{(t+1)} = \begin{cases} \mathbf{U}_i^{(t)} & \text{if } f(\mathbf{U}_i^{(t)}) < f(\mathbf{X}_i^{(t)}) \\ \mathbf{X}_i^{(t)} & \text{otherwise}. \end{cases} \tag{22}$$

WSO phase: Use the best solutions from DE as the initial population for WSO to refine the solutions and find the global optimum. **Velocity update:** Update the velocity of each individual based on the best positions found so far (as shown in equation (23)).

$$\mathbf{V}_i^{(t)} = \mu \mathbf{V}_i^{(t-1)} + \text{wr} \cdot (\mathbf{wbest}_{v(i)}^{(t-1)} - \mathbf{X}_i^{(t-1)}), \tag{23}$$

where μ is an inertia parameter, wr is a weighting factor, and $\mathbf{wbest}_{v(i)}^{(t-1)}$ is a randomly chosen individual from the best solutions.

Position update: Update the position of each individual based on its velocity (as shown in equation (24)).

$$\mathbf{X}_i^{(t)} = \mathbf{X}_i^{(t-1)} + \frac{\mathbf{V}_i^{(t)}}{f}, \tag{24}$$

where f is a factor controlling the wavy motion.

Boundary check and correction: Ensure that individuals remain within the search space boundaries (as shown in equation (25)).

$$\mathbf{X}_i^{(t)} = \max(\mathbf{lb}, \min(\mathbf{X}_i^{(t)}, \mathbf{ub})). \tag{25}$$

Best solution update: Update the best positions found so far (as shown in equation (26)).

$$\mathbf{wbest}_i^{(t)} = \begin{cases} \mathbf{X}_i^{(t)} & \text{if } f(\mathbf{X}_i^{(t)}) < f(\mathbf{wbest}_i^{(t-1)}) \\ \mathbf{wbest}_i^{(t-1)} & \text{otherwise}. \end{cases} \tag{26}$$

Tracking the best solution:

$$\text{ccurve}(t) = \min_i f(\mathbf{X}_i^{(t)}), \tag{27}$$

where $\text{ccurve}(t)$ records the best objective function value found up to iteration t (as shown in equation (27)).

The hybrid WSODE algorithm as shown in Algorithm 1, combines the DE and WSO techniques to enhance optimization performance. In the initialization phase, the population is randomly initialized within the search space, covering a wide range of potential solutions (as shown in equation (19)). During the DE phase, a mutant vector is created by adding the weighted difference between two population vectors to a third vector (as shown in equation (20)). A trial vector is then formed by combining elements from the target vector and the mutant vector (as shown in equation (21)). The better vector between the trial vector and the target vector is selected based on their fitness values (as shown in equation (22)). This process is repeated for a predefined number of iterations to explore the search space.

In the WSO phase, as shown in Figure 1, the velocity of each individual is updated based on the best positions found so far (as shown in equation (23)). The position of each individual is then updated based on its velocity (as shown in equation (24)). A boundary check and correction ensure that individuals remain within the search space boundaries (as shown in equation (25)). The best positions found so far are updated based on the fitness values (as shown in equation (26)). This phase continues for the remaining number of iterations to refine the solutions and find the global optimum. Throughout the iterations, the algorithm records the best objective function value found up to each iteration (as shown in equation (27)).

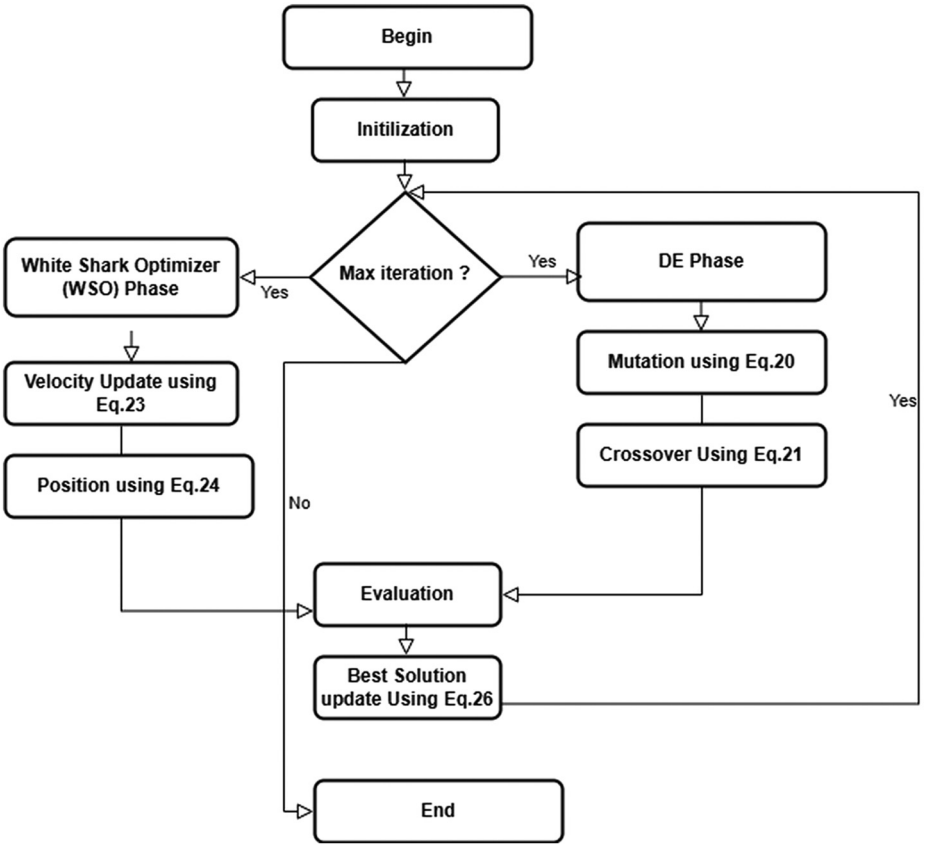

Figure 1: WSODE Flowchart. Source: Created by the authors.

Algorithm 1. Pseudocode and steps of WSODE algorithm

Initialization:
Initialize the population randomly within the search space using equation (19)
Differential Evolution (DE) Phase:
for t = 1 to t_{DE} **do**
    for each individual i in the population **do**
        Mutation: Create a mutant vector using equation (20).
        Crossover: Create a trial vector using equation (21).
        Selection: Select the better vector between the trial vector and the target vector using equation (22).
    end for
end for
White Shark Optimizer (WSO) Phase:
for $t = t_{\mathrm{DE}} + 1$ to $t_{\max}$
    for each individual i in the population **do**
        Velocity Update: Update the velocity of each individual using equation (23).
        Position Update: Update the position of each individual using equation (24).
        Boundary Check and Correction: Ensure that individuals remain within the search space boundaries using equation (25).
        Best Solution Update: Update the best positions found so far using equation (26).
    end for
end for
Output:
Return the best solution found and the convergence curve.

3.2 Exploration, exploitation, and local optima avoidance features of WSODE

WSODE adeptly balances exploration and exploitation through its integration of DE and the WSO. Each phase of the algorithm is tailored to enhance either exploration or exploitation, ensuring a comprehensive search of the solution space and effective refinement of potential solutions.

The exploration capabilities of WSODE are primarily driven by the DE component. DE is renowned for its ability to traverse the search space extensively, preventing the algorithm from getting trapped in local optima. This phase involves the mutation operation, where DE generates mutant vectors by adding the weighted differences between randomly selected population vectors to another vector (equation (20)). The crossover operation combines elements from the target vector and the mutant vector to produce a trial vector (equation (21)). This helps maintain diversity in the population and explores different combinations of solutions. Finally, DE ensures that only the best solutions are carried forward by selecting the better vector between the trial and target vectors (equation (22)). This selection process guarantees that the population evolves towards better solutions.

The exploitation capabilities of WSODE are primarily harnessed through the WSO component, which fine-tunes the solutions obtained from the DE phase. In the WSO phase, the velocity of each individual is updated based on the best positions found so far (equation (23)). The position of each individual is then updated based on its velocity (equation (24)). To ensure that individuals remain within the feasible search space, a boundary check and correction mechanism are employed (equation (25)). The best positions are continually updated based on the fitness values of the solutions, ensuring convergence towards the global optimum (equation (26)). Local optima avoidance is a crucial feature of WSODE. The DE phase introduces diversity through its mutation operation (equation (20)), which consistently generates new solutions from different areas of the search space, reducing the likelihood of the population getting stuck in suboptimal regions. Additionally, the crossover

operation (equation (21)) further enhances this diversity by combining different vectors, ensuring the algorithm explores a wide range of solutions.

In the WSO phase, local optima avoidance is enhanced by the use of the velocity updates (equation (8)), which drive individuals toward both global and local best solutions while maintaining a degree of randomness through the weighting factor wr. The wavy motion (equation (9)) also helps introduce nonlinearity into the position updates, allowing the individuals to avoid settling into local optima. Moreover, the adaptive prey-sensing mechanism (equation (12)) enables the sharks to dynamically adjust their positions based on their proximity to the global best, ensuring that they continue exploring rather than prematurely converging. The "fishing school" mechanism (equation (13)) further enhances the algorithm's ability to escape local optima by allowing individuals to adjust their positions relative to neighboring solutions, fostering collaboration within the population and encouraging exploration in different directions.

3.3 Solving single objective optimization problems using WSODE

WSODE is adept at solving single objective optimization problems, where the goal is to find the best solution that minimizes or maximizes a given objective function. The formulation of the objective function and the step-by-step process of solving such problems using WSODE are outlined below.

A single objective optimization problem can be mathematically formulated as:

$$\min_{\mathbf{x}} f(\mathbf{X}) \quad \text{or} \quad \max_{\mathbf{x}} f(\mathbf{X}),$$

where $f(\mathbf{X})$ is the objective function to be optimized, and $\mathbf{X}$ represents the vector of decision variables within the search space defined by the lower and upper bounds **lb** and **ub**, respectively.

The algorithm begins by initializing the population randomly within the search space, ensuring a diverse set of initial solutions as described by equation (19). In the DE phase, mutant vectors are generated using equation (20), and trial vectors are created by combining elements from the target and mutant vectors as per equation (21). The better vectors are then selected based on their fitness values, ensuring only the best solutions are carried forward, as formalized by equation (22).

In the WSO phase, the velocities of individuals are updated based on the best positions found so far (equation (23)), and their positions are updated accordingly (equation (24)). A boundary check and correction ensure solutions remain within search space boundaries (equation (25)), and the best solutions are updated based on their fitness values (equation (26)).

The convergence of the optimization process is tracked by recording the best objective function value found up to each iteration using equation (27). The algorithm concludes by returning the best solution found and the convergence curve, indicating the progression of the optimization process.

3.4 Computational complexity analysis of WSODE

The computational complexity of WSODE is analyzed by examining both the DE and WSO phases. Each phase contributes to the overall complexity, which depends on the population size N, the dimensionality of the problem dim, and the maximum number of iterations itemax.

DE phase complexity: In the DE phase, the algorithm starts by initializing a population of N individuals, each with dim dimensions. The main operations during each iteration include mutation, crossover, and selection:

Mutation involves selecting three random vectors from the population and applying a mutation strategy. This operation takes constant time per individual, i.e., $O(1)$, since it only requires arithmetic operations.

Crossover is performed over the dim dimensions for each individual. The algorithm iterates over all the dimensions, combining the target and mutant vectors, as shown in equation (3). This step has a complexity of $O(dim)$.

Selection compares the trial and target vectors based on their fitness values, and the better vector is retained for the next generation. The selection step is performed for each individual, and the fitness is evaluated in constant time, i.e., $O(1)$, as shown in equation (4).

The DE phase runs for $DE\_itermax = \frac{\text{itemax}}{2}$ iterations. Therefore, the total complexity of the DE phase is $O(N \times \dim \times \frac{\text{itemax}}{2})$, as shown in equation (28).

$$O(N \times \dim \times \frac{\text{itemax}}{2}). \tag{28}$$

WSO phase complexity: After the DE phase, WSO takes over and refines the solutions further. The main operations in the WSO phase include velocity updates, position updates, and fitness evaluations.

Velocity update is performed for each individual by calculating the difference between the current position and the best-known solutions, scaled by a random weighting factor. This operation is performed over all dim dimensions and has a complexity of $O(dim)$.

Position update involves updating each individual's position based on the velocity and wavy motion. This step also runs over dim dimensions and has a complexity of $O(dim)$

Fitness evaluation is carried out after updating the positions. The fitness of each individual is evaluated in constant time, i.e., $O(1)$, assuming the objective function evaluation is constant.

The WSO phase runs for itemax iterations, so the total complexity of the WSO phase is $O(N \times \dim \times \text{itemax})$, as shown in in the following equation:

$$O(N \times \dim \times \text{itemax}). \tag{29}$$

Overall complexity: The total computational complexity of WSODE is the sum of the complexities of both the DE and WSO phases. This is represented as:

$$O\left(N \times \dim \times \frac{\text{itemax}}{2}\right) + O(N \times \dim \times \text{itemax}). \tag{30}$$

Simplifying the expression results in the overall complexity of WSODE

$$O(N \times \dim \times \text{itemax}), \tag{31}$$

where N is the population size, dim is the dimensionality of the problem, and itemax is the total number of iterations.

4 Data description

4.1 Institute of electrical and electronics engineers (IEEE) congress on evolutionary computation CEC2022 benchmark description

The assessment of WSODE efficacy leveraged a comprehensive array of benchmark functions from the CEC2022 competition. These functions are crafted to probe the capabilities and flexibility of evolutionary computation algorithms in diverse optimization environments. The suite includes various types of functions: unimodal, multimodal, hybrid, and composition. Unimodal functions, exemplified by the Shifted and Full Rotated Zakharov function (F1), evaluate the fundamental search capabilities and convergence attributes of the algorithms. In contrast, multimodal functions, such as the Shifted and Full Rotated Levy function (F5), present multiple local optima, thus testing the algorithms' global search proficiency. Hybrid functions, with Hybrid function 3 (F8) as an instance, integrate aspects of different problem domains to reflect more intricate and practical optimization challenges. Furthermore, composition functions, notably Composition function 4 (F12), combine various problem landscapes into a singular test scenario, assessing the adaptability and resilience of the algorithms.

4.2 IEEE congress on evolutionary computation CEC2021 benchmark description

The CEC2021 benchmark functions are a set of standardized test functions designed to evaluate and compare the performance of optimization algorithms. These functions encompass various types of optimization challenges, including unimodal, multimodal, hybrid, and composition functions, each presenting unique complexities and characteristics. Unimodal functions have a single global optimum, while multimodal functions contain multiple local optima, making them challenging for optimization algorithms to navigate. Hybrid functions combine features from different types of functions, and composition functions blend multiple sub-functions to create highly complex landscapes. The primary goal of these benchmarks is to provide a rigorous and diverse set of problems that can thoroughly test the robustness, efficiency, and accuracy of optimization techniques in a controlled and consistent manner.

4.3 IEEE congress on evolutionary computation CEC2017 benchmark description

The CEC2017 benchmark suite, launched at the IEEE congress on evolutionary computation in 2017, provides a sophisticated array of test functions specifically designed to evaluate and enhance the capabilities of optimization algorithms. This suite is a significant update from previous iterations, incorporating new challenges that accurately reflect the evolving complexities found in real-world optimization scenarios. The suite is systematically organized into different categories including unimodal, multimodal, hybrid, and composition test functions, each tailored to assess distinct aspects of algorithmic performance. Unimodal functions within the suite test the algorithms' ability to refine solutions in relatively simple environments, focusing on the depth of exploitation required to achieve optimal results. Multimodal functions, by contrast, challenge algorithms on their exploratory capabilities, essential for identifying global optima in landscapes populated with numerous local optima. Hybrid functions examine the versatility of algorithms in handling a mix of these environments, while composition functions are designed to test the algorithms' proficiency in managing a combination of several complex scenarios concurrently. The design of the CEC2017 functions aims to rigorously evaluate not just the accuracy and speed of algorithms in reaching optimal solutions, but also their robustness, scalability, and adaptability in response to dynamic and noisy environments. This comprehensive testing is crucial for advancing metaheuristic algorithms and other evolutionary computation techniques, ensuring they are sufficiently robust and versatile for practical applications. The CEC2017 benchmark suite thus stands as a vital resource for the optimization community, offering a structured and challenging environment for the continuous evaluation and refinement of algorithms. It plays a pivotal role in driving the innovation and development of advanced optimization methods that are capable of addressing the complex and dynamic challenges present in various sectors.

5 Testing and performance

5.1 Setting parameters for benchmark testing

The setting of parameters is essential for the uniform evaluation of optimization algorithms using the benchmark functions established by the CEC competitions in 2022, 2021, and 2017. These established parameters ensure a standardized environment that facilitates comparative analysis of different evolutionary algorithms. Table 1 provides a summary of these settings across all benchmarks.

The chosen population size of 30 and dimensionality of 10 strike an effective balance between computational manageability and the complexity needed for significant testing. A limit of 1,000 function evaluations ensures adequate iterations for the algorithms to demonstrate their potential for convergence, without

Table 1: Standard parameter configurations for CEC benchmarks

Parameter	Value
Population size	30
Maximum function evaluations	1,000
Dimensionality (D)	10
Search range	$[-100,\ 100]^D$
Rotation	Included for all rotating functions
Shift	Included for all shifting functions

imposing undue computational demands. The specified search range of $[-100,\ 100]^D$ provides a wide and uniform testing space across all dimensions.

The benchmark functions frequently incorporate rotation and shifting to enhance the complexity, better mimicking real-world optimization scenarios. Excluding noise focuses the results on the algorithms' ability to adeptly navigate complex environments, rather than dealing with random variations. This structured setting enables an equitable comparison among different algorithms, showcasing their strengths and weaknesses within a broad spectrum of benchmark tests (Table 2).

Table 2: Detailed parameters for comparative algorithm analysis

Algorithm	Parameter
MFO	Gradually decreases from −1 to −2
SHIO	No additional parameters required
FOX	Modularity = 0.01, Exponent = v, Switching likelihood = 0.8
HHO	U = 0.05, V = 0.05, L = 0.05
WSO	Adjustment constant a = [2, 0]
DA	w = 0.9, c = 0.1
SCA	r1 = random(0, 1), r2 = random(0, 1), r3 = random(0, 1), r4 = random(0,1)

In conducting a detailed statistical evaluation of various optimization algorithms, we utilized key statistical metrics including the mean, standard deviation (STD). The mean is crucial as it represents the central tendency, summarizing the average results achieved by the algorithms across multiple trials and providing an overview of their general performance levels. The STD is employed to measure the extent of variation or dispersion from the mean, which sheds light on the reliability and uniformity of the results from different tests. These measures are essential for determining the stability and predictability of the algorithms' effectiveness.

5.2 Discussion of WSODE results on IEEE congress on evolutionary computation CEC 2022 benchmarks

The WSODE algorithm demonstrates outperforming performance across the CEC2022 benchmark suite, designed to evaluate optimization algorithms on complex, high-dimensional, multi-modal, and deceptive landscapes, which present significant challenges for conventional metaheuristics. As shown in Table 3, WSODE consistently outperforms or performs comparably to state-of-the-art algorithms such as WSO, grey wolf optimizer (GWO), whale optimization algorithm (WOA), moth-flame optimization (MFO), fox optimization algorithm (FOX), success history intelligent optimizer (SHIO), dung beetle optimization, oriolus heuristic optimization (OHO), four vector intelligent metaheuristic (FVIM), and spotted hyena optimizer (SHO). The results showcase WSODE's capability to balance exploration and exploitation effectively, which is essential for avoiding premature convergence and achieving high-quality solutions across diverse problem landscapes.

In Function F1, which assesses unimodal performance where algorithms can effectively test their convergence capabilities, WSODE achieves the best performance with a mean value of 3.00×10^2, ranking first overall. This result indicates WSODE's superior convergence speed and precision, as it consistently finds the global optimum more reliably than other algorithms. The close ranking in Function F2, where WSODE ranks second with a mean value of 4.06×10^2, reflects its robustness and stability in handling various optimization scenarios, outclassing other popular algorithms like GWO, WOA, and MFO. This result indicates WSODE's adaptability to challenges posed by slightly more complex unimodal landscapes, demonstrating its reliability in diverse optimization tasks.

