IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 March 2024, accepted 11 April 2024, date of publication 18 April 2024, date of current version 26 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3390670

RESEARCH ARTICLE

Leaf in Wind Optimization: A New Metaheuristic Algorithm for Solving Optimization Problems

NING FANG[1] **AND QI CAO**[2]
[1]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[2]Longyan Tobacco Industrial Company Ltd., Longyan 364021, China

Corresponding author: Qi Cao (caoqi_82@163.com)

ABSTRACT This study introduces a novel metaheuristic algorithm, Leaf in Wind Optimization, inspired by the natural phenomenon of falling leaves in the wind. The proposed method simulates the motion response of leaves in varying intensities of wind by establishing models for light wind-driven blades and strong wind-driven blades. The algorithm incorporates motion modes of linear translation and spiral rotation induced by wind, offering a hybrid search framework suitable for both strategies. This approach enables enhanced exploration and exploitation of the search space. The algorithm's performance was evaluated using three challenging benchmark test sets, CEC 2017, CEC 2019 and CEC 2022, as well as an engineering practical problem. Its effectiveness was assessed through comparison with 10 random optimization algorithms, namely: Tree Seed Algorithm, Multi-Verse Optimizer, Salp Swarm Algorithm, Artificial Ecosystem-based Optimization, Hunger Games Search, Fox Optimizer, Spider Wasp Optimizer, AOBLMOA, Enhanced Snake Optimizer, and IbI Logic Algorithm. In the comprehensive testing conducted, the proposed algorithm consistently outperformed other optimizers in approximately 82% of comparisons. Through examination of convergence curves and statistical data, it is evident that Leaf in Wind Optimization demonstrates superior potential compared to the alternative optimizers under consideration.

INDEX TERMS Metaheuristic algorithm, optimization, benchmark functions.

I. INTRODUCTION

Metaheuristic optimization algorithms are effective tools for addressing optimization problems characterized by high dimensionality and strong interdependencies [1], [2]. These algorithms have emerged as prominent approaches in the field of optimization, employing random processes and group search strategies to identify approximate solutions that closely approximate the optimal solution within a reasonable computational timeframe. Consequently, they have been successfully employed in diverse application domains. Song et al. introduced the APSO/DU method as a solution to portfolio optimization problems [3]. Shinde and Pawar employed the TLBO method to address the Trajectory Optimization of an Industrial Robot problem [4]. Kumar and Sharma also utilized the TLBO method to tackle the Wind Farm Layout Optimization Problem [5]. Liu and Zhang developed an Immersive Intelligent English Teaching (IET) system by employing the Neural Network (NN) algorithm of Particle Swarm Optimization (PSO) [6]. Osei-Kawakye et al. proposed the HPSOCSA-CIS algorithm to solve the feature selection problem [7]. With the proliferation of data volume and escalating intricacy of computational tasks, there is a growing need for enhanced computing efficiency. Optimization algorithms must not only address diverse practical complex problems [8], [9], [10], but also strive for superior optimization strategies to minimize computational complexity, expedite convergence, and circumvent local optima. This is crucial for enhancing computational efficiency and successfully accomplishing larger and more intricate computational tasks.

Many novel metaheuristic optimization algorithms have been developed, drawing inspiration from established algorithms, such as particle swarm optimization (PSO) [11], Artistic Bee Colony (ABC) [12], and bat-inspired algorithm (BA) [13]. Researchers have adopted diverse approaches,

The associate editor coordinating the review of this manuscript and approving it for publication was Bijoy Chand Chatterjee.

converting distinct logic, behaviors, and states into search strategies and employing mathematical models to represent them [14], [15], [16], [17]. Moreover, the performance of these newly proposed optimization algorithms continues to improve. In this vein, Sunday Oladejo et al. introduced a metaheuristic algorithm called the deep-sleep optimizer (DSO) [18]. The deep-sleep optimizer emulates the sleep patterns of humans and replicates the fluctuations in stress levels during sleep. In their study, Trojovská et al. introduced the Drawer Algorithm (DA), which simulates the selection of objects from various drawers to achieve an optimal combination [19]. Rezvani et al. proposed the Bedbug Metaheuristic Algorithm (BMHA), which is inspired by the clustering behavior of bedbugs in nature, both static and dynamic, and models the social interactions among bedbugs during their search for food [20]. Givi et al. introduced the Red Panda Optimization (RPO) algorithm, which emulates the innate actions of small pandas in their natural habitat, including foraging and seeking refuge in trees for rest [21].

Natural phenomena offer valuable insights into optimization algorithms [22]. Numerous researchers have observed and translated these phenomena into mathematical models to optimize problems, yielding highly satisfactory outcomes. Consequently, these optimization algorithms present influential concepts that foster the development of novel algorithms. Several optimization algorithms have been proposed in recent years, all of which are based on natural phenomena.

The Tree-seed algorithm (TSA) establishes a correspondence between the positions of trees and seeds within a D-dimensional search space, thereby providing a potential solution to an optimization problem [23]. Throughout the iterative process, one or more seeds were consistently generated from the trees and the positions of the trees were substituted with those of the superior seed positions. The generation of new seeds in each iteration is regulated by a parameter known as the search trend control. As the ST value decreases, the rate of convergence decreases, whereas the capacity for a global search strengthens.

Cheraghalipour et al. introduced the Tree Growth Algorithm (TGA) as a means of simulating competitive dynamics among trees in their quest for light and nutrients [24]. The algorithm partitions the population into four distinct groups, with one designated as the "best tree group." These superior trees, benefiting from favorable growth conditions, afforded the opportunity to thrive further. Once their light requirements are satisfied, their competition shifts towards acquiring food resources. Given the relatively sluggish growth rate of trees, those that exhibit robust development tend to possess greater height and smoother morphology, ultimately reflecting their advanced age compared with their counterparts. This phenomenon can be attributed to the deceleration of the growth rate in aging trees compared to younger trees, as well as the intensified competition for nutrients at the root level. Another group of trees engages in light intensity competition, whereby they reposition themselves towards the nearest optimal tree to maximize exposure to light. Additionally, a subset of weak trees exhibiting slow growth or other factors outlined in the preceding section was selectively removed and replaced by forest farmers with new trees. In the breeding group, the superior trees commence the process of reproduction and generate new plants because of their favorable growth. Given their proximity to the mother tree, they inherit certain factors associated with a specific location.

Wind-driven optimization (WDO) technology is an iterative heuristic global optimization algorithm that operates using a population-based approach [25]. Its fundamental principle involves the exploration of infinitesimal gas blocks in one-dimensional space, adhering to Newton's second law of motion, which is commonly employed to describe the movement of gas blocks within Earth's atmosphere. In contrast to particle-based algorithms of similar nature, WDO incorporates supplementary terms within its velocity update equations, encompassing forces such as gravity and Coriolis forces. This inclusion enhances the resilience of the algorithm and offers additional flexibility for precise adjustments. According to Bayraktar et al., the conventional Wind Driven Optimization (WDO) algorithm offers a straightforward and efficient metaheuristic optimization approach. However, the inherent coefficients of the algorithm introduce unforeseen intricacy, particularly for inexperienced users. To address this complexity and enable automated coefficient selection, researchers have proposed an adaptive wind-driven optimization (AWDO) method [26].

Shehadeh introduced a novel metaheuristic optimization approach known as the Chernobyl Disaster Optimizer (CDO) [27]. The underlying principles and concepts of this method are inspired by the catastrophic explosion of the Chernobyl nuclear reactor core. Within the CDO framework, radioactivity is generated because of nuclear instability, leading to the emission of various types of radiation from the atomic nucleus. Among these, the γ, β, and α particles were the most prevalent. These particles traverse from the point of explosion (high pressure) to the point of lower pressure (human foothold), posing a significant risk to human health. CDO exhibits the ability to adhere to the human body following a nuclear explosion while also simulating the effects of nuclear radiation.

Luo drew inspiration from the natural water flow pattern and introduced a Water Flow Optimization Algorithm (WFO) [28]. This algorithm emulates the hydraulic behavior of water particles as they move from higher to lower ground by employing two operators: 1) laminar flow and 2) turbulence. Similarly, Shehadeh et al. merged the Gravity Search Algorithm (GSA) with the Sperm Swarm Optimization (SSO) algorithm to create a hybrid optimization algorithm known as hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) [29]. The fundamental concept of this algorithm entails the amalgamation of the utilization capability found in SSO and the exploration capability present in GSA, thereby integrating the advantages of both algorithms. Zhao et al. introduced an optimization algorithm known as artificial ecosystem-based optimization(AEO), which draws inspiration from the energy flow within Earth's ecosystems [30]. The AEO, as a population-based optimization algorithm, emulates three distinct behaviors exhibited by living organisms: production, consumption, and decomposition. In their study, Azizi et al. introduced the Energy Valley Optimizer (EVO) algorithm, a newly developed metaheuristic algorithm that draws inspiration from sophisticated physical principles pertaining to stability and various particle-decay modes [31].

The author draws inspiration from these algorithms to utilize the movement patterns of fallen leaves in the wind to address continuous optimization problems. LiWO emerged as a sudden epiphany for the author, prompting them to closely observe the behavior of fallen leaves under windy conditions. Despite the consistent outcome of leaves descending to the ground, the varying intensities of gentle or strong winds result in diverse trajectories that ultimately lead to their dispersal within a defined spatial range. These phenomena serve as the fundamental basis of LiWO. Within the LiWO framework, fallen leaves, referred to as candidate solutions, manifest distinct associations with wind across various stages of descent, thereby facilitating the enhanced exploration and exploitation of the search space. The efficacy of LiWO was assessed using an engineering problem and three arduous benchmarks derived from the Evolutionary Computing Conference (CEC), namely CEC 2017, CEC 2019, and CEC 2022. This study aimed to assess the optimization ability of LiWO by comparing it with 10 other randomly selected optimization techniques. The selected algorithms for comparison encompass recently published algorithms as well as algorithms associated with trees. Based on the analysis of the convergence curve and statistical data, LiWO emerged as the most promising among the competing optimizers. The primary contributions of this study are as follows:

- A novel metaheuristic algorithm, named Leaf in Wind Optimization (LiWO), was introduced, drawing inspiration from the factors such as wind force, translation, and rotation that influence the movement of falling leaves in the wind.
- Drawing upon the observed characteristics of leaf movement, the breeze driven leaf strategy and the strong wind driven leaf strategy was modeled to effectively complete global search and local search in LiWO.
- Hybrid search behavior has been proposed, wherein the movement of individual leaf is comprised of one or more superimposed simple motions, enabling a more efficient exploration and utilization of the search space.
- By utilizing three rigorous CEC benchmarks and an engineering problem to assess the performance of LiWO in addressing a variety of optimization problems with diverse characteristics, experimental findings demonstrate that LiWO outperforms other comparable optimization algorithms.

The subsequent section of this article is organized as follows. The second section presents a comprehensive overview of the proposed LiWO, encompassing its implications, mathematical models, and pseudocode. The third section presents the results and discussion, wherein the new algorithm was evaluated on a test set comprising 51 test programs. By employing an extensive test set encompassing all the genuine parameter single-objective optimization test sets from CEC2017, CEC2019, CEC2022 and an engineering problem to validate the algorithm, the likelihood of encountering overfitting issues is somewhat mitigated when compared with the utilization of a limited number of test sets. Finally, Section IV concludes the paper and offers suggestions for prospective research.

II. LEAF IN WIND OPTIMIZATION

A. INSPIRATION

Many plant-based optimization algorithms have established corresponding optimization algorithms by simulating certain growth characteristics or effects of plants. This indicates that there are many adaptive or self-regulated phenomena in natural phenomena. Falling leaves in the wind are one of the natural phenomena. Under the influence of wind, falling leaves have obvious dynamic characteristics and can perfectly complete the process of falling to the ground. This is inevitably an adaptive motion.

It is imperative to acknowledge the intricate nature of leaf descent. Indeed, in everyday circumstances, individuals witness diverse patterns of motion when releasing lightweight and delicate objects such as leaves. Occasionally, the trajectory of a descending piece of paper may appear haphazard, while at other times it exhibits a high degree of regularity. This can be attributed to the multitude of external forces, including lift, friction, and gravity, which influence the process of leaves descending to the ground. Leaves undergo continuous postural adjustments and descend from a considerable height under the influence of external forces. The act of landing represents the ultimate state of a leaf, whereas falling signifies sequential stages of leaf movement. Drawing inspiration from this phenomenon, it is plausible to perceive the descent process as an optimization procedure.

