processes

Article

Dynamic Chaotic Opposition-Based Learning-Driven Hybrid Aquila Optimizer and Artificial Rabbits Optimization Algorithm: Framework and Applications

Yangwei Wang, Yaning Xiao, Yanling Guo and Jian Li *

College of Mechanical and Electrical Engineering, Northeast Forestry University, Harbin 150040, China
* Correspondence: lijian499@163.com

Abstract: Aquila Optimizer (AO) and Artificial Rabbits Optimization (ARO) are two recently developed meta-heuristic optimization algorithms. Although AO has powerful exploration capability, it still suffers from poor solution accuracy and premature convergence when addressing some complex cases due to the insufficient exploitation phase. In contrast, ARO possesses very competitive exploitation potential, but its exploration ability needs to be more satisfactory. To ameliorate the above-mentioned limitations in a single algorithm and achieve better overall optimization performance, this paper proposes a novel chaotic opposition-based learning-driven hybrid AO and ARO algorithm called CHAOARO. Firstly, the global exploration phase of AO is combined with the local exploitation phase of ARO to maintain the respective valuable search capabilities. Then, an adaptive switching mechanism (ASM) is designed to better balance the exploration and exploitation procedures. Finally, we introduce the chaotic opposition-based learning (COBL) strategy to avoid the algorithm fall into the local optima. To comprehensively verify the effectiveness and superiority of the proposed work, CHAOARO is compared with the original AO, ARO, and several state-of-the-art algorithms on 23 classical benchmark functions and the IEEE CEC2019 test suite. Systematic comparisons demonstrate that CHAOARO can significantly outperform other competitor methods in terms of solution accuracy, convergence speed, and robustness. Furthermore, the promising prospect of CHAOARO in real-world applications is highlighted by resolving five industrial engineering design problems and photovoltaic (PV) model parameter identification problem.

Keywords: Aquila Optimizer; Artificial Rabbits Optimization; adaptive switching mechanism; chaotic opposition-based learning; industrial engineering design; photovoltaic model

check for updates

Citation: Wang, Y.; Xiao, Y.; Guo, Y.; Li, J. Dynamic Chaotic Opposition-Based Learning-Driven Hybrid Aquila Optimizer and Artificial Rabbits Optimization Algorithm: Framework and Applications. *Processes* **2022**, *10*, 2703. https://doi.org/10.3390/pr10122703

Academic Editors: Amir H. Gandomi and Laith Abualigah

Received: 21 November 2022
Accepted: 12 December 2022
Published: 14 December 2022

1. Introduction

Optimization can be considered the process of finding the best solution among all candidates for a particular problem so as to maximize profits, efficiency, and performance with limited resource consumption [1]. Optimization problems widely exist in different disciplines and engineering fields, such as feature selection [2], industrial design [3], Internet of Things task scheduling [4], data clustering [5], and aerospace control [6,7], which makes the study on optimization techniques become a hot topic and have drawn the attention of many scholars. Deterministic mathematical programming methodology commonly requires the target function of an optimization problem to be convex and differentiable. Nevertheless, over the past few decades, the complexity of real-life optimization problems has increased dramatically. These conventional optimization methods struggle to find optimal or near-optimal solutions effectively when facing the challenges of large-scale, multimodal, and non-convex search domains [8]. Therefore, to accomplish the desired goal, meta-heuristic algorithms (MAs) are gradually becoming very popular as powerful tools to solve such intractable global optimization problems [9]. The meta-heuristic algorithm is a kind of stochastic algorithm, which assumes the optimization problem as a black box and then iteratively adopts different random operators to sample the search domain for better decision variables. Compared with traditional techniques, MAs have the unique advantages of conceptual simplicity, high flexibility, and no gradient information required [10]. Benefiting from these advantages, MAs are always capable of showing excellent performance in various scientific and industrial application scenarios. This drives the continuous interest of more worldwide scholars devoted to MAs.

MAs generally build mathematical optimization models by simulating various stochastic occurrences in nature. Based on different design philosophies, MAs can be classified into four dominant categories [11]: evolution-based algorithms, physics-based algorithms, swarm-based algorithms, and human-based algorithms. Evolutionary algorithms mimic the laws of natural selection in biology and use operators like selection, crossover, and mutation to evolve the initial population toward the global optimum. Physics-based algorithms are derived from physical phenomena in the universe and update the search agent with formulas borrowed from physical theories. Swarm-based algorithms are inspired by the social behavior within a group of animals, plants, or other organisms. The common feature of these algorithms is the sharing of biological information of all individuals in the optimization process. The last category of methods, human-based algorithms, originates from human cooperative behavior and activities in the community. Table 1 shows the details of some well-known optimization paradigms belonging to these four classes of algorithms. The core components of MAs are global exploration and local exploitation. In the early iterations, a well-organized optimizer would explore the entire search space as much as possible to locate promising regions where the global optimal solution may exist. Then, in the later stage, more local exploitation is performed to improve the final quality of the solution based on the previously obtained space information. Generally speaking, it is critical for MAs to maintain a good balance between exploration and exploitation, which is related to the effectiveness of algorithms in solving complex optimization problems [12].

Table 1. Classification of meta-heuristic algorithms.

Classification	Algorithm	Inspiration	Year	Reference
Evolutionary	Genetic Algorithm (GA)	Evolutionary concepts	1992	[13]
	Differential Evolution (DE)	Darwin's theory of evolution	1997	[14]
	Biogeography-Based Optimization (BBO)	Biogeography regarding the migration of species	2008	[15]
	Tree Growth Algorithm (TGA)	Competition of trees for food and light	2018	[16]
Physics-based	Simulated Annealing (SA)	Annealing process in metallurgy	1983	[17]
	Gravity Search Algorithm (GSA)	Law of gravity and mass interactions	2009	[18]
	Black Hole Algorithm (BHA)	Black hole phenomenon	2013	[19]
	Multi-Verse Optimizer (MVO)	Multi-verse theory	2015	[20]
	Sine Cosine Algorithm (SCA)	Sine/cosine functions	2016	[21]
	Henry gas solubility optimization (HGSO)	Huddling behavior of gas	2019	[22]
	Arithmetic Optimization Algorithm (AOA)	Distribution behavior of arithmetic operators in mathematics	2021	[23]
Swarm-based	Particle Swarm Optimization (PSO)	Foraging behavior of bird flocks	1995	[24]
	Ant Colony Optimization (ACO)	Foraging behavior of some ant species	2006	[25]
	Cuckoo Search (CS)	Breed behavior of certain cuckoo species	2011	[26]
	Grey Wolf Optimizer (GWO)	Leadership hierarchy and hunting mechanism of grey wolves	2014	[27]
	Ant Lion Optimizer (ALO)	Hunting mechanism of antlions	2015	[28]
	Moth-Flame Optimization (MFO)	Navigation of moths in nature	2015	[29]
	Whale Optimization Algorithm (WOA)	Social behavior of humpback whales	2016	[30]
	Salp Swarm Algorithm (SSA)	Swarming behavior of salps	2017	[31]
	Harris Hawks Optimization (HHO)	Cooperative behavior and chasing style of Harris' hawks	2019	[32]
	Tunicate Swarm Algorithm (TSA)	Jet propulsion behavior of tunicates	2020	[33]

Table 1. *Cont.*

Classification	Algorithm	Inspiration	Year	Reference
	Slime Mould Algorithm (SMA)	Oscillation mode of slime mould	2020	[34]
	Reptile Search Algorithm (RSA)	Hunting behavior of Crocodiles	2022	[35]
	Golden Jackal Optimization (GJO)	Hunting behavior of golden jackals	2022	[36]
Human-based	Teaching Learning-Based Optimization (TLBO)	Teaching and learning in a classroom	2011	[37]
	Harmony Search (HS)	Behavior of a music orchestra	2013	[38]
	Collective Decision Optimization (CDO)	Decision-making characteristics of humans	2017	[39]
	Political Optimizer (PO)	Multi-phased process of politics	2020	[40]

Though MAs play an increasingly prominent role in computational science, some skepticism also emerges: since there are already many famous MAs like those mentioned above, why is it necessary to present new algorithms as well as further innovations? That is because, according to the No-Free-Lunch (NFL) [41] theorem, no one algorithm can be guaranteed to work for all optimization problems. In fact, the average performance of the original optimizers is almost the same, and most of them still have some drawbacks to be eliminated, such as poor solution accuracy, slow convergence speed, and ease of falling into local optima. Hence, motivated by the NFL theorem, apart from developing new MAs, lots of scholars try to improve existing optimization algorithms by employing some helpful measures. Currently, there are three popular trends in the improvement of existing MAs [42]: (1) Embed one or more search mechanisms into the algorithm, (2) Hybridize two or more algorithms, and (3) Hybridize two or more algorithms with further enhancement by one or more search mechanisms. Among them, the third approach is highly favored because it can effectively promote valuable information exchange and diversity between search agents in the optimization process of the hybrid algorithm; meanwhile, the improvement strategies would also assist in boosting the overall performance [3]. For example, Zhang et al. [43] presented a chaotic hybridization optimizer of SCA and HHO, namely CSCAHHO. In CSCAHHO, the stable convergence ability of SCA and the fast convergence speed of HHO were fully retained. Besides, chaotic mapping was used to improve the randomness of the algorithm. Compared with basic SCA, HHO, and five advanced algorithms, it was proven that CSCAHHO could provide better convergence accuracy and stability. Cheng et al. [44] integrated PSO and GWO into an efficient optimization approach known as IPSO-GWO. To further increase population diversity and avoid the local optimum, Logistic mapping and adaptive inertial weight were embedded in this hybrid algorithm. In a case study of global path planning for mobile robots, IPSO-GWO was able to find the optimal path with a faster convergence speed than traditional methods. In [45], the authors proposed a hybrid LSMA-TLBO algorithm. First, SMA was combined with TLBO to balance the exploration and exploitation, and then Lévy flight was introduced into the hybrid algorithm to improve its global search capability further. Experimental results showed that LSMA-TLBO has superior performance over other competitors on 33 well-known benchmark functions and six engineering design problems. Moreover, Liu et al. [3] constructed a novel improved hybrid technique (HAGSA) based on the AOA and Golden Sine Algorithm (GSA). The Brownian mutation was also employed to enhance the local exploitation competence of HAGSA.

In this study, we center on two state-of-the-art swarm intelligence optimization algorithms, namely Aquila Optimizer (AO) and Artificial Rabbits Optimization (ARO). The AO algorithm simulates the Aquila's behavior during the process of hunting the prey, which was first proposed by Abualigah et al. [46] in 2021. Preliminary studies have demonstrated that AO has quite a few advantages, such as easy implementation, stable global exploration capability, and unique search ways. In view of this, AO is widely applied to scientific research and real-world optimization problems [47,48]. However, as with other MAs, the canonical AO inevitably suffers from poor convergence accuracy and proneness to fall into local optima in certain cases, mainly due to its inadequate exploitation phase [49,50]. Consequently, many variants of AO were suggested to enhance its searchability for global optimization. Yu et al. [51] proposed a modified AO-based method called mAO, which assimilated opposition-based learning and restart strategy to strengthen the exploration capability of the algorithm and employed chaotic local search to boost the exploitation trend. Compared with the original AO and nine other algorithms, mAO can obtain better results on 29 benchmark functions and five constrained engineering design issues. Zhao et al. [49] presented a simplified AO by removing the two exploitation strategies and keeping only the position update formula of the exploration phase for the whole iteration. The effectiveness of the proposed method was validated by a series of comprehensive experiments. In [52], an improved version of AO, namely IAO, was developed for solving continuous numerical optimization problems. First, a novel search control factor was used to refine the hunting strategies. Furthermore, opposition-based learning and Gaussian mutation were integrated to enhance the general search performance of IAO. Wang et al. [50] combined the exploration phase of AO and the exploitation phase of HHO into a new hybrid optimizer termed as IHAOHHO. Additionally, random opposition-based learning and nonlinear escaping energy parameter were introduced to help the algorithm avoid local optima and accelerate convergence. Experimental findings revealed that IHAOHHO shows excellent performance on benchmark function optimization tests. In [53], a new hybrid meta-heuristic optimization technique named AGO was raised based on the AO and Grasshopper Optimization Algorithm (GOA). The proposed AGO was applied to optimize the motor geometry of Surface Inset Permanent Magnet Synchronous Motor (SI-PMSM) to minimize the total core loss. The results indicated that AGO could effectively reduce core losses, thereby increasing the power density of the motor. Also, Zhang et al. [54] merged the merits of AO and AOA to design the AOAOA algorithm. Simulation experiments on 27 benchmark test functions and three practical engineering applications fully verified the superior robustness and convergence accuracy of AOAOA.