For Function F3, which includes deceptive elements to challenge the search strategy, WSODE ranks first with a mean value of 6.00×10^2 and exhibits a notably low STD, underscoring its ability to avoid deception traps and maintain consistency in identifying optimal solutions. In Functions F4 and F5, characterized by multi-modality and complex local optima configurations, WSODE maintains top-tier performance, securing second place in both functions. This performance, with a mean value of 8.16×10^2 for Function F4 and a leading value of 9.00×10^2 in Function F5, highlights WSODE's resilience and adaptability to the intricate landscape of local optima.

In the highly rugged landscape of Function F6, WSODE secures first place with a mean value of 1.80×10^3, surpassing competitors like WOA, FOX, and SHO. This outcome underscores WSODE's capability to efficiently explore the search space and identify high-quality solutions in highly complex, multi-modal scenarios. Furthermore, WSODE consistently ranks first or second in Functions F7, F8, and F9, where the landscape complexity increases with the addition of narrow, deep basins and deceptive valleys. Specifically, WSODE ranks first in Function F7 with a mean value of 2.01×10^3, displaying its robustness and consistency in complex, nonlinear landscapes.

For the complex multi-modal scenarios in Functions F10 and F11, WSODE achieves first place with mean values of 2.51×10^3 and 2.60×10^3, respectively. These results validate WSODE's strength in maintaining a strong balance between exploration of diverse areas and exploitation of promising solutions, ensuring that it does not converge prematurely on suboptimal points. Finally, WSODE ranks first in Function F12 with a mean value of 2.86×10^3, further cementing its effectiveness in tackling high-dimensional, complex optimization problems with deceptive and rugged landscapes (Table 3).

The Wilcoxon signed-rank test results for the CEC 2022 benchmarks (Table 4) show that the WSODE optimizer demonstrates robust performance across most comparisons. Although WSODE faces challenges from DE, with only 1 win, 5 losses, and 6 ties, it performs better against GWO, achieving 8 wins, 1 loss, and 3 ties. WSODE exhibits dominance over WOA and butterfly optimization algorithm (BOA), winning all 12 functions in both cases without any losses or ties. Similarly, WSODE outperforms MFO with 6 wins, 2 losses, and 4 ties, and shows a strong advantage over SHIO with 11 wins and 1 tie. Against coot optimization algorithm (COA), WSODE secures 11 wins and suffers only 1 loss. In flawless comparisons, WSODE wins all functions against OHO (12 wins), SCA (11 wins, 1 tie), and golden jackal optimization (GJO) (11 wins, 1 tie). Against SHO, WSODE achieves 10 wins, 1 loss, and 1 tie.

5.3 Discussion of the WSODE results on IEEE congress on evolutionary computation CEC 2021

The WSODE algorithm demonstrates significant performance improvements across the CEC2021 benchmark suite, which includes diverse optimization landscapes with varying properties, designed to test algorithms on different aspects such as multi-modality, ruggedness, separability, and deceptive traps. Table 5 presents a detailed comparison of WSODE's performance against other leading optimizers, including WSO, covariance matrix adaptation evolution strategy (CMAES), particle swarm optimization (PSO), multi-verse optimizer (MVO), dandelion optimizer (DO), MFO, SHIO, simplified dolphin echolocation (SDE), bat algorithm (BAT), and FOX, across ten test functions (C1 to C10). These results highlight WSODE's adaptability and superior ability to navigate complex optimization landscapes.

Function C1 is a high-dimensional unimodal function, aimed at evaluating an algorithm's convergence speed and precision in simpler landscapes without local minima. WSODE achieves a mean value of 3.45×10^{-33}

Table 3: WSODE comparison results on IEEE congress on evolutionary computation 2022 with FES = 1,000 and 30 independent runs

Function	Measurments	WSODE	WSO	DE	GWO	WOA	MFO	BOA	SHIO	COA	OHO	SCA	GJO	SHO
F1	Mean	3.00×10^{2}	3.00×10^{2}	4.26×10^{2}	1.07×10^{3}	1.56×10^{4}	6.66×10^{3}	7.95×10^{3}	3.61×10^{3}	3.34×10^{2}	1.57×10^{4}	1.09×10^{3}	9.90×10^{2}	1.99×10^{3}
	Std	6.56×10^{-14}	6.11×10^{-4}	8.47×10^{1}	1.31×10^{3}	6.72×10^{3}	7.21×10^{3}	2.10×10^{3}	2.82×10^{3}	9.73×10^{1}	4.41×10^{3}	2.40×10^{2}	1.26×10^{3}	2.61×10^{3}
	SEM	2.08×10^{-14}	1.93×10^{-4}	1.26×10^{2}	4.15×10^{2}	2.12×10^{3}	2.28×10^{3}	6.65×10^{2}	8.93×10^{2}	3.08×10^{1}	1.39×10^{3}	7.60×10^{1}	3.99×10^{2}	8.26×10^{2}
	Rank	1	2	4	6	12	10	11	9	3	13	7	5	8
F2	Mean	4.06×10^{2}	4.00×10^{2}	4.09×10^{2}	4.24×10^{2}	4.40×10^{2}	4.17×10^{2}	2.07×10^{3}	4.43×10^{2}	4.13×10^{2}	2.54×10^{3}	4.53×10^{2}	4.43×10^{2}	4.37×10^{2}
	Std	3.73×10^{00}	1.26×10^{00}	0.00×10^{00}	3.17×10^{1}	3.58×10^{1}	2.67×10^{1}	9.58×10^{2}	2.84×10^{1}	2.06×10^{1}	9.54×10^{2}	1.29×10^{1}	2.95×10^{1}	3.41×10^{1}
	SEM	1.18×10^{00}	3.99×10^{-1}	8.92×10^{00}	1.00×10^{1}	1.13×10^{1}	8.44×10^{00}	3.03×10^{2}	8.97×10^{00}	6.51×10^{00}	3.02×10^{2}	4.07×10^{00}	9.32×10^{00}	1.08×10^{1}
	Rank	2	1	3	6	8	5	12	9	4	13	11	10	7
F3	Mean	6.00×10^{2}	6.00×10^{2}	6.01×10^{2}	6.01×10^{2}	6.30×10^{2}	6.01×10^{2}	6.39×10^{2}	6.05×10^{2}	6.05×10^{2}	6.60×10^{2}	6.19×10^{2}	6.07×10^{2}	6.12×10^{2}
	Std	8.50×10^{-11}	3.08×10^{-1}	6.76×10^{-1}	6.88×10^{-1}	1.56×10^{1}	4.99×10^{-1}	5.95×10^{00}	6.18×10^{00}	1.36×10^{1}	4.09×10^{00}	3.88×10^{00}	7.71×10^{00}	4.76×10^{00}
	SEM	2.69×10^{-11}	9.75×10^{-2}	1.36×10^{-1}	2.18×10^{-1}	4.94×10^{00}	1.58×10^{-1}	1.88×10^{00}	1.95×10^{00}	4.29×10^{00}	1.29×10^{00}	1.23×10^{00}	2.44×10^{00}	1.50×10^{00}
	Rank	1	2	4	5	11	3	12	6	7	13	10	8	9
F4	Mean	8.16×10^{2}	8.08×10^{2}	8.20×10^{2}	8.16×10^{2}	8.42×10^{2}	8.35×10^{2}	8.47×10^{2}	8.16×10^{2}	8.29×10^{2}	8.44×10^{2}	8.41×10^{2}	8.34×10^{2}	8.23×10^{2}
	Std	7.26×10^{00}	6.41×10^{00}	1.35×10^{00}	3.26×10^{00}	1.82×10^{1}	1.35×10^{1}	9.93×10^{00}	4.08×10^{00}	7.47×10^{00}	4.03×10^{00}	6.09×10^{00}	1.27×10^{1}	8.54×10^{00}
	SEM	2.30×10^{00}	2.03×10^{00}	1.96×10^{1}	1.03×10^{00}	5.77×10^{00}	4.27×10^{00}	3.14×10^{00}	1.29×10^{00}	2.36×10^{00}	1.27×10^{00}	1.93×10^{00}	4.01×10^{00}	2.70×10^{00}
	Rank	2	1	5	3	11	9	13	4	7	12	10	8	6
F5	Mean	9.00×10^{2}	9.01×10^{2}	9.00×10^{2}	9.06×10^{2}	1.46×10^{3}	1.03×10^{3}	1.27×10^{3}	9.28×10^{2}	9.06×10^{2}	1.57×10^{3}	1.00×10^{3}	9.79×10^{2}	1.03×10^{3}
	Std	0.00×10^{00}	8.30×10^{-1}	1.31×10^{-5}	1.10×10^{1}	3.67×10^{2}	2.89×10^{2}	5.85×10^{1}	2.89×10^{1}	8.31×10^{00}	5.45×10^{1}	3.67×10^{1}	3.10×10^{1}	9.57×10^{1}
	SEM	0.00×10^{00}	2.62×10^{-1}	7.99×10^{-6}	3.48×10^{00}	1.16×10^{2}	9.13×10^{1}	1.85×10^{1}	9.15×10^{00}	2.63×10^{00}	1.72×10^{1}	1.16×10^{1}	9.81×10^{00}	3.03×10^{1}
	Rank	1	3	2	5	12	10	11	6	4	13	8	7	9
F6	Mean	1.80×10^{3}	1.81×10^{3}	2.04×10^{3}	5.72×10^{3}	2.80×10^{3}	5.13×10^{3}	6.38×10^{7}	4.16×10^{3}	3.73×10^{3}	7.30×10^{8}	1.40×10^{6}	8.36×10^{3}	5.17×10^{3}
	Std	4.51×10^{-1}	4.68×10^{00}	3.69×10^{2}	2.26×10^{3}	9.74×10^{2}	2.34×10^{3}	1.17×10^{8}	2.32×10^{3}	1.90×10^{3}	9.77×10^{8}	8.78×10^{5}	2.57×10^{3}	1.20×10^{3}
	SEM	1.43×10^{-1}	1.48×10^{00}	2.35×10^{2}	7.14×10^{2}	3.08×10^{2}	7.40×10^{2}	3.69×10^{7}	7.33×10^{2}	6.02×10^{2}	3.09×10^{8}	2.78×10^{5}	8.12×10^{2}	3.78×10^{2}
	Rank	1	2	3	9	4	7	12	6	5	13	11	10	8
F7	Mean	2.01×10^{3}	2.02×10^{3}	2.00×10^{3}	2.03×10^{3}	2.08×10^{3}	2.02×10^{3}	2.08×10^{3}	2.04×10^{3}	2.02×10^{3}	2.13×10^{3}	2.06×10^{3}	2.04×10^{3}	2.03×10^{3}
	Std	9.58×10^{00}	8.67×10^{00}	5.37×10^{-5}	9.85×10^{00}	2.22×10^{1}	9.14×10^{-1}	1.33×10^{1}	1.25×10^{1}	8.15×10^{00}	5.11×10^{00}	1.34×10^{1}	9.76×10^{00}	1.12×10^{1}
	SEM	3.03×10^{00}	2.74×10^{00}	3.59×10^{-5}	3.11×10^{00}	7.02×10^{00}	2.89×10^{-1}	4.20×10^{00}	3.97×10^{00}	2.58×10^{00}	1.62×10^{00}	4.24×10^{00}	3.09×10^{00}	3.55×10^{00}
	Rank	2	4	1	6	12	5	11	9	3	13	10	8	7
F8	Mean	2.20×10^{3}	2.21×10^{3}	2.20×10^{3}	2.23×10^{3}	2.23×10^{3}	2.22×10^{3}	2.28×10^{3}	2.23×10^{3}	2.22×10^{3}	2.43×10^{3}	2.23×10^{3}	2.23×10^{3}	2.22×10^{3}
	Std	9.80×10^{-1}	9.49×10^{00}	2.90×10^{00}	4.25×10^{00}	4.28×10^{00}	4.27×10^{00}	6.70×10^{1}	3.29×10^{00}	7.89×10^{00}	1.27×10^{2}	2.81×10^{00}	3.50×10^{00}	1.83×10^{00}
	SEM	3.10×10^{-1}	3.00×10^{00}	4.89×10^{00}	1.34×10^{00}	1.35×10^{00}	1.35×10^{00}	2.12×10^{1}	1.04×10^{00}	2.50×10^{00}	4.03×10^{1}	8.88×10^{-1}	1.11×10^{00}	5.79×10^{-1}
	Rank	1	3	2	7	11	6	12	9	4	13	10	8	5
F9	Mean	2.53×10^{3}	2.53×10^{3}	2.53×10^{3}	2.56×10^{3}	2.59×10^{3}	2.53×10^{3}	2.76×10^{3}	2.60×10^{3}	2.53×10^{3}	2.84×10^{3}	2.56×10^{3}	2.58×10^{3}	2.59×10^{3}
	Std	0.00×10^{00}	1.48×10^{-4}	6.91×10^{00}	2.85×10^{1}	4.09×10^{1}	6.35×10^{00}	6.26×10^{1}	3.76×10^{1}	2.11×10^{-5}	8.77×10^{1}	1.64×10^{1}	3.06×10^{1}	3.86×10^{1}
	SEM	0.00×10^{00}	4.67×10^{-5}	2.33×10^{2}	9.00×10^{00}	1.29×10^{1}	2.01×10^{00}	1.98×10^{1}	1.19×10^{1}	6.66×10^{-6}	2.77×10^{1}	5.18×10^{00}	9.68×10^{00}	1.22×10^{1}

(*Continued*)

Table 3: *Continued*

Function	Measurments	WSODE	WSO	DE	GWO	WOA	MFO	BOA	SHIO	COA	OHO	SCA	GJO	SHO
	Rank	1	3	5	6	9	4	12	11	2	13	7	8	10
F10	Mean	2.51×10^{3}	2.54×10^{3}	2.53×10^{3}	2.56×10^{3}	2.54×10^{3}	2.53×10^{3}	2.52×10^{3}	2.53×10^{3}	2.55×10^{3}	2.81×10^{3}	2.52×10^{3}	2.59×10^{3}	2.55×10^{3}
	Std	3.33×10^{1}	5.57×10^{1}	2.14×10^{-2}	5.96×10^{1}	6.97×10^{1}	5.39×10^{1}	4.80×10^{1}	5.39×10^{1}	6.15×10^{1}	2.27×10^{2}	6.25×10^{-1}	6.03×10^{1}	6.75×10^{1}
	SEM	1.05×10^{1}	1.76×10^{1}	1.00×10^{2}	1.88×10^{1}	2.20×10^{1}	1.70×10^{1}	1.52×10^{1}	1.70×10^{1}	1.95×10^{1}	7.17×10^{1}	1.98×10^{-1}	1.91×10^{1}	2.13×10^{1}
	Rank	1	7	4	11	8	5	2	6	9	13	3	12	10
F11	Mean	2.60×10^{3}	2.65×10^{3}	2.65×10^{3}	2.88×10^{3}	2.78×10^{3}	2.78×10^{3}	2.98×10^{3}	2.82×10^{3}	2.73×10^{3}	4.04×10^{3}	2.82×10^{3}	2.84×10^{3}	2.80×10^{3}
	Std	4.29×10^{-13}	1.01×10^{2}	8.65×10^{1}	2.29×10^{2}	1.31×10^{2}	1.65×10^{2}	1.84×10^{2}	1.84×10^{2}	1.60×10^{2}	3.40×10^{2}	1.46×10^{2}	2.01×10^{2}	2.21×10^{2}
	SEM	1.36×10^{-13}	3.20×10^{1}	5.05×10^{1}	7.25×10^{1}	4.14×10^{1}	5.22×10^{1}	5.82×10^{1}	5.81×10^{1}	5.07×10^{1}	1.08×10^{2}	4.61×10^{1}	6.36×10^{1}	7.00×10^{1}
	Rank	1	2	3	11	6	5	12	8	4	13	9	10	7
F12	Mean	2.86×10^{3}	2.87×10^{3}	2.86×10^{3}	2.87×10^{3}	2.90×10^{3}	2.86×10^{3}	2.92×10^{3}	2.88×10^{3}	2.86×10^{3}	3.21×10^{3}	2.87×10^{3}	2.87×10^{3}	2.88×10^{3}
	Std	1.24×10^{00}	3.74×10^{00}	1.28×10^{00}	2.23×10^{00}	5.89×10^{1}	1.08×10^{00}	2.88×10^{1}	1.66×10^{1}	2.43×10^{00}	1.86×10^{2}	1.38×10^{00}	1.35×10^{1}	1.48×10^{1}
	SEM	3.91×10^{-1}	1.18×10^{00}	1.64×10^{2}	7.07×10^{-1}	1.86×10^{1}	3.40×10^{-1}	9.10×10^{00}	5.26×10^{00}	7.68×10^{-1}	5.89×10^{1}	4.36×10^{-1}	4.26×10^{00}	4.69×10^{00}
	Rank	1	6	4	5	11	2	12	9	3	13	7	8	10

Table 4: WSODE Wilcoxon signed rank sum (SRS) test results on IEEE congress on evolutionary computation 2022 with FES = 1,000 and 30 independent runs