The motion of the descending foliage demonstrates intriguing oscillations, rolling, and spiral motions, alongside unpredictable transitions between these phases, encompassing random occurrences, sudden changes in velocity, and similar phenomena [32], [33]. Numerous computational methods exist for replicating the state of falling leaves, with a significant portion involving the calculation of forces and moments exerted on the leaves [34], [35], [36]. In this regard, the leaves are regarded as rigid entities with a negligible thickness assumed to be zero, which is solely characterized by their length and mass. By applying the principles of fluid mechanics, the forces and moments exerted by the wind on the leaves can be determined. Subsequently, the degree of positional and attitudinal changes in leaves in a three-dimensional space can be calculated, enabling a comprehensive understanding of the leaf falling process over time. Optimization problems can be effectively addressed by simulating diverse leaf movements during the descent.

B. GENERAL OPTIMIZATION PROBLEMS

Optimization refers to the systematic procedure of identifying the most favorable solution from a pool of potential solutions in an optimization problem guided by specific criteria. The realm of optimization encompasses a variety of problems that entail either the maximization or minimization of processes. In the context of Equations (1) and (2), where S denotes the search space, $F, F \in S$, represents the collection of satisfactory solutions for S, f denotes the objective function or fitness, and the act of minimizing or maximizing entails the identification of $X \in F$. Equations (1) and (2) serve to define minimization and maximization, respectively.

$$f(X^*) \leq f(X) \quad \forall X \in F \tag{1}$$

$$f(X^*) \geq f(X) \quad \forall X \in F \tag{2}$$

Group optimization technology utilizes a stochastic generation process to generate a candidate solution $X_i, i = 1, 2, \ldots, N$, from a search space S, which is bounded by a lower bound LB and an upper bound UB. The size of the search space is contingent upon the particular optimization problem at hand. In addition, specific rules are employed to facilitate the movement of N individuals within this search space. The movement process terminates when either a predetermined number of iterations or the convergence conditions are satisfied. During each movement, the global best solution, X_{gb}, is determined by applying Equation (3) and Equation (4) to the population.

$$X_{gb} = \min\{f(X_i)\} \tag{3}$$

$$X_{gb} = \max\{f(X_i)\} \tag{4}$$

In a broad context, the maximization problem can be transformed into a minimization problem using Equation (5). Consequently, the ensuing discourse focuses on minimizing this problem.

$$X_{gb} = \max\{f(X_i)\} = \min\{-f(X_i)\} \tag{5}$$

C. ALGORITHM

This section presents leaf optimization algorithms that draw inspiration from leaf movement. In this context, the solution space is utilized as the domain for leaf motion, with each leaf representing an individual engaged in optimization. In practical scenarios, various factors, including gravity and wind, contribute to the diverse movements of the leaves, such as fluttering, rolling, and spiraling. However, within the optimized motion space, our focus is solely on the influence of wind, as we posit it to be the primary factor responsible for the continuous movement of leaves in space. When leaves manifest sporadically within a dynamic environment, two distinct types of winds emerge: mild breeze and forceful gust. These winds materialize haphazardly, propelling the motion of the leaves, albeit never concurrently.

Let X denote the set of all leaves, N denote the total count of leaves, and D represent the dimensionality of the leaf motion space, which corresponds to the dimensionality of the solution vector for the optimization problem. The interrelations among these three variables are expressed as Equation (6). Leaves exhibit continuous movement owing to the impact of either mild or intense winds, with the origin of the wind being the highest fitness position X_{gb} attained by the leaves during their motion, while intense winds are assumed to be randomly generated within the spatial domain. The subsequent sections present a separate introduction to the distinct impacts of the two wind types on the leaves.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^D \\ x_2^1 & x_2^2 & \cdots & x_2^D \\ \vdots & \vdots & \vdots & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^D \end{bmatrix} \tag{6}$$

1) BREEZE DRIVEN LEAF STRATEGY

In the presence of a mild breeze, the wind force D_{wind}^t uniformly impacts all the leaves and can be mathematically represented as.

$$D_{wind}^t = X_{gb} - X_{m_1} \tag{7}$$

where X_{m_1} represents randomly selected leaves.

Wind force D_{wind}^t serves as a representation of both the direction and distance that each individual leaf traverses. To accurately depict the inherent randomness in the motion of each leaf, Equation (8) is employed to update the positional coordinates of the leaves.

$$X_{new}^1 = X_i + r_1 \cdot D_{wind}^t \tag{8}$$

where r_1 denotes a uniformly distributed random variable in the interval [0, 1], X_i represents the i-th leaf, and X_{new}^1 represents the updated position of the i-th leaf resulting from the impact of a mild breeze.

Leaves propelled by a mild breeze exhibit both translational and spiral motions. However, spiral motion was not observed in any dimension within the multidimensional space. To account for this, a probability mechanism was introduced. When the probability $p_1 = 0.1$, Equations (9) and (10) are employed to signify that the i-th leaf, following the translational motion induced by Equation (8), experiences spiral motion in the j-th dimension as a result of the influence exerted by the gentle breeze.

$$x_{new}^{1,j} = x_{new}^{1,j} + C_1 \cdot d_{wind}^s \cdot \sin(\varphi) \cdot \varphi \tag{9}$$

$$d_{wind}^s = \left| x_{gb}^j - x_{m_2}^j \right| \tag{10}$$

where φ is a uniformly distributed random number within the interval $[0, 2\pi]$, denoting the deviation in the rotational motion of the leaf. X_{m_2} signifies other leaves that are randomly chosen, whereas d^s_{wind} represents the magnitude of the impact of the breeze on the spiral motion in the j-th dimension. C_1 is the factor that determines the adaptive amplitude of movement, which is determined by Equations (11) and (12).

$$C_1 = \exp(\omega^5) - 1 \tag{11}$$

$$\omega = \omega_{\max} - t \cdot ((\omega_{\max} - \omega_{\min})/T_{\max}) \tag{12}$$

where $\omega_{\max}$ takes the value of 0.9 while $\omega_{\min}$ is assigned a value of 0, with these specific values being derived from the numerical parameters utilized in the classical Particle Swarm Optimization (PSO) [11] algorithm to characterize the adaptive adjustment mechanism. t represents the current number of iterations, and $T_{\max}$ represents the total number of iterations.

If the j-th dimension of the newly calculated position X^1_{new} surpasses the range of the search space, the data in that particular dimension will be reverted back to its initial state prior to the movement, as depicted in Equation (13).

$$x^{1,j}_{new} = x^j_i \tag{13}$$

2) STRONG WIND DRIVEN LEAF STRATEGY

In contrast to the effect of gentle breezes on foliage, it has been postulated that the repercussions of forceful winds on leaves manifest in the alteration of a specific unidimensional spatial location. It should be noted that not all dimensions were affected, but only one dimension. Initially, the unaltered state of the novel position of the i-th leaf is maintained.

$$X^2_{new} = X_i \tag{14}$$

Next, a dimension j impacted by strong winds was chosen at random, causing the position $x^{2,j}_{new}$ of the leaves to experience a unidirectional displacement. The resulting movement can be represented using Equations (15) and (16):

$$x^{2,j}_{new} = x^{2,j}_{new} + d^{st}_{wind} \cdot r_2 \tag{15}$$

$$d^{st}_{wind} = x^j_{m_3} - x^j_i \tag{16}$$

where r_2 is a uniformly distributed random number within the range of [0, 1], and d^{st}_{wind} represents the magnitude of strong winds influenced by the i-th leaf and other randomly selected leaves X_{m_3}.

To ensure the preservation of leaf position diversity in the face of strong winds, a probability mechanism was employed to randomly reset the position in the j-th dimension, as shown in Equation (17).

$$x^{2,j}_{new} = \begin{cases} LB^j + r_3 \cdot (UB^j - LB^j) & if\ r_4 < p_2 \\ x^{2,j}_{new} & otherwise \end{cases} \tag{17}$$

where p_2 represents the reset probability, with a value of 0.1. r_3 and r_4 are two uniformly distributed random numbers within the interval [0, 1].

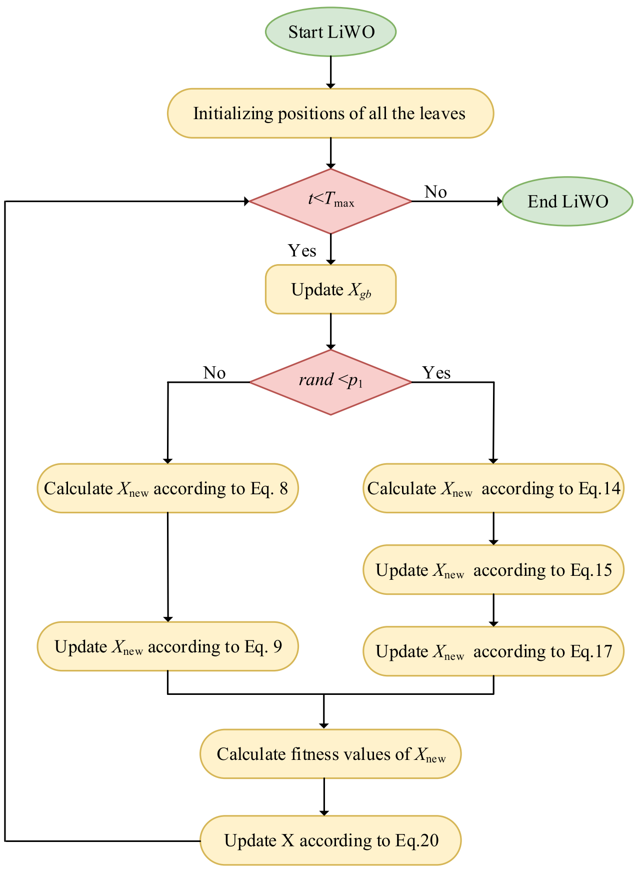
Start LiWO
Initializing positions of all the leaves
$t<T_{\max}$
No
End LiWO
Yes
Update X_{gb}
No
$rand<p_1$
Yes
Calculate X_{new} according to Eq. 8
Calculate X_{new} according to Eq.14
Update X_{new} according to Eq.15
Update X_{new} according to Eq. 9
Update X_{new} according to Eq.17
Calculate fitness values of X_{new}
Update X according to Eq.20

FIGURE 1. Flowchart of LiWO.

3) FLOWCHART

This section presents the fundamental procedure for LiWO. LiWO employs only two strategies for updating the leaf positions, rendering the overall algorithm process relatively straightforward. Initially, all leaves are uniformly distributed within the search space; thus, Equation (18) is employed for the initialization of the leaf positions.

$$X_i = LB + r_5 \cdot (UB - LB) \tag{18}$$

where r_5 is a uniformly distributed random numbers within the [0, 1] interval.

Figure 1 illustrates the comprehensive procedure for LiWO.

During each iteration, the leaves in space are subjected to the influence of strong and gentle winds, which occur randomly through a probabilistic mechanism. Assuming a probability p_3 of generating a gentle breeze with a specific value of 0.3, the new position of the leaves will be updated in a random manner, following Equation (19).

$$X_{new} = \begin{cases} X^1_{new} & if\ r_6 < p_3 \\ X^2_{new} & otherwise \end{cases} \tag{19}$$

where r_6 is a uniformly distributed random numbers within the [0, 1] interval.

The determination of the final update method for the leaf position is based on the fitness value, as illustrated in the subsequent equation.

$$X_i = \begin{cases} X_{new} & if\ f(X_{new}) < f(X_i) \\ X_i & otherwise \end{cases} \tag{20}$$

The update rule for the global optimal solution resembles the update method employed for leaf position, as depicted in the subsequent equation.

$$X_{gb} = \begin{cases} X_{new} & if\ f(X_{new}) < f(X_{gb}) \\ X_{gb} & otherwise \end{cases} \tag{21}$$

The pseudocode of LiWO is as follows in Algorithm 1.