In 2022, Wang et al. [55] first proposed the ARO algorithm by modeling the survival strategies of rabbits in nature. The strategy of rabbits searching for food away from their nests represents the exploration capability of the algorithm, whereas the strategy of rabbits randomly choosing one burrow to shelter from predators reflects the exploitation capability of the algorithm. Despite its strong local exploitation potential and good search efficiency, ARO is plagued by the unstable exploration phase so that it tends to fall into the local optimum when processing some complex or high-latitude problems [56]. Since this algorithm has been proposed only a short time ago, there is not any study on the improvement of ARO.

Given the above discussion and the encouragement of NFL theorem, this paper attempts to hybridize AO and ARO to make full use of their respective advantages and then proposes a novel chaotic opposition-based learning-driven hybrid AO and ARO optimizer for solving complex global optimization problems, namely CHAOARO. To our knowledge, such hybridization has never been used before. First, the exploration strategy of AO is integrated into ARO to achieve better overall search performance. Then, an adaptive switching mechanism (ASM) is designed to establish a robust exploration-exploitation balance in the hybrid algorithm. Finally, chaotic opposition-based learning (COBL) is used to update the current best solution to increase the population diversity and avoid the local minimum stagnation. On the basis of various metrics, the performance of the proposed CHAOARO is compared with those of the original AO, ARO, and other existing MAs using a total of thirty-three benchmark functions, including seven unimodal, six multimodal, ten fix-dimension multimodal, and ten modern CEC2019 benchmark functions. Moreover, five industrial engineering design problems and the parameter identification problem of the photovoltaic (PV) model are employed to test the applicability of CHAOARO in solving real-life optimization problems. Experimental results indicate that the proposed work performs better than other comparison algorithms in terms of solution accuracy, convergence speed, and stability. The main contributions of this paper can be shortened as follows:

- A new hybrid meta-heuristic algorithm, CHAOARO, is proposed based on AO and ARO for global optimization;
- The ASM and COBL strategies are adopted to synergistically improve the global exploration and local exploitation capabilities of the hybrid algorithm;
- The performance of CHAOARO is thoroughly compared with that of AO, ARO, and several state-of-the-art optimizers on 23 classical benchmark functions and 10 IEEE CEC2019 test functions;
- Five constrained engineering optimization problems and PV cell/module parameter identification problem are considered to highlight the applicability of CHAOARO in addressing real-life optimization tasks;
- Experimental results demonstrate that CHAOARO can significantly outperform other competitor algorithms in most cases.

The structure of this paper is organized as follows: A brief review of the original AO and ARO is presented in Section 2. Section 3 details the two novel search operators employed, namely ASM and COBL, as well as the framework of the proposed CHAOARO algorithm in this paper. In Section 4, a series of comparison experiments based on the 23 classical benchmark functions and IEEE CEC2019 test suite are carried out to fully validate the superiority of the proposed technique. In Sections 5 and 6, the proposed CHAOARO is applied to solve five common industrial engineering design problems and identify the optimal parameters for PV system, respectively. The conclusion and potential future research are given in Section 7.

2. Preliminary Knowledge

2.1. Aquila Optimizer (AO)

AO is a novel swarm-based meta-heuristic algorithm proposed by Abualigah et al. [46] in 2021 which mimics the intelligent hunting behaviors of the Aquila, a famous genus of bird of prey found in the Northern Hemisphere. Aquila uses its breakneck flight speed and powerful claws to attack the intended prey, and it is able to switch between different predation methods depending on the prey species, including (1) High soar with vertical stoop, (2) Contour flight with short glide attack, (3) Low flight with slow descent attack, and (4) Walking and grabbing prey. Accordingly, in the mathematical model of the AO algorithm, the first two hunting strategies of Aquila are defined as the exploration phase, whereas the last two hunting strategies belong to the exploitation phase. To achieve a smooth transition between global exploration and local exploitation, each phase of the algorithm is performed or not based on the condition that if $t \leq \left(\frac{2}{3}\right) * T$, the exploration steps are executed; otherwise, the exploitation steps will be executed, where t is the current iteration and T is the maximum number of iterations. In the following, the four strategies involved in the mathematical model of AO are described.

2.1.1. Expanded Exploration: High Soar with Vertical Stoop

In the first strategy, Aquila conducts a preliminary search at high altitude to detect the target. Once the best hunting area is determined, Aquila will dive vertically toward the prey. This behavior is modeled as follows:

$$X_i(t+1) = X_{best}(t) \times \left(1 - \frac{t}{T}\right) + X_m(t) - X_{best}(t) \times r_1 \tag{1}$$

where $X_i(t+1)$ denotes the candidate position vector of i-th Aquila in the next iteration $t+1$. $X_{best}(t)$ denotes the best solution obtained so far. $X_m(t)$ represents the mean position value of all individuals in the population, which can be calculated by Equation (2). r_1 is a random value between 0 and 1.

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{2}$$

where N refers to the population size, and $X_i(t)$ denotes the position vector of i-th Aquila in the current iteration.

2.1.2. Narrowed Exploration: Contour Flight with Short Glide Attack

This is the most common hunting strategy utilized by Aquila. When the prey area is located, Aquila will shift from soaring high to hovering above the target prey and look for a suitable opportunity to attack. At this moment, the position update formula is shown as:

$$X_i(t+1) = X_{best}(t) \times \mathrm{LF}(D) + X_r(t) + (y - x) \times r_2 \tag{3}$$

where $X_r(t)$ denotes the position of a random Aquila individual. D denotes the dimension size of the given problem. r_2 is a random number between 0 and 1. $\mathrm{LF}(\cdot)$ stands for the Lévy flight distribution function, which is expressed as follows:

$$\mathrm{LF}(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}},\ \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(1+\beta) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)^{1/\beta} \tag{4}$$

where u and v are random numbers in the range of 0 and 1, $\Gamma(\cdot)$ is the gamma function, and β is a constant equal to 1.5. In Equation (3), y and x are employed to depict the contour spiral shape, which can be calculated using Equation (5).

$$\begin{cases} x = (R + U \times D_1) \times \sin(-\omega \times D_1 + \frac{3\times\pi}{2}) \\ y = (R + U \times D_1) \times \cos(-\omega \times D_1 + \frac{3\times\pi}{2}) \end{cases} \tag{5}$$

where R means a fixed number of search cycles between 1 and 20, U denotes a small value fixed to 0.00565, D_1 is integer numbers from 1 to the dimension size (D), and ω equals 0.005.

2.1.3. Expanded Exploitation: Low Flight with Slow Descent Attack

In the third strategy, when the location of the prey has been roughly specified, Aquila descends vertically and then makes an initial attack on the prey to observe its reaction. This predation behavior can be simulated as in Equation (6).

$$X_i(t+1) = (X_{best}\ (t) - X_m(t)) \times \alpha - r_3 + ((ub - lb) \times r_4 + lb) \times \delta \tag{6}$$

where α and δ are the exploitation adjustment coefficients fixed to 0.1. r_3 and r_4 are random numbers within the interval $[0,\ 1]$. ub and lb are the upper and lower bounds of the search domain, respectively.

2.1.4. Narrowed Exploitation: Walking and Grabbing Prey

In this step, Aquila comes to the land, follows the random escape trajectory of the target prey in pursuit, and finally launches a precise attack. The mathematical representation of this behavior is given by:

$$X_i(t+1) = QF \times X_{best}(t) - G_1 \times X_i(t) \times r_5 - G_2 \times \mathrm{LF}(D) + G_1 \times r_6 \tag{7}$$

$$\begin{cases} QF(t) = t^{\frac{2\times r_7 - 1}{(1-T)^2}} \\ G_1 = 2 \times r_8 - 1 \\ G_2 = 2 \times (1 - \frac{t}{T}) \end{cases} \tag{8}$$

where QF denotes the quality function used to balance the search strategy, G_1 denotes the motion parameter of Aquila in tracking the absconding prey, which is a random number in the range of -1 and 1, G_2 denotes the flight slope of Aquila in tracking the absconding prey, which decreases linearly from 2 to 0, r_5, r_6, r_7, r_8 are all random numbers between 0 and 1.

The flow chart of the original AO algorithm is presented in Figure 1.

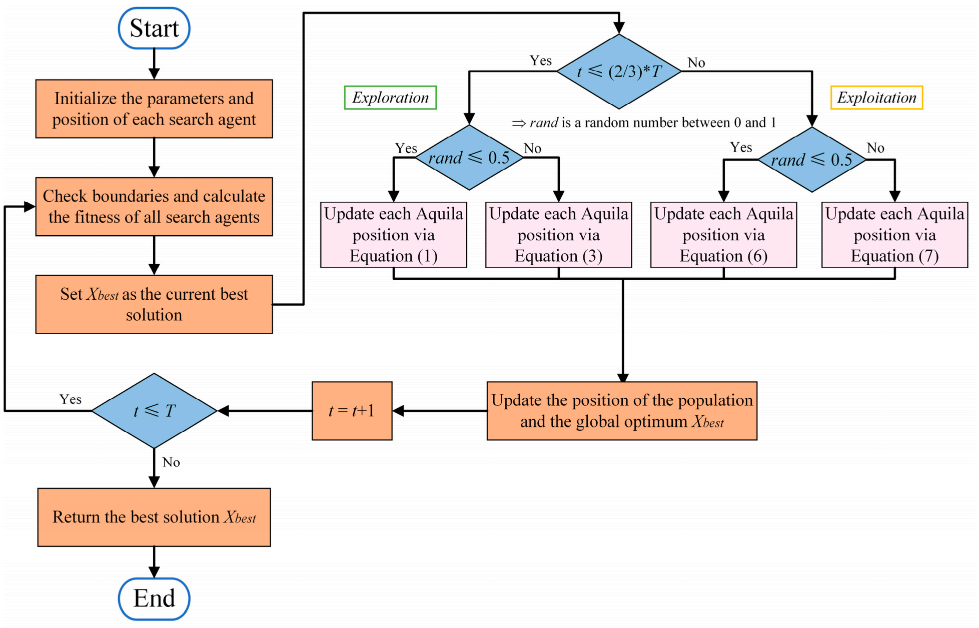
Start
Initialize the parameters and position of each search agent
Check boundaries and calculate the fitness of all search agents
Set Xbest as the current best solution
Yes
t ≤ (2/3)*T
No
Exploration
Exploitation
⇒ rand is a random number between 0 and 1
Yes
rand ≤ 0.5
No
Yes
rand ≤ 0.5
No
Update each Aquila position via Equation (1)
Update each Aquila position via Equation (3)
Update each Aquila position via Equation (6)
Update each Aquila position via Equation (7)
Update the position of the population and the global optimum Xbest
t = t+1
Yes
t ≤ T
No
Return the best solution Xbest
End

Figure 1. Flow chart of AO.