Function	WSO	DE	GWO	WOA	MFO	BOA	SHIO	COA	OHO	SCA	GJO	SHO
F1	0.557743	0.24519	0.001114	1.73×10^{-6}	2.13×10^{-6}	1.73×10^{-6}	0.000831	1.73×10^{-6}	1.73×10^{-6}	4.73×10^{-6}	5.31×10^{-5}	0.000115
	T+: 261, T-: 204	T+: 176, T-: 289	T+: 391, T-: 74	T+: 465, T-: 0	T+: 463, T-: 2	T+: 465, T-: 0	T+: 395, T-: 70	T+: 465, T-: 0	T+: 465, T-: 0	T+: 455, T-: 10	T+: 429, T-: 36	T+: 420, T-: 45
F2	1.13×10^{-5}	0.271155	8.92×10^{-5}	1.73×10^{-6}	0.031603	1.73×10^{-6}	1.73×10^{-6}	6.98×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	3.88×10^{-6}	2.88×10^{-6}
	T+: 446, T-: 19	T+: 179, T-: 286	T+: 423, T-: 42	T+: 465, T-: 0	T+: 337, T-: 128	T+: 465, T-: 0	T+: 465, T-: 0	T+: 451, T-: 14	T+: 465, T-: 0	T+: 465, T-: 0	T+: 457, T-: 8	T+: 460, T-: 5
F3	1.73×10^{-6}	0.016566	0.001709	1.73×10^{-6}	9.32×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T-: 0	T+: 116, T-: 349	T+: 385, T-: 80	T+: 465, T-: 0	T+: 448, T-: 17	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0
F4	0.000306	1.73×10^{-6}	0.000174	0.001382	0.025637	1.73×10^{-6}	0.813017	0.000332	1.73×10^{-6}	2.88×10^{-6}	0.349346	5.31×10^{-5}
	T+: 408, T-: 57	T+: 465, T-: 0	T+: 50, T-: 415	T+: 388, T-: 77	T+: 124, T-: 341	T+: 465, T-: 0	T+: 244, T-: 221	T+: 58, T-: 407	T+: 465, T-: 0	T+: 460, T-: 5	T+: 278, T-: 187	T+: 36, T-: 429
F5	9.32×10^{-6}	0.003854	0.00016	1.73×10^{-6}	8.47×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	4.73×10^{-6}	1.73×10^{-6}
	T+: 448, T-: 17	T+: 92, T-: 373	T+: 416, T-: 49	T+: 465, T-: 0	T+: 449, T-: 16	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 455, T-: 10	T+: 465, T-: 0
F6	1.24×10^{-5}	0.440522	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 445, T-: 20	T+: 195, T-: 270	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0
F7	2.88×10^{-6}	0.097772	0.00439	1.92×10^{-6}	0.829013	1.73×10^{-6}	2.6×10^{-6}	0.003162	1.73×10^{-6}	1.73×10^{-6}	6.98×10^{-6}	2.13×10^{-6}
	T+: 460, T-: 5	T+: 152, T-: 313	T+: 371, T-: 94	T+: 464, T-: 1	T+: 222, T-: 243	T+: 465, T-: 0	T+: 461, T-: 4	T+: 376, T-: 89	T+: 465, T-: 0	T+: 465, T-: 0	T+: 451, T-: 14	T+: 463, T-: 2
F8	0.007731	0.003379	0.135908	2.6×10^{-6}	0.002105	1.73×10^{-6}	2.84×10^{-5}	0.001382	1.73×10^{-6}	1.92×10^{-6}	0.00873	0.658331
	T+: 362, T-: 103	T+: 90, T-: 375	T+: 305, T-: 160	T+: 461, T-: 4	T+: 83, T-: 382	T+: 465, T-: 0	T+: 436, T-: 29	T+: 388, T-: 77	T+: 465, T-: 0	T+: 464, T-: 1	T+: 360, T-: 105	T+: 211, T-: 254
F9	1.73×10^{-6}	0.047162	2.35×10^{-6}	2.13×10^{-6}	0.765519	1.73×10^{-6}	1.73×10^{-6}	0.000222	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T-: 0	T+: 136, T-: 329	T+: 462, T-: 3	T+: 463, T-: 2	T+: 247, T-: 218	T+: 465, T-: 0	T+: 465, T-: 0	T+: 412, T-: 53	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0	T+: 465, T-: 0
F10	0.016566	0.152861	0.14139	0.002255	0.097772	0.003379	0.000283	0.000332	1.73×10^{-6}	0.059836	0.012453	4.45×10^{-5}
	T+: 349, T-: 116	T+: 302, T-: 163	T+: 304, T-: 161	T+: 381, T-: 84	T+: 313, T-: 152	T+: 375, T-: 90	T+: 409, T-: 56	T+: 407, T-: 58	T+: 465, T-: 0	T+: 324, T-: 141	T+: 354, T-: 111	T+: 431, T-: 34
F11	0.075213	0.008217	0.557743	0.000174	0.001197	2.88×10^{-6}	8.19×10^{-5}	0.002585	1.73×10^{-6}	0.00016	9.71×10^{-5}	0.00049
	T+: 319, T-: 146	T+: 104, T-: 361	T+: 261, T-: 204	T+: 415, T-: 50	T+: 390, T-: 75	T+: 460, T-: 5	T+: 424, T-: 41	T+: 379, T-: 86	T+: 465, T-: 0	T+: 416, T-: 49	T+: 422, T-: 43	T+: 402, T-: 63
F12	2.13×10^{-6}	0.382034	0.001709	2.35×10^{-6}	0.110926	1.73×10^{-6}	1.92×10^{-6}	0.000283	1.73×10^{-6}	1.49×10^{-5}	8.92×10^{-5}	1.92×10^{-6}
	T+: 463, T-: 2	T+: 275, T-: 190	T+: 385, T-: 80	T+: 462, T-: 3	T+: 155, T-: 310	T+: 465, T-: 0	T+: 464, T-: 1	T+: 409, T-: 56	T+: 465, T-: 0	T+: 443, T-: 22	T+: 423, T-: 42	T+: 464, T-: 1
Total	+:10, -:0, =:2	+:1, -:5, =:6	+:8, -:1, =:3	+:12, -:0, =:0	+:6, -:2, =:4	+:12, -:0, =:0	+:11, -:0, =:1	+:11, -:1, =:0	+:12, -:0, =:0	+:11, -:0, =:1	+:11, -:0, =:1	+:10, -:1, =:1

Table 5: WSODE comparison results on IEEE congress on evolutionary computation 2021 with FES = 1,000 and 30 independent runs

Function		WSODE	WSO	CMAES	PSO	MVO	DO	MFO	SHIO	SDE	BAT	FOX
C1	Mean	3.45×10^{-33}	8.74×10^{-6}	2.28×10^{-54}	9.61×10^{-14}	6.18×10^{3}	4.48×10^{-3}	7.00×10^{3}	1.09×10^{-156}	1.56×10^{-2}	9.20×10^{-22}	1.48×10^{-8}
	Std	1.04×10^{-27}	1.43×10^{-5}	4.28×10^{-54}	1.17×10^{-13}	3.99×10^{3}	4.98×10^{-3}	4.83×10^{3}	3.38×10^{-156}	4.95×10^{-2}	1.29×10^{-45}	3.30×10^{-8}
	SEM	3.28×10^{-28}	4.51×10^{-6}	1.35×10^{-54}	3.72×10^{-14}	1.26×10^{3}	1.57×10^{-3}	1.53×10^{3}	1.07×10^{-156}	1.56×10^{-2}	4.07×10^{-46}	1.48×10^{-8}
	Rank	3	7	2	5	10	8	11	1	9	4	6
C2	Mean	6.77×10^{2}	1.02×10^{00}	9.13×10^{2}	9.60×10^{2}	6.88×10^{2}	5.50×10^{00}	9.83×10^{2}	1.78×10^{2}	6.80×10^{2}	7.21×10^{2}	9.81×10^{2}
	Std	2.33×10^{2}	1.54×10^{00}	5.90×10^{2}	1.61×10^{2}	3.60×10^{2}	6.21×10^{00}	2.32×10^{2}	1.83×10^{2}	3.79×10^{2}	1.96×10^{2}	2.19×10^{2}
	SEM	7.37×10^{1}	4.88×10^{-1}	1.87×10^{2}	5.09×10^{1}	1.14×10^{2}	1.96×10^{00}	7.34×10^{1}	5.78×10^{1}	1.20×10^{2}	6.21×10^{1}	9.81×10^{1}
	Rank	4	1	8	9	6	2	11	3	5	7	10
C3	Mean	3.06×10^{1}	5.73×10^{00}	3.11×10^{1}	3.26×10^{1}	3.34×10^{1}	5.52×10^{00}	3.30×10^{1}	3.83×10^{1}	2.51×10^{1}	4.24×10^{1}	3.13×10^{1}
	Std	2.37×10^{00}	1.13×10^{1}	8.20×10^{00}	5.07×10^{00}	8.01×10^{00}	8.64×10^{00}	9.44×10^{00}	1.55×10^{1}	4.45×10^{00}	6.60×10^{00}	1.95×10^{00}
	SEM	7.51×10^{-1}	3.58×10^{00}	2.59×10^{00}	1.60×10^{00}	2.53×10^{00}	2.73×10^{00}	2.98×10^{00}	4.89×10^{00}	1.41×10^{00}	2.09×10^{00}	8.71×10^{-1}
	Rank	4	2	5	7	9	1	8	10	3	11	6
C4	Mean	1.86×10^{00}	1.37×10^{00}	1.93×10^{00}	1.95×10^{00}	1.90×10^{00}	2.97×10^{-1}	1.89×10^{00}	1.93×10^{00}	1.96×10^{00}	1.93×10^{00}	1.91×10^{00}
	Std	2.34×10^{-1}	5.32×10^{-1}	3.02×10^{-1}	2.35×10^{-1}	5.66×10^{-1}	4.02×10^{-1}	7.09×10^{-1}	6.64×10^{-1}	2.90×10^{-1}	5.69×10^{-1}	4.90×10^{-1}
	SEM	7.39×10^{-2}	1.68×10^{-1}	9.55×10^{-2}	7.44×10^{-2}	1.79×10^{-1}	1.27×10^{-1}	2.24×10^{-1}	2.10×10^{-1}	9.16×10^{-2}	1.80×10^{-1}	2.19×10^{-1}
	Rank	3	2	7	10	5	1	4	8	11	9	6
C5	Mean	1.46×10^{00}	1.01×10^{00}	7.23×10^{3}	3.90×10^{2}	4.22×10^{3}	4.70×10^{00}	2.81×10^{2}	3.04×10^{00}	1.36×10^{1}	2.35×10^{1}	1.24×10^{1}
	Std	3.54×10^{00}	4.11×10^{-1}	5.31×10^{3}	3.15×10^{2}	2.54×10^{3}	5.89×10^{00}	2.51×10^{2}	5.37×10^{00}	1.39×10^{1}	3.90×10^{1}	5.64×10^{00}
	SEM	1.12×10^{00}	1.30×10^{-1}	1.68×10^{3}	9.96×10^{1}	8.02×10^{2}	1.86×10^{00}	7.95×10^{1}	1.70×10^{00}	4.41×10^{00}	1.23×10^{1}	2.52×10^{00}
	Rank	2	1	11	9	10	4	8	3	6	7	5
C6	Mean	3.32×10^{-1}	7.07×10^{-1}	4.01×10^{1}	7.09×10^{00}	4.93×10^{1}	1.21×10^{00}	2.27×10^{1}	3.42×10^{00}	1.03×10^{1}	1.10×10^{00}	8.18×10^{00}
	Std	2.96×10^{-1}	3.83×10^{-1}	3.13×10^{1}	7.78×10^{00}	5.42×10^{1}	1.46×10^{00}	1.48×10^{1}	3.38×10^{00}	9.73×10^{00}	5.64×10^{-1}	9.67×10^{00}
	SEM	9.36×10^{-2}	1.21×10^{-1}	9.89×10^{00}	2.46×10^{00}	1.71×10^{1}	4.62×10^{-1}	4.68×10^{00}	1.07×10^{00}	3.08×10^{00}	1.78×10^{-1}	4.32×10^{00}
	Rank	1	2	10	6	11	4	9	5	8	3	7
C7	Mean	3.36×10^{-1}	3.94×10^{-1}	1.96×10^{3}	1.57×10^{2}	1.45×10^{3}	4.39×10^{00}	2.70×10^{1}	5.04×10^{-1}	1.78×10^{00}	5.92×10^{00}	1.78×10^{00}
	Std	2.81×10^{-1}	1.87×10^{-1}	6.92×10^{2}	2.24×10^{2}	1.23×10^{3}	1.11×10^{1}	3.82×10^{1}	6.72×10^{-1}	1.26×10^{00}	7.88×10^{00}	6.01×10^{-1}
	SEM	8.88×10^{-2}	5.91×10^{-2}	2.19×10^{2}	7.09×10^{1}	3.88×10^{2}	3.50×10^{00}	1.21×10^{1}	2.12×10^{-1}	3.99×10^{-1}	2.49×10^{00}	2.69×10^{-1}
	Rank	1	2	11	9	10	6	8	3	5	7	4
C8	Mean	4.07×10^{-16}	8.98×10^{00}	6.56×10^{00}	7.77×10^{1}	5.48×10^{2}	1.22×10^{1}	1.47×10^{2}	1.99×10^{1}	2.57×10^{1}	1.03×10^{2}	4.04×10^{1}
	Std	5.90×10^{-16}	7.82×10^{00}	7.58×10^{00}	6.95×10^{1}	3.42×10^{2}	3.87×10^{1}	2.52×10^{2}	3.27×10^{1}	1.94×10^{1}	1.60×10^{2}	2.26×10^{1}
	SEM	1.87×10^{-16}	2.47×10^{00}	2.40×10^{00}	2.20×10^{1}	1.08×10^{2}	1.22×10^{1}	7.98×10^{1}	1.03×10^{1}	6.13×10^{00}	5.06×10^{1}	1.01×10^{1}
	Rank	1	3	2	8	11	4	10	5	6	9	7
C9	Mean	1.60×10^{-14}	1.41×10^{-3}	3.55×10^{-15}	3.13×10^{-9}	6.47×10^{-1}	4.53×10^{-8}	8.88×10^{-14}	9.77×10^{-14}	7.73×10^{-10}	5.21×10^{-1}	2.16×10^{-10}

(*Continued*)

Table 5: *Continued*

Function		WSODE	WSO	CMAES	PSO	MVO	DO	MFO	SHIO	SDE	BAT	FOX
	Std	8.16×10^{-15}	4.40×10^{-3}	4.59×10^{-15}	4.11×10^{-9}	1.39×10^{00}	2.84×10^{-8}	0.00×10^{00}	2.81×10^{-15}	8.86×10^{-10}	1.65×10^{00}	3.58×10^{-10}
	SEM	2.58×10^{-15}	1.39×10^{-3}	1.45×10^{-15}	1.30×10^{-9}	4.38×10^{-1}	8.99×10^{-9}	0.00×10^{00}	8.88×10^{-16}	2.80×10^{-10}	5.21×10^{-1}	1.60×10^{-10}
	Rank	2	9	1	7	11	8	3	4	6	10	5
C10	Mean	4.90×10^{1}	4.93×10^{1}	4.92×10^{1}	6.06×10^{1}	4.91×10^{1}	5.19×10^{1}	4.98×10^{1}	5.46×10^{1}	4.91×10^{1}	5.00×10^{1}	4.90×10^{1}
	Std	2.11×10^{-1}	7.45×10^{-1}	1.91×10^{-1}	1.95×10^{1}	5.48×10^{-1}	2.16×10^{1}	2.19×10^{00}	1.03×10^{1}	2.36×10^{-1}	3.06×10^{00}	1.90×10^{-1}
	SEM	6.66×10^{-2}	2.36×10^{-1}	6.05×10^{-2}	6.18×10^{00}	1.73×10^{-1}	6.85×10^{00}	6.93×10^{-1}	3.25×10^{00}	7.47×10^{-2}	9.68×10^{-1}	8.49×10^{-2}
	Rank	1	6	5	11	4	9	7	10	3	8	2

Table 6: WSODE Wilcoxon Signed rank results on IEEE Congress on Evolutionary Computation 2021 with FES = 1,000 and 30 independent runs

Function	WSO	CSAES	PSO	MVO	DO	MFO	SHIO	SDE	BAT	FOX
F1	0.478125	0.001713	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	0.001507	0.000103	8.86×10^{-5}	0.000189	0.00078
	T+: 124, T−: 86	T+: 189, T−: 21	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 190, T−: 20	T+: 209, T−: 1	T+: 210, T−: 0	T+: 205, T−: 5	T+: 195, T−: 15
F2	0.002821	8.86×10^{-5}	8.86×10^{-5}	0.079322	8.86×10^{-5}	0.000189	0.005734	8.86×10^{-5}	8.86×10^{-5}	0.000103
	T+: 185, T−: 25	T+: 210, T−: 0	T+: 210, T−: 0	T+: 152, T−: 58	T+: 210, T−: 0	T+: 205, T−: 5	T+: 179, T−: 31	T+: 210, T−: 0	T+: 210, T−: 0	T+: 209, T−: 1
F3	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	0.000103	8.86×10^{-5}	8.86×10^{-5}	0.000103	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}
	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 209, T−: 1	T+: 210, T−: 0	T+: 210, T−: 0	T+: 209, T−: 1	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0
F4	0.411465	0.006425	0.00014	0.601213	8.86×10^{-5}	0.575486	0.262722	8.86×10^{-5}	0.000103	0.295878
	T+: 127, T−: 83	T+: 32, T−: 178	T+: 207, T−: 3	T+: 119, T−: 91	T+: 210, T−: 0	T+: 90, T−: 120	T+: 135, T−: 75	T+: 210, T−: 0	T+: 209, T−: 1	T+: 133, T−: 77
F5	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	0.000103	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}
	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 209, T−: 1	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0
F6	0.156004	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}
	T+: 143, T−: 67	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0
F7	0.000681	0.00014	8.86×10^{-5}	0.167184	8.86×10^{-5}	8.86×10^{-5}	0.156004	8.86×10^{-5}	8.86×10^{-5}	0.00014
	T+: 196, T−: 14	T+: 207, T−: 3	T+: 210, T−: 0	T+: 142, T−: 68	T+: 210, T−: 0	T+: 210, T−: 0	T+: 143, T−: 67	T+: 210, T−: 0	T+: 210, T−: 0	T+: 207, T−: 3
F8	0.001507	0.108427	8.86×10^{-5}	0.881293	8.86×10^{-5}	8.86×10^{-5}	0.000593	8.86×10^{-5}	8.86×10^{-5}	0.000681
	T+: 190, T−: 20	T+: 148, T−: 62	T+: 210, T−: 0	T+: 101, T−: 109	T+: 210, T−: 0	T+: 210, T−: 0	T+: 197, T−: 13	T+: 210, T−: 0	T+: 210, T−: 0	T+: 196, T−: 14
F9	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}	0.007189	8.86×10^{-5}	8.86×10^{-5}	0.001019	8.86×10^{-5}	8.86×10^{-5}	8.86×10^{-5}
	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0	T+: 177, T−: 33	T+: 210, T−: 0	T+: 210, T−: 0	T+: 193, T−: 17	T+: 210, T−: 0	T+: 210, T−: 0	T+: 210, T−: 0
F10	0.217957	0.167184	0.002495	0.125859	0.601213	0.002821	0.331723	0.000103	0.135357	0.000681
	T+: 138, T−: 72	T+: 142, T−: 68	T+: 186, T−: 24	T+: 146, T−: 64	T+: 91, T−: 119	T+: 185, T−: 25	T+: 131, T−: 79	T+: 209, T−: 1	T+: 145, T−: 65	T+: 196, T−: 14
Total	+:6, −:0, =:4	+:7, −:1, =:2	+:10, −:0, =:0	+:5, −:0, =:5	+:9, −:0, =:1	+:9, −:0, =:1	+:7, −:0, =:3	+:10, −:0, =:0	+:9, −:0, =:1	+:9, −:0, =:1

with a STD of 1.04×10^{-27}, ranking third overall. This performance demonstrates WSODE's strong convergence capability, as it reaches near-global optima, though WSO and MFO achieve slightly better results on this function.

In Function C2, a unimodal function with added separability, WSODE ranks fourth with a mean value of 6.77×10^{2} and a STD of 2.33×10^{2}. The separable nature of C2 allows algorithms to optimize variables independently, and WSODE's competitive ranking highlights its proficiency in tackling such challenges, despite the close competition with WSO and CMAES.

Function C3 presents a multi-modal, deceptive landscape, introducing numerous local optima to test an algorithm's exploration efficiency. WSODE ranks fourth with a mean value of 3.06×10^{1} and a STD of 2.37, showing its capability to maintain a strong balance between exploration and exploitation, even though WSO and MFO exhibit slightly better results in this context. In Function C4, which adds ruggedness to the multi-modal landscape, WSODE ranks third with a mean value of 1.86 and a STD of 2.34×10^{-1}, underscoring its ability to handle rugged, complex search spaces where robust exploration is necessary.

Function C5 introduces nonseparability and a high degree of variable interaction, challenging algorithms to explore global optima without decomposing the function. WSODE ranks second with a mean value of 1.46 and a STD of 3.54, outperformed only by WSO. This result reflects WSODE's effective handling of nonseparable problems, outperforming traditional algorithms like CMAES, PSO, and MFO, which tend to struggle with such intricacies.

For Function C6, a deceptive and highly rugged multi-modal function, WSODE ranks first with a mean value of 3.32×10^{-1} and a STD of 2.96×10^{-1}. This result emphasizes WSODE's superior exploration abilities, as it effectively navigates the deceptive traps to locate high-quality solutions. Similarly, for Function C7, which features complex landscapes with narrow, deep basins, WSODE also ranks first with a mean value of 3.36×10^{-1} and a STD of 2.81×10^{-1}. This performance highlights WSODE's robustness in locating optima in tightly constrained areas of the search space.

Function C8, a high-dimensional, multimodal landscape with numerous local optima, poses a challenge for exploitation-dominant algorithms. WSODE achieves the best performance with a mean value of 4.07×10^{-16} and a STD of 5.90×10^{-16}, outperforming all other algorithms. This result underscores WSODE's excellent balance of exploration and exploitation, as it avoids becoming trapped in local optima and successfully converges to global solutions.

In Function C9, which introduces deceptive features to test resilience against premature convergence, WSODE ranks second with a mean value of 1.60×10^{-14} and a STD of 8.16×10^{-15}. Although CMAES performs slightly better, WSODE's close ranking highlights its robust adaptability to deceptive optimization challenges, maintaining high-quality results consistently across functions.