Algorithm 1 Pseudocode of LiWO

```
Start LiWO
1.  Set the number of leaves ($N$) and the total number of iterations ($T_{\max}$)
2.  Initialization process of leaves' locations $X$
3.  For $t = 1$: $T_{\max}$
4.      Update $X_{gb}$ according to Eq.21
5.      If $rand < p_1$
6.        For $i = 1$: $N$
7.          Calculate new location of the $i$th leaf using Eq. 8
8.          For $j = 1$: $D$
9.            If $rand < p_2$
10.             Calculate new location of the $i$th leaf using Eq.9
11.           End If
12.         End for
13.       End for
14.     Else
15.       For $i = 1$: $N$
16.         Assign new location of the $i$th leaf using Eq.14
17.         Randomly choose $j$ and calculate new location of the $j$th dimension of the $i$th leaf using Eq.15
18.         If $rand < p_3$
19.           Calculate new location of the $j$th dimension of the $i$th using Eq.17
20.         End if
21.       End for
22.     End if
23.     Calculate fitness values for all new leaves
24.     Update $X$ according to Eq.20
25. End for
End LiWO
```

4) COMPUTATIONAL COMPLEXITY

The computational complexity of the LiWO is primarily contingent on the computational complexity of its fitness function. The fitness function was evaluated during the population initialization and leaf-position update stages. During initialization, the fitness function is computed approximately N times, with a complexity denoted as $O(N)$. Complexity arising solely from the fitness function was not considered in this analysis. The computational complexity of the fitness function during the leaf position update stage is contingent on both the total number of iterations $T_{\max}$ executed by the algorithm and the population size N represented by the number of leaves. Within each iteration of $T_{\max}$, the number of function calculations induced by N leaves leads to the complexity of $O(N \times T_{\max})$ for the updated solution. The cumulative computational complexity, encompassing both the initialization and update processes, amounts to $O(N \times (T_{\max} + 1))$ for LiWO. The subsequent chapters will employ diverse benchmark functions to assess and validate the efficacy of LiWO.

III. SIMULATION AND ANALYSIS

A. SIMULATION SETTINGS

This section presents a comprehensive analysis of the LiWO algorithm. In the present study, 51 benchmark test sets comprising real parameter single-objective optimization problems were sourced from the CEC2017, CEC2019, and CEC2022 test sets to validate the algorithm. These test sets have gained significant popularity for evaluating the efficacy of optimization algorithms. Specifically, the CEC2017 test set consisted of 29 benchmarks, CEC2019 test set encompassed 10 benchmarks, and CEC2022 test set comprised 12 benchmarks. To facilitate the presentation of the outcomes, the benchmark function of the CEC2017 test set was labeled as AF1-AF29. Specifically, AF1 and AF2 represent unimodal functions, AF3-AF9 denote multimodal functions, AF10-AF19 signify hybrid functions, and AF20-AF29 correspond to composite functions. Similarly, the benchmark function for the CEC2019 test set is denoted as BF1-BF10. Furthermore, the CEC2022 test set function is designated as CF1-CF12, where CF1 represents a unimodal function, CF2-CF5 represent multimodal functions, CF6-CF8 denote mixed functions, and CF9-CF12 correspond to composite and multimodal functions. Unimodal functions are distinguished by the absence of local minima with only a global minimum present, thereby serving as a means of assessing the convergence capability of the algorithm. Conversely, multimodal functions possess local extremum points, which are employed to evaluate the capacity of the algorithm to escape from local optima. A mixed function, on the other hand, encompasses three or more benchmark functions following rotation or displacement, with each sub function being assigned a specific weight. Composite functions exhibit the distinctive feature of comprising a minimum of three mixed or benchmark functions that undergo rotation and shifting operations. Each subfunction within the composite function is accompanied by an added bias value and weight. Consequently, the presence of these composite functions increases the complexity of the optimization algorithm.

The experiments were carried out using MATLAB R2020a on a personal computer equipped with an Intel (R) Core (TM) i5-7200U CPU @ 2.50GHz 2.71 GHz and a Microsoft Windows 10 Enterprise 64-bit operating system.

For the algorithm validation, 30 independent runs were conducted to account for the randomness of the algorithm. The conclusions were derived from the results of these runs, including the best solution (BEST), mean (AVG), and standard deviation (STD) obtained for each iteration. This study examines the various dimensions of the test set functions, including the 10, 30, 50, and 100 dimensions of the CEC2017 test set, the 10 dimensions of the CEC2019 test set, and the 10 and 20 dimensions of the CEC2022 test set. To ensure fairness in the comparison, each run sets the population size and maximum function computation times of all algorithms to 30 and 5000 × 30, respectively. Consequently, each iteration consisted of only 30 function computations, equivalent to the particle swarm size, resulting in a total of 5000 iterations.

To ascertain the superiority of the algorithm, a comparative analysis was conducted using ten recently published algorithms: Tree-seed algorithm (TSA) [23], Multi-verse optimizer (MVO) [37], Salp swarm algorithm (SSA) [38], Artificial ecosystem-based optimization (AEO) [30], Hunger games search (HGS) [39], Fox optimizer (FOX). [40], Spider wasp optimizer (SWO) [17], AOBLMOA [41], Enhanced snake optimizer (ESO) [42], and IbI Logics Algorithm (ILA) [14]. It is important to note that the source code for each algorithm has been made publicly available by its respective authors. The parameters utilized during the computation process are derived from the original code, with specific values detailed in Table 1.

TABLE 1. The algorithm information and parameter settings.

No.	Algorithms	Parameters settings
1	TSA	Search tendency (ST) : 0.1
2	MVO	Wormhole existence probability (WEP) : Min(WEP) =0.2 and Max(WEP) = 1 Exploitation accuracy over the iterations: 6
3	SSA	NA
4	AEO	r_1, r_2 andr_3: rand (0, 1)
5	HGS	Parameter l: 0.08
6	FOX	Variable c_1: 0.18 Variable c_2: 0.82
7	SWO	Trade-off probability TR: 0.3 Crossover probability CR: 0.2 Minimum population size $N_{\min}$: 20
8	AOBLMOA	Individual optimal attraction coefficient α_1: 1 Population optimal attraction coefficient α_2: 1.5 Attraction coefficient α_3: 1.5 Gravity coefficient g: decreasing in the interval [0.4, 0.9] Development adjustment parameter α: 0.1 Development adjustment parameter δ: 0.1
9	ESO	Threshold T_1 : 0.25 Threshold T_2 : 0.6 Constant C_1 : 0.5 Constant C_2 : 0.05 Constant C_3 : 2
10	ILA	Number of models : 5 Maximum percentage of iterations in stage 1 : 0.33 Maximum percentage of iterations in stage 2 : 0.33 Minimum boundary for the parameters of IbI : 0.4 Maximum boundary for the parameters of IbI : 0.6 Number of replications for clustering : 10 Number of attempts to converge the classification :100

B. OPTIMIZATION ACCURACY

All the comparison algorithms were implemented on three test set functions, and the results of 30 runs, which can be obtained in the supplementary file, were subjected to Wilcoxon signed-rank tests. The test results with the significance levels of $\alpha = 0.05$are displayed in Tables 2-4, with '+', '≈', and '-' denoting the superiority, approximation, and inferiority of our proposed method compared with the comparison method, respectively. The statistical frequencies of these three outcomes are provided below the tables in the format of 'Win/Similar/Loss' format. The dimensions of the test function are denoted as D10, D20, D30, D50, and D100 in the table. Based on the comparative findings, it is evident that the LiWO algorithm demonstrates commendable performance across all three test sets.

On the CEC2017 test set, the LiWO algorithm achieved 246 victories, 42 draws, and only two losses on the 10-dimensional dataset, demonstrating a noteworthy enhancement in the performance of the LiWO algorithm on low-dimensional data. In the results obtained with a dimension of 30, the LiWO algorithm secured 250 victories, 26 draws, and 14 losses against its opponent, thereby attaining a substantial improvement in performance. Furthermore, as dimensionality increases, the frequency of draws experiences a significant decrease, whereas the number of losses exhibits a slight increase. The aforementioned observation suggests that, as the dimensionality increases, the optimization of the test function becomes more challenging, thereby facilitating a clearer assessment of the relative merits and drawbacks of different algorithms. In the case of a 50-dimensional scenario, the LiWO algorithm emerged victorious in 245 instances, achieved a draw in 29 instances, and suffered defeat in 16. Similarly, in the 100-dimensional scenario, the LiWO algorithm secured victory in 231 instances, experienced a draw in 14 instances, and encountered a defeat in 45 instances. In terms of various dimensions and functions, the LiWO algorithm demonstrates superior performance compared to the other algorithms employed for comparison, thereby signifying a substantial enhancement in performance on the CEC2017 test set. Upon closer examination, the SWO algorithm exhibited commendable competitiveness among all the algorithms utilized for comparison. Notably, the LiWO algorithm performed comparably well in numerous test functions, with a slight increase in the number of successes as the dimensionality increased. This observation suggests that the LiWO algorithm possesses advantages over the SWO algorithm in the lower to medium dimensions. Furthermore, the test results of the MVO, AEO, SSA, and FOX algorithms exhibited marginal enhancements in the context of 100 dimensions. This observation further substantiates the

TABLE 2. Wilcoxon's Signed Rank test under the CEC2017 benchmarks.

Function	TSA	MVO	SSA	AEO	HGS	FOX	SWO	AOBLMOA	ESO	ILA
	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100	10/30/50/100
AF1	+/+/+/+	+/+/+/-	+/+/≈/-	+/+/≈/-	+/+/+/+	+/+/≈/-	≈/≈/≈/-	+/+/+/+	+/+/+/+	+/+/+/+
AF2	≈/+/+/+	+/-/-/-	≈/-/-/+	≈/-/-/-	+/+/+/+	≈/-/-/+	≈/-/-/-	≈/+/+/+	+/+/+/+	+/+/+/+
AF3	+/+/+/+	+/+/+/-	+/+/+/-	≈/+/≈/≈	+/+/+/+	+/+/+/-	≈/≈/≈/-	+/+/+/+	+/+/+/+	+/+/+/+
AF4	+/+/+/+	+/+/+/-	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/≈/-	+/+/+/+	+/+/+/+	+/+/+/+
AF5	≈/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/-/-/+	+/+/+/+	+/+/+/+	+/+/+/+
AF6	+/+/+/+	+/+/+/≈	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/≈	+/+/+/+	+/+/+/+	+/+/+/+
AF7	+/+/+/+	+/+/+/-	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/+/+/-	+/+/+/+	+/+/+/+	+/+/+/+
AF8	≈/-/≈/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/≈/≈/-	+/+/+/+	+/+/+/+	+/+/+/+
AF9	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+
AF10	+/+/+/+	+/+/+/-	+/+/+/-	+/+/+/-	+/+/+/+	+/+/+/-	+/≈/≈/-	+/+/+/+	+/+/+/+	+/+/+/+
AF11	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/-	+/+/+/+	+/+/+/+	≈/+/+/-	+/+/+/+	+/+/+/+	+/+/+/+
AF12	+/+/≈/≈	+/+/+/+	+/+/+/+	+/+/+/≈	+/+/+/+	+/+/+/+	≈/+/+/≈	+/+/+/+	+/+/+/+	+/+/+/+
AF13	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/≈/+/+	+/+/+/+	+/+/+/+	+/+/+/+
AF14	+/+/+/-	+/+/+/+	+/+/+/+	+/+/+/-	+/+/+/+	+/+/+/+	≈/≈/+/-	+/+/+/+	+/+/+/+	+/+/+/+
AF15	+/+/+/+	+/+/+/≈	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/≈/≈/≈	+/+/+/+	+/+/+/+	+/+/+/+
AF16	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/≈/≈/≈	+/+/+/+	+/+/+/+	+/+/+/+
AF17	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	≈/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+
AF18	+/+/+/-	+/+/+/+	+/+/+/+	+/+/+/-	+/+/+/+	+/+/+/+	-/+/+/-	+/+/+/+	+/+/+/+	+/+/+/+
AF19	≈/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/≈/≈/≈	+/+/+/+	+/+/+/+	+/+/+/+
AF20	≈/+/+/+	+/+/+/-	+/+/+/+	≈/+/+/+	+/+/+/+	+/+/+/+	≈/≈/-/-	+/+/+/+	+/+/+/+	≈/+/+/+
AF21	≈/+/+/+	+/+/+/+	+/+/+/+	+/≈/+/+	+/+/+/+	+/+/+/+	≈/≈/+/+	+/+/+/+	+/≈/≈/+	+/≈/+/+
AF22	+/+/+/+	+/≈/≈/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/≈/≈/+	+/+/+/+	+/+/+/+	+/+/+/+
AF23	+/+/≈/+	+/-/-/-	+/-/-/+	≈/+/+/+	+/+/-/+	+/+/+/+	≈/-/-/≈	+/+/+/+	+/-/-/+	-/≈/-/+
AF24	+/+/+/+	+/≈/-/-	+/+/≈/-	+/+/+/-	+/+/+/+	+/-/≈/-	≈/≈/≈/-	+/+/+/+	+/+/+/+	≈/+/+/+
AF25	≈/+/+/+	+/≈/≈/-	+/≈/≈/+	≈/+/+/+	+/+/+/+	+/+/+/≈	≈/≈/≈/≈	+/+/+/+	≈/+/+/+	≈/-/-/+
AF26	≈/≈/+/+	≈/+/-/-	≈/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/≈/+	+/+/+/+	+/+/+/+	+/+/+/+
AF27	+/+/+/+	+/+/≈/-	+/+/≈/-	≈/≈/+/-	+/+/+/+	+/-/≈/-	≈/≈/+/-	+/+/+/+	+/+/+/+	+/+/+/+
AF28	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/≈/≈/+	+/+/+/+	+/+/+/+	+/+/+/+
AF29	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/+	+/+/+/≈	+/+/+/+	+/+/+/+	+/+/+/+
Wilcoxon-D10	21 | 8 | 0	28 | 1 | 0	27 | 2 | 0	23 | 6 | 0	29 | 0 | 0	28 | 1 | 0	9 | 19 | 1	28 | 1 | 0	28 | 1 | 0	25 | 3 | 1
Wilcoxon-D30	27 | 1 | 1	24 | 3 | 2	26 | 1 | 2	26 | 2 | 1	29 | 0 | 0	26 | 0 | 3	10 | 16 | 3	29 | 0 | 0	27 | 1 | 1	26 | 2 | 1
Wilcoxon-D50	26 | 3 | 0	22 | 3 | 4	23 | 4 | 2	26 | 2 | 1	28 | 0 | 1	25 | 3 | 1	12 | 13 | 4	29 | 0 | 0	27 | 1 | 1	27 | 0 | 2
Wilcoxon-D100	26 | 1 | 2	15 | 2 | 12	24 | 0 | 5	19 | 2 | 8	29 | 0 | 0	23 | 1 | 5	8 | 8 | 13	29 | 0 | 0	29 | 0 | 0	29 | 0 | 0