2.2. *Artificial Rabbits Optimization (ARO)*

As a new gradient-free meta-heuristic algorithm developed by Wang et al. [55] in 2022, ARO simulates the survival skills of rabbits in nature. Rabbits are herbivores, which primarily eat grass and leafy weeds. To avoid predators detecting their own nests, rabbits would not consume the grass surrounding the holes; instead, they often look for food away from the nest. This detour foraging strategy is defined as exploration in ARO. Moreover, to further reduce the likelihood of being captured by predators or hunters, rabbits are adept at digging many holes for their nests and then randomly select one as a shelter. This random hiding strategy is considered as exploitation in ARO. Due to their lower level in the food chain, rabbits need to run fast to avoid the danger from numerous predators, which will lead to a decrease in their energy, so rabbits have to adaptively shift between detour foraging and random hiding based on the energy status. With the above knowledge about the biological habits of rabbits, the mathematical model of ARO is constructed, including exploration, transition from exploration to exploitation, and exploitation. Subsequently, we will briefly outline each phase in ARO.

2.2.1. Detour Foraging (Exploration)

In ARO, it is assumed that each rabbit in the population has its own region with some grass and burrows. During foraging activities, the rabbit tends to randomly move to the far-away areas of other individuals in search for food and overlooks what lies close at hand, just like an old Chinese proverb says: “A rabbit doesn’t eat grass near its own nest”. This behavior is called detour foraging, and its mathematical model is represented as follows:

$$X_i(t+1) = X_j(t) + A \times \left(X_i(t) - X_j(t)\right) + round(0.5 \times (0.05 + R_1)) \times n_1, \quad i,j = 1,\ldots,N \text{ and } i \neq j \tag{9}$$

$$A = L \times c \tag{10}$$

$$L = \left(e - e^{\left(\frac{t-1}{T}\right)^2}\right) \times \sin(2\pi R_2) \tag{11}$$

$$c(k) = \begin{cases} 1, \ if\ k == g(l) \\ 0, \ otherwise \end{cases} \quad k = 1,\ldots,D \text{ and } l = 1,\ldots,\lceil R_3 \times D \rceil \tag{12}$$

$$g = randperm(D) \tag{13}$$

$$n_1 \sim N(0,1) \tag{14}$$

where $X_i(t+1)$ is the candidate position of the i-th rabbit in the next iteration $t+1$. $X_i(t)$ and $X_j(t)$ denotes the position of the i-th rabbit and the j-th rabbit in the current iteration t, respectively. N denotes the population size. t is the current iteration. T is the maximum number of iterations. D represents the dimension size of the specific problem. $\lceil \cdot \rceil$ stands for the ceiling function. $round(\cdot)$ signifies rounding to the nearest integer. $randperm(\cdot)$ represents a randomly selected integer between 1 and D. R_1, R_2, and R_3 are all random number in the interval [0, 1]. n_1 follows the standard normal distribution. L represents the length of the movement step while conducting the detour foraging.

2.2.2. Transition from Exploration to Exploitation

In ARO, rabbits are inclined to implement continual detour foraging in the early stage of the iteration, whereas in the later stages of the search, they frequently execute random hiding. To maintain a good balance between exploration and exploitation, a concept of rabbit energy E is introduced, which will gradually decrease over time. The formula for the energy factor E is as follows:

$$E(t) = 4(1 - \frac{t}{T}) \ln \frac{1}{R_4} \tag{15}$$

where R_4 is a random number between 0 and 1. The value of the energy coefficient E varies in the interval [0, 2]. When $E > 1$, it means that the rabbit has lots of energy to randomly explore the foraging area of other different individuals so that the detour foraging occurs, and this phase is defined as the exploration. In the case of $E \leq 1$, it indicates that the rabbit has less energy for physical activity, so it needs to perform random hiding to escape from predation, and the ARO algorithm enters the exploitation phase.

2.2.3. Random Hiding (Exploitation)

Rabbits are usually confronted with chase and attack from predators. To survive, they would dig a number of different holes around the nest for shelter. In ARO, at each iteration, a rabbit always generates D burrows along the dimension of the search space and then randomly selects one among them for hiding to decrease the probability of being captured. The mathematical model of this behavior is simulated as follows:

$$X_i(t+1) = X_i(t) + A \times (R_5 \times b_{i,r}(t) - X_i(t)) \tag{16}$$

$$b_{i,r}(t) = X_i(t) + H \times g_r(k) \times X_i(t) \tag{17}$$

$$g_r(k) = \begin{cases} 1, \; if \; k == \lceil R_6 \times D \rceil \\ 0, \; otherwise \end{cases} \tag{18}$$

$$H = \frac{T-t+1}{T} \times n_2 \tag{19}$$

$$n_2 \sim N(0,1) \tag{20}$$

where the parameter A can be calculated using Equations (10)–(13), $b_{i,r}(t)$ represents a randomly selected burrow of the i-th rabbit from D burrows used for hiding in the current iteration t, R_5 and R_6 are two random numbers between 0 and 1, and n_2 follows the standard normal distribution.

The flow chart of the original ARO algorithm is presented in Figure 2.

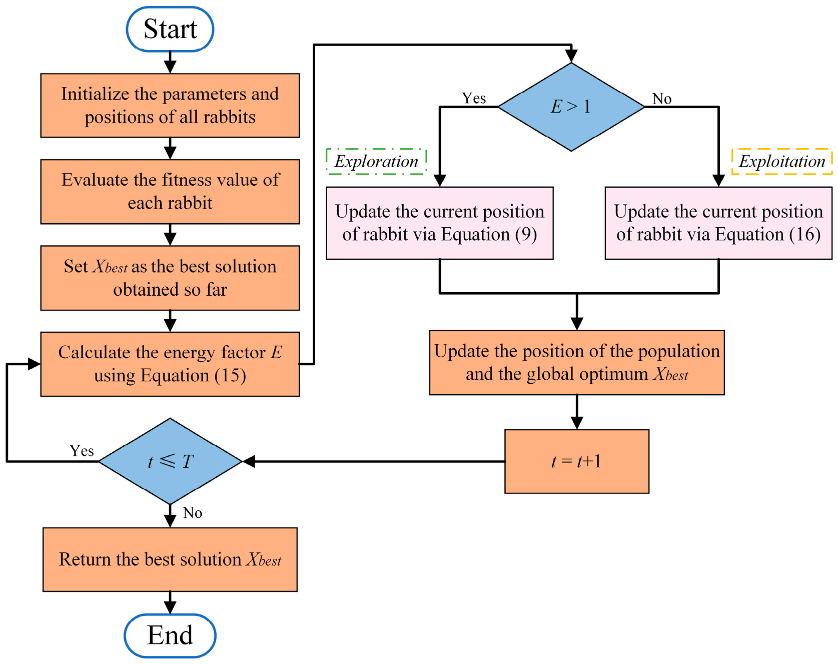
Start
Initialize the parameters and positions of all rabbits
Evaluate the fitness value of each rabbit
Set X_{best} as the best solution obtained so far
Calculate the energy factor E using Equation (15)
$E > 1$
Yes
No
Exploration
Exploitation
Update the current position of rabbit via Equation (9)
Update the current position of rabbit via Equation (16)
Update the position of the population and the global optimum X_{best}
$t = t+1$
$t \leqslant T$
Yes
No
Return the best solution X_{best}
End

Figure 2. Flow chart of ARO.

3. The Proposed CHAOARO Algorithm

3.1. Hybridization of AO with ARO Algorithms

In the exploration phase of AO, the algorithm simulates the predatory behavior of Aquila rapidly pursuing target prey within a wide flight area. When updating the positions of search agents, the current global optimal position is added directly to improve searchability and accelerate the convergence (see Equations (1) and (3)). Nevertheless, at the later stage, the selected search domain cannot be exploited thoroughly, and the weak escape mechanism of Lévy flight makes the algorithm easily fall into local optima (see Equation (7)). As demonstrated in earlier studies: the convergence curve of AO remains the same during later iterations, especially on the multi-modal benchmark functions [49]. Most of the quality results achieved throughout the optimization process are likely to come from the contributions of exploration. Thus, the AO algorithm has an excellent global exploration capability, but meanwhile, its exploitation phase is still inadequate. On the contrary, the experimental findings of the ARO algorithm indicate that the defects of poor population diversity and slow convergence speed exist in the early exploration phase, and the detour foraging mechanism cannot provide sufficient volatility for search agents to explore the whole search space as much as possible (see Equation (9)). As the number of iterations increases, the energy coefficient E of the rabbit gradually decreases, and ARO enters the development stage. The random hiding behavior makes the individuals in the swarm continuously move closer to the neighborhood of the global optimal point, which significantly improves the solution accuracy (see Equation (16)). So, ARO owns good local exploitation ability.

Based on the above characteristic analysis, we consider the framework of ARO as the main body and preliminarily hybridize the exploration phase of AO with it to give full play to the strengths of the two basic algorithms and preserve more robust global and local search capabilities as well as faster convergence speed.

3.2. Adaptive Switching Mechanism (ASM)

For most bio-inspired optimizers, how to more effectively balance exploration and exploration is key to improving the algorithm's performance. Exploration is a process of leaving any local region and subsequently exploring unknown spaces, while exploitation refers to the process of probing a local region to find a promising solution. Generally, exploration should be executed in the early stages of the algorithm, whereas exploitation is implemented in the later stages [57]. A successful hybrid algorithm needs to be equipped with effective switching mechanisms to reasonably balance the relationship between exploration and exploitation in the search for the global optimum [58]. In order to better balance the exploration process of AO and the exploitation phase of ARO, another parameter needs to be involved in the new combined optimizer to guide the search direction of individuals. The random number may be an option [54], however, considering the starvation ratio F of vultures in the African Vultures Optimization Algorithm (AVOA) [59] can lead the algorithm to transit smoothly from exploration to exploitation, thus providing competitive performance even when solving challenging optimization problems, therefore, an adaptive switching mechanism (ASM) is proposed based on this. The mathematical representation for ASM is as follows:

$$F = (2 \times rand + 1) \times z \times (1 - \frac{t}{T}) + g \tag{21}$$

$$g = h \times (\sin^{w}(\frac{\pi}{2} \times \frac{t}{T}) + \cos(\frac{\pi}{2} \times \frac{t}{T}) - 1) \tag{22}$$

where $rand$ denotes a random value in the interval $[0,\ 1]$, z is a random number between -1 and 1, t and T are the current number of iterations and the maximum iteration, respectively, h is a random number between -2 and 2, and w is a constant equal to 2.5. Figure 3 illustrates the dynamic behavior of F over 1000 iterations during the optimization operation. As per the AVOA algorithm, when $|F| \geq 1$, vultures look for food in different regions (exploration), and if $|F| < 1$, vultures search for food near the optimal solution (exploitation). The ASM based on the F-value ensures that the algorithm focuses on global exploration in the early iterations whilst retaining the possibility of local search. As the value of F gradually decreases, the algorithm performs more local exploitation in the later stage.

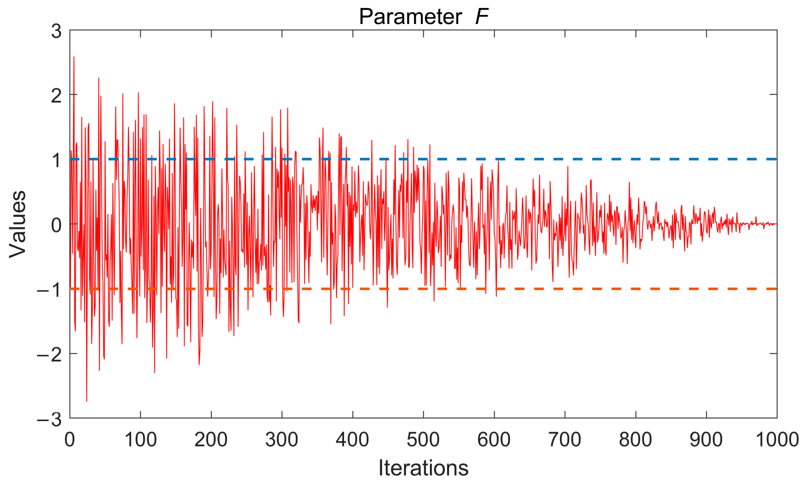

Figure 3. Dynamic behavior of F during 1000 iterations.