Finally, Function C10, a nonseparable, high-dimensional problem with deceptive traps and ruggedness, represents one of the most challenging functions in the CEC2021 suite. WSODE ranks first with a mean value of 4.90×10^{1} and a STD of 2.11×10^{-1}, demonstrating its strong optimization capabilities in handling complex, rugged, and deceptive landscapes.

As shown in Table 6, the Wilcoxon signed-rank test results demonstrate that the WSODE optimizer consistently achieves significant performance advantages over other optimizers on the CEC 2021 benchmark set. WSODE secures a majority of significant wins across comparisons, achieving flawless or near-flawless results against several algorithms, including WOA, BOA, and OHO, with no losses. While WSODE demonstrates robustness across all comparisons, it encounters minimal challenges from GWO and MFO, which manage to achieve a single loss and several ties, respectively.

5.4 WSODE comparison results on IEEE congress on evolutionary computation 2017

The performance of WSODE was evaluated against several prominent optimizers, including WSO, CMAES, COA, reptile search algorithm (RSA), biogeography-based optimization (BBO), african vultures optimization algorithm (AVOA), SDE, SCA, WOA, DO, MFO, SHIO, and AOA, on the benchmark functions provided by the IEEE CEC 2017 competition (F1–F15). As detailed in Table 7, WSODE exhibits superior performance across a majority of the tested functions, achieving the lowest mean values in numerous cases, which underscores its high optimization efficacy.

Table 7: WSODE comparison results on IEEE congress on evolutionary computation 2017 (F1–F15) with FES = 1,000 and 30 independent runs

Function	Statistics	WSODE	WSO	DE	CMAES	COA	RSA	BBO	AVOA	SDE	SCA	WOA	DO	MFO	SHIO	AOA
F1	Mean	1.00×10^{2}	1.20×10^{2}	1.14×10^{2}	4.44×10^{9}	4.01×10^{3}	1.54×10^{10}	3.25×10^{9}	6.10×10^{9}	5.19×10^{9}	8.16×10^{8}	6.97×10^{5}	3.65×10^{3}	1.17×10^{7}	6.81×10^{7}	1.61×10^{10}
	Std	8.99×10^{-15}	4.50×10^{1}	2.49×10^{1}	1.25×10^{9}	1.74×10^{3}	3.01×10^{9}	7.52×10^{8}	2.81×10^{9}	3.46×10^{9}	2.48×10^{8}	5.02×10^{5}	3.23×10^{3}	2.86×10^{7}	1.58×10^{8}	3.37×10^{9}
	SEM	3.67×10^{-15}	1.84×10^{1}	1.44×10^{1}	5.10×10^{8}	7.09×10^{2}	1.35×10^{9}	3.36×10^{8}	1.26×10^{9}	1.55×10^{9}	1.01×10^{8}	2.05×10^{5}	1.32×10^{3}	1.17×10^{7}	6.47×10^{7}	1.51×10^{9}
	Rank	1	3	2	11	5	14	10	13	12	9	6	4	7	8	15
F2	Mean	2.00×10^{2}	2.00×10^{2}	2.01×10^{2}	2.83×10^{11}	2.00×10^{2}	1.08×10^{13}	1.39×10^{11}	1.24×10^{11}	1.78×10^{11}	5.22×10^{7}	2.55×10^{4}	2.00×10^{2}	5.37×10^{7}	1.33×10^{7}	2.96×10^{13}
	Std	8.84×10^{-10}	4.25×10^{-6}	2.42×10^{00}	2.44×10^{11}	1.38×10^{-3}	1.27×10^{13}	2.26×10^{11}	1.69×10^{11}	2.28×10^{11}	4.40×10^{7}	2.15×10^{4}	2.48×10^{-3}	1.27×10^{8}	2.06×10^{7}	4.21×10^{13}
	SEM	3.61×10^{-10}	1.73×10^{-6}	1.40×10^{00}	9.98×10^{10}	5.62×10^{-4}	5.68×10^{12}	1.01×10^{11}	7.55×10^{10}	1.02×10^{11}	1.80×10^{7}	8.77×10^{3}	1.01×10^{-3}	5.17×10^{7}	8.40×10^{6}	1.88×10^{13}
	Rank	1	2	5	13	3	14	11	10	12	8	6	4	9	7	15
F3	Mean	3.00×10^{2}	3.00×10^{2}	4.03×10^{2}	3.42×10^{4}	3.03×10^{2}	1.70×10^{4}	3.59×10^{4}	4.21×10^{4}	2.58×10^{4}	1.51×10^{3}	1.26×10^{3}	3.00×10^{2}	2.42×10^{3}	5.81×10^{3}	3.94×10^{4}
	Std	2.54×10^{-14}	8.62×10^{-2}	4.23×10^{1}	1.23×10^{4}	3.58×10^{00}	2.33×10^{3}	7.81×10^{3}	2.12×10^{4}	6.12×10^{3}	6.53×10^{2}	6.48×10^{2}	4.27×10^{-3}	4.22×10^{3}	3.95×10^{3}	2.69×10^{4}
	SEM	1.04×10^{-14}	3.52×10^{-2}	1.03×10^{2}	5.01×10^{3}	1.46×10^{00}	1.04×10^{3}	3.49×10^{3}	9.48×10^{3}	2.74×10^{3}	2.67×10^{2}	2.64×10^{2}	1.74×10^{-3}	1.72×10^{3}	1.61×10^{3}	1.20×10^{4}
	Rank	1	3	5	12	4	10	13	15	11	7	6	2	8	9	14
F4	Mean	4.01×10^{2}	4.02×10^{2}	4.06×10^{2}	6.08×10^{2}	4.04×10^{2}	1.86×10^{3}	6.07×10^{2}	9.49×10^{2}	9.50×10^{2}	4.36×10^{2}	4.27×10^{2}	4.05×10^{2}	4.29×10^{2}	4.19×10^{2}	1.41×10^{3}
	Std	5.60×10^{-1}	1.55×10^{00}	6.81×10^{-1}	1.22×10^{2}	1.30×10^{00}	3.67×10^{2}	7.56×10^{1}	3.71×10^{2}	1.65×10^{2}	3.37×10^{00}	4.24×10^{1}	7.43×10^{-1}	2.77×10^{1}	3.01×10^{1}	4.69×10^{2}
	SEM	2.29×10^{-1}	6.31×10^{-1}	6.18×10^{00}	4.99×10^{1}	5.31×10^{-1}	1.64×10^{2}	3.38×10^{1}	1.66×10^{2}	7.36×10^{1}	1.38×10^{00}	1.73×10^{1}	3.03×10^{-1}	1.13×10^{1}	1.23×10^{1}	2.10×10^{2}
	Rank	1	2	5	11	3	15	10	12	13	9	7	4	8	6	14
F5	Mean	5.15×10^{2}	5.11×10^{2}	5.11×10^{2}	5.68×10^{2}	5.15×10^{2}	6.36×10^{2}	5.71×10^{2}	6.02×10^{2}	6.06×10^{2}	5.47×10^{2}	5.56×10^{2}	5.34×10^{2}	5.31×10^{2}	5.28×10^{2}	6.21×10^{2}
	Std	9.27×10^{00}	7.32×10^{00}	2.03×10^{00}	1.82×10^{1}	8.85×10^{00}	2.01×10^{1}	4.36×10^{00}	2.16×10^{1}	1.29×10^{1}	9.57×10^{00}	9.63×10^{00}	1.25×10^{1}	2.10×10^{1}	7.67×10^{00}	1.48×10^{1}
	SEM	3.79×10^{00}	2.99×10^{00}	1.07×10^{1}	7.44×10^{00}	3.61×10^{00}	8.98×10^{00}	1.95×10^{00}	9.66×10^{00}	5.76×10^{00}	3.91×10^{00}	3.93×10^{00}	5.10×10^{00}	8.56×10^{00}	3.13×10^{00}	6.62×10^{00}
	Rank	3	2	1	10	4	15	11	12	13	8	9	7	6	5	14
F6	Mean	6.00×10^{2}	6.01×10^{2}	6.01×10^{2}	6.30×10^{2}	6.03×10^{2}	6.65×10^{2}	6.38×10^{2}	6.47×10^{2}	6.46×10^{2}	6.15×10^{2}	6.42×10^{2}	6.07×10^{2}	6.01×10^{2}	6.04×10^{2}	6.62×10^{2}
	Std	1.66×10^{-10}	7.60×10^{-1}	1.05×10^{1}	2.92×10^{1}	6.70×10^{00}	1.03×10^{1}	1.12×10^{1}	1.78×10^{1}	1.56×10^{1}	3.99×10^{00}	1.25×10^{1}	6.59×10^{00}	5.68×10^{-1}	4.82×10^{00}	6.06×10^{00}
	SEM	6.78×10^{-11}	3.10×10^{-1}	1.11×10^{1}	1.19×10^{1}	2.74×10^{00}	4.62×10^{00}	5.02×10^{00}	7.98×10^{00}	6.96×10^{00}	1.63×10^{00}	5.11×10^{00}	2.69×10^{00}	2.32×10^{-1}	1.97×10^{00}	2.71×10^{00}
	Rank	1	3	3	9	5	15	10	13	12	8	11	7	2	6	14
F7	Mean	7.28×10^{2}	7.24×10^{2}	7.32×10^{2}	7.34×10^{2}	7.62×10^{2}	8.26×10^{2}	8.59×10^{2}	8.69×10^{2}	8.59×10^{2}	7.71×10^{2}	7.83×10^{2}	7.51×10^{2}	7.41×10^{2}	7.39×10^{2}	8.78×10^{2}
	Std	5.34×10^{00}	9.55×10^{-1}	5.43×10^{00}	3.84×10^{00}	2.05×10^{1}	1.32×10^{1}	2.09×10^{1}	2.91×10^{1}	3.36×10^{1}	9.68×10^{00}	2.95×10^{1}	1.26×10^{1}	2.09×10^{1}	1.73×10^{1}	2.71×10^{1}
	SEM	2.18×10^{00}	3.90×10^{-1}	2.22×10^{1}	1.57×10^{00}	8.38×10^{00}	5.88×10^{00}	9.36×10^{00}	1.30×10^{1}	1.50×10^{1}	3.95×10^{00}	1.21×10^{1}	5.16×10^{00}	8.51×10^{00}	7.07×10^{00}	1.21×10^{1}
	Rank	2	1	3	4	8	11	13	14	12	9	10	7	6	5	15
F8	Mean	8.09×10^{2}	8.05×10^{2}	8.11×10^{2}	8.19×10^{2}	8.23×10^{2}	8.77×10^{2}	8.74×10^{2}	8.85×10^{2}	8.84×10^{2}	8.39×10^{2}	8.26×10^{2}	8.26×10^{2}	8.25×10^{2}	8.23×10^{2}	8.90×10^{2}
	Std	8.79×10^{00}	1.16×10^{00}	2.06×10^{00}	9.06×10^{00}	7.69×10^{00}	1.07×10^{1}	4.89×10^{00}	9.47×10^{00}	1.57×10^{1}	8.91×10^{00}	1.06×10^{1}	9.13×10^{00}	7.23×10^{00}	1.05×10^{1}	2.25×10^{1}
	SEM	3.59×10^{00}	4.73×10^{-1}	1.06×10^{1}	3.70×10^{00}	3.14×10^{00}	4.80×10^{00}	2.19×10^{00}	4.23×10^{00}	7.00×10^{00}	3.64×10^{00}	4.33×10^{00}	3.73×10^{00}	2.95×10^{00}	4.27×10^{00}	1.01×10^{1}
	Rank	2	1	3	4	6	12	11	14	13	10	9	8	7	5	15
F9	Mean	9.00×10^{2}	9.01×10^{2}	9.00×10^{2}	9.00×10^{2}	9.02×10^{2}	2.25×10^{3}	1.77×10^{3}	2.42×10^{3}	1.97×10^{3}	9.98×10^{2}	1.14×10^{3}	9.00×10^{2}	9.31×10^{2}	1.09×10^{3}	2.53×10^{3}
	Std	0.00×10^{00}	1.37×10^{00}	0.00×10^{00}	0.00×10^{00}	3.59×10^{00}	5.61×10^{2}	4.40×10^{2}	1.36×10^{3}	2.73×10^{2}	5.30×10^{1}	2.28×10^{2}	1.82×10^{-1}	6.84×10^{1}	2.31×10^{2}	1.17×10^{3}
	SEM	0.00×10^{00}	5.60×10^{-1}	0.00×10^{00}	0.00×10^{00}	1.47×10^{00}	2.51×10^{2}	1.97×10^{2}	6.07×10^{2}	1.22×10^{2}	2.16×10^{1}	9.31×10^{1}	7.42×10^{-2}	2.79×10^{1}	9.41×10^{1}	5.25×10^{2}

(*Continued*)

Table 7: *Continued*

Function	Statistics	WSODE	WSO	DE	CMAES	COA	RSA	BBO	AVOA	SDE	SCA	WOA	DO	MFO	SHIO	AOA
	Rank	1	5	1	1	6	13	11	14	12	8	10	4	7	9	15
F10	Mean	1.93×10^{3}	1.58×10^{3}	1.68×10^{3}	2.68×10^{3}	1.73×10^{3}	3.45×10^{3}	3.32×10^{3}	3.05×10^{3}	3.20×10^{3}	2.36×10^{3}	2.11×10^{3}	1.78×10^{3}	1.77×10^{3}	2.05×10^{3}	3.36×10^{3}
	Std	3.23×10^{2}	4.01×10^{2}	1.45×10^{2}	2.47×10^{2}	1.78×10^{2}	2.00×10^{2}	3.04×10^{2}	3.05×10^{2}	2.41×10^{2}	1.85×10^{2}	5.53×10^{2}	2.75×10^{2}	3.60×10^{2}	2.64×10^{2}	3.41×10^{2}
	SEM	1.32×10^{2}	1.64×10^{2}	5.80×10^{2}	1.01×10^{2}	7.28×10^{1}	8.95×10^{1}	1.36×10^{2}	1.36×10^{2}	1.08×10^{2}	7.53×10^{1}	2.26×10^{2}	1.12×10^{2}	1.47×10^{2}	1.08×10^{2}	1.52×10^{2}
	Rank	6	1	2	10	3	15	13	11	12	9	8	5	4	7	14
F11	Mean	1.10×10^{3}	1.11×10^{3}	1.10×10^{3}	1.44×10^{3}	1.18×10^{3}	1.52×10^{4}	2.53×10^{3}	4.30×10^{3}	2.36×10^{3}	1.21×10^{3}	1.18×10^{3}	1.13×10^{3}	1.17×10^{3}	1.17×10^{3}	5.85×10^{3}
	Std	5.47×10^{-1}	6.33×10^{00}	1.48×10^{00}	3.08×10^{2}	8.08×10^{1}	1.45×10^{4}	8.30×10^{2}	2.05×10^{3}	8.93×10^{2}	2.40×10^{1}	3.83×10^{1}	1.23×10^{1}	1.17×10^{2}	2.87×10^{1}	1.48×10^{3}
	SEM	2.23×10^{-1}	2.58×10^{00}	2.55×10^{00}	1.26×10^{2}	3.30×10^{1}	6.50×10^{3}	3.71×10^{2}	9.17×10^{2}	3.99×10^{2}	9.81×10^{00}	1.56×10^{1}	5.01×10^{00}	4.76×10^{1}	1.17×10^{1}	6.62×10^{2}
	Rank	1	3	2	10	7	15	12	13	11	9	8	4	5	6	14
F12	Mean	1.25×10^{3}	1.64×10^{3}	1.60×10^{4}	1.07×10^{8}	1.74×10^{4}	2.84×10^{8}	2.31×10^{8}	3.89×10^{8}	1.47×10^{8}	8.19×10^{6}	5.10×10^{6}	2.40×10^{5}	2.75×10^{6}	5.06×10^{5}	1.31×10^{9}
	Std	6.96×10^{1}	1.37×10^{2}	2.07×10^{3}	8.48×10^{7}	2.10×10^{4}	2.09×10^{8}	1.13×10^{8}	2.42×10^{8}	6.52×10^{7}	4.26×10^{6}	6.64×10^{6}	2.95×10^{5}	4.23×10^{6}	4.56×10^{5}	5.28×10^{8}
	SEM	2.84×10^{1}	5.60×10^{1}	1.48×10^{4}	3.46×10^{7}	8.55×10^{3}	9.33×10^{7}	5.05×10^{7}	1.08×10^{8}	2.92×10^{7}	1.74×10^{6}	2.71×10^{6}	1.20×10^{5}	1.73×10^{6}	1.86×10^{5}	2.36×10^{8}
	Rank	1	2	3	10	4	13	12	14	11	9	8	5	7	6	15
F13	Mean	1.30×10^{3}	1.32×10^{3}	1.31×10^{3}	4.64×10^{5}	4.37×10^{3}	4.91×10^{7}	5.68×10^{6}	8.95×10^{5}	7.25×10^{6}	3.27×10^{4}	2.03×10^{4}	1.03×10^{4}	1.33×10^{4}	1.15×10^{4}	1.05×10^{8}
	Std	1.74×10^{00}	5.97×10^{00}	1.88×10^{00}	1.04×10^{6}	2.60×10^{3}	3.56×10^{7}	7.37×10^{6}	9.03×10^{5}	7.69×10^{6}	2.28×10^{4}	1.67×10^{4}	9.22×10^{3}	1.22×10^{4}	6.30×10^{3}	1.18×10^{8}
	SEM	7.12×10^{-1}	2.44×10^{00}	9.61×10^{00}	4.25×10^{5}	1.06×10^{3}	1.59×10^{7}	3.29×10^{6}	4.04×10^{5}	3.44×10^{6}	9.32×10^{3}	6.81×10^{3}	3.76×10^{3}	5.00×10^{3}	2.57×10^{3}	5.26×10^{7}
	Rank	1	3	2	10	4	14	12	11	13	9	8	5	7	6	15
F14	Mean	1.40×10^{3}	1.42×10^{3}	1.40×10^{3}	4.73×10^{3}	1.56×10^{3}	3.22×10^{5}	2.66×10^{4}	8.05×10^{3}	4.39×10^{4}	1.63×10^{3}	2.26×10^{3}	1.52×10^{3}	2.65×10^{3}	2.78×10^{3}	5.90×10^{5}
	Std	7.89×10^{00}	2.09×10^{00}	6.06×10^{-1}	2.33×10^{3}	9.17×10^{1}	3.40×10^{5}	2.41×10^{4}	5.82×10^{3}	4.99×10^{4}	1.45×10^{2}	1.45×10^{3}	1.31×10^{2}	5.18×10^{2}	1.91×10^{3}	6.84×10^{5}
	SEM	3.22×10^{00}	8.52×10^{-1}	4.81×10^{-1}	9.53×10^{2}	3.74×10^{1}	1.52×10^{5}	1.08×10^{4}	2.60×10^{3}	2.23×10^{4}	5.90×10^{1}	5.91×10^{2}	5.34×10^{1}	2.11×10^{2}	7.80×10^{2}	3.06×10^{5}
	Rank	1	3	2	10	5	14	12	11	13	6	7	4	8	9	15
F15	Mean	1.50×10^{3}	1.51×10^{3}	1.50×10^{3}	4.42×10^{3}	1.96×10^{3}	9.26×10^{4}	6.29×10^{4}	1.98×10^{4}	4.17×10^{5}	2.30×10^{3}	6.52×10^{3}	2.25×10^{3}	4.06×10^{3}	6.26×10^{3}	3.18×10^{6}
	Std	2.43×10^{-1}	1.37×10^{1}	3.28×10^{-1}	8.64×10^{2}	4.07×10^{2}	1.01×10^{5}	6.11×10^{4}	9.49×10^{3}	6.20×10^{5}	7.66×10^{2}	3.24×10^{3}	1.36×10^{3}	2.27×10^{3}	8.60×10^{3}	4.77×10^{6}
	SEM	9.90×10^{-2}	5.59×10^{00}	9.72×10^{-1}	3.53×10^{2}	1.66×10^{2}	4.53×10^{4}	2.73×10^{4}	4.24×10^{3}	2.77×10^{5}	3.13×10^{2}	1.32×10^{3}	5.56×10^{2}	9.28×10^{2}	3.51×10^{3}	2.13×10^{6}
	Rank	1	3	2	8	4	13	12	11	14	6	10	5	7	9	15

For instance, WSODE consistently ranks first or second on functions F1–F4, F6, F9, and F1–F15, demonstrating its ability to find optimal or near-optimal solutions with outperforming precision. The low STDs and standard errors associated with WSODE's results indicate a high level of stability and reliability, as the algorithm produces consistent outputs across multiple runs. Specifically, the mean value for function F1 using WSODE is 1.00×10^2 with an almost negligible STD, while other algorithms like CMAES and COA exhibit significantly higher mean values and larger variances, indicating less stable performance. Furthermore, the comparative analysis reveals that while some traditional algorithms like CMAES occasionally deliver competitive results, they generally fall behind WSODE due to higher mean optimization values and greater variability. For example, in function F3, CMAES has a mean value of 3.42×10^4 with a high STD, compared to WSODE's mean of 3.00×10^2 with minimal variation, highlighting WSODE's superior performance. Additionally, COA and RSA often show substantial performance degradation, particularly in functions with complex, multimodal landscapes. Moreover, WSODE's performance is notable for its robustness across different types of optimization problems. It excels not only in unimodal functions but also in complex multimodal, hybrid, and composition functions, reflecting its versatile optimization capability. The consistently high ranks across diverse benchmark functions affirm WSODE's adaptability and effectiveness.