TABLE 3. Wilcoxon's Signed Rank test under the CEC2019 benchmarks.

Function	TSA	MVO	SSA	AEO	HGS	FOX	SWO	AOBLMOA	ESO	ILA
BF1	+	+	+	≈	+	≈	+	≈	≈	≈
BF2	+	+	+	+	+	+	+	+	+	+
BF3	+	+	≈	≈	≈	+	≈	≈	+	+
BF4	+	+	+	+	+	+	≈	+	+	+
BF5	+	+	+	+	+	+	≈	+	+	+
BF6	≈	+	+	+	+	+	≈	+	+	+
BF7	+	+	+	+	+	+	≈	+	+	+
BF8	+	+	+	+	+	+	≈	+	+	+
BF9	+	≈	+	+	+	+	≈	+	+	+
BF10	+	+	+	≈	≈	≈	+	≈	≈	+
Wilcoxon	9 | 1 | 0	9 | 1 | 0	9 | 1 | 0	7 | 3 | 0	8 | 2 | 0	8 | 2 | 0	3 | 7 | 0	7 | 3 | 0	8 | 2 | 0	9 | 1 | 0

notion that each optimization algorithm possesses distinct suitability, thereby affirming the theory that there are no universally superior solutions.

In the CEC2019 test set, the testing dimension was limited to 10 dimensions. The LiWO algorithm exhibited remarkable performance enhancement on this test set, as evidenced by

TABLE 4. Wilcoxon's Signed Rank test under the CEC2022 benchmarks.

Function	TSA	MVO	SSA	AEO	HGS	FOX	SWO	AOBLMOA	ESO	ILA
Dimension	10/20	10/20	10/20	10/20	10/20	10/20	10/20	10/20	10/20	10/20
CF1	≈/+	+/+	≈/≈	≈/≈	+/+	≈/+	≈/≈	≈/+	+/+	+/+
CF2	+/+	+/≈	+/+	≈/≈	+/+	+/+	-/≈	+/+	≈/+	+/+
CF3	≈/+	+/+	+/+	+/+	+/+	+/+	≈/≈	+/+	+/+	+/+
CF4	+/+	+/-	+/≈	+/+	+/+	+/+	+/-	+/+	≈/-	+/+
CF5	+/-	≈/-	+/≈	+/+	+/+	+/+	+/-	+/+	≈/-	+/≈
CF6	+/+	+/+	+/+	+/+	+/+	+/+	≈/+	+/+	+/+	+/+
CF7	+/+	+/+	+/+	+/+	+/+	+/+	≈/+	+/+	+/+	+/+
CF8	+/+	+/+	+/+	+/+	+/+	+/+	≈/≈	+/+	+/+	+/+
CF9	+/+	+/+	+/+	+/+	+/+	+/+	+/+	+/+	+/+	+/+
CF10	+/+	+/+	≈/+	+/+	+/+	+/+	+/+	+/+	+/+	+/+
CF11	≈/≈	+/+	+/+	≈/≈	+/+	+/+	≈/≈	+/+	+/≈	+/+
CF12	-/-	-/≈	≈/+	+/+	≈/+	+/+	+/≈	+/+	+/+	≈/+
Wilcoxon- D10	8 | 3 | 1	10 | 1 | 1	9 | 3 | 0	9 | 3 | 0	11 | 1 | 0	11 | 1 | 0	5 | 6 | 1	11 | 1 | 0	9 | 3 | 0	11 | 1 | 0
Wilcoxon- D20	9 | 1 | 2	8 | 2 | 2	9 | 3 | 0	9 | 3 | 0	12 | 0 | 0	12 | 0 | 0	4 | 6 | 2	12 | 0 | 0	9 | 1 | 2	11 | 1 | 0

its 77 victories, 23 ties, and 0 losses compared with other methods. Notably, the SWO algorithm continued to demonstrate robust competitiveness, although the LiWO algorithm narrowly surpassed its performance. It is worth mentioning that the alternative algorithms can only achieve comparable results to the LiWO algorithm for a maximum of three functions. In the CEC2022 test set, 240 comparisons were conducted, wherein 189 comparisons resulted in wins, 40 comparisons were deemed similar, and only 11 comparisons were found to be inferior to their respective opponents. Notably, the LiWO algorithm exhibited commendable performance on the CEC2022 dataset.

Furthermore, Tables 2-4 assessed the algorithm's optimization capabilities across dimensions of 10, 20, 30, 50, and 100. As the dimensions escalate and optimization challenges become more complex, algorithm scalability becomes a crucial factor. While the SWO algorithm exhibits notable strengths in this regard, the LiWO algorithm also shows competitive performance and delivers superior optimization outcomes in certain functions, including AF9, AF17, AF29, BF10, and CF10.

In conclusion, considering the aspect of optimization accuracy, the LiWO algorithm demonstrated significant competitiveness in effectively addressing unimodal, multimodal, mixed, and composite functions, exhibiting a commendable overall performance.

C. CONVERGENCE ANALYSIS

To gain a comprehensive understanding of the efficacy of the LiWO algorithm, the 30-dimensional outcomes of the CEC2017 test set were specifically chosen for examination. Table 5 presents the mean and standard deviation of the global optimal solution for each algorithm executed 30 times, whereas the bottom of the table displays the mean rank obtained from the Friedman test. Remarkably, the LiWO algorithm achieved the lowest mean rank value among all the algorithms, signifying that our proposed approach outperformed the other comparative algorithms in a statistically significant manner. This outcome aligns with the results of the Wilcoxon signed-rank test. Based on the findings presented in Table 5, it is evident that AF1 and AF2 in the CEC2017 test set function are unimodal functions. The LiWO algorithm can identify the optimal solution for AF1 multiple times within 30 operations, indicating its suitability for addressing functions resembling AF1. Conversely, the performance of the LiWO algorithm on AF2 was not optimal. Conversely, the SSA exhibits contrasting characteristics, repeatedly attaining the optimal solution for AF2, but failing to do so for AF1. The AF3-AF9 function is classified as a multimodal function, and the LiWO algorithm demonstrates a commendable performance in solving it. Although many algorithms are capable of achieving optimal results, our algorithm also achieves results that closely approximates the best solution for functions in which the optimal solution has not been obtained. On the other hand, the AF10-AF19 functions are categorized as mixed functions, and their optimal solutions pose greater difficulty compared to the previous functions. The LiWO algorithm exhibited relatively strong convergence, as evidenced by the superiority of its average of 30 obtained solutions over most comparison algorithms. The comparison algorithms demonstrated their efficacy in handling mixed functions and exhibited exceptional performance in certain functions. The composite function AF20-AF29 poses the highest level of difficulty among all functions, making it challenging to attain the optimal solution through comparative methods. Among the discovered optimal solutions, the LiWO algorithm outperformed other algorithms. In terms of the standard deviation, the TSA algorithm displays remarkable convergence ability, indicating its strong convergence capacity, although its optimization ability is not exceptional. Furthermore, the MVO, SSA, SWO, ESO, and ILA algorithms demonstrated efficacy in achieving favorable outcomes on specific test functions. In conclusion, we assert that the LiWO algorithm is highly competitive.

This part presents the convergence curves of the proposed LiWO algorithm and 10 other optimizers for four categories of test functions (Figure 2): unimodal, multimodal, hybrid, and composite. These test functions are defined

TABLE 5. **Optimization results for CEC2017 test suit with dimension of 30.**

Method	AF1		AF2		AF3		AF4		AF5	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
LiWO	**1.0000E+02**	**0.0000E+00**	2.7217E+02	1.5159E+02	**3.3665E+02**	3.2724E+01	**4.6809E+02**	1.4506E+01	5.0000E+02	2.0618E-03
TSA	6.6816E+03	6.7888E+03	7.4012E+04	1.2822E+04	3.9786E+02	**1.3084E+01**	6.0821E+02	**1.1041E+01**	5.0000E+02	1.8992E-03
MVO	2.4083E+04	8.9520E+03	2.0174E+02	7.3845E-01	3.8744E+02	1.7758E+01	4.9821E+02	2.7332E+01	5.1386E+02	1.1338E+01
SSA	5.5652E+03	5.3570E+03	**2.0000E+02**	**0.0000E+00**	3.9300E+02	2.4805E+01	5.3671E+02	4.6210E+01	5.3629E+02	1.1637E+01
AEO	4.8442E+03	5.2800E+03	2.0002E+02	2.2482E-02	3.7893E+02	3.1387E+01	5.5505E+02	3.6829E+01	5.1958E+02	7.9345E+00
HGS	1.0204E+09	1.0884E+09	2.0722E+04	8.4263E+03	4.3423E+02	3.7568E+01	5.3399E+02	2.9298E+01	5.1200E+02	4.6891E+00
FOX	1.6826E+03	8.6458E+02	2.0001E+02	6.1655E-03	3.6982E+02	2.8406E+01	7.0624E+02	1.8418E+01	5.7254E+02	8.9226E+00
SWO	**1.0000E+02**	1.8257E-07	2.0103E+02	2.6570E+00	3.4450E+02	3.7593E+01	4.7674E+02	1.4881E+01	**5.0000E+02**	**1.9117E-04**
AOBLMOA	5.9077E+03	6.9208E+03	7.4755E+04	1.0993E+04	4.0858E+02	1.3923E+01	6.9371E+02	2.4810E+01	5.5853E+02	5.9165E+00
ESO	1.3683E+06	1.5082E+06	1.5154E+04	4.5991E+03	4.0530E+02	2.6796E+01	5.3277E+02	2.2980E+01	5.1874E+02	9.2122E+00
ILA	1.6567E+08	6.3781E+07	1.4362E+04	4.1666E+03	4.5366E+02	2.0913E+01	5.8072E+02	4.7495E+01	5.3303E+02	1.0191E+01