3.3. Chaotic Opposition-Based Learning (COBL)

Opposition-based learning (OBL) is a powerful optimization technique in the field of intelligence computation first proposed by Tizhoosh [60]. In general, MAs start with some initial random solutions and try to continuously approach the global optimum through iterative calculation. The search procedure terminates once certain pre-defined conditions are met. If no related advance information about the solution is available, it may take quite a long time to converge. As illustrated in Figure 4, the main principle of OBL is to simultaneously evaluate the fitness values of the current solution and its corresponding opposite solution, then retain the dominant individual to continue with the next iteration, thus effectively strengthening the population diversity. It turns out that the generated opposite candidate solution has almost a 50% higher probability of being close to the global optimum than the current solution. Therefore, OBL has been widely implemented to enhance the optimization performance of many basic MAs [61,62]. The mathematical model for OBL is presented as follows:

$$\hat{X} = lb + ub - X \tag{23}$$

where $\hat{X}$ denotes the generated opposite solution, X is the current solution, ub and lb represent the upper and lower bounds of the search domain, respectively.

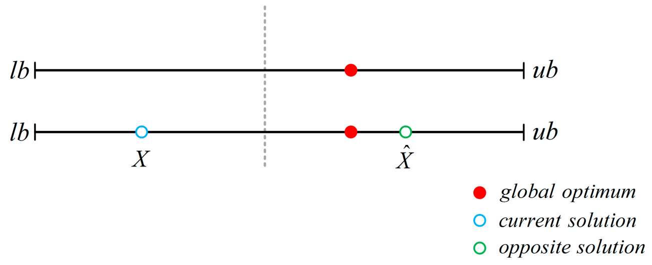

Figure 4. Principle of the traditional opposition-based learning.

As can be seen from Equation (23), OBL can only yield the opposite solution at a fixed location, which works well in the early stage of optimization, but as the search process continues, it may occur that the opposite solution falls near the local optimum, and other individuals will quickly move towards this region, resulting in premature convergence and poor solution accuracy. To this end, reference [63] proposed a random opposition-based learning (ROBL) strategy by introducing the random perturbation to modify Equation (23) as follows:

$$\hat{X} = lb + ub - rand \cdot X \tag{24}$$

where $rand$ is a random number between 0 and 1. Although ROBL can enhance the population diversity of the algorithm and help avoid local optima to some extent, the algorithm's convergence speed is still not satisfactory.

Chaos is a dynamic behavior found in nonlinear systems with three essential characteristics of chaos: ergodicity, regularity, and randomness [64]. Compared with random search, which mainly relies on probability distributions, the chaotic map can thoroughly investigate the search space at a much higher speed benefiting from its dynamic properties. To further improve population diversity and global convergence speed, this paper combines traditional OBL together with chaotic maps and proposes a chaotic opposition-based learning (COBL) strategy. The mathematical formula is given as follows:

$$\widehat{X^{co}} = lb + ub - \varphi \cdot X \tag{25}$$

where $\widehat{X^{co}}$ denotes the generated inverse solution of X, and φ is the chaotic map value.

In our paper, ten common chaotic maps are used to combine with OBL, including Chebyshev map, circle map, gauss map, iterative map, logistic map, piecewise map, sine map, singer map, sinusoidal map, and tent map. The images of these chaotic maps are shown in Figure 5, and the specific equations are listed in Table 2. In the next section, we will test in detail which map is more suitable to be employed for boosting the optimization performance of the proposed algorithm.

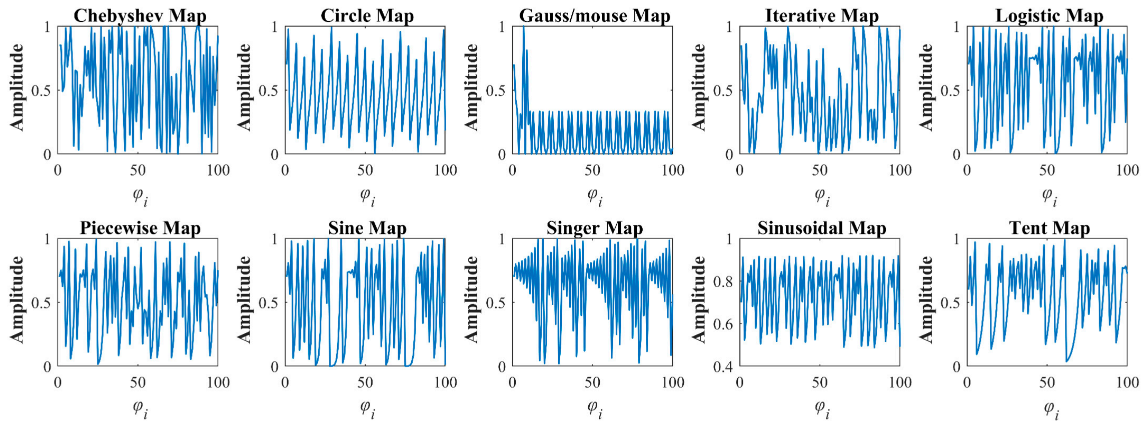

Figure 5. Visualization of ten commonly used chaotic maps.

Table 2. Ten chaotic maps.

No	Map Name	Equation
CM1	Chebyshev	$\varphi_{i+1} = \cos\left(i \cos^{-1}(\varphi_i)\right)$
CM2	Circle	$\varphi_{i+1} = \mathrm{mod}\left(\varphi_i + b - \left(\frac{a}{2\pi}\right)\sin(2\pi\varphi_i), 1\right); a = 0.5, b = 0.2$
CM3	Gauss/mouse	$\varphi_{i+1} = \begin{cases} 1, & \varphi_i = 0 \\ \frac{1}{\mathrm{mod}(\varphi_i,\, 1)}, & \text{otherwise} \end{cases}$
CM4	Iterative	$\varphi_{i+1} = \sin\left(\frac{a\pi}{\varphi_i}\right), a = 0.7$
CM5	Logistic	$\varphi_{i+1} = a\varphi_i(1 - \varphi_i), a = 4$
CM6	Piecewise	$\varphi_{i+1} = \begin{cases} \varphi_i/P, 0 \le \varphi_i < P \\ \frac{\varphi_i - P}{0.5 - P}, P \le \varphi_i \le 0.5 \\ \frac{1 - P - \varphi_i}{0.5 - P}, 0.5 \le \varphi_i < 1 - P \\ \frac{1 - \varphi_i}{P}, 1 - P \le \varphi_i < 1 \end{cases}, P = 0.4$
CM7	Sine	$\varphi_{i+1} = \frac{a}{4}\sin(\pi\varphi_i), a = 4$
CM8	Singer	$\varphi_{i+1} = \mu\left(7.86\varphi_i - 23.31\varphi_i^2 + 28.75\varphi_i^3 - 13.301875]\varphi_i^4\right), \mu = 1.07$
CM9	Sinusoidal	$\varphi_{i+1} = a\varphi_i \sin(\pi\varphi_i), a = 2.3$
CM10	Tent	$\varphi_{i+1} = \begin{cases} \varphi_i/0.7, & \varphi_i < 0.7 \\ \frac{10}{3}(1 - \varphi_i), & \varphi_i \ge 0.7 \end{cases}$

3.4. Detailed Design of CHAOARO

On the basis of the above Sections 3.1–3.3, the proposed methodology is summarized as follows. First, the exploration phase of AO is hybridized with the exploitation phase of ARO to achieve a more stable overall optimization performance. Then, ASM is designed to control the smooth switch from exploration to exploitation, which improves the efficiency of the algorithm in finding the most promising domain. In addition, AO and ARO share a common drawback of local optima stagnation. For this reason, the COBL strategy is utilized to update the current optimal solution before the next iteration calculation to further enhance the population diversity and local optima avoidance. All these operations significantly boost the convergence speed, solution accuracy, and robustness of both single algorithms. Finally, this new hybrid version of Aquila Optimizer and Artificial Rabbits Optimization driven by chaotic opposition-based learning can be abbreviated as CHAOARO. Figure 6 presents the flow chart of CHAOARO, and its pseudo-code is described in Algorithm 1.

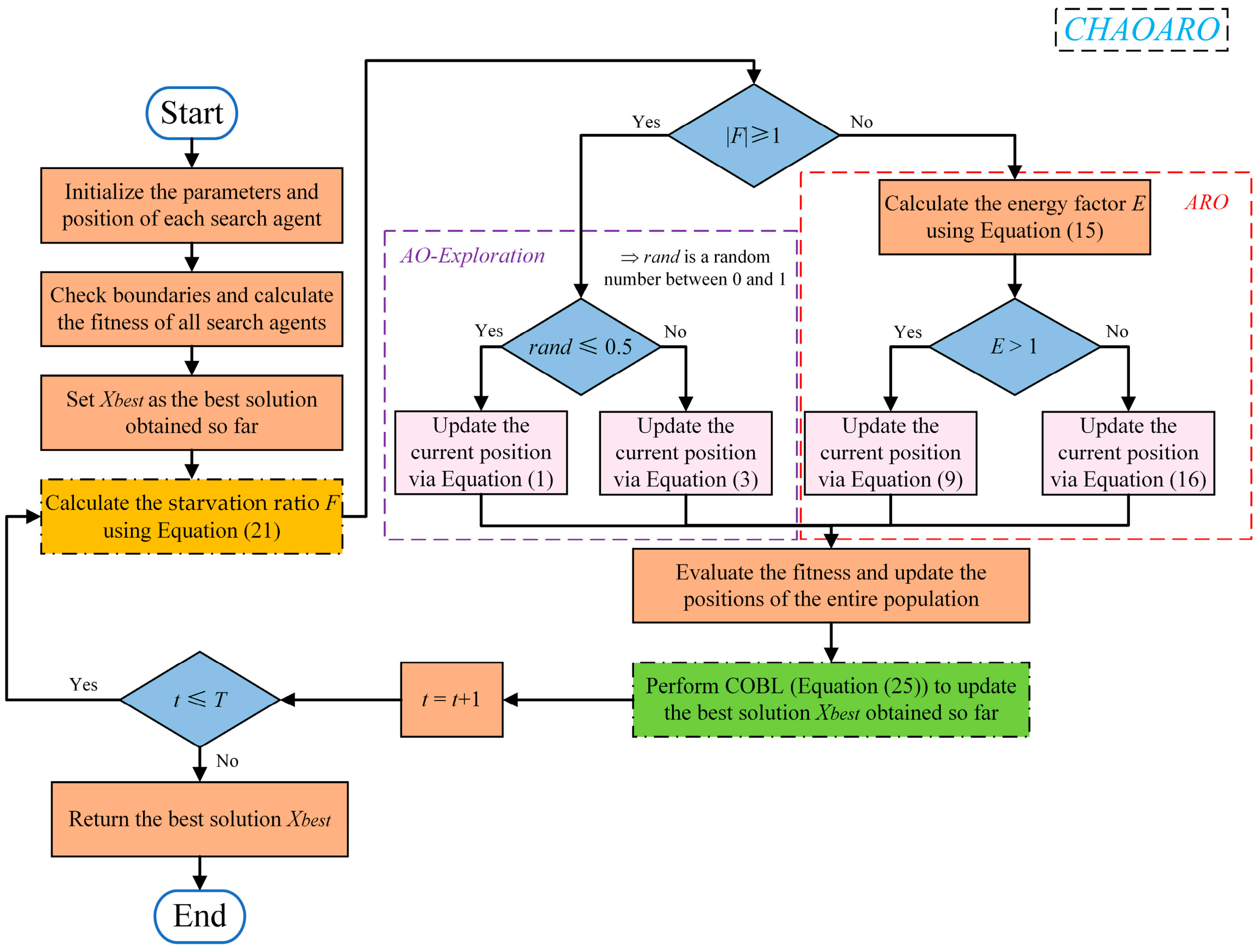
CHAOARO
Start
Initialize the parameters and position of each search agent
Check boundaries and calculate the fitness of all search agents
Set Xbest as the best solution obtained so far
Calculate the starvation ratio F using Equation (21)
Yes
|F|≥1
No
AO-Exploration
⇒ rand is a random number between 0 and 1
Yes
rand ≤ 0.5
No
Update the current position via Equation (1)
Update the current position via Equation (3)
ARO
Calculate the energy factor E using Equation (15)
Yes
E > 1
No
Update the current position via Equation (9)
Update the current position via Equation (16)
Evaluate the fitness and update the positions of the entire population
Perform COBL (Equation (25)) to update the best solution Xbest obtained so far
t = t+1
Yes
t ≤ T
No
Return the best solution Xbest
End

Figure 6. Flow chart of the proposed CHAOARO.