Moreover, the performance of WSODE on the benchmark functions F16–F30, as outlined in Table 8, showcases its robustness, whereas, it consistently achieves high ranks across these functions, reflecting its effective optimization capabilities. For functions like F16 and F17, WSODE secures the first rank with the lowest mean values and relatively low STDs, indicating stable and reliable performance. In function F16, WSODE's mean value is 1.60×10^3 with a STD of 4.61, outperforming other optimizers such as CMAES and RSA, which show higher mean values and greater variability. Similarly, in function F17, WSODE's mean is 1.72×10^3, maintaining its lead with minimal variation. In more complex functions like F18 and F19, WSODE continues to demonstrate its superiority by maintaining low mean values compared to other algorithms. For instance, in function F18, WSODE achieves a mean of 1.80×10^3 with a STD of 8.20, significantly outperforming algorithms like CMAES and RSA, which exhibit much higher mean values and STDs, highlighting WSODE's consistency in handling complex optimization landscapes.

WSODE also excels in functions F20–F23, often ranking in the top positions. For example, in function F20, WSODE ranks first with a mean value of 2.00×10^3, while in function F21, it maintains a competitive edge with a mean of 2.29×10^3. These results indicate that WSODE can effectively navigate various optimization challenges, providing reliable solutions with low variability. In functions F24–F27, WSODE's performance remains robust, frequently ranking among the top algorithms. For instance, in function F24, WSODE secures the second rank with a mean value of 2.70×10^3, demonstrating its ability to outperform other methods like CMAES and COA. Similarly, in function F26, WSODE's mean value of 2.91×10^3 and low STD underscore its optimization strength. The evaluation of WSODE on functions F28–F30 further confirms its effectiveness. In function F28, WSODE ranks third with a mean value of 3.25×10^3, while in function F30, it achieves the first rank with a mean of 3.58×10^3, significantly outperforming other algorithms which display higher mean values and greater STDs.

As shown in Table 9, the Wilcoxon signed-rank test results for the CEC 2017 benchmark set reveal that the WSODE optimizer consistently demonstrates superior performance over several compared optimization algorithms. WSODE achieves dominant results against most algorithms, notably outperforming WOA and BOA with near-perfect outcomes, underscoring its significant advantage. Additionally, WSODE displays a strong competitive edge over SHIO and COA, securing substantial wins with minimal or no losses. The optimizer also achieves flawless performance against OHO and SCA, winning all comparisons without ties or losses. Although WSODE faces some challenges from GWO, MFO, and SHO, which manage to achieve a few losses or ties.

5.5 WSODE convergence curve

The convergence curves for functions F1–F9 over CEC2022 as shown in Figure 2, demonstrate the robust and consistent performance of WSODE. The curves generally show a steady decline in the best value obtained so far, indicating continuous progress towards optimal solutions. Functions F1, F5, and F9 exhibit smooth and

Table 8: WSODE comparison results on IEEE congress on evolutionary computation 2017 (F16–F30) with FES = 1,000 and 30 independent runs

Function	Statistics	WSODE	WSO	DE	CMAES	COA	RSA	BBO	AVOA	SDE	SCA	WOA	DO	MFO	SHIO	AOA
F16	Mean	1.60×10^{3}	1.60×10^{3}	1.61×10^{3}	2.06×10^{3}	1.61×10^{3}	2.38×10^{3}	2.14×10^{3}	2.23×10^{3}	2.27×10^{3}	1.69×10^{3}	1.82×10^{3}	1.75×10^{3}	1.70×10^{3}	1.81×10^{3}	2.40×10^{3}
	Std	4.61×10^{00}	2.80×10^{00}	3.48×10^{-1}	5.65×10^{1}	3.97×10^{00}	1.84×10^{2}	1.23×10^{2}	6.71×10^{1}	1.03×10^{2}	4.56×10^{1}	1.08×10^{2}	1.22×10^{2}	6.13×10^{1}	1.45×10^{2}	1.95×10^{2}
	SEM	1.88×10^{00}	1.14×10^{00}	8.67×10^{-1}	2.31×10^{1}	1.62×10^{00}	8.23×10^{1}	5.51×10^{1}	3.00×10^{1}	4.59×10^{1}	1.86×10^{1}	4.42×10^{1}	4.99×10^{1}	2.50×10^{1}	5.93×10^{1}	8.73×10^{1}
	Rank	1	2	3	10	4	14	11	12	13	5	9	7	6	8	15
F17	Mean	1.72×10^{3}	1.75×10^{3}	1.74×10^{3}	1.78×10^{3}	1.73×10^{3}	2.12×10^{3}	2.07×10^{3}	2.00×10^{3}	2.05×10^{3}	1.78×10^{3}	1.84×10^{3}	1.76×10^{3}	1.74×10^{3}	1.82×10^{3}	2.17×10^{3}
	Std	1.68×10^{1}	4.44×10^{00}	1.02×10^{00}	1.76×10^{1}	1.28×10^{1}	6.29×10^{1}	1.09×10^{2}	9.93×10^{1}	9.74×10^{1}	1.11×10^{1}	8.08×10^{1}	1.96×10^{1}	1.97×10^{1}	6.51×10^{1}	9.25×10^{1}
	SEM	6.85×10^{00}	1.81×10^{00}	1.15×10^{00}	7.18×10^{00}	5.24×10^{00}	2.81×10^{1}	4.89×10^{1}	4.44×10^{1}	4.36×10^{1}	4.54×10^{00}	3.30×10^{1}	7.99×10^{00}	8.05×10^{00}	2.66×10^{1}	4.14×10^{1}
	Rank	1	5	4	8	2	14	13	11	12	7	10	6	3	9	15
F18	Mean	1.80×10^{3}	1.83×10^{3}	1.81×10^{3}	8.36×10^{6}	1.16×10^{4}	2.78×10^{8}	2.71×10^{7}	4.84×10^{6}	2.93×10^{7}	1.51×10^{5}	1.05×10^{4}	2.28×10^{4}	2.38×10^{4}	1.53×10^{4}	6.68×10^{8}
	Std	8.20×10^{00}	1.42×10^{1}	1.48×10^{00}	7.71×10^{6}	9.42×10^{3}	3.01×10^{8}	3.89×10^{7}	5.80×10^{6}	1.70×10^{7}	1.15×10^{5}	1.55×10^{4}	1.57×10^{4}	1.31×10^{4}	8.99×10^{3}	4.16×10^{8}
	SEM	3.35×10^{00}	5.78×10^{00}	1.65×10^{00}	3.15×10^{6}	3.85×10^{3}	1.35×10^{8}	1.74×10^{7}	2.59×10^{6}	7.60×10^{6}	4.70×10^{4}	6.34×10^{3}	6.41×10^{3}	5.34×10^{3}	3.67×10^{3}	1.86×10^{8}
	Rank	1	3	2	11	5	14	12	10	13	9	4	7	8	6	15
F19	Mean	1.90×10^{3}	1.90×10^{3}	1.90×10^{3}	5.85×10^{4}	2.26×10^{3}	7.05×10^{6}	1.05×10^{5}	1.98×10^{5}	9.07×10^{5}	3.32×10^{3}	1.69×10^{4}	3.37×10^{3}	1.29×10^{4}	7.92×10^{3}	1.01×10^{8}
	Std	4.97×10^{-2}	1.82×10^{00}	2.56×10^{-2}	6.00×10^{4}	2.13×10^{2}	7.53×10^{6}	1.30×10^{5}	1.89×10^{5}	1.69×10^{6}	1.35×10^{3}	1.76×10^{4}	3.37×10^{3}	1.28×10^{4}	6.44×10^{3}	1.89×10^{8}
	SEM	2.03×10^{-2}	7.41×10^{-1}	1.68×10^{-2}	2.45×10^{4}	8.69×10^{1}	3.37×10^{6}	5.82×10^{4}	8.45×10^{4}	7.58×10^{5}	5.52×10^{2}	7.18×10^{3}	1.38×10^{3}	5.24×10^{3}	2.63×10^{3}	8.47×10^{7}
	Rank	1	3	2	10	4	14	11	12	13	5	9	6	8	7	15
F20	Mean	2.00×10^{3}	2.02×10^{3}	2.02×10^{3}	2.21×10^{3}	2.02×10^{3}	2.43×10^{3}	2.25×10^{3}	2.35×10^{3}	2.43×10^{3}	2.11×10^{3}	2.16×10^{3}	2.06×10^{3}	2.05×10^{3}	2.13×10^{3}	2.33×10^{3}
	Std	6.34×10^{-1}	9.29×10^{00}	1.80×10^{-1}	8.16×10^{1}	1.17×10^{1}	1.19×10^{2}	8.06×10^{1}	9.98×10^{1}	2.06×10^{1}	3.88×10^{1}	8.64×10^{1}	4.89×10^{1}	2.38×10^{1}	9.94×10^{1}	7.77×10^{1}
	SEM	2.59×10^{-1}	3.79×10^{00}	1.04×10^{-1}	3.33×10^{1}	4.77×10^{00}	5.33×10^{1}	3.60×10^{1}	4.46×10^{1}	9.21×10^{00}	1.58×10^{1}	3.53×10^{1}	2.00×10^{1}	9.73×10^{00}	4.06×10^{1}	3.47×10^{1}
	Rank	1	2	3	10	4	15	11	13	14	7	9	6	5	8	12
F21	Mean	2.29×10^{3}	2.25×10^{3}	2.30×10^{3}	2.28×10^{3}	2.28×10^{3}	2.43×10^{3}	2.37×10^{3}	2.34×10^{3}	2.37×10^{3}	2.26×10^{3}	2.34×10^{3}	2.29×10^{3}	2.30×10^{3}	2.33×10^{3}	2.34×10^{3}
	Std	4.84×10^{1}	5.95×10^{1}	2.39×10^{1}	0.00×10^{00}	5.86×10^{1}	1.88×10^{1}	1.27×10^{1}	6.62×10^{1}	8.60×10^{1}	6.48×10^{1}	6.68×10^{1}	6.39×10^{1}	6.48×10^{1}	8.20×10^{00}	4.33×10^{1}
	SEM	1.97×10^{1}	2.43×10^{1}	2.00×10^{2}	0.00×10^{00}	2.39×10^{1}	8.41×10^{00}	5.66×10^{00}	2.96×10^{1}	3.85×10^{1}	2.65×10^{1}	2.73×10^{1}	2.61×10^{1}	2.65×10^{1}	3.35×10^{00}	1.94×10^{1}
	Rank	5	1	8	4	3	15	14	10	13	2	12	6	7	9	11
F22	Mean	2.29×10^{3}	2.30×10^{3}	2.30×10^{3}	2.82×10^{3}	2.30×10^{3}	3.47×10^{3}	2.68×10^{3}	2.87×10^{3}	2.71×10^{3}	2.36×10^{3}	2.58×10^{3}	2.29×10^{3}	2.26×10^{3}	2.31×10^{3}	3.03×10^{3}
	Std	3.62×10^{1}	4.97×10^{-1}	3.05×10^{-2}	7.75×10^{2}	8.29×10^{-1}	1.81×10^{2}	1.65×10^{2}	1.75×10^{2}	1.02×10^{2}	1.88×10^{1}	6.46×10^{2}	2.80×10^{1}	4.10×10^{1}	7.22×10^{00}	2.92×10^{2}
	SEM	1.48×10^{1}	2.03×10^{-1}	1.00×10^{2}	3.16×10^{2}	3.38×10^{-1}	8.10×10^{1}	7.36×10^{1}	7.84×10^{1}	4.55×10^{1}	7.68×10^{00}	2.64×10^{2}	1.14×10^{1}	1.67×10^{1}	2.95×10^{00}	1.30×10^{2}
	Rank	2	5	4	12	6	15	10	13	11	8	9	3	1	7	14
F23	Mean	2.61×10^{3}	2.61×10^{3}	2.61×10^{3}	2.69×10^{3}	2.62×10^{3}	2.80×10^{3}	2.70×10^{3}	2.70×10^{3}	2.71×10^{3}	2.66×10^{3}	2.66×10^{3}	2.63×10^{3}	2.63×10^{3}	2.64×10^{3}	2.79×10^{3}
	Std	4.85×10^{00}	5.17×10^{00}	4.86×10^{00}	7.15×10^{00}	4.99×10^{00}	9.77×10^{1}	1.88×10^{1}	2.27×10^{1}	1.84×10^{1}	8.77×10^{00}	1.80×10^{1}	8.59×10^{00}	1.41×10^{1}	1.96×10^{1}	4.46×10^{1}
	SEM	1.98×10^{00}	2.11×10^{00}	3.15×10^{2}	2.92×10^{00}	2.04×10^{00}	4.37×10^{1}	8.41×10^{00}	1.02×10^{1}	8.22×10^{00}	3.58×10^{00}	7.34×10^{00}	3.51×10^{00}	5.77×10^{00}	8.01×10^{00}	2.00×10^{1}
	Rank	1	2	3	10	4	15	11	12	13	8	9	6	5	7	14
F24	Mean	2.70×10^{3}	2.62×10^{3}	2.75×10^{3}	2.81×10^{3}	2.75×10^{3}	2.92×10^{3}	2.81×10^{3}	2.85×10^{3}	2.83×10^{3}	2.74×10^{3}	2.79×10^{3}	2.79×10^{3}	2.71×10^{3}	2.75×10^{3}	2.95×10^{3}
	Std	9.70×10^{1}	1.30×10^{2}	2.25×10^{-1}	9.91×10^{00}	5.41×10^{00}	3.79×10^{1}	4.20×10^{1}	2.66×10^{1}	3.27×10^{1}	8.48×10^{1}	3.16×10^{1}	2.05×10^{1}	8.34×10^{1}	9.42×10^{00}	7.25×10^{1}
	SEM	3.96×10^{1}	5.29×10^{1}	3.47×10^{2}	4.05×10^{00}	2.21×10^{00}	1.69×10^{1}	1.88×10^{1}	1.19×10^{1}	1.46×10^{1}	3.46×10^{1}	1.29×10^{1}	8.37×10^{00}	3.40×10^{1}	3.85×10^{00}	3.24×10^{1}

(*Continued*)

Table 8: *Continued*

Function	Statistics	WSODE	WSO	DE	CMAES	COA	RSA	BBO	AVOA	SDE	SCA	WOA	DO	MFO	SHIO	AOA
	Rank	2	1	5	10	7	14	11	13	12	4	8	9	3	6	15
F25	Mean	2.92×10^{3}	2.91×10^{3}	2.93×10^{3}	3.16×10^{3}	2.91×10^{3}	3.51×10^{3}	3.14×10^{3}	3.46×10^{3}	3.19×10^{3}	2.97×10^{3}	2.93×10^{3}	2.93×10^{3}	2.93×10^{3}	2.96×10^{3}	3.55×10^{3}
	Std	2.61×10^{1}	2.33×10^{1}	2.62×10^{1}	6.19×10^{1}	2.34×10^{1}	2.26×10^{2}	1.17×10^{2}	2.46×10^{2}	1.49×10^{2}	8.76×10^{00}	2.68×10^{1}	2.46×10^{1}	2.51×10^{1}	3.97×10^{1}	1.98×10^{2}
	SEM	1.07×10^{1}	9.52×10^{00}	4.30×10^{2}	2.53×10^{1}	9.54×10^{00}	1.01×10^{2}	5.25×10^{1}	1.10×10^{2}	6.68×10^{1}	3.58×10^{00}	1.10×10^{1}	1.00×10^{1}	1.02×10^{1}	1.62×10^{1}	8.85×10^{1}
	Rank	3	1	4	11	2	14	10	13	12	9	6	5	7	8	15
F26	Mean	2.91×10^{3}	2.89×10^{3}	2.92×10^{3}	4.36×10^{3}	3.24×10^{3}	4.41×10^{3}	3.97×10^{3}	4.16×10^{3}	3.49×10^{3}	3.08×10^{3}	3.47×10^{3}	2.99×10^{3}	3.05×10^{3}	3.39×10^{3}	4.51×10^{3}
	Std	1.92×10^{1}	4.85×10^{1}	1.44×10^{1}	5.85×10^{2}	3.69×10^{2}	4.11×10^{2}	5.44×10^{2}	4.59×10^{2}	1.06×10^{2}	2.92×10^{1}	6.25×10^{2}	2.84×10^{2}	1.72×10^{2}	5.04×10^{2}	3.38×10^{2}
	SEM	7.86×10^{00}	1.98×10^{1}	3.15×10^{2}	2.39×10^{2}	1.51×10^{2}	1.84×10^{2}	2.43×10^{2}	2.05×10^{2}	4.73×10^{1}	1.19×10^{1}	2.55×10^{2}	1.16×10^{2}	7.01×10^{1}	2.06×10^{2}	1.51×10^{2}
	Rank	2	1	3	13	7	14	11	12	10	6	9	4	5	8	15
F27	Mean	3.09×10^{3}	3.10×10^{3}	3.09×10^{3}	3.12×10^{3}	3.13×10^{3}	3.42×10^{3}	3.19×10^{3}	3.22×10^{3}	3.14×10^{3}	3.10×10^{3}	3.13×10^{3}	3.10×10^{3}	3.10×10^{3}	3.14×10^{3}	3.31×10^{3}
	Std	2.99×10^{00}	1.10×10^{1}	6.10×10^{-1}	7.06×10^{00}	5.37×10^{1}	8.29×10^{1}	1.01×10^{1}	6.98×10^{1}	2.85×10^{1}	1.49×10^{00}	4.41×10^{1}	4.11×10^{00}	5.57×10^{00}	4.45×10^{1}	5.60×10^{1}
	SEM	1.22×10^{00}	4.51×10^{00}	3.92×10^{2}	2.88×10^{00}	2.19×10^{1}	3.71×10^{1}	4.53×10^{00}	3.12×10^{1}	1.28×10^{1}	6.07×10^{-1}	1.80×10^{1}	1.68×10^{00}	2.27×10^{00}	1.82×10^{1}	2.51×10^{1}
	Rank	1	6	2	7	9	15	12	13	11	5	8	4	3	10	14
F28	Mean	3.25×10^{3}	3.15×10^{3}	3.33×10^{3}	3.47×10^{3}	3.20×10^{3}	3.82×10^{3}	3.67×10^{3}	3.66×10^{3}	3.61×10^{3}	3.26×10^{3}	3.44×10^{3}	3.26×10^{3}	3.33×10^{3}	3.37×10^{3}	3.78×10^{3}
	Std	1.66×10^{2}	1.16×10^{2}	1.40×10^{2}	7.51×10^{1}	1.63×10^{2}	2.24×10^{2}	1.39×10^{2}	1.23×10^{2}	1.69×10^{2}	3.20×10^{1}	1.53×10^{2}	1.35×10^{2}	1.35×10^{2}	1.03×10^{2}	1.83×10^{2}
	SEM	6.78×10^{1}	4.73×10^{1}	5.27×10^{2}	3.07×10^{1}	6.66×10^{1}	1.00×10^{2}	6.22×10^{1}	5.51×10^{1}	7.55×10^{1}	1.31×10^{1}	6.27×10^{1}	5.49×10^{1}	5.51×10^{1}	4.22×10^{1}	8.19×10^{1}
	Rank	3	1	6	10	2	15	13	12	11	5	9	4	7	8	14
F29	Mean	3.14×10^{3}	3.16×10^{3}	3.17×10^{3}	3.35×10^{3}	3.19×10^{3}	3.75×10^{3}	3.51×10^{3}	3.48×10^{3}	3.51×10^{3}	3.24×10^{3}	3.32×10^{3}	3.21×10^{3}	3.20×10^{3}	3.23×10^{3}	3.68×10^{3}
	Std	3.36×10^{00}	1.39×10^{1}	6.72×10^{00}	9.33×10^{1}	4.71×10^{1}	1.29×10^{2}	1.02×10^{2}	1.89×10^{2}	1.00×10^{2}	4.61×10^{1}	1.09×10^{2}	4.55×10^{1}	3.86×10^{1}	6.34×10^{1}	1.40×10^{2}
	SEM	1.37×10^{00}	5.66×10^{00}	2.74×10^{2}	3.81×10^{1}	1.92×10^{1}	5.78×10^{1}	4.56×10^{1}	8.44×10^{1}	4.47×10^{1}	1.88×10^{1}	4.46×10^{1}	1.86×10^{1}	1.58×10^{1}	2.59×10^{1}	6.27×10^{1}
	Rank	1	2	3	10	4	15	12	11	13	8	9	6	5	7	14
F30	Mean	3.58×10^{3}	1.40×10^{5}	1.46×10^{4}	8.86×10^{5}	2.68×10^{5}	1.00×10^{8}	1.44×10^{7}	9.75×10^{6}	2.86×10^{7}	9.40×10^{5}	1.78×10^{6}	3.95×10^{5}	5.11×10^{5}	4.93×10^{5}	5.90×10^{7}
	Std	2.12×10^{2}	3.34×10^{5}	3.02×10^{3}	1.28×10^{-10}	4.90×10^{5}	5.67×10^{7}	8.14×10^{6}	4.94×10^{6}	1.86×10^{7}	7.39×10^{5}	1.03×10^{6}	5.80×10^{5}	4.11×10^{5}	9.49×10^{5}	2.79×10^{7}
	SEM	8.65×10^{1}	1.36×10^{5}	1.16×10^{4}	5.21×10^{-11}	2.00×10^{5}	2.54×10^{7}	3.64×10^{6}	2.21×10^{6}	8.32×10^{6}	3.02×10^{5}	4.21×10^{5}	2.37×10^{5}	1.68×10^{5}	3.88×10^{5}	1.25×10^{7}
	Rank	1	3	2	8	4	15	12	11	13	9	10	5	7	6	14