Method	AF6		AF7		AF8		AF9		AF10	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
LiWO	**6.9948E+02**	1.3008E+01	**7.6474E+02**	1.7116E+01	9.9926E+02	1.6911E+02	**3.0252E+03**	3.4418E+02	**1.0448E+03**	3.3129E+01
TSA	8.4649E+02	**1.1813E+01**	9.1051E+02	**1.2472E+01**	**8.0278E+02**	**2.3354E+00**	8.0248E+03	3.3094E+02	1.1557E+03	**2.2713E+01**
MVO	7.2963E+02	2.7179E+01	8.0147E+02	2.9241E+01	2.8154E+03	2.1334E+03	4.1193E+03	6.2638E+02	1.1776E+03	5.4329E+01
SSA	7.7657E+02	3.7599E+01	8.1739E+02	2.8370E+01	3.4599E+03	1.2830E+03	4.7306E+03	6.3826E+02	1.1794E+03	5.8551E+01
AEO	9.2146E+02	5.8649E+01	8.2719E+02	3.1975E+01	3.6307E+03	8.4184E+02	4.6174E+03	6.7282E+02	1.1137E+03	3.8867E+01
HGS	8.3646E+02	5.3599E+01	8.2177E+02	2.1139E+01	3.3076E+03	9.9873E+02	3.9982E+03	4.8388E+02	1.2402E+03	3.6446E+02
FOX	1.2224E+03	1.4999E+01	8.9843E+02	1.6821E+01	5.3290E+03	5.5808E+01	5.5205E+03	6.3817E+02	1.1922E+03	5.7920E+01
SWO	7.1317E+02	1.8752E+01	7.7585E+02	1.7290E+01	9.9236E+02	1.4827E+02	3.6013E+03	4.5164E+02	1.0545E+03	2.8959E+01
AOBLMOA	1.1936E+03	4.0428E+01	8.7876E+02	1.8894E+01	4.6100E+03	1.3594E+03	5.3757E+03	**2.5105E+02**	1.1041E+03	4.0817E+01
ESO	7.6443E+02	4.7567E+01	8.0523E+02	2.0107E+01	2.4113E+03	1.5562E+03	3.6592E+03	1.1187E+03	1.1432E+03	3.7955E+01
ILA	8.2524E+02	2.6454E+01	8.6413E+02	2.4494E+01	2.9855E+03	1.4016E+03	4.5771E+03	5.1725E+02	1.2686E+03	4.2615E+01

Method	AF11		AF12		AF13		AF14		AF15	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
LiWO	**8.3749E+03**	**9.8126E+03**	**1.2874E+03**	**4.5576E+01**	**1.3560E+03**	2.5586E+01	1.4422E+03	2.8323E+01	**2.0822E+03**	2.1278E+02
TSA	3.5031E+06	1.7189E+06	6.4201E+03	6.3268E+03	4.5894E+04	2.7190E+04	3.0417E+03	2.4627E+03	3.0661E+03	**1.3901E+02**
MVO	4.5479E+06	3.1905E+06	1.2016E+05	9.3867E+04	7.8665E+03	5.7697E+03	2.4244E+04	1.3682E+04	2.3476E+03	2.9196E+02
SSA	3.2310E+06	2.8054E+06	9.6653E+04	5.5860E+04	1.1011E+04	8.6014E+03	5.6474E+04	3.4491E+04	2.5288E+03	3.3712E+02
AEO	4.0847E+04	3.7223E+04	1.4405E+04	1.2442E+04	3.2720E+03	2.3751E+03	6.4378E+03	9.0807E+03	2.6501E+03	3.0759E+02
HGS	9.6551E+06	1.6189E+07	6.2250E+04	3.2078E+04	9.0564E+04	1.0717E+05	1.5742E+04	1.2252E+04	2.5711E+03	3.0876E+02
FOX	1.5490E+06	1.4526E+06	1.0932E+05	7.9707E+04	5.9803E+03	2.6976E+03	2.9591E+04	3.5821E+04	3.3796E+03	4.7136E+02
SWO	2.2335E+04	1.6689E+04	1.4278E+03	1.1300E+02	1.3637E+03	**1.5770E+01**	**1.4380E+03**	**2.3576E+01**	2.1596E+03	2.2501E+02
AOBLMOA	2.3879E+06	9.9352E+05	2.1215E+08	1.1484E+09	3.8775E+05	7.3541E+05	2.7822E+03	1.2757E+03	3.1735E+03	4.6610E+02
ESO	8.7948E+06	7.1962E+06	3.7221E+05	3.7119E+05	5.6056E+03	3.3691E+03	3.3889E+04	2.8985E+04	2.5481E+03	2.4448E+02
ILA	3.4364E+07	1.9485E+07	5.6237E+05	6.6799E+05	2.9554E+04	4.7749E+04	5.2732E+04	2.8479E+04	2.5769E+03	2.4284E+02

in 30 dimensions. An analysis of the convergence curve reveals that the LiWO algorithm initially lacks strong competitiveness within approximately 100 iterations. However, it gradually demonstrates promising optimization capability in subsequent iterations, leading to improved function-optimal solutions. During the iterative process, the algorithms exhibited distinct trajectories for obtaining the global best solution. For instance, SSA demonstrates a step-like optimization process, whereas the MVO algorithm shows a step-like descent process in the later stages of iteration. This descent process suggests that the population particles of the MVO algorithm encounter spatial positions that effectively

TABLE 5. ***(Continued.)*** **Optimization results for CEC2017 test suit with dimension of 30.**

Method	AF16		AF17		AF18		AF19		AF20	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
LiWO	1.7875E+03	**1.1215E+02**	**1.7391E+03**	**1.8210E+01**	**1.8219E+03**	**7.7520E+00**	2.0986E+03	1.0581E+02	**2.2597E+03**	4.5463E+01
TSA	2.1298E+03	1.3422E+02	2.6826E+06	1.1669E+06	4.1490E+03	2.5736E+03	2.4368E+03	1.4315E+02	2.3973E+03	**1.1757E+01**
MVO	2.0074E+03	1.8076E+02	2.4152E+05	1.9793E+05	6.9684E+04	9.6912E+04	2.2876E+03	1.5592E+02	2.2986E+03	3.1982E+01
SSA	1.9982E+03	1.6564E+02	1.9413E+05	1.5089E+05	5.0227E+05	2.4691E+05	2.3315E+03	1.4579E+02	2.3189E+03	3.0913E+01
AEO	2.1002E+03	2.1636E+02	2.3962E+04	2.1462E+04	8.0839E+03	9.9582E+03	2.3534E+03	1.8549E+02	2.3375E+03	3.4017E+01
HGS	2.0876E+03	1.9815E+02	6.6184E+05	8.2514E+05	2.2466E+04	1.9958E+04	2.4135E+03	2.0557E+02	2.3229E+03	2.7370E+01
FOX	2.6163E+03	3.7769E+02	1.9981E+05	1.2299E+05	2.0322E+05	1.1775E+05	2.9223E+03	2.6927E+02	2.5887E+03	5.4012E+01
SWO	**1.7836E+03**	1.1991E+02	6.4949E+03	5.7996E+03	1.8277E+03	9.4706E+00	**2.0942E+03**	1.1867E+02	2.2741E+03	1.7400E+01
AOBLMOA	2.6158E+03	2.4922E+02	2.0259E+06	9.1442E+06	4.9390E+03	1.9427E+03	2.7338E+03	2.5801E+02	2.4176E+03	4.5997E+01
ESO	1.9826E+03	1.3718E+02	1.2639E+05	5.4371E+04	5.3454E+05	9.5935E+05	2.2665E+03	1.3779E+02	2.3068E+03	2.4275E+01
ILA	1.9643E+03	1.4628E+02	3.4107E+05	3.4087E+05	7.4868E+05	7.0571E+05	2.2951E+03	**9.4866E+01**	2.3107E+03	7.6656E+01

Method	AF21		AF22		AF23		AF24		AF25	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
LiWO	2.7203E+03	1.0890E+03	**2.6288E+03**	1.8932E+01	2.8578E+03	3.3071E+01	2.7868E+03	1.5720E+00	3.9478E+03	8.0382E+02
TSA	4.5866E+03	3.0984E+03	2.7513E+03	**8.9668E+00**	2.9156E+03	**1.1945E+01**	2.7874E+03	2.0317E-01	5.5262E+03	**1.1264E+02**
MVO	5.2531E+03	1.0908E+03	2.6350E+03	2.6446E+01	**2.8003E+03**	2.1260E+01	2.7870E+03	3.1973E+00	4.3831E+03	3.8862E+02
SSA	4.2943E+03	1.9191E+03	2.6690E+03	3.9282E+01	2.8167E+03	2.8360E+01	2.8010E+03	2.0455E+01	4.1929E+03	1.0503E+03
AEO	2.8850E+03	1.5577E+03	2.7732E+03	6.3195E+01	2.9638E+03	7.3496E+01	2.7953E+03	1.0546E+01	5.0732E+03	1.5046E+03
HGS	5.3290E+03	1.2916E+03	2.6747E+03	2.3824E+01	2.8891E+03	5.4710E+01	2.8563E+03	2.8808E+01	4.7477E+03	4.2650E+02
FOX	7.4037E+03	1.0544E+03	3.4209E+03	1.8657E+02	3.5462E+03	1.1422E+02	2.7872E+03	9.9368E+00	8.0376E+03	1.4052E+03
SWO	2.3139E+03	6.2285E+02	2.6321E+03	2.6734E+01	2.8155E+03	2.3994E+01	2.7873E+03	1.1822E+00	3.8356E+03	1.0017E+03
AOBLMOA	5.2900E+03	2.4065E+03	2.9949E+03	1.1496E+02	3.1898E+03	1.2336E+02	2.8007E+03	4.9149E+00	6.3840E+03	2.1564E+03
ESO	**2.2107E+03**	**3.7498E+00**	2.6957E+03	4.7266E+01	2.8295E+03	2.8464E+01	2.8132E+03	2.0811E+01	6.1763E+03	8.6746E+02
ILA	2.2770E+03	2.0132E+01	2.7217E+03	3.3857E+01	2.8664E+03	2.9708E+01	2.8614E+03	2.5925E+01	**3.2635E+03**	5.3486E+02

Method	AF26		AF27		AF28		AF29		Mean rank
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	
LiWO	**3.1101E+03**	1.1906E+01	3.0712E+03	6.3846E+01	**3.3812E+03**	**1.1033E+02**	**5.0477E+03**	**2.1147E+02**	**1.7586**
TSA	3.1134E+03	**6.8541E+00**	3.1266E+03	1.5192E+01	4.0768E+03	1.3186E+02	3.3911E+04	2.2541E+04	7.0000
MVO	3.1179E+03	1.3119E+01	3.1233E+03	2.2895E+01	3.7273E+03	1.7638E+02	1.3633E+06	9.1043E+05	5.1379
SSA	3.1438E+03	2.7036E+01	3.1226E+03	4.2764E+01	3.8871E+03	2.6009E+02	2.1246E+06	1.4950E+06	6.1379
AEO	3.1992E+03	4.2506E+01	3.0811E+03	4.6651E+01	3.8474E+03	2.4028E+02	8.2132E+03	2.2962E+03	5.9310
HGS	3.1254E+03	1.5261E+01	3.2320E+03	6.7757E+01	3.6814E+03	1.9301E+02	4.3469E+05	4.6630E+05	7.2069
FOX	4.0498E+03	4.7088E+02	**3.0137E+03**	2.9720E+01	4.6514E+03	2.1073E+02	6.2981E+05	4.1556E+05	8.2414
SWO	3.1212E+03	1.1186E+01	3.0666E+03	5.3790E+01	3.4363E+03	1.2881E+02	6.8443E+03	6.9780E+02	2.1034
AOBLMOA	3.4740E+03	1.1030E+02	3.1587E+03	**1.2915E+01**	4.6663E+03	3.1826E+02	2.0885E+05	2.5315E+05	8.6897
ESO	3.1656E+03	2.9685E+01	3.1514E+03	2.0759E+01	4.1271E+03	2.0567E+02	6.3065E+05	9.2159E+05	6.1034
ILA	3.1678E+03	2.3509E+01	3.2373E+03	2.3239E+01	3.9207E+03	1.6007E+02	7.1086E+06	7.1818E+06	7.6897

guide the search. The majority of the algorithms demonstrate a relatively straightforward decline during the convergence process, devoid of notable fluctuations post-convergence. The convergence of the LiWO algorithm displays minimal disparity when confronted with unimodal, multimodal, mixed, and composite functions, all of which exhibit seamless downward trajectories. This observation further suggests that the LiWO algorithm possesses a certain level of universality, rendering it capable of effectively addressing a wide range of function-optimization problems.