Computational complexity is an important metric to evaluate the time consumption of an algorithm when solving optimization problems. With the pseudo-code shown in Algorithm 1, it can be concluded that the computational complexity of the proposed CHAOARO is related to the population size (N), the dimension space of problems (D), and the maximum number of iterations (T). In the initialization process, the positions of all search agents are randomly generated in the search space, which requires a computational complexity of $O(N)$. Afterward, throughout the iteration procedure, it takes $O(N \times T + N \times D \times T)$ to carry out the fitness evaluation and position update. Accordingly, the total computational complexity of CHAOARO should be $O(N + NT + NDT)$. Compared with the canonical AO and ARO algorithms, the computational complexity of the proposed method does not increase.

Algorithm 1. Pseudo-code of the proposed CHAOARO

```
1. Initialize the population size $N$ and the maximum iterations $T$
2. Initialize the position of each search agent $X_i (i = 1, 2, \cdots, N)$
3. While $t \leq T$
4.   Check if the position goes beyond the search limits and adjust it
5.   Evaluate the fitness values of all search agents
6.   Set $X_{best}$ as the best solution obtained so far
7.   For each $X_i$
8.     Calculate the starvation ratio $F$ using Equation (21)   //ASM
9.     If $|F| \geq 1$ then   //Exploration of AO
10.      If $rand \leq 0.5$ then   $\Longrightarrow rand$ is a random number between 0 and 1
11.        Update the search agent's position using Equation (1)
12.      Else
13.        Update the search agent's position using Equation (3)
14.      End If
15.    Else
16.      Calculate the energy factor $E$ using Equation (15)
17.      If $E > 1$ then
18.        Update the search agent's position using Equation (9) //Detour foraging of ARO
19.      Else
20.        Update the search agent's position using Equation (16) //Random hiding of ARO
21.      End If
22.    End If
23.    Generate the opposite solution of $X_{best}$ using Equation (25),
       Select the one with better fitness into the next generation   //COBL
24.  End For
25.  $t = t + 1$
26. End While
27. Return $X_{best}$
```

4. Experimental Results and Discussion

In this section, a series of systematic experimental studies are conducted on 23 classical benchmark functions (IEEE CEC2005 test suite) and the IEEE CEC2019 test set to comprehensively investigate the performance of the proposed CHAOARO method. In order to illustrate the superiority of CHAOARO, seven state-of-the-art MAs, and two improved algorithms are employed for comparison analysis, namely AO [46], GWO [27], WOA [30], SCA [21], TSA [33], GJO [36], ARO [55], Weighted Chimp Optimization Algorithm (WChOA) [65], and Dynamic Arithmetic Optimization Algorithm (DAOA) [66]. Table 3 lists the main parameters used in each algorithm, which are the same as those recommended in the original research papers. Note that the parameter settings appearing in the position update model (Equations (1) and (3)) for the exploration phase of AO are also inherited into CHAOARO. In the experiment, for all involved algorithms, the population size is fixed to 30, and the maximum iteration is set as 30 for a fair comparison.

Table 3. Parameter settings for CHAOARO and other selected competitor algorithms.

Algorithm	Parameter Setting
AO [46]	$U = 0.00565; R = 10; \omega = 0.005; \alpha = 0.1; \delta = 0.1; G_1 \in [-1,\ 1]; G_2 = [2,\ 0]$
GWO [27]	$a = [2,\ 0]$
WOA [30]	$b = 1; a_1 = [2,\ 0]; a_2 = [-2, -1]$
SCA [21]	$a = 2$
TSA [33]	$P_{min} = 1; P_{max} = 4$
GJO [36]	$c_1 = 1.5$
ARO [55]	—
WChOA [65]	$f = [2.5,\ 0];\ M = Gauss\ chaotic\ value$
DAOA [66]	$\alpha = 25; \mu = 0.001$
CHAOARO	$U = 0.00565; R = 10; \omega = 0.005$

Through 30 independent runs, the obtained average fitness (Avg) and standard deviation (Std) results are recorded as evaluation criteria, where the average fitness value reflects the optimization accuracy of an algorithm, which can be calculated as follows:

$$\text{Avg} = \frac{1}{times} \sum_{i=1}^{times} O_i \tag{26}$$

where *times* stands for the total number of runs, and O_i denotes the outcome of the i-th operation. The closer the average fitness is to the theoretical optimal solution, the better the searchability of the algorithm. On the other hand, the standard deviation reveals the departure degree of the experimental data, and the smaller the standard deviation, the higher the stability of the algorithm. The mathematical formula for the standard deviation is as follows:

$$\text{Std} = \sqrt{\frac{1}{times - 1} \sum_{i=1}^{times} (O_i - \text{Avg})^2} \tag{27}$$

Besides, two non-parametric statistical techniques, including the Friedman ranking test [67] and the Wilcoxon rank-sum test [68], are further performed to check whether CHAOARO is significantly different from other comparison methods. All the experiments are implemented in MATLAB R2017a with Microsoft Windows 10 system, and the hardware platform configuration of the computer is Intel (R) Core (TM) i5-10300H CPU @ 2.50 GHz and 16 GB RAM.

4.1. Experiment 1: Classical Benchmark Functions

To verify the effectiveness of the proposed CHAOARO in solving simple numerical optimization problems, we select 23 classical benchmark functions with different characteristics from [10] for testing. These benchmark functions can be classified into three categories: unimodal, multimodal, and fix-dimension multimodal functions. Unimodal functions (F_1~F_7) have only one global optimum, which can be utilized to estimate the exploitation propensity of the algorithm. For multimodal and fix-dimension multimodal functions (F_8~F_{23}), they contain a large number of local optima and are therefore usually adopted to examine the exploration and local optima avoidance abilities of the algorithm. Table 4 provides the details of the 23 classical benchmark functions.

First, the impact of ten different combinations of the chaotic map and opposition-based learning on the performance of the proposed CHAOARO algorithm is studied. Afterward, we compare CHAOARO with other selected advanced algorithms in turn with respect to numerical results, convergence behavior, boxplot, computational consumption, Wilcoxon rank-sum test, and scalability in the dimensional space.

4.1.1. Chaotic Map Selection Analysis

The COBL strategy designed in Section 3.3 integrates chaotic maps and traditional opposition-based learning to prevent the algorithm from getting trapped in the local optimum during iterations. To confirm which chaotic map in Table 2 should be used, this part tests the optimization performance of CHAOARO with ten different chaotic maps on 23 classical benchmark functions. After 30 independent runs, the average fitness and standard deviation results obtained are listed in Table 5.

As can be clearly seen from Table 5, in most test cases, CHAOARO based on CM3 (gauss/mouse map) performs better than that using other chaotic maps. When solving unimodal functions F_1~F_7, CM3 (gauss/mouse map) ranks first among all its peers, especially on F_1~F_4, which can consistently find the theoretical optimal value (0). For multimodal and fix-dimension multimodal functions F_8~F_{23}, the gauss/mouse map also provides satisfactory solutions. Eventually, CHAOARO with gauss/mouse map obtains the minimum Friedman mean ranking value of 1.4348. This indicates that the gauss/mouse map has the best effect in improving the comprehensive performance of the algorithm; therefore, it is selected to generate chaotic map values φ for the COBL strategy in this paper.

Table 4. Characteristics of the 23 classical benchmark functions (UM: unimodal, MM: multimodal, FM: fix-dimension multimodal, Dim: dimension, Range: search boundaries, $F_{\min}$: theoretical optimal value).

Function	**Type**	**Dim (*D*)**	**Range**	$F_{\min}$
$F_1(x) = \sum_{i=1}^{D} x_i^2$	UM	30	$[-100, 100]$	0
$F_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	UM	30	$[-10, 10]$	0
$F_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{D} x_j)^2$	UM	30	$[-100, 100]$	0
$F_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le D\}$	UM	30	$[-100, 100]$	0
$F_5(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	UM	30	$[-30, 30]$	0
$F_6(x) = \sum_{i=1}^{D} (\lvert x_i + 0.5 \rvert)^2$	UM	30	$[-100, 100]$	0
$F_7(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	UM	30	$[-1.28, 1.28]$	0
$F_8(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	MM	30	$[-500, 500]$	$-418.9829 \times D$
$F_9(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	MM	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	MM	30	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	MM	30	$[-600, 600]$	0
$F_{12}(x) =$ $\frac{\pi}{D}\left\{10\sin(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}, u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	MM	30	$[-50, 50]$	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{D}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_D - 1)^2[1 + \sin^2(2\pi x_n)]\right\} +$ $\sum_{i=1}^{D} u(x_i, 5, 100, 4)$	MM	30	$[-50, 50]$	0
$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25}(j + \sum_{i=1}^{2}(x_i - a_{ij})^6)^{-1})^{-1}$	FM	2	$[-65, 65]$	0.998
$F_{15}(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	FM	4	$[-5, 5]$	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	FM	2	$[-5, 5]$	-1.0316
$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	FM	2	$[-5, 5]$	0.398
$F_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $\times\left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_2 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	FM	2	$[-2, 2]$	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	FM	3	$[-1, 2]$	-3.8628
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	FM	6	$[0, 1]$	-3.32
$F_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	FM	4	$[0, 10]$	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	FM	4	$[0, 10]$	-10.4028
$F_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	FM	4	$[0, 10]$	-10.5363

4.1.2. Evaluation of Exploitation and Exploration

Based on the properties of the unimodal, multimodal, and fix-dimension multimodal benchmark functions described earlier, in this subsection, we perform a systematic evaluation of the exploitation and exploration propensities of the proposed optimizer. The specific parameter settings have been shown in Table 3. After 30 runs on the 30-dimensional test functions F_1~F_{23}, the average fitness and standard deviation values obtained by CHAOARO and other competitor algorithms are recorded in Table 6.

Table 5. Comparison results of different chaotic maps on 23 benchmark functions.