Table 9: WSODE Wilcoxon signed rank sum test results on IEEE congress on evolutionary computation 2017 (F1–F30) with FES = 1,000 and 30 independent runs

Function	WSO	DE	CMAES	COA	RSAA	BBO	SDE	SCA	WOA	DO	MFO	AOA
F1	1.73×10^{-6}	0.009271	1.92×10^{-6}	1.73×10^{-6}	0.000189	1.73×10^{-6}	1.73×10^{-6}	5.22×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 106, T−: 359	T+: 464, T−: 1	T+: 465, T−: 0	T+: 414, T−: 51	T+: 465, T−: 0	T+: 465, T−: 0	T+: 454, T−: 11	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F2	0.000616	0.926255	1.24×10^{-5}	1.73×10^{-6}	4.29×10^{-6}	1.73×10^{-6}	1.64×10^{-5}	4.29×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}
	T+: 399, T−: 66	T+: 228, T−: 237	T+: 445, T−: 20	T+: 465, T−: 0	T+: 456, T−: 9	T+: 465, T−: 0	T+: 442, T−: 23	T+: 456, T−: 9	T+: 465, T−: 0	T+: 465, T−: 0	T+: 464, T−: 1	T+: 465, T−: 0
F3	0.338856	0.171376	7.51×10^{-5}	1.73×10^{-6}	2.88×10^{-6}	1.73×10^{-6}	0.00532	3.18×10^{-6}	1.73×10^{-6}	3.41×10^{-5}	0.013975	0.000205
	T+: 279, T−: 186	T+: 166, T−: 299	T+: 425, T−: 40	T+: 465, T−: 0	T+: 460, T−: 5	T+: 465, T−: 0	T+: 368, T−: 97	T+: 459, T−: 6	T+: 465, T−: 0	T+: 434, T−: 31	T+: 352, T−: 113	T+: 413, T−: 52
F4	1.02×10^{-5}	0.085896	0.000261	1.92×10^{-6}	8.92×10^{-5}	1.73×10^{-6}	4.73×10^{-6}	0.000174	1.73×10^{-6}	1.73×10^{-6}	8.47×10^{-6}	1.73×10^{-6}
	T+: 447, T−: 18	T+: 149, T−: 316	T+: 410, T−: 55	T+: 464, T−: 1	T+: 423, T−: 42	T+: 465, T−: 0	T+: 455, T−: 10	T+: 415, T−: 50	T+: 465, T−: 0	T+: 465, T−: 0	T+: 449, T−: 16	T+: 465, T−: 0
F5	1.13×10^{-5}	0.000148	0.020671	2.13×10^{-6}	0.465283	1.73×10^{-6}	2.35×10^{-6}	0.280214	1.73×10^{-6}	1.73×10^{-6}	0.021827	2.37×10^{-5}
	T+: 446, T−: 19	T+: 417, T−: 48	T+: 120, T−: 345	T+: 463, T−: 2	T+: 197, T−: 268	T+: 465, T−: 0	T+: 462, T−: 3	T+: 285, T−: 180	T+: 465, T−: 0	T+: 465, T−: 0	T+: 344, T−: 121	T+: 438, T−: 27
F6	1.73×10^{-6}	0.000283	6.89×10^{-5}	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 56, T−: 409	T+: 426, T−: 39	T+: 465, T−: 0	T+: 464, T−: 1	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F7	2.6×10^{-6}	1.8×10^{-5}	0.84508	1.73×10^{-6}	0.465283	1.73×10^{-6}	6.34×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.13×10^{-5}	2.13×10^{-6}
	T+: 461, T−: 4	T+: 441, T−: 24	T+: 242, T−: 223	T+: 465, T−: 0	T+: 197, T−: 268	T+: 465, T−: 0	T+: 452, T−: 13	T+: 464, T−: 1	T+: 465, T−: 0	T+: 465, T−: 0	T+: 446, T−: 19	T+: 463, T−: 2
F8	0.002585	0.002105	0.000125	6.34×10^{-6}	0.115608	2.13×10^{-6}	0.95899	0.110926	1.73×10^{-6}	1.92×10^{-6}	0.95899	0.089718
	T+: 379, T−: 86	T+: 382, T−: 83	T+: 46, T−: 419	T+: 452, T−: 13	T+: 156, T−: 309	T+: 463, T−: 2	T+: 230, T−: 235	T+: 155, T−: 310	T+: 465, T−: 0	T+: 464, T−: 1	T+: 235, T−: 230	T+: 150, T−: 315
F9	2.35×10^{-6}	0.000529	0.003379	1.73×10^{-6}	1.02×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	2.35×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 462, T−: 3	T+: 64, T−: 401	T+: 375, T−: 90	T+: 465, T−: 0	T+: 447, T−: 18	T+: 465, T−: 0	T+: 465, T−: 0	T+: 462, T−: 3	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F10	0.289477	0.051931	1.64×10^{-5}	0.082206	1.73×10^{-6}	8.47×10^{-6}	0.016566	0.703564	1.73×10^{-6}	0.015658	0.002957	3.52×10^{-6}
	T+: 284, T−: 181	T+: 138, T−: 327	T+: 23, T−: 442	T+: 148, T−: 317	T+: 0, T−: 465	T+: 449, T−: 16	T+: 116, T−: 349	T+: 214, T−: 251	T+: 465, T−: 0	T+: 350, T−: 115	T+: 88, T−: 377	T+: 7, T−: 458
F11	1.73×10^{-6}	0.002415	1.73×10^{-6}	1.73×10^{-6}	2.13×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 380, T−: 85	T+: 465, T−: 0	T+: 465, T−: 0	T+: 463, T−: 2	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F12	0.002415	0.599936	0.000453	1.73×10^{-6}	0.000283	1.73×10^{-6}	2.13×10^{-6}	1.36×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	3.41×10^{-5}	2.6×10^{-5}
	T+: 380, T−: 85	T+: 207, T−: 258	T+: 403, T−: 62	T+: 465, T−: 0	T+: 409, T−: 56	T+: 465, T−: 0	T+: 463, T−: 2	T+: 444, T−: 21	T+: 465, T−: 0	T+: 465, T−: 0	T+: 434, T−: 31	T+: 437, T−: 28
F13	1.73×10^{-6}	0.038723	1.73×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 132, T−: 333	T+: 465, T−: 0	T+: 464, T−: 1	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F14	1.73×10^{-6}	0.040702	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 332, T−: 133	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F15	1.73×10^{-6}	0.797098	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 245, T−: 220	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F16	0.000664	0.95899	0.14139	2.35×10^{-6}	0.280214	1.73×10^{-6}	0.000174	0.002255	1.73×10^{-6}	1.73×10^{-6}	0.001287	0.000136
	T+: 398, T−: 67	T+: 235, T−: 230	T+: 304, T−: 161	T+: 462, T−: 3	T+: 285, T−: 180	T+: 465, T−: 0	T+: 415, T−: 50	T+: 381, T−: 84	T+: 465, T−: 0	T+: 465, T−: 0	T+: 389, T−: 76	T+: 418, T−: 47
F17	0.404835	1.73×10^{-6}	0.797098	0.002957	0.004114	2.13×10^{-6}	0.008217	0.628843	1.73×10^{-6}	0.000148	0.643517	0.012453
	T+: 273, T−: 192	T+: 0, T−: 465	T+: 220, T−: 245	T+: 377, T−: 88	T+: 93, T−: 372	T+: 463, T−: 2	T+: 361, T−: 104	T+: 256, T−: 209	T+: 465, T−: 0	T+: 417, T−: 48	T+: 255, T−: 210	T+: 111, T−: 354
F18	2.6×10^{-6}	5.79×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}

(*Continued*)

Table 9: *Continued*

Function	WSO	DE	CMAES	COA	RSAA	BBO	SDE	SCA	WOA	DO	MFO	AOA
	T+: 461, T−: 4	T+: 37, T−: 428	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F19	1.73×10^{-6}	0.221022	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 292, T−: 173	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0	T+: 465, T−: 0
F20	0.000306	4.07×10^{-5}	0.059836	3.18×10^{-6}	0.062683	1.73×10^{-6}	6.34×10^{-6}	0.165027	1.73×10^{-6}	2.88×10^{-6}	0.001484	0.036826
	T+: 408, T−: 57	T+: 33, T−: 432	T+: 324, T−: 141	T+: 459, T−: 6	T+: 142, T−: 323	T+: 465, T−: 0	T+: 452, T−: 13	T+: 300, T−: 165	T+: 465, T−: 0	T+: 460, T−: 5	T+: 387, T−: 78	T+: 334, T−: 131
F21	0.110926	0.014795	0.688359	0.000174	0.614315	0.078647	0.00042	0.130592	2.35×10^{-6}	0.033269	0.102011	0.040702
	T+: 310, T−: 155	T+: 351, T−: 114	T+: 252, T−: 213	T+: 415, T−: 50	T+: 257, T−: 208	T+: 318, T−: 147	T+: 404, T−: 61	T+: 306, T−: 159	T+: 462, T−: 3	T+: 336, T−: 129	T+: 312, T−: 153	T+: 332, T−: 133
F22	1.73×10^{-6}	0.165027	2.6×10^{-6}	1.92×10^{-6}	3.11×10^{-5}	1.73×10^{-6}	1.73×10^{-6}	0.00049	1.73×10^{-6}	2.13×10^{-6}	9.71×10^{-5}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 165, T−: 300	T+: 461, T−: 4	T+: 464, T−: 1	T+: 435, T−: 30	T+: 465, T−: 0	T+: 465, T−: 0	T+: 402, T−: 63	T+: 465, T−: 0	T+: 463, T−: 2	T+: 422, T−: 43	T+: 465, T−: 0
F23	3.18×10^{-6}	0.038723	0.797098	7.69×10^{-6}	0.054463	6.98×10^{-6}	2.13×10^{-6}	0.893644	1.73×10^{-6}	1.73×10^{-6}	0.000189	1.73×10^{-6}
	T+: 459, T−: 6	T+: 333, T−: 132	T+: 245, T−: 220	T+: 450, T−: 15	T+: 139, T−: 326	T+: 451, T−: 14	T+: 463, T−: 2	T+: 239, T−: 226	T+: 465, T−: 0	T+: 465, T−: 0	T+: 414, T−: 51	T+: 465, T−: 0
F24	0.781264	0.000205	0.703564	1.92×10^{-6}	0.370935	0.000359	2.6×10^{-6}	0.078647	1.92×10^{-6}	1.73×10^{-6}	3.11×10^{-5}	0.002957
	T+: 246, T−: 219	T+: 413, T−: 52	T+: 214, T−: 251	T+: 464, T−: 1	T+: 276, T−: 189	T+: 406, T−: 59	T+: 461, T−: 4	T+: 318, T−: 147	T+: 464, T−: 1	T+: 465, T−: 0	T+: 435, T−: 30	T+: 377, T−: 88
F25	0.00016	0.393334	0.075213	1.92×10^{-6}	0.205888	1.73×10^{-6}	0.003854	0.028486	1.73×10^{-6}	1.73×10^{-6}	0.000388	0.000571
	T+: 416, T−: 49	T+: 191, T−: 274	T+: 319, T−: 146	T+: 464, T−: 1	T+: 294, T−: 171	T+: 465, T−: 0	T+: 373, T−: 92	T+: 339, T−: 126	T+: 465, T−: 0	T+: 465, T−: 0	T+: 405, T−: 60	T+: 400, T−: 65
F26	1.24×10^{-5}	0.909931	0.000205	1.73×10^{-6}	9.71×10^{-5}	1.73×10^{-6}	4.29×10^{-6}	0.000205	1.73×10^{-6}	1.73×10^{-6}	3.18×10^{-6}	2.13×10^{-6}
	T+: 445, T−: 20	T+: 227, T−: 238	T+: 413, T−: 52	T+: 465, T−: 0	T+: 422, T−: 43	T+: 465, T−: 0	T+: 456, T−: 9	T+: 413, T−: 52	T+: 465, T−: 0	T+: 465, T−: 0	T+: 459, T−: 6	T+: 463, T−: 2
F27	1.73×10^{-6}	0.14139	0.008217	1.73×10^{-6}	0.477947	1.73×10^{-6}	2.35×10^{-6}	0.000453	1.73×10^{-6}	1.73×10^{-6}	2.6×10^{-5}	1.73×10^{-6}
	T+: 465, T−: 0	T+: 161, T−: 304	T+: 361, T−: 104	T+: 465, T−: 0	T+: 198, T−: 267	T+: 465, T−: 0	T+: 462, T−: 3	T+: 403, T−: 62	T+: 465, T−: 0	T+: 465, T−: 0	T+: 437, T−: 28	T+: 465, T−: 0
F28	0.028486	0.416534	6.32×10^{-5}	1.36×10^{-5}	0.019569	1.92×10^{-6}	0.001114	0.006424	1.73×10^{-6}	6.89×10^{-5}	5.22×10^{-6}	3.52×10^{-6}
	T+: 339, T−: 126	T+: 193, T−: 272	T+: 427, T−: 38	T+: 444, T−: 21	T+: 346, T−: 119	T+: 464, T−: 1	T+: 391, T−: 74	T+: 365, T−: 100	T+: 465, T−: 0	T+: 426, T−: 39	T+: 454, T−: 11	T+: 458, T−: 7
F29	4.07×10^{-5}	0.289477	0.749871	2.35×10^{-6}	0.010444	1.73×10^{-6}	7.69×10^{-6}	0.000136	1.73×10^{-6}	2.6×10^{-6}	0.000771	0.00016
	T+: 432, T−: 33	T+: 181, T−: 284	T+: 217, T−: 248	T+: 462, T−: 3	T+: 357, T−: 108	T+: 465, T−: 0	T+: 450, T−: 15	T+: 418, T−: 47	T+: 465, T−: 0	T+: 461, T−: 4	T+: 396, T−: 69	T+: 416, T−: 49
F30	0.130592	0.22888	0.035009	1.02×10^{-5}	0.004682	1.73×10^{-6}	0.001197	0.000616	1.73×10^{-6}	1.73×10^{-6}	0.003609	2.16×10^{-5}
	T+: 306, T−: 159	T+: 174, T−: 291	T+: 335, T−: 130	T+: 447, T−: 18	T+: 370, T−: 95	T+: 465, T−: 0	T+: 390, T−: 75	T+: 399, T−: 66	T+: 465, T−: 0	T+: 465, T−: 0	T+: 374, T−: 91	T+: 439, T−: 26
Total	+:24, −:0, =:6	+:8, −:7, =:15	+:18, −:3, =:9	+:29, −:0, =:1	+:18, −:2, =:10	+:29, −:0, =:1	+:28, −:1, =:1	+:22, −:0, =:8	+:30, −:0, =:0	+:30, −:0, =:0	+:26, −:1, =:3	+:27, −:2, =:1

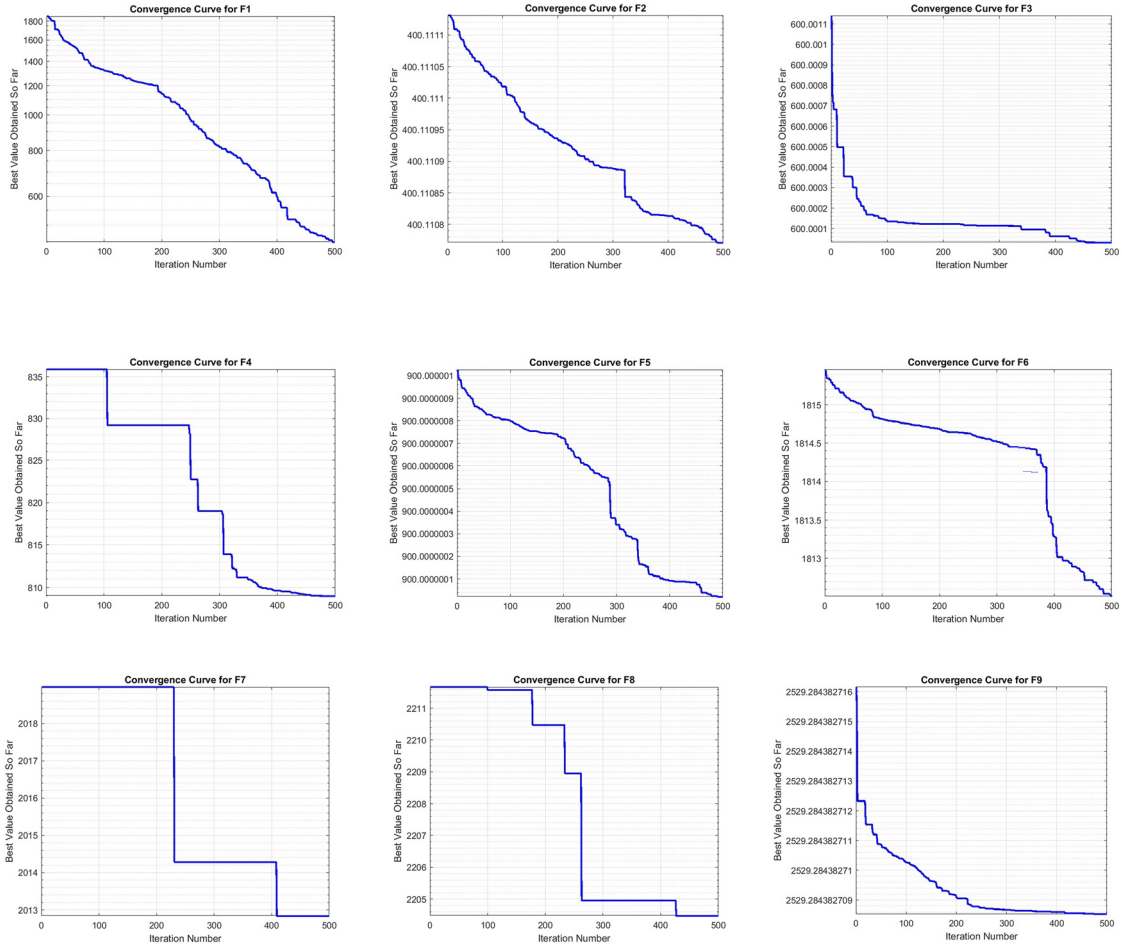

Figure 2: Convergence curve analysis over CEC2022 benchmark functions (F1–F9). Source: Created by the authors.

steady declines, reflecting stable optimization. Functions F4, F7, and F10 display step-like patterns, suggesting significant improvements at specific iterations. Moreover, WSODE effectively balances exploration and exploitation, achieving reliable convergence across various optimization tasks.