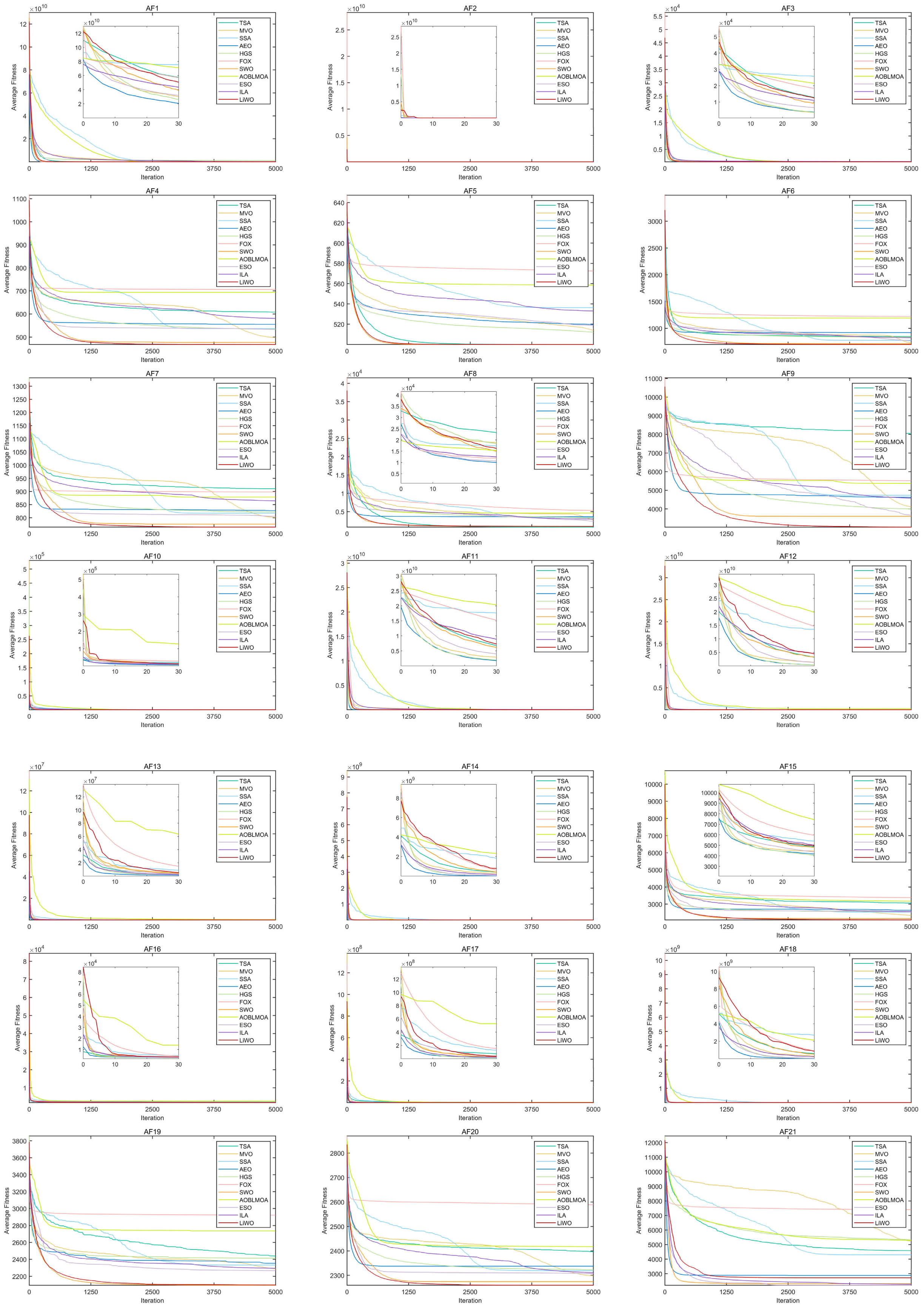

FIGURE 2. The convergence curves of LiWO and the other 10 metaheuristic algorithms on CEC2017 test suit with dimension of 30.

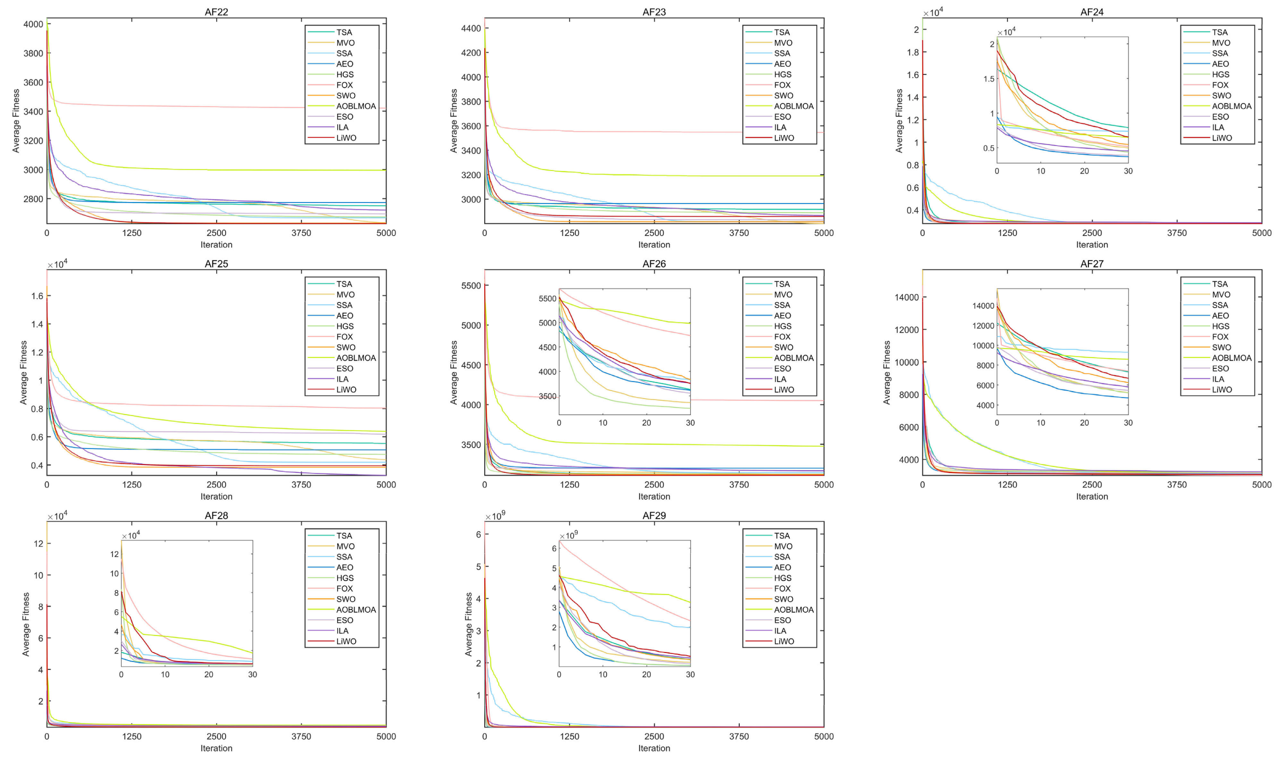

FIGURE 2. *(Continued.)* **The convergence curves of LiWO and the other 10 metaheuristic algorithms on CEC2017 test suit with dimension of 30.**

D. STATISTICAL RESULTS

Box plots are commonly employed to visualize the discrete distribution of data, with their primary benefit being their resistance to the influence of outliers. In Figure 3, statistical box plots are depicted for the 29 functions within the 30 dimensions of the CEC2017 test set. These plots illustrate the distribution of outcomes obtained from 11 algorithms applied to a specific problem, thereby facilitating an examination of algorithm stability. These examples encompass a variety of functions, including unimodal and multimodal functions as well as hybrid and composite functions. As the complexity of the functions increases, the accuracy of the convergence results for all the algorithms becomes more dispersed. Among the depicted graphs, LiWO demonstrates superior optimization performance, as indicated by its lower box-plot positions. Specifically, the third to ninth graphs in Figure 3 represent box plots of multimodal functions characterized by intricate topological structures that contribute to the emergence of an unstable state in the distribution of results across all algorithms. The box plots presented in subsequent sections reveal that certain algorithms may yield solutions of notably inferior quality. For instance, the FOX algorithm exhibits substantial deviations from the optimal solution on AF26, whereas the AOBLMOA algorithm demonstrates significant deviations on AF17, and the ILA algorithm exhibits notable deviations on AF29. In contrast, the box plot for LiWO displays a flat shape and occupies the lowest position, indicating superior robustness and stability.

E. PARAMETER SENSITIVITY

This section presents an analysis of the influence exerted by three primary parameters -p_1, p_2, and p_3, on the performance of the algorithm. It is important to note that all three parameters have probabilistic connotations. Consequently, to assess the efficacy of the optimization algorithm in optimizing diverse functions across varying values, ten potential values were chosen for each parameter. The convergence sensitivity of the algorithm was analyzed using the unimodal function F1, multimodal function F4, mixed function F11, and composite function F21 based on the CEC 2017 test suite. The test suite encompasses four distinct dimensions: D=10, D=30, D=50, and D=100, allowing for the calculation of the impact of various parameter values. The remaining parameters remained unchanged from those in the previous configuration. Considering the diverse levels of impact exerted by dimensions and functional types on function optimization, Equations (22) - (24) are employed to standardize the errors of each optimization outcome and aggregate a cumulative sum of errors. This outcome can aid in the selection of suitable parameter values, and the corresponding calculation results are presented in Tables 6–8.

$$TE(v) = \sum_{i=1}^{f_N} \sum_{j=1}^{D_N} \Delta f_{i,j}(v) / \Delta f_{i,j}^{\max} \tag{22}$$

$$\Delta f_{i,j}(v) = \left(\frac{1}{N_{run}} \sum_{k=1}^{N_{run}} f_{i,j,k}(v) \right) - f_i^* \tag{23}$$

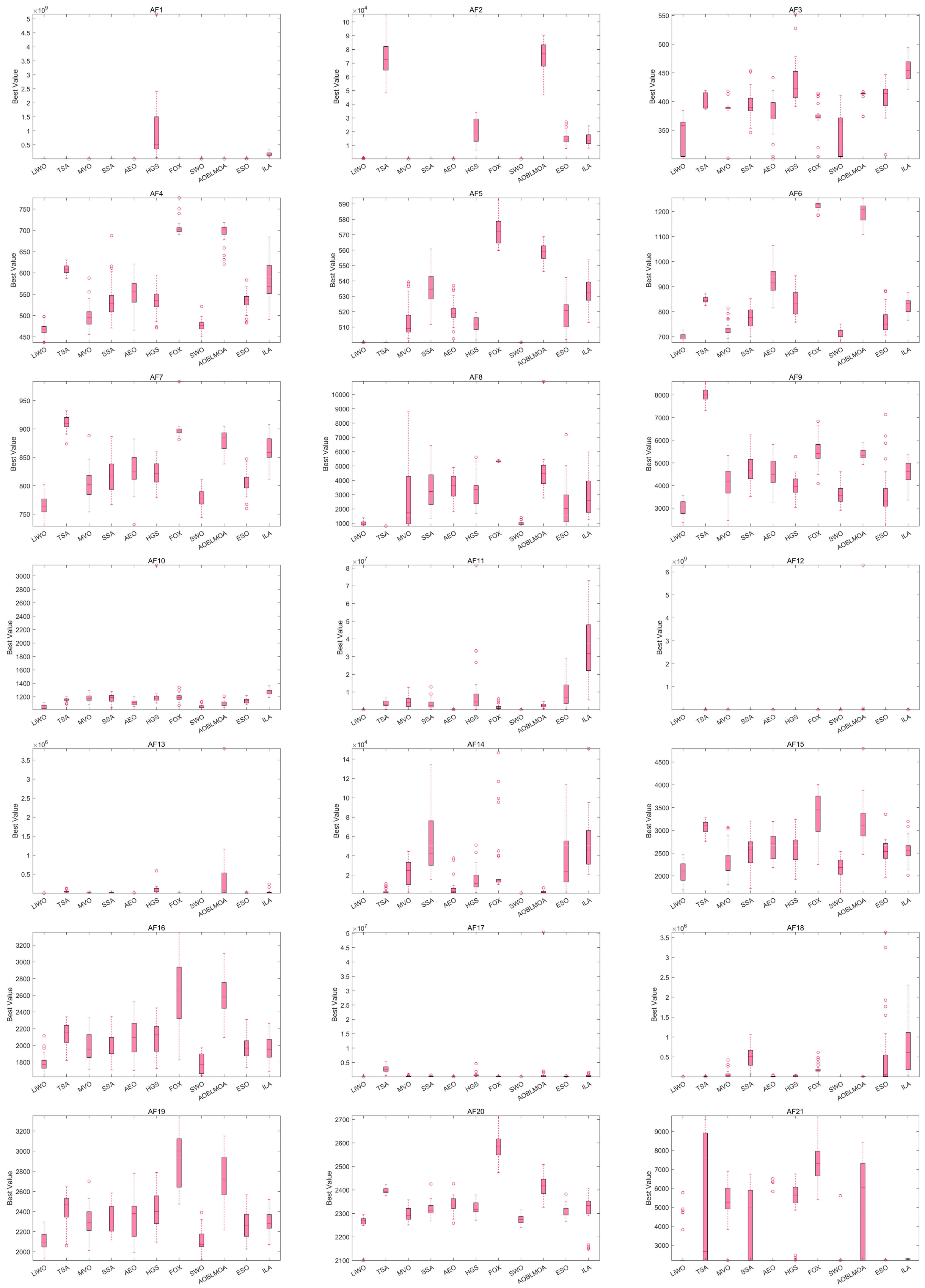

FIGURE 3. Box plot of the CEC2017 test suit with dimension 30.