F_n	Criteria	CM1	CM2	CM3	CM4	CM5	CM6	CM7	CM8	CM9	CM10
	Avg	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	7.26×10^{-288}	1.84×10^{-165}	**0.00×10^{0}**
F_1	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
	Rank	1	1	1	1	1	1	1	9	10	1
	Avg	3.68×10^{-286}	2.81×10^{-230}	**0.00×10^{0}**	5.71×10^{-248}	1.21×10^{-310}	3.49×10^{-203}	1.07×10^{-305}	1.03×10^{-144}	3.75×10^{-84}	8.25×10^{-204}
F_2	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	1.21×10^{-144}	6.69×10^{-84}	**0.00×10^{0}**
	Rank	4	6	1	5	2	8	3	9	10	7
	Avg	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	2.97×10^{-286}	5.18×10^{-165}	**0.00×10^{0}**
F_3	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
	Rank	1	1	1	1	1	1	1	9	10	1
	Avg	2.59×10^{-286}	7.95×10^{-231}	**0.00×10^{0}**	3.07×10^{-248}	5.02×10^{-311}	1.12×10^{-203}	3.84×10^{-306}	5.29×10^{-145}	2.75×10^{-81}	2.05×10^{-204}
F_4	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	8.98×10^{-145}	4.23×10^{-81}	**0.00×10^{0}**
	Rank	4	6	1	5	2	8	3	9	10	7
	Avg	1.19×10^{-2}	3.11×10^{-2}	**3.21×10^{-3}**	3.04×10^{-2}	8.96×10^{-3}	8.00×10^{-3}	5.91×10^{-3}	3.71×10^{-3}	1.65×10^{-2}	1.37×10^{-2}
F_5	Std	3.74×10^{-2}	1.35×10^{-1}	**3.70×10^{-3}**	7.81×10^{-2}	3.68×10^{-2}	2.69×10^{-2}	8.18×10^{-3}	5.05×10^{-3}	6.09×10^{-2}	3.63×10^{-2}
	Rank	6	10	1	9	5	4	3	2	8	7
	Avg	4.45×10^{-6}	3.24×10^{-6}	**2.13×10^{-6}**	6.26×10^{-6}	3.12×10^{-6}	3.52×10^{-6}	3.51×10^{-6}	2.32×10^{-6}	3.74×10^{-6}	4.28×10^{-6}
F_6	Std	8.39×10^{-6}	4.45×10^{-6}	**2.86×10^{-6}**	1.76×10^{-5}	4.16×10^{-6}	3.94×10^{-6}	5.40×10^{-6}	2.70×10^{-6}	9.22×10^{-6}	4.85×10^{-6}
	Rank	9	4	1	10	3	6	5	2	7	8
	Avg	1.81×10^{-4}	2.44×10^{-4}	**1.76×10^{-4}**	2.10×10^{-4}	2.65×10^{-4}	2.03×10^{-4}	2.03×10^{-4}	1.80×10^{-4}	2.28×10^{-4}	2.00×10^{-4}
F_7	Std	1.77×10^{-4}	1.90×10^{-4}	**1.74×10^{-4}**	1.63×10^{-4}	2.40×10^{-4}	2.41×10^{-4}	2.03×10^{-4}	1.38×10^{-4}	1.81×10^{-4}	1.55×10^{-4}
	Rank	3	9	1	7	10	6	5	2	8	4
	Avg	−9051.0832	−9152.4981	**−9400.3304**	−9281.9785	−9290.1890	−8903.7512	−9024.0377	−9007.2763	−9003.1787	−9282.5212
F_8	Std	8.87×10^{2}	9.51×10^{2}	**7.22×10^{2}**	7.99×10^{2}	8.87×10^{2}	5.49×10^{2}	6.66×10^{2}	1.17×10^{3}	6.82×10^{2}	9.29×10^{2}
	Rank	6	5	1	4	2	10	7	8	9	3
	Avg	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
F_9	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
	Rank	1	1	1	1	1	1	1	1	1	1
	Avg	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**	**8.88×10^{-16}**
F_{10}	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
	Rank	1	1	1	1	1	1	1	1	1	1
	Avg	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
F_{11}	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
	Rank	1	1	1	1	1	1	1	1	1	1
	Avg	2.87×10^{-7}	5.11×10^{-7}	1.75×10^{-7}	2.22×10^{-7}	3.24×10^{-7}	2.55×10^{-7}	2.38×10^{-7}	**1.62×10^{-7}**	2.53×10^{-7}	3.50×10^{-7}
F_{12}	Std	5.34×10^{-7}	1.37×10^{-6}	2.26×10^{-7}	2.36×10^{-7}	7.49×10^{-7}	2.77×10^{-7}	3.87×10^{-7}	**2.23×10^{-7}**	3.81×10^{-7}	4.45×10^{-7}
	Rank	7	10	2	3	8	6	4	1	5	9
	Avg	2.73×10^{-6}	1.79×10^{-6}	1.75×10^{-6}	3.86×10^{-6}	3.09×10^{-6}	**1.40×10^{-6}**	1.99×10^{-6}	1.92×10^{-6}	1.73×10^{-6}	2.92×10^{-6}
F_{13}	Std	5.59×10^{-6}	2.13×10^{-6}	2.31×10^{-6}	6.32×10^{-6}	5.80×10^{-6}	**1.63×10^{-6}**	2.19×10^{-6}	3.43×10^{-6}	1.80×10^{-6}	5.76×10^{-6}
	Rank	7	4	3	10	9	1	6	5	2	8

Table 5. *Cont.*

F_n	Criteria	CM1	CM2	CM3	CM4	CM5	CM6	CM7	CM8	CM9	CM10
	Avg	1.97×10^{0}	**9.98×10^{-1}**	**9.98×10^{-1}**	1.13×10^{0}	1.13×10^{0}	1.84×10^{0}	1.59×10^{0}	1.20×10^{0}	1.20×10^{0}	1.06×10^{0}
F_{14}	Std	2.97×10^{0}	1.75×10^{-14}	**7.14×10^{-17}**	5.03×10^{-1}	5.24×10^{-1}	2.74×10^{0}	2.18×10^{0}	6.05×10^{-1}	2.35×10^{-1}	3.62×10^{-1}
	Rank	10	2	1	4	5	9	8	7	6	3
	Avg	3.38×10^{-4}	3.08×10^{-4}	3.08×10^{-4}	**3.07×10^{-4}**	3.52×10^{-4}	3.42×10^{-4}	3.42×10^{-4}	3.69×10^{-4}	3.38×10^{-4}	**3.07×10^{-4}**
F_{15}	Std	1.68×10^{-4}	8.34×10^{-8}	1.55×10^{-8}	4.31×10^{-8}	2.35×10^{-4}	1.68×10^{-4}	1.86×10^{-4}	2.35×10^{-4}	1.67×10^{-4}	**2.41×10^{-9}**
	Rank	6	4	3	2	9	7	8	10	5	1
	Avg	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**
F_{16}	Std	5.76×10^{-16}	6.12×10^{-16}	**5.21×10^{-16}**	5.83×10^{-16}	5.53×10^{-16}	6.12×10^{-16}	5.61×10^{-16}	5.98×10^{-16}	5.90×10^{-16}	5.83×10^{-16}
	Rank	4	9	1	5	2	9	3	8	7	5
	Avg	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**	**3.98×10^{-1}**
F_{17}	Std	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**
	Rank	1	1	1	1	1	1	1	1	1	1
	Avg	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**	**3.00×10^{0}**
F_{18}	Std	1.53×10^{-15}	1.32×10^{-15}	1.26×10^{-15}	2.18×10^{-15}	6.39×10^{-16}	1.31×10^{-15}	**5.71×10^{-16}**	1.42×10^{-15}	1.40×10^{-15}	1.37×10^{-15}
	Rank	9	5	3	10	2	4	1	8	7	6
	Avg	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**	**-3.8628**
F_{19}	Std	2.64×10^{-15}	2.60×10^{-15}	2.60×10^{-15}	2.60×10^{-15}	2.67×10^{-15}	2.64×10^{-15}	2.67×10^{-15}	**2.55×10^{-15}**	2.60×10^{-15}	2.64×10^{-15}
	Rank	6	2	2	2	9	6	9	1	2	6
	Avg	-3.2784	-3.2744	**-3.2982**	-3.2824	-3.2902	-3.2586	-3.2744	-3.2823	-3.2902	-3.2863
F_{20}	Std	5.83×10^{-2}	5.92×10^{-2}	**4.84×10^{-2}**	5.83×10^{-2}	5.35×10^{-2}	6.03×10^{-2}	5.92×10^{-2}	5.70×10^{-2}	5.35×10^{-2}	5.54×10^{-2}
	Rank	7	9	1	5	2	10	8	6	2	4
	Avg	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**	**-10.1532**
F_{21}	Std	**5.56×10^{-15}**	5.63×10^{-15}	**5.56×10^{-15}**	7.05×10^{-10}	5.63×10^{-15}	5.58×10^{-15}	1.75×10^{-14}	**5.56×10^{-15}**	5.69×10^{-15}	5.58×10^{-15}
	Rank	1	6	1	10	6	4	9	1	8	4
	Avg	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**	**-10.4029**
F_{22}	Std	4.66×10^{-16}	5.71×10^{-16}	**0.00×10^{0}**	2.35×10^{-10}	3.30×10^{-16}	**0.00×10^{0}**	3.30×10^{-16}	3.32×10^{-15}	1.81×10^{-15}	4.66×10^{-16}
	Rank	5	7	1	10	3	1	3	9	8	5
	Avg	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**	**-10.5364**
F_{23}	Std	1.68×10^{-15}	1.85×10^{-9}	1.64×10^{-15}	1.71×10^{-15}	1.68×10^{-15}	1.65×10^{-15}	**1.55×10^{-15}**	1.68×10^{-15}	1.62×10^{-15}	1.65×10^{-15}
	Rank	6	10	3	9	6	4	1	6	2	4
Friedman Mean Rank		4.6087	4.9565	1.4348	5.0435	3.9565	4.7391	4.0000	5.0435	5.6522	4.2174
Final Ranking		5	7	1	8	2	6	3	8	10	4

The best values obtained have been highlighted in **boldface**.

Table 6. Comparison results of CHAOARO and other algorithms on 23 benchmark functions.