5.6 WSODE search history

The search history plots for functions F1–F9 over CEC2022 Figure 3, provides insights into the exploration and exploitation behavior of WSODE. These plots illustrate how the algorithm samples the search space over time. For all functions, the search history reveals a high concentration of solutions around the optimal regions, represented by dense clusters of points, particularly in the central areas. This clustering indicates that WSODE effectively narrows down the search to promising areas of the solution space. Additionally, the presence of outlier points suggests that WSODE maintains a level of exploration to avoid premature convergence and ensure a comprehensive search. Furthermore, the search history demonstrates that WSODE balances exploration and exploitation efficiently, consistently converging towards optimal solutions across various functions.

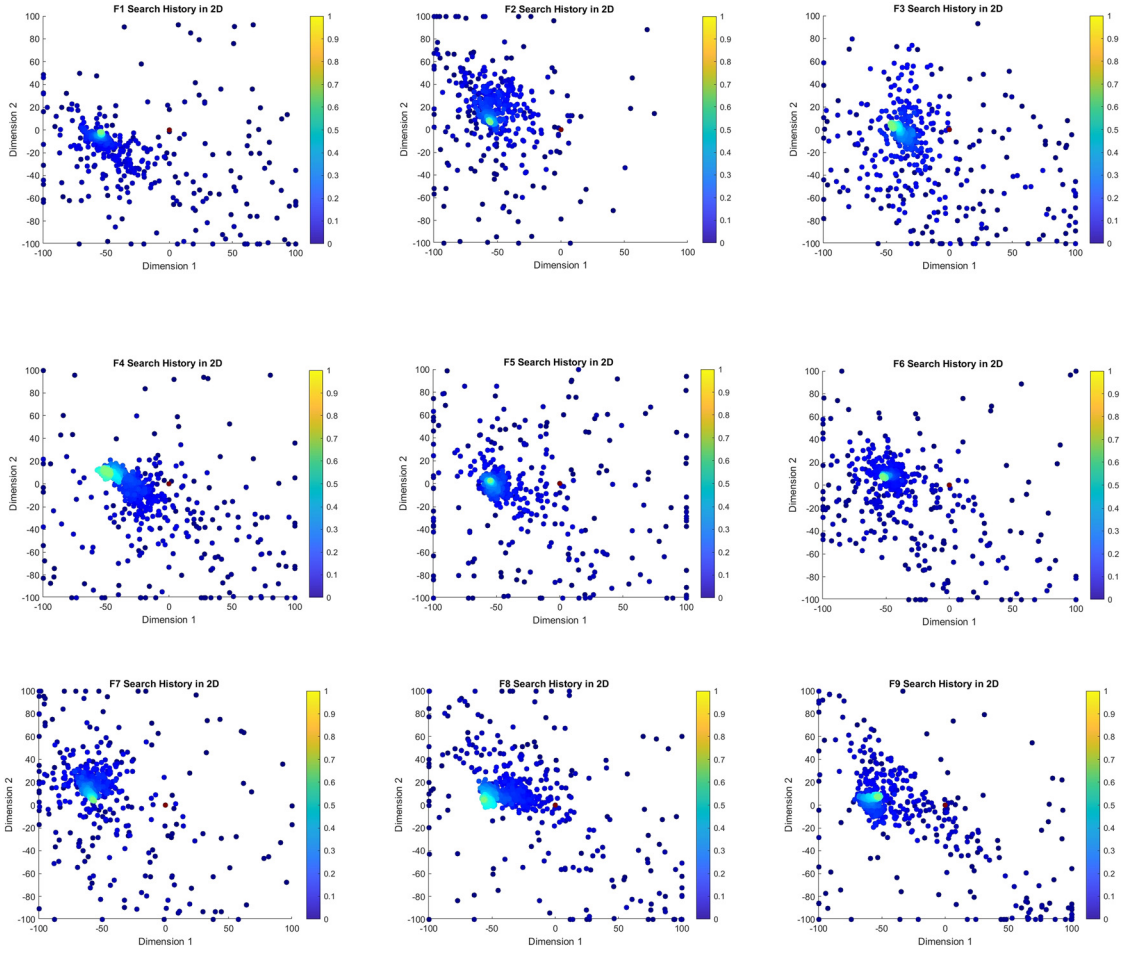

Figure 3: Search history analysis over CEC2022 benchmark functions (F1–F9). Source: Created by the authors.

5.7 WSODE heatmap analyis

The sensitivity analysis using heatmaps for functions F1 through F9 over CEC2022 (Figure 4), reveals important insights into the performance of WSODE under varying conditions of search agents and iterations. As seen in the heat maps, the performance tends to improve with an increase in the number of search agents and iterations, particularly for more complex functions like F1, F6, and F11. For instance, in F1, a clear trend of improved performance is noticeable as the number of iterations increases, especially when the number of search agents is high. Similarly, for simpler functions like F3, performance stabilizes with fewer agents and iterations, indicating the robustness of the WSODE in efficiently solving these functions without requiring extensive computational resources. This analysis highlights the adaptability and efficiency of WSODE across different optimization landscapes.

6 Training MLPs neural network using WSODE

The foundational and most critical step in training an MLP using meta-heuristic techniques is the problem representation. Specifically, the problem of training MLPs must be formulated in a manner that is compatible with meta-heuristic approaches. As previously discussed, the primary variables involved in training an MLP are the synaptic weights and the biases. The objective of the training process is to identify a set of values for these weights and biases that yield the highest accuracy in terms of classification, approximation, or prediction.

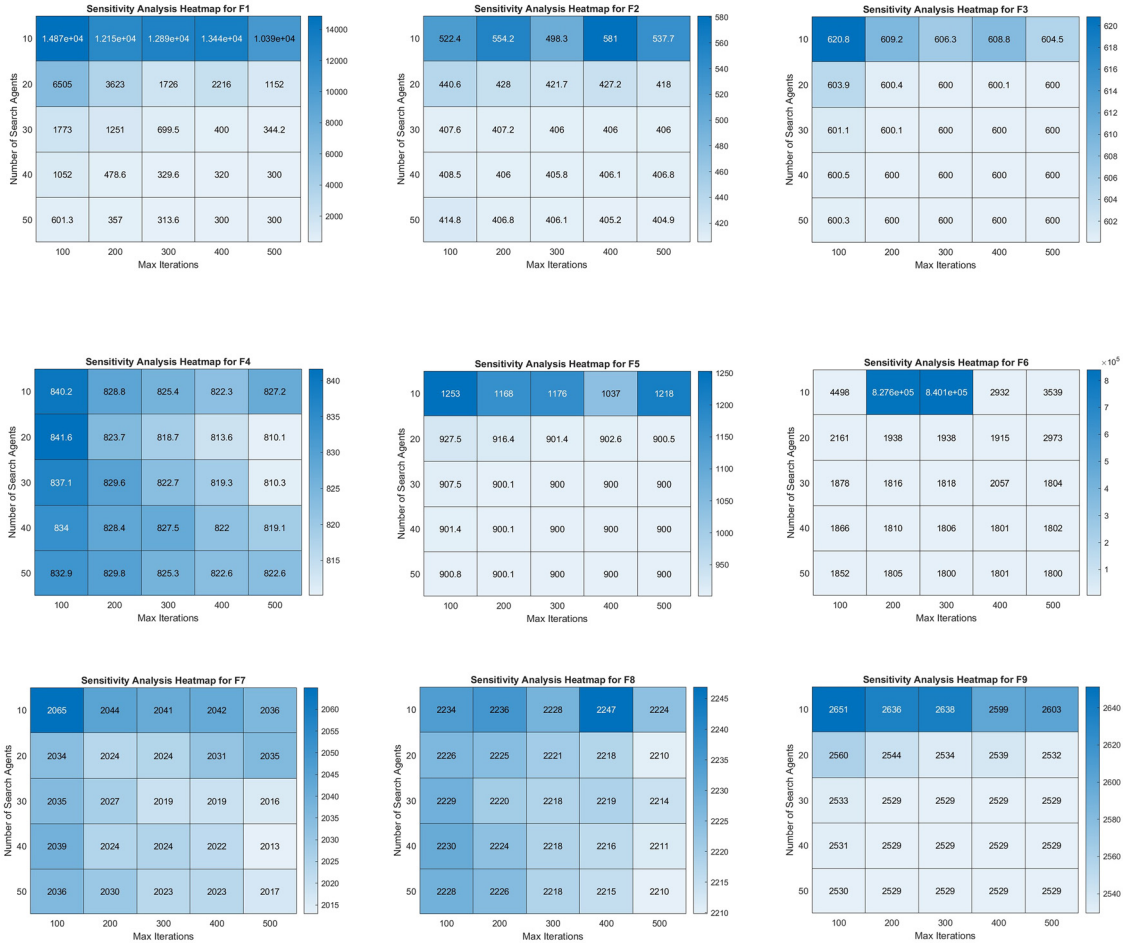

Figure 4: Sensitivity analysis using heatmap over CEC2022 benchmark functions (F1–F9). Source: Created by the authors.

Consequently, the variables in this context are the weights and biases. The WSODE algorithm handles these variables in the form of a vector. For the purposes of this algorithm, the variables of an MLP are represented as illustrated in the following equation:

$$\vec{U} = \{\vec{X}, \beta\} = \{X_{1,1}, X_{1,2}, \ldots, X_{m,k}, \beta_1, \beta_2, \ldots, \beta_k\}, \tag{32}$$

where m denotes the number of input nodes, X_{ij} signifies the synaptic weight from the ith input node to the jth hidden node, and β_j represents the bias (threshold) of the jth hidden node.

Following the definition of the parameters, it is crucial to establish the objective function for the WSODE algorithm. The primary aim in training an MLP is to maximize accuracy in tasks such as classification, approximation, or prediction for both training and testing datasets. To assess the performance of an MLP, the MSE is a widely utilized metric. This metric measures the deviation between the desired output and the actual output produced by the MLP for a given set of training samples. The MSE is calculated as the sum of the squared differences between the actual output and the target output for all output nodes across all training samples. Formally, it is expressed as:

$$\text{MSE} = \sum_{i=1}^{p} (y_{j,i} - t_{j,i})^2, \tag{33}$$

where p denotes the number of output nodes. The term $t_{j,i}$ represents the target output of the ith input unit for the jth training sample, while $y_{j,i}$ signifies the actual output of the ith input unit when the jth training sample is presented to the MLP. By minimizing the MSE, the WSODE algorithm seeks to decrease the disparity between

the MLP's predicted outputs and the actual target values, thereby improving the accuracy of the MLP in its assigned tasks.

It is evident that for an MLP to be effective, it must adapt to the entire set of training samples. Therefore, the performance of an MLP is assessed based on the average MSE across all training samples, as illustrated in the following equation:

$$\text{MSE} = \frac{1}{q}\sum_{j=1}^{q}\sum_{i=1}^{p}(y_{j,i} - t_{j,i})^2, \tag{34}$$

where q represents the number of training samples, p denotes the number of output nodes, $t_{j,i}$ is the target output of the ith input unit for the jth training sample, and $y_{j,i}$ is the actual output of the ith input unit when the jth training sample is applied.

Thus, the problem of training an MLP is formulated by defining the variables and the average MSE for the WSODE algorithm as shown in the following equation:

$$\text{Minimize: } F(\vec{U}) = \text{MSE}. \tag{35}$$

In this context, the WSODE algorithm provides the MLP with synaptic weights and biases and receives the average MSE for all training samples. The WSODE algorithm iteratively adjusts the weights and biases to minimize the average MSE across all training samples.

In the integrated WSODE algorithm for training an MLP, the process begins with initializing a population of search agents, referred to as sharks, which represent potential solutions in the optimization process. Each shark's position corresponds to a set of synaptic weights and biases for the MLP, initialized randomly within specified bounds defined by lower (lb) and upper bounds (ub). The algorithm starts by setting up initial parameters for the DE phase, such as the mutation factor (F) and crossover probability (CR).

During the DE phase, sharks are selected for mutation, where a new solution (offspring) is generated by combining the positions of three randomly chosen sharks. A crossover operation then determines whether the mutated offspring should replace the current position of the shark, based on the crossover probability CR. The fitness of each shark is evaluated using an objective function f_{obj}, typically assessing the MSE across all training samples to quantify the discrepancy between the desired and actual outputs produced by the MLP.

After the DE phase, the algorithm transitions to the WSO phase, where sharks adjust their positions based on historical best positions (p_{best}) and the global best position (g_{best}) found during the optimization process. This phase utilizes movement velocity (v) and adaptive parameters ($f_{\min}$, $f_{\max}$) to balance exploration and exploitation within the search space. The WSO phase aims to refine the positions of sharks further, enhancing the convergence toward optimal solutions.

Throughout the iterative optimization process, sharks continually update their positions based on the evaluated fitness, aiming to minimize the MSE and improve the MLP's performance. Convergence criteria, such as reaching a specified number of iterations (itemax) or achieving MSE below a defined threshold, determine the termination of the algorithm.

Once convergence is reached, the optimized set of synaptic weights and biases (g_{best}) represents the refined parameters for the MLP. These optimized parameters are then used to update the MLP, ensuring that it incorporates the best-performing configuration found by the WSODE algorithm. The performance of the MLP can subsequently be evaluated on unseen data or utilized for further tasks, such as testing and validation in classification, approximation, or prediction scenarios.

The WSODE algorithm guides the adjustment of weights and biases towards increasingly refined MLP configurations discovered throughout its iterations, fostering continual improvement in model performance. This iterative refinement process capitalizes on the best-performing MLP solutions identified at each step. While WSODE's stochastic nature precludes a deterministic assurance of discovering the optimal MLP configuration for a given dataset, the algorithm's strategy of leveraging evolved solutions progressively reduces the overall average MSE across the population over time. Consequently, with sufficient iteration, WSODE converges toward solutions that significantly outperform initial random configurations, demonstrating its efficacy in enhancing MLP performance through iterative evolution.

6.1 WSODE-based MLP results and discussion

In this section, we present the benchmarking results of the WSODE-based MLP trainer using five standard classification datasets sourced from the University of California at Irvine (UCI) Machine Learning Repository [52].

7 WSODE-based MLP experimental setup

The experimental setup for evaluating the WSODE-based MLP trainer involves several critical steps and assumptions to ensure a comprehensive and fair comparison with other algorithms. The optimization process begins by generating random initial weights and biases within the range of $[-10,\ 10]$ for all datasets. This range is selected to provide a broad search space, allowing the algorithm to explore diverse potential solutions effectively.

In addition to the initial weight and bias range, several other parameters and assumptions are integral to the WSODE algorithm and the comparative algorithms. These parameters are crucial for defining the behavior and performance of the algorithms during the optimization process. The specific assumptions and parameters for the WSODE algorithm, along with those for the comparative algorithms used in this study, are summarized in Table 10.

Table 10: Classification and regression datasets

Dataset	Number of attributes	Number of samples	Number of classes
Wine	13	178	3
Abalone	8	4,177	3
Hepatitis	19	155	2
Breast cancer	9	699	2
Housing	13	506	1
Banknote authentication	4	1,372	2

Key parameters and assumptions

- **Population size:** The number of candidate solutions (or individuals) in the population for each algorithm. This affects the diversity and convergence rate of the algorithms.
- **Number of iterations:** The total number of iterations or generations for which the algorithms run. This parameter determines the computational budget and the potential for finding optimal solutions.
- **Learning rate:** The step size used during the optimization process, influencing how quickly the algorithms adjust the weights and biases.
- **Crossover and mutation rates:** Specific to evolutionary algorithms, these rates determine how new candidate solutions are generated from existing ones.
- **Fitness evaluation:** The criteria used to assess the performance of each candidate solution. For MLP training, this is typically based on the MSE.
- **Algorithm-specific parameters:** Unique parameters pertinent to each algorithm, such as inertia weight in PSO or differential weight in DE.

Table 10 provides a detailed overview of these parameters and assumptions for the WSODE algorithm and the comparative algorithms used in the experiments. This comprehensive setup ensures a robust evaluation of the WSODE-based MLP trainer's performance against established benchmarks.

As it can be seen in Tables 10 and 11, the specifications of the datasets are as follows: the Wine dataset has 178 samples, 13 attributes, and 3 classes. The Abalone dataset is more challenging, featuring 8 attributes, 4,177 samples, and 3 classes. The Hepatitis dataset includes 19 attributes, 155 samples, and 2 classes. Additionally, the

Table 11: Datasets MLP structure

Dataset	Number of attributes	MLP structure
Wine	13	13-27-3
Abalone	8	8-17-3
Hepatitis	19	19-39-2
Breast cancer	9	9-19-2
Housing	13	13-27-1
Banknote authentication	4	4-9-2

Breast cancer dataset comprises 9 attributes, 699 samples, and 2 classes. The Housing dataset includes 13 attributes, 506 samples, and 1 output variable. Finally, the Banknote authentication dataset includes 4 attributes, 1,372 samples, and 2 classes. These classification and regression datasets were deliberately chosen with varying numbers of samples and levels of difficulty to effectively test the performance of the proposed WSODE-based MLP trainer. WSODE results are compared with those obtained using FVIM, SCA, WOA, DE, and MFO for verification.

As it can be seen in Table 11, the notation "13-27-3" in the MLP structure column represents the architecture of the MLP network used for each dataset. Specifically, this format describes the number of neurons in each layer of the network. For example, "13-27-3" corresponds to an MLP with three layers: the first layer has 13 neurons (matching the number of input attributes), the hidden layer has 27 neurons, and the output layer has 3 neurons. This structure is chosen to adapt to the specific characteristics of each dataset. Thus, each entry in this column defines the input layer size, hidden layer size, and output layer size for the MLP model.

7.1 Repeated trials and statistical analysis

Each dataset is evaluated 30 times using each algorithm to ensure robust and reliable results. The statistical metrics reported include the average (AVE) and STD of the best MSEs obtained in the final iteration of each algorithm. A lower average and STD of MSE indicate superior performance. The statistical results are expressed as $\text{AVE} \pm \text{STD}$. Additionally, the highest classification rates or lowest test errors achieved by each algorithm during these runs are also reported as a comparative metric.

7.2 Normalization procedure

Normalization is a crucial preprocessing step for MLPs, especially when dealing with datasets containing attributes with varying ranges. In this study, min-max normalization is employed, which is defined by the following equation:

$$Y = \frac{(X - X_{\min}) \times (D_{\max} - D_{\min})}{(X_{\max} - X_{\min})} + D_{\min}, \tag{36}$$

where X represents the original value, $X_{\min}$ and $X_{\max}$ denote the minimum and maximum values of X respectively, and $D_{\min}$ and $D_{\max}$ are the new minimum and maximum values to which X is scaled.