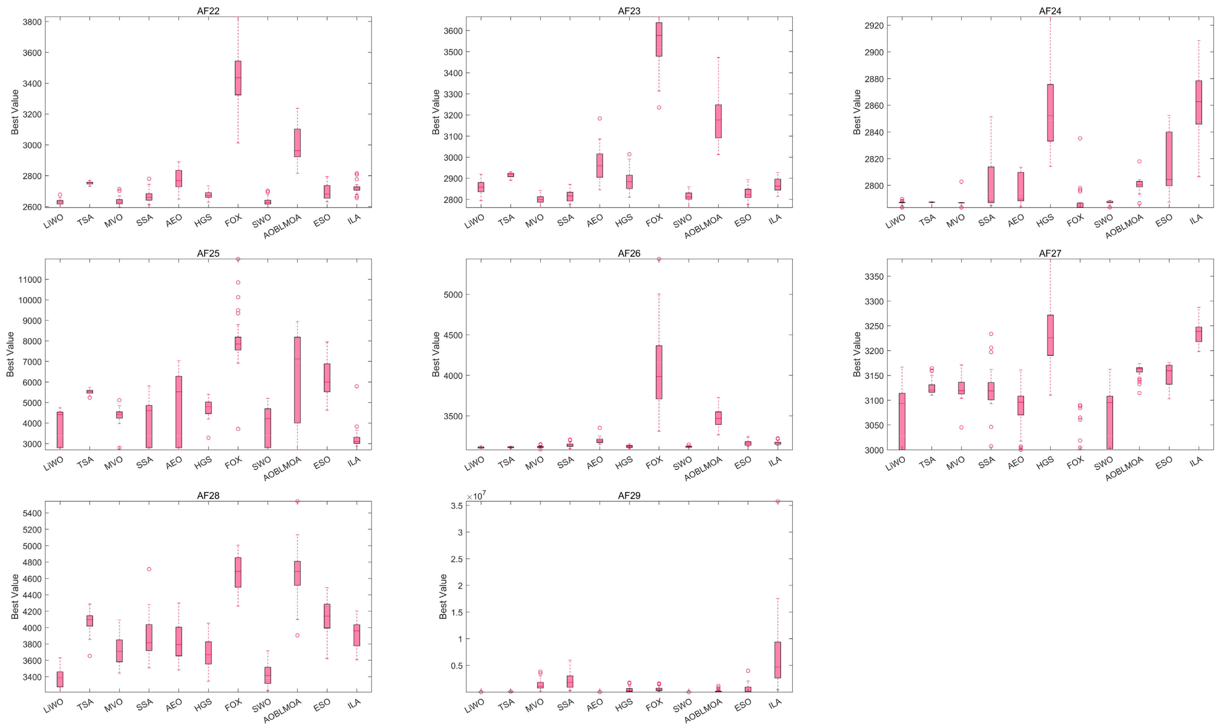

FIGURE 3. *(Continued.)* **Box plot of the CEC2017 test suit with dimension 30.**

TABLE 6. Normalized error sum for four types of functions with different values of p_1.

	p_1									
	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Normalized error sum	11	8.8	9.5	9.4	9.6	9.1	9.5	10	11	10.5

TABLE 7. Normalized error sum for four types of functions with different values of p_2.

	p_2									
	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Normalized error sum	11.3	9.2	9.6	9.3	11	9.6	10	10	9.6	11

$$\Delta f_{i,j}^{\max} = \max(\Delta f_{i,j}(v)) \tag{24}$$

where v represents the parameter value, $\Delta f_{i,j}(v)$ represents the difference between the average optimal solution and the optimal solution of the function after 30 runs under the v parameter, f_i^* represents the optimal solution of the i-th function, and $\Delta f_{i,j}^{\max}$ represents the maximum difference obtained under different v.

Table 6 presents the comprehensive variation in the parameter p_1 across various values. The findings reveal that deviation is minimized when p_1 assumes a value of 0.1. Notably, the test outcome with an p_1 value of 0.05 exhibits a significantly higher deviation compared to that with a value of 0.1, thereby suggesting the indispensability of spiral motion throughout the optimization process. Moreover, the value of 0.1 further signifies that the occurrence of spiral motion does not necessitate frequent repetition, aligning with the observed behavior of falling leaves in real-life scenarios.

Table 7 presents the comprehensive variation in parameter p_2 across various values, revealing that p_2 attains the lowest normalization error when set to 0.1. p_2, denoting the reset probability, serves to enhance population diversity and prevent the algorithm from converging to local minima. The experimental results further underscore the significance of the reset operation.

Table 8 presents the comprehensive variation in the parameter p_3 across various values. The findings revealed that p_3 attains the lowest normalization error at a value of 0.3. p_3 denotes the probability of a gentle breeze being employed to enhance the exploration capability of the algorithm. A value of 0.3 signifies the necessity for the algorithm to sustain a consistent level of global exploration throughout the entire optimization duration, while ensuring that the local utilization surpasses the exploration capacity of the tested function.

F. TIME CONSUMPTION ANALYSIS

In order to elucidate the time complexity of the LiWO algorithm, the total time required for evaluating 10 benchmark functions BF1-BF10 on the CEC2019 test suite is normalized by the maximum time consumption. The normalized data is depicted in Figure 4. Analysis of the figure

TABLE 8. Normalized error sum for four types of functions with different values of p_3.

	p_3									
	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Normalized error sum	10.9	11.4	9.7	9.2	9.3	10.3	10.4	9.6	9.9	11.9

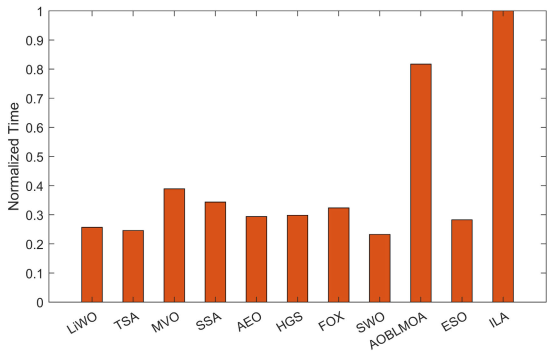

FIGURE 4. Bar plot of the time consumptions under CEC2019 test suit.

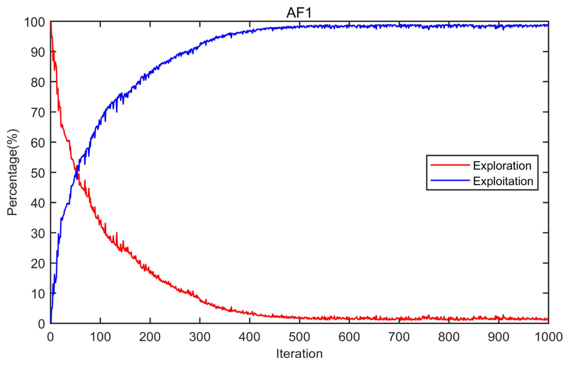

FIGURE 5. Exploration and exploitation of LiWO on AF1 of CEC2017.

reveals that the time consumption of the LiWO algorithm is comparable to that of the SWO and TSA algorithms, but significantly greater than that of the AOBLMOA and ILA algorithms. This suggests that the LiWO algorithm exhibits competitive computational complexity.

G. EXPLORATION AND EXPLOITATION ANALYSIS

This section examines the exploration and exploitation performance of the LiWO algorithm using the unimodal function AF1 and mixed function AF22 from the CEC2017 test set as case studies. Figures 5 and 6 depict the distribution of exploration and exploitation [43] within the population of the LiWO algorithm over the course of the iteration period. The unimodal function AF1 demonstrates a rapid increase in development proportion, suggesting strong exploitation capability of the algorithm. Conversely, the composite function F22 shows a gradual decrease in exploration proportion, indicating proficient global exploration ability of the algorithm.

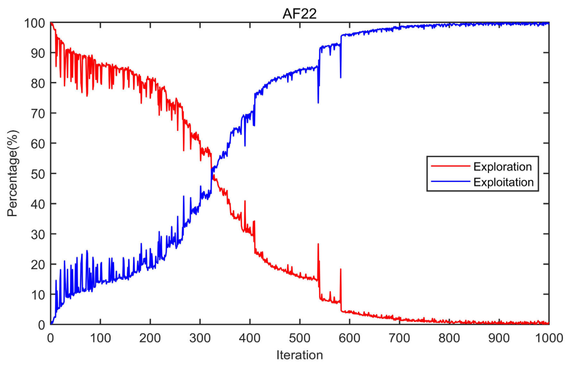

FIGURE 6. Exploration and exploitation of LiWO on AF22 of CEC2017.

TABLE 9. Comparison for the GTCD Problem.

Algorithm	BEST	AVG	STD
LiWO	**2964895.4173**	**2964895.4173**	**4.73621514E-10**
TSA	**2964895.4173**	2989961.4137	0.0863702832
MVO	2965065.4111	2.0625409105	168269.19778
SSA	2967261.6625	3017381.1878	291035.56573
AEO	**2964895.4173**	2964906.6205	121.78859358
HGS	2964895.423	2965133.6059	2425.0012227
FOX	2970787.732	3083423.5256	167841.14325
SWO	**2964895.4173**	**2964895.4173**	6.50260599E-9
AOBLMOA	**2964895.4173**	**2964895.4173**	4.73621885E-10
ESO	2964896.0687	2966440.6792	8298.1522142
ILA	2964896.4262	2964908.1281	87.366341004

H. CONSTRAINED OPTIMIZATION PROBLEM

Gas transmission compressor design (GTCD) [41] is a real-world constrained optimization problem that optimizes the design of the gas transmission compressor and involves four design variables and one constraint condition. Its mathematical model is as follows:

$$\min f(x) = 8.61 \times 10^5 x_1^{1/2} x_2 x_3^{-2/3} x_4^{-1/2} + 3.69 \times 10^4 x_3 + 7.72 \times 10^8 x_1^{-1} x_2^{0.219} - 765.43 \times 10^6 x_1^{-1} \tag{25}$$

$$s.t. \quad g(x) = x_4 x_2^{-2} + x_2^{-2} - 1 \le 0 \tag{26}$$

where design variable ranges are $20 \le x_1 \le 50$, $1 \le x_2 \le 10$, $20 \le x_3 \le 50$, $0.1 \le x_4 \le 60$.

Table 9 presents the algorithms employed for calculating the optimal, average and standard deviation values related to the GTCD problem. The results indicate that the LiWO, TSA, AEO, SWO, and AOBLMOA algorithms achieved the optimal value compared to other results. Among these, the LiWO, SWO, and AOBLMOA algorithms obtained the optimal average value. Additionally, LiWO demonstrated the best standard deviation value. Therefore, LiWO demonstrates notable competitiveness in addressing engineering application issues.

IV. CONCLUSION

This study introduced LiWO, a novel metaheuristic algorithm rooted in a natural phenomenon, to address continuous optimization problems. The algorithm utilizes factors such as wind force, translation, and rotation that influence the movement of fallen leaves in the wind to inform the movement of population particles. The physical model of leaf trajectory is translated into a motion model of particles within the solution space. Drawing upon the observed characteristics of leaf movement, the algorithm incorporates the Breeze-driven leaf strategy and the Strong Wind-driven leaf strategy to effectively blend the two approaches and achieve the optimization goal. Utilizing three iterations of the computational evolutionary computation benchmarks, specifically CEC 2017, CEC 2019, and CEC 2022, in conjunction with ten prominent optimization algorithms, this study assesses the efficacy and efficiency of the LiWO algorithm. Across all comparative analyses, LiWO demonstrated a success rate of 82% and a failure rate of merely 6%. Furthermore, the LiWO algorithm successfully attained the optimal value in a constrained optimization process test. While LiWO has demonstrated strong optimization competitiveness, opportunities for improvement remain in optimizing mixed and higher dimensional functions. Our future research will concentrate on establishing a hierarchical process for leaf falling and incorporating additional physical models that impact leaf falling. Furthermore, our research will also prioritize the development of binary and multi-objective variants of LiWO to enhance its performance evaluation.