F_n	Criteria	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
F_1	Avg	2.48×10^{-103}	1.58×10^{-27}	3.55×10^{-72}	4.81×10^{1}	2.04×10^{-194}	3.60×10^{-54}	4.10×10^{-58}	6.78×10^{-281}	8.21×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	Std	1.36×10^{-102}	3.76×10^{-27}	1.94×10^{-71}	1.54×10^{2}	$\mathbf{0.00 \times 10^{0}}$	9.28×10^{-54}	2.08×10^{-57}	$\mathbf{0.00 \times 10^{0}}$	4.32×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	Rank	4	8	5	10	3	7	6	2	9	1
F_2	Avg	2.33×10^{-54}	9.47×10^{-17}	2.50×10^{-51}	1.71×10^{-2}	1.20×10^{-100}	1.97×10^{-32}	2.43×10^{-32}	5.44×10^{-145}	1.18×10^{6}	$\mathbf{0.00 \times 10^{0}}$
	Std	1.27×10^{-53}	6.62×10^{-17}	8.78×10^{-51}	2.71×10^{-2}	3.13×10^{-100}	2.56×10^{-32}	7.41×10^{-32}	7.88×10^{-145}	5.23×10^{6}	$\mathbf{0.00 \times 10^{0}}$
	Rank	4	8	5	9	3	6	7	2	10	1
F_3	Avg	3.34×10^{-102}	9.76×10^{-6}	3.82×10^{4}	8.88×10^{3}	2.12×10^{-182}	2.19×10^{-17}	8.18×10^{-40}	3.76×10^{-190}	1.88×10^{3}	$\mathbf{0.00 \times 10^{0}}$
	Std	1.83×10^{-101}	1.46×10^{-5}	1.46×10^{4}	5.24×10^{3}	$\mathbf{0.00 \times 10^{0}}$	6.63×10^{-17}	4.39×10^{-39}	$\mathbf{0.00 \times 10^{0}}$	6.16×10^{2}	$\mathbf{0.00 \times 10^{0}}$
	Rank	4	7	10	9	3	6	5	2	8	1
F_4	Avg	1.08×10^{-51}	7.69×10^{-7}	4.74×10^{1}	3.94×10^{1}	8.50×10^{-92}	2.27×10^{-16}	1.32×10^{-23}	3.50×10^{-137}	1.25×10^{1}	$\mathbf{0.00 \times 10^{0}}$
	Std	4.12×10^{-51}	6.64×10^{-7}	3.12×10^{1}	1.27×10^{1}	3.06×10^{-91}	3.94×10^{-16}	6.84×10^{-23}	1.68×10^{-136}	5.92×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	Rank	4	7	10	9	3	6	5	2	8	1
F_5	Avg	9.19×10^{-3}	2.71×10^{1}	2.81×10^{1}	6.27×10^{4}	2.86×10^{1}	2.77×10^{1}	3.79×10^{0}	2.90×10^{1}	1.22×10^{3}	$\mathbf{2.66 \times 10^{-3}}$
	Std	3.46×10^{-3}	8.65×10^{-1}	4.52×10^{-1}	1.63×10^{5}	4.04×10^{-1}	8.07×10^{-1}	8.90×10^{0}	2.05×10^{-3}	1.39×10^{3}	$\mathbf{4.89 \times 10^{-4}}$
	Rank	2	4	6	10	7	5	3	8	9	1
F_6	Avg	1.83×10^{-4}	8.21×10^{-1}	4.18×10^{-1}	1.92×10^{1}	6.30×10^{0}	2.58×10^{0}	1.63×10^{-3}	2.39×10^{0}	8.07×10^{0}	$\mathbf{1.53 \times 10^{-6}}$
	Std	2.56×10^{-4}	4.40×10^{-1}	2.75×10^{-1}	3.49×10^{1}	8.02×10^{-1}	4.71×10^{-1}	7.53×10^{-4}	2.93×10^{-1}	2.86×10^{0}	$\mathbf{2.28 \times 10^{-6}}$
	Rank	2	5	4	10	8	7	3	6	9	1
F_7	Avg	9.23×10^{-5}	2.05×10^{-3}	2.17×10^{-3}	1.03×10^{-1}	$\mathbf{7.55 \times 10^{-5}}$	5.59×10^{-4}	8.47×10^{-4}	1.52×10^{-4}	9.77×10^{-2}	8.52×10^{-5}
	Std	9.80×10^{-5}	1.12×10^{-3}	2.10×10^{-3}	1.09×10^{-1}	$\mathbf{7.24 \times 10^{-5}}$	3.97×10^{-4}	7.51×10^{-4}	1.39×10^{-4}	3.64×10^{-2}	8.29×10^{-5}
	Rank	3	7	8	10	1	5	6	4	9	2
F_8	Avg	-6.45×10^{3}	-5.98×10^{3}	$\mathbf{-1.05 \times 10^{4}}$	-3.70×10^{3}	-3.35×10^{3}	-4.28×10^{3}	-9.05×10^{3}	-2.49×10^{3}	-7.33×10^{3}	-1.03×10^{4}
	Std	2.07×10^{3}	8.50×10^{2}	$\mathbf{3.02 \times 10^{2}}$	1.67×10^{3}	4.30×10^{2}	1.26×10^{3}	8.17×10^{2}	4.40×10^{2}	7.00×10^{2}	4.89×10^{2}
	Rank	5	6	1	8	9	7	3	10	4	2
F_9	Avg	$\mathbf{0.00 \times 10^{0}}$	2.63×10^{0}	7.52×10^{0}	4.41×10^{1}	3.50×10^{1}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	5.34×10^{1}	$\mathbf{0.00 \times 10^{0}}$
	Std	$\mathbf{0.00 \times 10^{0}}$	2.70×10^{0}	4.12×10^{1}	4.34×10^{1}	5.76×10^{1}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	1.33×10^{1}	$\mathbf{0.00 \times 10^{0}}$
	Rank	1	6	7	9	8	1	1	1	10	1
F_{10}	Avg	$\mathbf{8.88 \times 10^{-16}}$	1.03×10^{-13}	4.80×10^{-15}	1.22×10^{1}	4.56×10^{-15}	7.28×10^{-15}	$\mathbf{8.88 \times 10^{-16}}$	4.20×10^{-15}	3.13×10^{0}	$\mathbf{8.88 \times 10^{-16}}$
	Std	$\mathbf{0.00 \times 10^{0}}$	1.73×10^{-14}	3.00×10^{-15}	9.12×10^{0}	6.49×10^{-16}	1.45×10^{-15}	$\mathbf{0.00 \times 10^{0}}$	9.01×10^{-16}	7.69×10^{-1}	$\mathbf{0.00 \times 10^{0}}$
	Rank	1	8	6	10	5	7	1	4	9	1
F_{11}	Avg	$\mathbf{0.00 \times 10^{0}}$	1.95×10^{-3}	6.10×10^{-3}	9.62×10^{-1}	2.60×10^{-3}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	1.06×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	Std	$\mathbf{0.00 \times 10^{0}}$	5.13×10^{-3}	3.34×10^{-2}	5.54×10^{-1}	8.20×10^{-3}	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	$\mathbf{0.00 \times 10^{0}}$	2.53×10^{-2}	$\mathbf{0.00 \times 10^{0}}$
	Rank	1	6	8	9	7	1	1	1	10	1
F_{12}	Avg	1.09×10^{-6}	4.20×10^{-2}	2.12×10^{-2}	2.91×10^{4}	9.82×10^{-1}	2.42×10^{-1}	7.40×10^{-5}	1.68×10^{-1}	5.65×10^{0}	$\mathbf{2.25 \times 10^{-7}}$
	Std	1.01×10^{-6}	2.06×10^{-2}	1.71×10^{-2}	9.95×10^{4}	3.10×10^{-1}	1.04×10^{-1}	5.54×10^{-5}	3.32×10^{-2}	2.78×10^{0}	$\mathbf{3.24 \times 10^{-7}}$
	Rank	2	5	4	10	8	7	3	6	9	1
F_{13}	Avg	8.95×10^{-6}	6.09×10^{-1}	5.18×10^{-1}	8.05×10^{5}	2.53×10^{0}	1.71×10^{0}	5.38×10^{-3}	3.00×10^{0}	4.35×10^{0}	$\mathbf{2.11 \times 10^{-6}}$
	Std	1.43×10^{-5}	2.20×10^{-1}	2.41×10^{-1}	2.96×10^{6}	2.93×10^{-1}	2.18×10^{-1}	9.63×10^{-3}	2.30×10^{-5}	5.74×10^{0}	$\mathbf{3.41 \times 10^{-6}}$
	Rank	2	5	4	10	7	6	3	8	9	1

Table 6. *Cont.*

F_n	Criteria	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
	Avg	5.13×10^{0}	5.33×10^{0}	3.29×10^{0}	2.05×10^{0}	1.17×10^{1}	6.08×10^{0}	$\mathbf{9.98 \times 10^{-1}}$	1.71×10^{0}	2.72×10^{0}	$\mathbf{9.98 \times 10^{-1}}$
F_{14}	Std	5.01×10^{0}	4.56×10^{0}	3.51×10^{0}	1.90×10^{0}	4.96×10^{0}	4.64×10^{0}	4.12×10^{-17}	1.79×10^{0}	1.27×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	Rank	7	8	6	4	10	9	2	3	5	1
	Avg	3.29×10^{-4}	2.44×10^{-3}	6.59×10^{-4}	1.03×10^{-3}	8.27×10^{-3}	3.12×10^{-3}	3.22×10^{-4}	4.11×10^{-2}	1.09×10^{-2}	$\mathbf{3.12 \times 10^{-4}}$
F_{15}	Std	2.60×10^{-5}	6.08×10^{-3}	4.16×10^{-4}	3.67×10^{-4}	1.76×10^{-2}	6.88×10^{-3}	6.45×10^{-5}	3.91×10^{-2}	8.31×10^{-3}	$\mathbf{2.26 \times 10^{-5}}$
	Rank	3	6	4	5	8	7	2	10	9	1
	Avg	−1.0314	**−1.0316**	**−1.0316**	**−1.0316**	−1.0242	**−1.0316**	**−1.0316**	−1.0031	−0.9228	**−1.0316**
F_{16}	Std	1.07×10^{-3}	2.12×10^{-8}	2.70×10^{-10}	3.71×10^{-5}	1.36×10^{-2}	2.61×10^{-7}	5.68×10^{-16}	7.48×10^{-3}	2.82×10^{-1}	$\mathbf{5.61 \times 10^{-16}}$
	Rank	7	4	3	6	8	5	2	9	10	1
	Avg	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	4.01×10^{-1}	4.00×10^{-1}	$\mathbf{3.98 \times 10^{-1}}$	$\mathbf{3.98 \times 10^{-1}}$	1.19×10^{0}	1.00×10^{0}	$\mathbf{3.98 \times 10^{-1}}$
F_{17}	Std	2.96×10^{-5}	7.64×10^{-7}	1.50×10^{-5}	3.25×10^{-3}	2.57×10^{-3}	8.90×10^{-5}	$\mathbf{0.00 \times 10^{0}}$	8.70×10^{-1}	7.97×10^{-1}	$\mathbf{0.00 \times 10^{0}}$
	Rank	5	3	4	8	7	6	1	10	9	1
	Avg	1.66×10^{1}	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{3.00 \times 10^{0}}$	1.05×10^{1}	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{3.00 \times 10^{0}}$	$\mathbf{3.00 \times 10^{0}}$	3.00×10^{0}
F_{18}	Std	1.38×10^{1}	5.16×10^{-5}	1.59×10^{-4}	1.12×10^{-4}	2.22×10^{1}	6.59×10^{-6}	1.47×10^{-15}	1.21×10^{-4}	3.18×10^{-4}	$\mathbf{1.28 \times 10^{-15}}$
	Rank	10	4	7	5	9	3	2	6	8	1
	Avg	−3.8384	−3.8611	−3.8563	−3.8546	−3.8595	−3.8606	**−3.8628**	−3.4620	−3.8112	**−3.8628**
F_{19}	Std	3.90×10^{-2}	2.64×10^{-3}	1.08×10^{-2}	3.41×10^{-3}	2.60×10^{-3}	3.49×10^{-3}	3.65×10^{-15}	3.35×10^{-1}	1.96×10^{-1}	$\mathbf{2.61 \times 10^{-15}}$
	Rank	8	3	6	7	5	4	2	10	9	1
	Avg	−3.2582	−3.2697	−3.2483	−2.9609	−3.1590	−3.1552	−3.2744	−1.7121	−3.2744	**−3.2824**
F_{20}	Std	7.46×10^{-2}	7.35×10^{-2}	9.16×10^{-2}	3.12×10^{-1}	1.66×10^{-1}	1.12×10^{-1}	5.92×10^{-2}	4.60×10^{-1}	5.92×10^{-2}	$\mathbf{5.70 \times 10^{-2}}$
	Rank	5	4	6	9	7	8	2	10	2	1
	Avg	−10.1521	−8.8856	−8.2071	−2.8999	−7.1024	−8.7078	−9.6483	−0.9418	−5.8722	**−10.1532**
F_{21}	Std	1.11×10^{-3}	2.37×10^{0}	2.71×10^{0}	1.85×10^{0}	1.65×10^{0}	2.45×10^{0}	1.91×10^{0}	3.53×10^{-1}	3.02×10^{0}	$\mathbf{9.43 \times 10^{-5}}$
	Rank	2	4	6	9	7	5	3	10	8	1
	Avg	−10.4012	−10.4012	−7.2731	−2.9022	−6.5091	−9.2546	−9.8260	−1.3450	−5.8760	**−10.4029**
F_{22}	Std	2.14×10^{-3}	1.15×10^{-3}	3.28×10^{0}	1.75×10^{0}	2.06×10^{0}	2.34×10^{0}	1.77×10^{0}	7.23×10^{-1}	3.38×10^{0}	$\mathbf{4.66 \times 10^{-16}}$
	Rank	3	2	6	9	7	5	4	10	8	1
	Avg	−10.5348	−10.5346	−6.5171	−3.6233	−6.1982	−9.4430	−10.3130	−1.2424	−5.4826	**−10.5364**
F_{23}	Std	2.21×10^{-3}	1.02×10^{-3}	2.99×10^{0}	1.44×10^{0}	2.68×10^{0}	2.51×10^{0}	1.22×10^{0}	$4.17 \times 10{-}^{1}$	3.73×10^{0}	$\mathbf{1.84 \times 10^{-15}}$
	Rank	2	3	6	9	7	5	4	10	8	1
Friedman Mean Rank		3.7826	5.3478	5.7391	8.4348	6.3913	5.5652	3.0870	6.2609	8.2174	**1.0870**
Final Ranking		3	4	6	9	8	5	2	7	10	**1**

The best values obtained have been highlighted in **boldface**.