7.3 MLP structure

The architecture of the MLPs is another critical factor in the experimental setup. This study does not aim to determine the optimal number of hidden nodes; instead, the number of hidden nodes is set to $2N + 1$, where N

is the number of input features in the datasets. The specific structure of each MLP used for each dataset is detailed in Table 11.

7.4 Complexity and training challenges

As the size of the neural network increases, the number of weights and biases that need to be optimized also grows, leading to increased complexity in the training process. This necessitates efficient optimization algorithms capable of handling large-scale neural networks to ensure effective training and convergence.

The comprehensive setup and detailed parameters ensure a thorough evaluation of the WSODE-based MLP trainer against various benchmark algorithms, providing insights into its performance and effectiveness across different datasets.

7.5 Wine dataset

The wine dataset consists of 13 attributes, 178 samples, and 3 classes. The MLP trainers for this dataset have dimensions of 28. The results are summarized in Table 12.

Table 12: Experimental results for the wine dataset

Algorithm	MSE (AVE)	STD	Classification rate (%)
WSODE	0.003210	0.001540	98.89
FVIM	0.012300	0.004650	97.22
SCA	0.002430	0.001120	97.44
WOA	0.014500	0.005230	96.67
DE	0.008910	0.002780	97.78
MFO	0.005720	0.002130	98.33

An initial observation from the results is the high classification rate among all algorithms, which reflects the structured nature of the wine dataset. However, the average and STD of the MSE over 10 runs differ across the algorithms. Consistent with the results from other datasets, WSODE and SCA demonstrate superior performance in terms of avoiding local optima, as evidenced by the statistical outcomes of the MSEs. This finding underscores the high efficiency of WSODE in training MLPs.

7.6 Abalone dataset

The Abalone dataset, often used for regression tasks, consists of 8 attributes, 4,177 samples, and 3 classes. The MLP structure for this dataset is configured as 8-17-3, resulting in 173 variables. The performance results of various training algorithms are summarized in Table 13.

The results demonstrate that WSODE outperforms other algorithms, achieving the lowest MSE and the highest classification accuracy. Following FVIM, SCA and DE also show commendable performance. Given the complexity of the Abalone dataset and the corresponding MLP structure, these results provide strong evidence of the effectiveness of WSODE in training MLPs. The findings indicate that this algorithm excels in both local optima avoidance and accuracy.

Table 13: Experimental results for the Abalone dataset

Algorithm	MSE (AVE)	STD	Classification rate (%)
WSODE	0.1254	0.0152	67.33
FVIM	0.2150	0.0286	54.66
SCA	0.1457	0.0203	63.00
WOA	0.2401	0.0308	51.33
DE	0.1983	0.0257	58.66
MFO	0.1624	0.0221	61.33

7.7 Hepatitis dataset

The hepatitis dataset is a challenging dataset frequently used for classification tasks in the field of machine learning. It consists of 19 attributes, 155 samples, and 2 classes. The MLP structure employed for this dataset is configured as 19-39-2, resulting in 800 variables. The performance results of various training algorithms are summarized in Table 14.

Table 14: Experimental results for the hepatitis dataset

Algorithm	MSE (AVE)	STD	Classification rate (%)
WSODE	0.0147	0.0042	86.33
FVIM	0.113460	0.027235	75.33
SCA	0.052311	0.014638	83.33
WOA	0.205684	0.073775	70.66
DE	0.154320	0.042142	72.66
MFO	0.098567	0.036355	80.66

7.8 WSODE breast cancer dataset results

The results demonstrate that WSODE surpasses other algorithms, achieving the lowest MSE and the highest classification accuracy. Following WOA, SCA and MFO also exhibit commendable performance. Given the complexity of the hepatitis dataset and the corresponding MLP structure, these findings provide substantial evidence of WSODE's efficacy in training MLPs. The results highlight the algorithm's proficiency in avoiding both local optima and achieving high accuracy.

7.9 Breast cancer dataset

The breast cancer dataset is a widely studied dataset in machine learning. It includes 9 attributes, 699 samples, and 2 classes. The MLP structure for this dataset is configured as 9-19-2, resulting in 191 variables. The performance results of various training algorithms are summarized in Table 15.

The results indicate that WSODE outperforms other algorithms, achieving the lowest MSE and the highest classification accuracy. Following WSODE, SCA and MFO also show commendable performance. Given the complexity of the breast cancer dataset and the corresponding MLP structure, these results provide strong evidence of the efficacy of WSODE in training MLPs. The outcomes demonstrate that this algorithm excels in both local optima avoidance and accuracy.

Table 15: Experimental results for the breast cancer dataset

Algorithm	MSE (AVE)	STD	Classification rate (%)
WSODE	0.0154	0.0027	94.21
FVIM	0.238560	0.061823	82.24
SCA	0.075614	0.112784	91.78
WOA	0.423187	0.049831	89.53
DE	0.298432	0.046729	72.87
MFO	0.108945	0.031524	88.69

7.10 Housing dataset

The housing dataset, commonly used for regression tasks, consists of 13 attributes, 506 samples, and 1 output variable. The MLP structure for this dataset is configured as 13-27-1, resulting in 380 variables. The performance results of various training algorithms are summarized in Table 16.

Table 16: Experimental results for the housing dataset

Algorithm	MSE (AVE)	STD	Classification rate (%)
WSODE	0.0187	0.0029	89.67
FVIM	0.0924	0.0148	78.33
SCA	0.0456	0.0097	84.33
WOA	0.1125	0.0184	75.00
DE	0.0843	0.0126	80.67
MFO	0.0671	0.0109	82.00

The results reveal that WSODE surpasses other algorithms, achieving the lowest MSE and the highest classification accuracy. Following WSODE, SCA and MFO also exhibit strong performance. Given the complexity of the housing dataset and the corresponding MLP structure, these results provide robust evidence of the efficacy of WSODE in training MLPs. The findings indicate that this algorithm excels in both local optima avoidance and accuracy.

7.11 Banknote authentication dataset

As detailed in Table 10, the banknote authentication dataset consists of 4 inputs, 1,372 samples, and 1 output. The objective of the MLP trained on the banknote authentication dataset is to classify whether a given banknote is authentic based on the input features. Table 17 presents the statistical outcomes of the WSODE, FVIM,

Table 17: Experimental results for the Banknote authentication dataset

Algorithm	MSE (AVE)	STD	Classification rate (%)
WSODE	0.012340	0.010234	95.0
FVIM	0.084050	0.035945	70.50
SCA	0.001234	0.000567	88.0
WOA	0.120328	0.025268	62.50
DE	0.078739	0.011574	72.50
MFO	0.050228	0.039668	78.50

DE, SCA, WOA, and MFO algorithms on the banknote authentication dataset. Note that in the tables that follow, the training algorithms are denoted in the format algorithm.

The results indicate that WSODE achieved the best average MSE, suggesting that these two algorithms have superior capability in avoiding local optima compared to the others. WSODE managed to classify the banknotes with high accuracy. DE, being an evolutionary algorithm, demonstrates fair exploration capability. These findings highlight the competitive performance of the WSODE-based trainer when compared to the other algorithms.

8 Conclusion

This study proposed a hybrid WSODE algorithm, motivated by the need for advanced optimization techniques that can effectively balance exploration and exploitation, addressing complex and diverse optimization challenges. However, traditional optimization algorithms have shown limitations when applied to complex benchmark functions and real-world applications, such as engineering design and machine learning model training. To overcome these limitations, the WSODE algorithm integrates the exploration strengths of DE with the exploitation efficiency of the WSO, forming a hybrid solution optimized for robustness and adaptability. The effectiveness of WSODE was demonstrated through extensive testing on CEC2021 and CEC2022 benchmark functions, as well as the Spring Design Problem. Experimental results revealed that WSODE consistently achieves superior or comparable performance to several state-of-the-art optimization algorithms, particularly in metrics such as mean, STD, and standard error of the mean across most benchmark functions. The convergence curves and search history plots further confirm WSODE's capability to efficiently balance exploration and exploitation, achieving faster convergence rates while effectively avoiding local optima. Furthermore, WSODE's application to training MLPs across various datasets, each selected for its complexity and attribute count, demonstrated the hybrid algorithm's strong adaptability and performance. Results consistently showed that WSODE achieved the lowest MSE and the highest classification accuracy on the majority of datasets, reinforcing its robustness and efficiency. These findings establish WSODE as a versatile and effective solution for complex optimization tasks across benchmark functions and practical engineering and machine learning applications.

8.1 Future directions

Our future directions involve extending WSODE to solve multiobjective optimization problems. Multiobjective optimization often requires a balance between conflicting objectives, and adapting WSODE for this purpose could involve implementing a Pareto-based approach or incorporating nondominated sorting strategies. By developing a multiobjective WSODE, the algorithm could be applied to scenarios requiring trade-offs between objectives.

Funding information: No external funding was received for this research.

Author contributions: Hussam N. Fakhouri: conceived the research idea, contributed to algorithmic design, conducted experiments, and participated in manuscript preparation, experimentation, and manuscript review. Ahmad Sami Al-Shamayleh: contributed to methodology development, experimentation, and manuscript review. Abedelraouf Istiwi: provided critical insights on algorithm validation and data analysis, and assisted in manuscript editing. Sharif Naser Makhadmeh: performed result interpretation, comparative analysis, and manuscript proofreading. Faten Hamad: participated in benchmarking and statistical evaluations, and contributed to literature review. Sandi N. Fakhouri: provided assistance in algorithm coding and validation, and participated in experiments. Zaid Abdi Alkareem Alyasseri: reviewed the final manuscript, offered guidance on optimization benchmarks, and coordinated collaborative aspects of the study. All authors contributed substantially to the research and reviewed the final manuscript.

Conflict of interest: The authors declare that they have no conflicts of interest or competing interests that could have influenced the work reported in this article.

Data availability statement: All benchmark datasets used in this study are publicly available from their respective sources (CEC2022, CEC2021, and CEC2017 competition repositories), and the real-world datasets (Wine, Abalone, Hepatitis, Breast Cancer, Housing, and Banknote Authentication) are publicly accessible via the UCI Machine Learning Repository. Any additional data or code needed to replicate the experiments can be obtained from the corresponding author upon reasonable request.

References

[1] Abualigah L, Elaziz MA, Khasawneh AM, Alshinwan M, Ibrahim RA, Al-Qaness MA, et al. Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results. Neural Comput Appl. 2022;34:1–30, doi: 10.1007/s00521-021-06747-4.

[2] Suresh S, Lal S. Multilevel thresholding based on chaotic Darwinian particle swarm optimization for segmentation of satellite images. Appl Soft Comput. 2017;55:503–22. doi: 10.1016/j.asoc.2017.02.005.

[3] Abdel-Basset M, Abdel-Fatah L, Sangaiah AK. Metaheuristic algorithms: A comprehensive review. Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications. Academic Press; 2018. p. 185–231. doi: 10.1016/B978-0-12-813314-9.00010-4.

[4] Fakhouri HN, Awaysheh FM, Alawadi S, Alkhalaileh M, Hamad F. Four vector intelligent metaheuristic for data optimization. Computing. 2024;106:1–39. doi: 10.1007/s00607-024-01287-w.

[5] Fakhouri HN, Alawadi S, Awaysheh FM, Hamad F. Novel hybrid success history intelligent optimizer with gaussian transformation: Application in CNN hyperparameter tuning. Cluster Comput. 2023;27:1–23. doi: 10.1007/s10586-023-04161-0.

[6] Alba E, Nakib A, Siarry P. Metaheuristics for dynamic optimization. vol. 433. Heidelberg: Springer; 2013. doi: 10.1007/978-3-642-30665-5.

[7] Fakhouri HN, Alawadi S, Awaysheh FM, Alkhabbas F, Zraqou J. A cognitive deep learning approach for medical image processing. Scientif Reports. 2024;14(1):4539. doi: 10.1038/s41598-024-55061-1.

[8] Özkış A, Babalık A. A novel metaheuristic for multi-objective optimization problems: The multi-objective vortex search algorithm. Inform Sci. 2017;402:124–48. doi: 10.1016/j.ins.2017.03.026.

[9] Fakhouri HN, Ishtaiwi A, Makhadmeh SN, Al-Betar MA, Alkhalaileh M. Novel hybrid crayfish optimization algorithm and self-adaptive differential evolution for solving complex optimization problems. Symmetry. 2024;16(7):927. doi: 10.3390/sym16070927.

[10] Ayyarao EA. War strategy optimization algorithm: A new effective metaheuristic algorithm for global optimization. IEEE Access. 2022;10:25073–105. doi: 10.1109/ACCESS.2022.3153493.

[11] Abdel-Basset M, Mohamed R, Zidan M, Jameel M, Abouhawwash M. Mantis search algorithm: A novel bio-inspired algorithm for global optimization and engineering design problems. Comput Methods Appl Mech Eng. 2023;415:116200. doi: 10.1016/j.cma.2023.116200.

[12] Abdollahzadeh B, Gharehchopogh FS, Mirjalili S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. Comput Ind Eng. 2021;158:107408. doi: 10.1016/j.cie.2021.107408.

[13] Wang Z, Schafer BC. Machine learning to set meta-heuristic specific parameters for high-level synthesis design space exploration. In: 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE; 2020. p. 1–6. doi: 10.1109/DAC18072.2020.9218674.

[14] Talbi EG. Machine learning into metaheuristics: A survey and taxonomy. ACM Comput Surveys (CSUR). 2021;54(6):1–32. doi: 10.1145/3459664.

[15] Karimi-Mamaghan M, Mohammadi M, Meyer P, Karimi-Mamaghan AM, Talbi EG. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. Europ J Operat Res. 2022;296(2):393–422. doi: 10.1016/j.ejor.2021.04.032.

[16] Soler-Dominguez A, Juan AA, Kizys R. A survey on financial applications of metaheuristics. ACM Comput Surveys (CSUR). 2017;50(1):1–23. doi: 10.1145/3054133.

[17] Moscato P. On evolution, search, optimization, genetic algorithms and martial arts-towards memetic algorithms. C3P Rep. 1989;826:37.

[18] Fogel LJ, Owens AJ, Walsh MJ. Artificial intelligence through simulated evolution. New York: Wiley; 1966. doi: 10.1109/9780470544600.ch7.

[19] Rechenberg I. Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution. Stuttgart: Frommann-Holzboog; 1973.

[20] Holland J. Adaptation in natural and artificial systems: An introductory analysis with application to biology, control and artificial intelligence. Ann Arbor: University of Michigan Press; 1975. https://ci.nii.ac.jp/naid/10019844035/en/.

[21] Hillis WD. Co-evolving parasites improve simulated evolution as an optimization procedure. Phys D Nonlinear Phenom. 1990;42(1–3):228–34. doi: 10.1016/0167-2789(90)90076-2.
[22] Reynolds RG. An introduction to cultural algorithms. In: Proceedings of the 3rd Annual Conference on Evolutionary Programming. Singapore: World Scientific Publishing; 1994. p. 131–9. doi: 10.1142/9789814534116.
[23] Koza J. Genetic programming as a means for programming computers by natural selection. Stat Comput. 1994;4(2):87–112. doi: 10.1007/BF00175355.
[24] Mühlenbein H, Paaß G. From recombination of genes to the estimation of distributions I. Binary parameters. In: Lecture Notes in Computer Science. Berlin: Springer; 1996. p. 178–87. doi: 10.1007/3-540-61723-X_982.
[25] Storn R, Price K. Differential evolution – simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997;11(4):341–59. doi: 10.1023/A:1008202821328.
[26] Ryan C, Collins J, Neill MO. Grammatical evolution: Evolving programs for an arbitrary language. In: Lecture Notes in Computer Science. Berlin: Springer; 1998. p. 83–96. doi: 10.1007/BFb0055930.
[27] Ferreira C. Gene expression programming in problem solving. In: Soft Computing and Industry. London: Springer London; 2002. p. 635–53. doi: 10.1007/978-1-4471-0123-9_54.
[28] Han KH, Kim JH. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans Evol Comput. 2002;6(6):580–93. doi: 10.1109/TEVC.2002.804320.
[29] Atashpaz-Gargari E, Lucas S. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In IEEE Congress on Evolutionary Computation; 2007. p. 4661–7. doi: 10.1109/CEC.2007.4425083.
[30] Civicioglu P. Transforming geocentric Cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput Geosci. 2012;46:229–47. doi: 10.1016/j.cageo.2011.12.011.
[31] Civicioglu P. Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput. 2013;219(15):8121–44. doi: 10.1016/j.amc.2013.02.017.
[32] Salimi H. Stochastic fractal search: A powerful metaheuristic algorithm. Knowl Based Syst. 2015;75:1–18. doi: 10.1016/j.knosys.2014.07.025.
[33] Dhivyaprabha TT, Subashini P, Krishnaveni M. Synergistic fibroblast optimization: A novel nature-inspired computational algorithm. Frontiers Inf Technol Electronic Eng. 2018;19:815–33. doi: 10.1631/FITEE.1601553.
[34] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B. 1996;26(1):29–41. doi: 10.1109/3477.484436.
[35] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science; 1995. p. 39–43. doi: 10.1109/MHS.1995.494215.
[36] Kennedy J, Eberhart RC. A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. vol. 5; 1997. p. 4104–8. doi: 10.1109/ICSMC.1997.637339.
[37] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optimiz. 2007;39(3):459–71. doi: 10.1007/s10898-007-9149-x.
[38] Yang XS. Engineering optimizations via nature-inspired virtual bee algorithms. Berlin: Springer; 2005. p. 317–23. doi: 10.1007/11499305_33.
[39] Zelinka I. SOMA self-organizing migrating algorithm. In: Berlin: Springer; 2004. p. 167–217. doi: 10.1007/978-3-540-39930-8_7.
[40] Eusuff MM, Lansey KE. Optimization of water distribution network design using the shuffled frog leaping algorithm. J Water Resources Plan Manag. 2003;129(3):210–25. doi: 10.1061/(ASCE)0733-9496(2003)129:3(210).
[41] Martin R, Wicker S. Termite: a swarm intelligent routing algorithm for mobile wireless ad-hoc networks. Berlin: Springer; 2006. p. 155–84. doi: 10.1007/978-3-540-34690-6_7.
[42] Li X. An optimizing method based on autonomous animats: Fish-Swarm algorithm. Syst Eng Pract. 2002;22(11):32–8.
[43] Yang XS. A new metaheuristic Bat-inspired algorithm. Berlin: Springer; 2010. p. 65–74. doi: 10.1007/978-3-642-12538-6_6.
[44] Yang XS, Deb S. Cuckoo search: recent advances and applications. Neural Comput Appl. 2014;24(1):169–74.
[45] Simon D. Biogeography-based optimization. IEEE Trans Evolut Comput. 2008;12(6):702–13. doi: 10.1109/TEVC.2008.919004.
[46] Mehrabian AR, Lucas C. A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform. 2006;1(4):355–66. doi: 10.1016/j.ecoinf.2006.07.003.
[47] Chandrashekar C, Krishnadoss P, Kedalu Poornachary V, Ananthakrishnan B, Rangasamy K. HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing. Appl Sci. 2023;13(6):3433. doi: 10.3390/app13063433.
[48] Suid MH, Ahmad MA, Nasir ANK, Ghazali MR, Jui JJ. Continuous-time Hammerstein model identification utilizing hybridization of augmented sine cosine algorithm and game-theoretic approach. Results Eng. 2024;23:102506. doi: 10.1016/j.rineng.2024.102506.
[49] Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-Qaness MA, Gandomi AH. Aquila optimizer: A novel meta-heuristic optimization algorithm. Comput Ind Eng. 2021;157:107250. doi: 10.1016/j.cie.2021.107250.
[50] Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH. Marine predators algorithm: A nature-inspired metaheuristic. Expert Syst Appl 2020;152:113377. doi: 10.1016/j.eswa.2020.113377.
[51] Mirjalili S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 2016;27(4):1053–73. doi: 10.1007/s00521-015-1920-1.
[52] Kelly M, Longjohn R, Nottingham K. The UCI machine learning repository; 2024. https://archive.ics.uci.edu.