REFERENCES

[1] M. Quiroz-Castellanos, L. G. de la Fraga, A. Lara, L. Trujillo, and O. Schütze, ''Numerical and evolutionary optimization 2021,'' *Math. Comput. Appl.*, vol. 28, no. 3, pp. 71–73, May 2023, doi: 10.3390/mca28030071.

[2] A. M. Deaconu, D. T. Cotfas, and P. A. Cotfas, ''Advanced optimization methods and applications,'' *Mathematics*, vol. 11, no. 9, pp. 2205–2211, May 2023, doi: 10.3390/math11092205.

[3] Y. Song, Y. Liu, H. Chen, and W. Deng, ''A multi-strategy adaptive particle swarm optimization algorithm for solving optimization problem,'' *Electronics*, vol. 12, no. 3, pp. 491–505, Jan. 2023, doi: 10.3390/electronics12030491.

[4] V. B. Shinde and P. J. Pawar, ''Trajectory optimization of an industrial robot using teaching-learning-based optimization,'' in *Advanced Engineering Optimization Through Intelligent Techniques* (Lecture Notes in Mechanical Engineering), R. V. Rao and J. Taler, Eds. Singapore: Springer, 2023, pp. 677–686, doi: 10.1007/978-981-19-9285-8_63.

[5] M. Kumar and A. Sharma, ''Wind farm layout optimization problem using teaching-learning-based optimization algorithm,'' in *Communication and Intelligent Systems* (Lecture Notes in Networks and Systems), vol. 689, H. Sharma, V. Shrivastava, K. K. Bharti, and L. Wang, Eds. Singapore: Springer, 2023, pp. 151–170, doi: 10.1007/978-981-99-2322-9_12.

[6] Y. Liu and C. Zhang, ''Neural network algorithm based on particle swarm optimization and design of immersive intelligent English teaching in college English teaching system,'' in *Proc. Int. Conf. Distrib. Comput. Electr. Circuits Electron. (ICDCECE)*, Apr. 2023, pp. 1–6, doi: 10.1109/ICDCECE57866.2023.10151018.

[7] J. Osei-kwakye, F. Han, A. A. Amponsah, Q.-H. Ling, and T. A. Abeo, ''A diversity enhanced hybrid particle swarm optimization and crow search algorithm for feature selection,'' *Int. J. Speech Technol.*, vol. 53, no. 17, pp. 20535–20560, Sep. 2023, doi: 10.1007/s10489-023-04519-2.

[8] N. D. Hieu, M. V. Linh, and P. D. Phong, ''A co-optimization algorithm utilizing particle swarm optimization for linguistic time series,'' *Mathematics*, vol. 11, no. 7, pp. 1597–1610, Mar. 2023, doi: 10.3390/math11071597.

[9] K. Rajesh Kumar and M. Vijayakumar, ''Optimization of cognitive femtocell network via oppositional beetle swarm optimization algorithm,'' *Intell. Autom. Soft Comput.*, vol. 36, no. 1, pp. 819–832, 2023, doi: 10.32604/iasc.2023.030961.

[10] S. Kankilic and E. Karpat, ''Optimization of multilayer absorbers using the bald eagle optimization algorithm,'' *Appl. Sci.*, vol. 13, no. 18, pp. 10301–10317, Sep. 2023, doi: 10.3390/app131810301.

[11] J. Kennedy and R. Eberhart, ''Particle swarm optimization,'' in *Proc. IEEE ICNN*, vol. 4, Perth, WA, USA, Jan. 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[12] D. Karaboga and B. Basturk, ''On the performance of artificial bee colony (ABC) algorithm,'' *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008, doi: 10.1016/j.asoc.2007.05.007.

[13] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari, ''Bat algorithm for constrained optimization tasks,'' *Neural Comput. Appl.*, vol. 22, no. 6, pp. 1239–1255, May 2013, doi: 10.1007/s00521-012-1028-9.

[14] M. Mirrashid and H. Naderpour, ''Incomprehensible but intelligible-in-time logics: Theory and optimization algorithm,'' *Knowl.-Based Syst.*, vol. 264, Mar. 2023, Art. no. 110305, doi: 10.1016/j.knosys.2023.110305.

[15] P. Trojovsky, M. Dehghani, and E. Milkova, ''Language education optimization: A new human-based metaheuristic algorithm for solving optimization problems,'' *Comput. Model. Eng. Sci.*, vol. 136, no. 2, pp. 1527–1573, 2023, doi: 10.32604/cmes.2023.025908.

[16] Y. Lin, A. A. Heidari, S. Wang, H. Chen, and Y. Zhang, ''An enhanced hunger games search optimization with application to constrained engineering optimization problems,'' *Biomimetics*, vol. 8, no. 5, pp. 441–477, Sep. 2023, doi: 10.3390/biomimetics8050441.

[17] M. Abdel-Basset, R. Mohamed, M. Jameel, and M. Abouhawwash, ''Spider wasp optimizer: A novel meta-heuristic optimization algorithm,'' *Artif. Intell. Rev.*, vol. 56, no. 10, pp. 11675–11738, Oct. 2023, doi: 10.1007/s10462-023-10446-y.

[18] S. O. Oladejo, S. O. Ekwe, L. A. Akinyemi, and S. A. Mirjalili, ''The deep sleep optimizer: A human-based metaheuristic approach,'' *IEEE Access*, vol. 11, pp. 83639–83665, 2023, doi: 10.1109/ACCESS.2023.3298105.

[19] E. Trojovská, M. Dehghani, and V. Leiva, ''Drawer algorithm: A new metaheuristic approach for solving optimization problems in engineering,'' *Biomimetics*, vol. 8, no. 2, pp. 239–273, Jun. 2023, doi: 10.3390/biomimetics8020239.

[20] K. Rezvani, A. Gaffari, and M. R. E. Dishabi, ''The bedbug meta-heuristic algorithm to solve optimization problems,'' *J. Bionic Eng.*, vol. 20, no. 5, pp. 2465–2485, Sep. 2023, doi: 10.1007/s42235-023-00356-8.

[21] H. Givi, M. Dehghani, and S. Hubálovský, ''Red panda optimization algorithm: An effective bio-inspired metaheuristic algorithm for solving engineering optimization problems,'' *IEEE Access*, vol. 11, pp. 57203–57227, 2023, doi: 10.1109/ACCESS.2023.3283422.

[22] S. Akyol and B. Alatas, ''Plant intelligence based metaheuristic optimization algorithms,'' *Artif. Intell. Rev.*, vol. 47, no. 4, pp. 417–462, Apr. 2017, doi: 10.1007/s10462-016-9486-6.

[23] M. S. Kiran, ''TSA: Tree-seed algorithm for continuous optimization,'' *Exp. Syst. Appl.*, vol. 42, no. 19, pp. 6686–6698, Nov. 2015, doi: 10.1016/j.eswa.2015.04.055.

[24] A. Cheraghalipour, M. Hajiaghaei-Keshteli, and M. M. Paydar, ''Tree growth algorithm (TGA): A novel approach for solving optimization problems,'' *Eng. Appl. Artif. Intell.*, vol. 72, pp. 393–414, Jun. 2018, doi: 10.1016/j.engappai.2018.04.021.

[25] Z. Bayraktar, M. Komurcu, J. A. Bossard, and D. H. Werner, ''The wind driven optimization technique and its application in electromagnetics,'' *IEEE Trans. Antennas Propag.*, vol. 61, no. 5, pp. 2745–2757, May 2013, doi: 10.1109/TAP.2013.2238654.

[26] Z. Bayraktar and M. Komurcu, ''Adaptive wind driven optimization,'' in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol.*, New York, NY, USA, 2016, pp. 124–127, doi: 10.4108/eai.3-12-2015.2262424.

[27] H. A. Shehadeh, ''Chernobyl disaster optimizer (CDO): A novel metaheuristic method for global optimization,'' *Neural Comput. Appl.*, vol. 35, no. 15, pp. 10733–10749, May 2023, doi: 10.1007/s00521-023-08261-1.

[28] K. Luo, ''Water flow optimizer: A nature-inspired evolutionary algorithm for global optimization,'' *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 7753–7764, Aug. 2022, doi: 10.1109/TCYB.2021.3049607.

[29] H. A. Shehadeh, ''A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization,'' *Neural Comput. Appl.*, vol. 33, no. 18, pp. 11739–11752, Sep. 2021, doi: 10.1007/s00521-021-05880-4.

[30] W. Zhao, L. Wang, and Z. Zhang, ''Artificial ecosystem-based optimization: A novel nature-inspired meta-heuristic algorithm,'' *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9383–9425, Jul. 2020, doi: 10.1007/s00521-019-04452-x.

[31] M. Azizi, U. Aickelin, H. A. Khorshidi, and M. B. Shishehgarkhaneh, ''Energy valley optimizer: A novel metaheuristic algorithm for global and engineering optimization,'' *Sci. Rep.*, vol. 13, no. 1, pp. 226–248, Jan. 2023, doi: 10.1038/s41598-022-27344-y.

[32] Y. Tanabe and K. Kaneko, ''Behavior of a falling paper,'' *Phys. Rev. Lett.*, vol. 73, no. 10, pp. 1372–1375, Sep. 1994, doi: 10.1103/physrevlett.73.1372.

[33] H. Zhong, S. Chen, and C. Lee, ''Experimental study of freely falling thin disks: Transition from planar zigzag to spiral,'' *Phys. Fluids*, vol. 23, no. 1, pp. 11702–11706, Jan. 2011, doi: 10.1063/1.3541844.

[34] C. Li, J. Qian, R. Tong, J. Chang, and J. Zhang, ''GPU based real-time simulation of massive falling leaves,'' *Comput. Vis. Media*, vol. 1, no. 4, pp. 351–358, Dec. 2015, doi: 10.1007/s41095-015-0025-1.

[35] T. Martin, N. Umetani, and B. Bickel, ''OmniAD: Data-driven omnidirectional aerodynamics,'' *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–12, Jul. 2015, doi: 10.1145/2766919.

[36] E. Quigley, Y. Yu, J. Huang, W. Lin, and R. Fedkiw, ''Real-time interactive tree animation,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 5, pp. 1717–1727, May 2018, doi: 10.1109/TVCG.2017.2661308.

[37] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, ''Multi-verse optimizer: A nature-inspired algorithm for global optimization,'' *Neural Comput. Appl.*, vol. 27, no. 2, pp. 495–513, Feb. 2016, doi: 10.1007/s00521-015-1870-7.

[38] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, ''Salp swarm algorithm: A bio-inspired optimizer for engineering design problems,'' *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017, doi: 10.1016/j.advengsoft.2017.07.002.

[39] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, ''Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts,'' *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114864, doi: 10.1016/j.eswa.2021.114864.

[40] H. Mohammed and T. Rashid, ''FOX: A FOX-inspired optimization algorithm,'' *Int. J. Speech Technol.*, vol. 53, no. 1, pp. 1030–1050, Jan. 2023, doi: 10.1007/s10489-022-03533-0.

[41] Y. Zhao, C. Huang, M. Zhang, and Y. Cui, ''AOBLMOA: A hybrid biomimetic optimization algorithm for numerical optimization and engineering design problems,'' *Biomimetics*, vol. 8, no. 4, pp. 381–420, Aug. 2023, doi: 10.3390/biomimetics8040381.

[42] L. Yao, P. Yuan, C.-Y. Tsai, T. Zhang, Y. Lu, and S. Ding, ''ESO: An enhanced snake optimizer for real-world engineering problems,'' *Exp. Syst. Appl.*, vol. 230, Nov. 2023, Art. no. 120594, doi: 10.1016/j.eswa.2023.120594.

[43] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, ''On the exploration and exploitation in popular swarm-based metaheuristic algorithms,'' *Neural Comput. Appl.*, vol. 31, no. 11, pp. 7665–7683, Nov. 2019, doi: 10.1007/s00521-018-3592-0.

NING FANG received the B.S. and Ph.D. degrees in electronic and information engineering from Beihang University, Beijing, China, in 2002 and 2007, respectively.

She is currently with the School of Electronic and Information Engineering, Beihang University. Her research interests include pattern recognition, evolutionary algorithms, and combinatorial optimization and their applications.

QI CAO received the bachelor's degree from Wuhan Polytechnic University, in 2004. He is currently the Deputy Chief Engineer of Longyan Tobacco Industry Company Ltd. He mainly engages in industrial automation and information construction and enterprise digital transformation.