As seen from Table 6, for unimodal functions F_1~F_7, as far as the average fitness is concerned, CHAOARO is able to precisely pinpoint the theoretical optimal solution (0) on F_1~F_4, whereas the obtained results of other comparison algorithms are not satisfactory. Meanwhile, it is noteworthy that the solution accuracy of the hybrid algorithm has a significant improvement over the basic AO and ARO. On functions F_5 and F_6, although CHAOARO fails to search for the global optimum, it still achieves the best average fitness value in the competition with the remaining nine algorithms. Regarding function F_7, the proposed CHAOARO also provides very competitive average fitness, which is second only to the best TSA. Additionally, CHAOARO achieves the smallest standard deviation among all algorithms on functions F_1~F_6, but marginally inferior to TSA on function F_7. Since the unimodal function has only one global optimal value, these experimental data indicate that CHAOARO has strong local exploitation potential. This is because the COBL strategy can effectively expand the unknown search domain and the hybrid operation facilitates the exchange of useful information among individuals in the population.

With regard to solving multimodal and fix-dimension multimodal functions F_8~F_{23}, CHAOARO maintains good convergence accuracy and robustness, which can reveal the best results on 15 out of 16 benchmark functions both in terms of average fitness and standard deviation. Especially on functions F_{12}, F_{13}, F_{15}, F_{20}, F_{21}, F_{22}, and F_{23}, CHAOARO has an overwhelming advantage compared with the basic AO and ARO, as well as other optimization methods. On functions F_{16}~F_{19}, some algorithms achieve the same average fitness as CHAOARO. However, the calculated standard deviation value of the proposed algorithm is the smallest among them. This highlights the superior stability of CHAOARO. Considering that the multimodal and fix-dimension multimodal functions are characterized by numerous local minima, these results prove the excellent exploration and local optima avoidance capabilities of CHAOARO. It can be explained that CHAOARO takes full advantage of the powerful exploration trend of AO, and the ASM mechanism can better balance the algorithm exploration and exploitation.

Moreover, the final Friedman mean ranking of CHAOARO obtained on 23 classical benchmark functions is 1.0870, which ranks first among these algorithms. Hence, it can be believed that the searchability of the hybrid algorithm proposed in this paper has been significantly enhanced.

4.1.3. Analysis of Convergence Behavior

To study the convergence behavior of the algorithm throughout the process of finding the global optimal solution, Figure 7 illustrates the convergence curves of AO, GWO, WOA, TSA, GJO, ARO, WChOA, DAOA, and CHAOARO on 23 classical benchmark functions. From Figure 7, it can be observed that the proposed CHAOARO can effectively reach the global optimal solution at the initial stage of the unimodal benchmark functions F_1~F_4 and shows the fastest convergence speed, while the original AO, ARO, and other comparison algorithms converge slowly with unsatisfactory convergence accuracy. On functions F_5 and F_6, CHAOARO has a similar convergence trend to AO in the early search process, but gradually AO falls into the local optima, while the proposed algorithm still maximizes the information in the search space to improve the quality of the solution. Finally, CHAOARO provides the best convergence accuracy among all these optimization methods. On function F_7, CHAOARO also gains a good ameliorate in the field of convergence accuracy and speed compared with AO and ARO. For multimodal functions F_8~F_{13}, the proposed CHAOARO continues its outstanding search performance. Specifically, on functions F_9 and F_{11}, CHAOARO is able to obtain the theoretical optimum with the minimum number of iterations among all algorithms. On functions F_8, F_{10}, F_{12}, and F_{13}, the convergence accuracy and speed of CHAOARO once again outperform the other competitors in varying degrees. For fix-dimension multimodal functions F_{14}~F_{23}, CHAOARO can quickly transfer from exploration to exploitation, converge to the global optimal value in the early searching stage, and not fall into the local optimum. Compared with its peers, CHAOARO possesses a significant superiority in terms of convergence accuracy and speed.

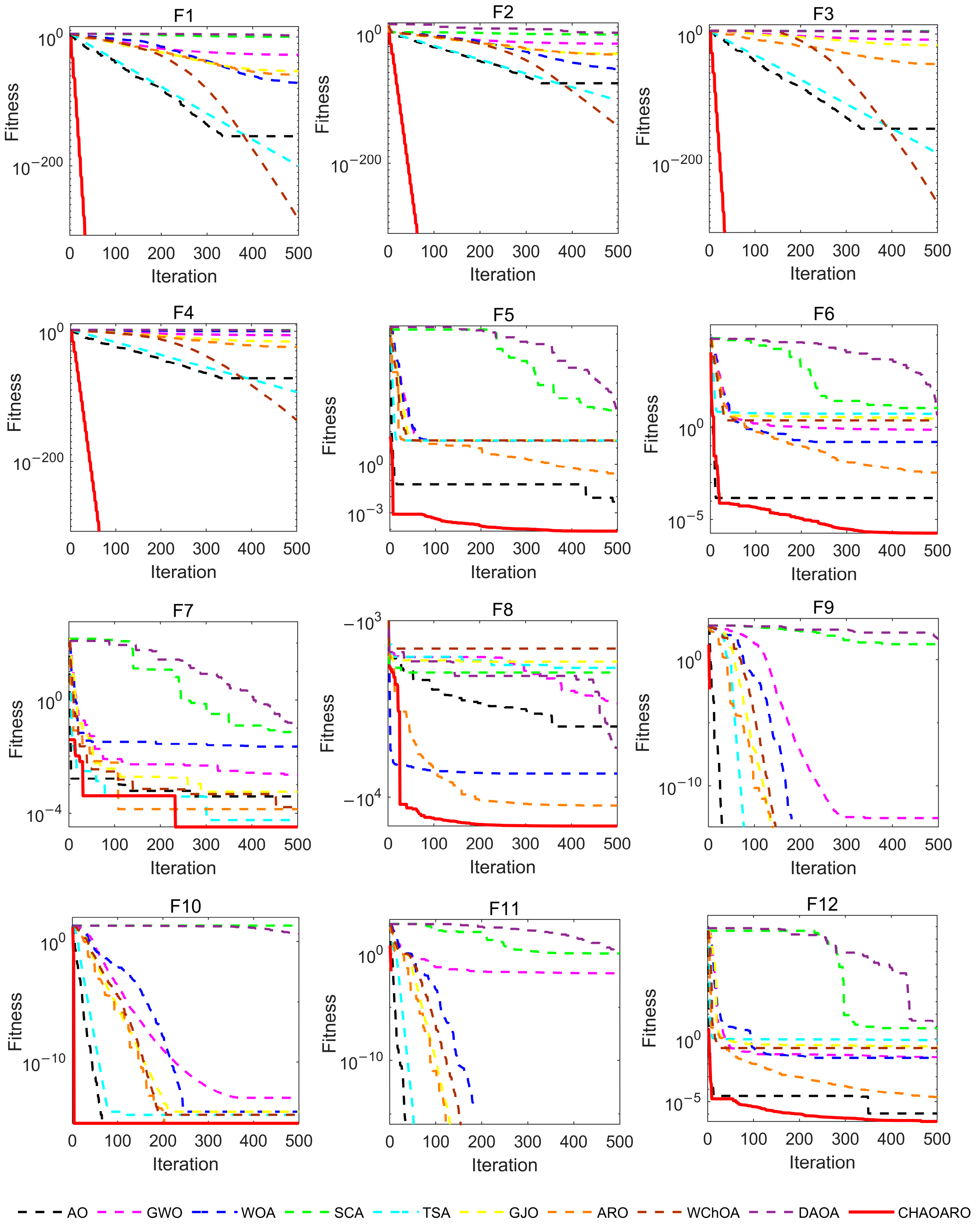

Figure 7. *Cont.*

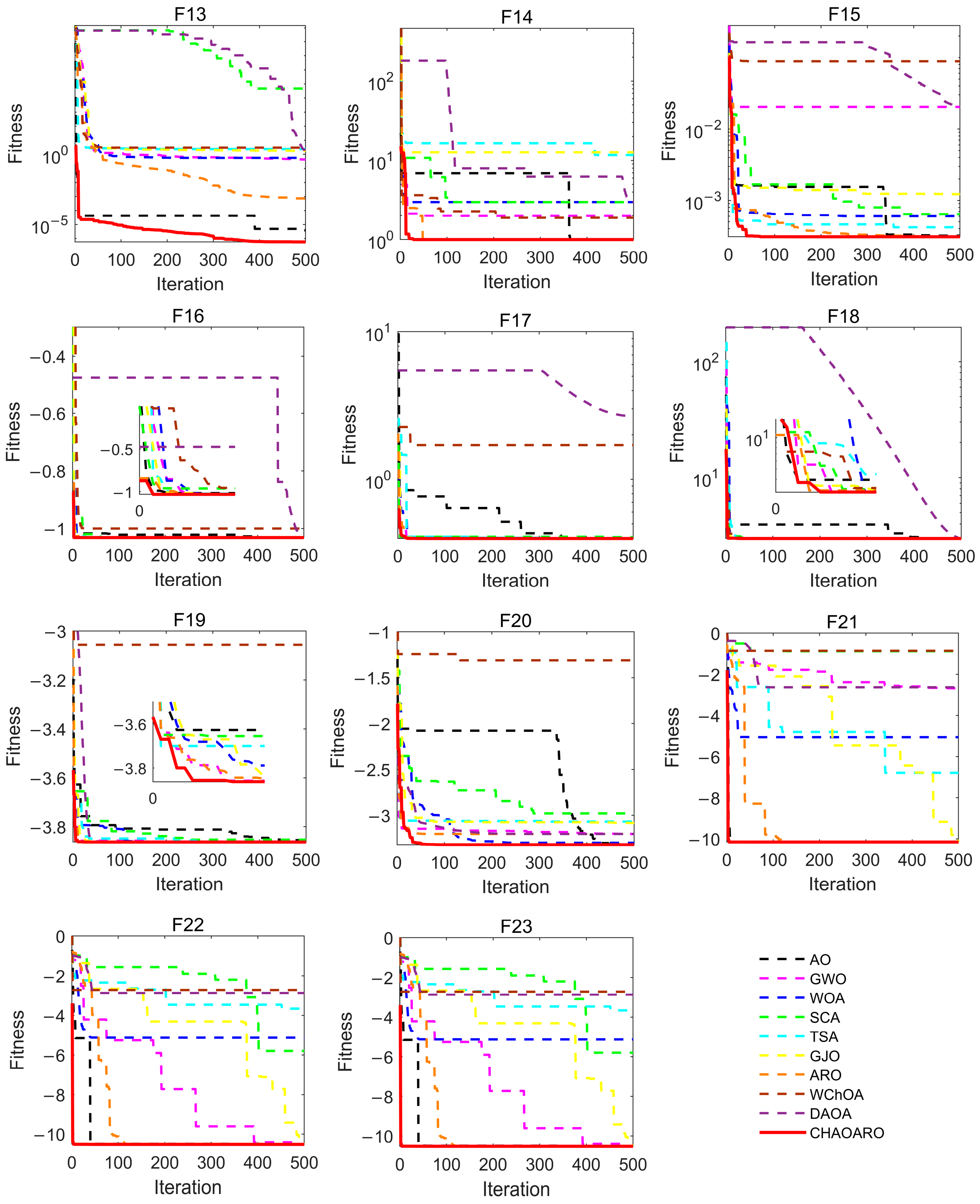

Figure 7. Convergence curves of CHAOARO and other comparison algorithms on 23 benchmark functions.

To sum up, the convergence pattern of the proposed CHAOARO is obviously improved, and it can rapidly and precisely locate an excellent solution for both unimodal and multimodal functions.

4.1.4. Boxplot Analysis

To better describe the consistency between the data obtained from 30 independent runs, in this subsection, the boxplot diagram is utilized to reflect the distribution characteristics of each algorithm. Figure 8 depicts the boxplots of AO, GWO, WOA, SCA, TSA, GJO, ARO, WChOA, DAOA, and CHAOARO on 12 representative benchmark functions. In this diagram, the center mark of each box signifies the median obtained, the lowest and largest points on the edges are the minimum and maximum values, respectively, and the symbol "+" denotes an outlier. From Figure 9, we can notice that the objective distribution of CHAOARO is narrower than that of comparison algorithms in most cases, which suggests that the proposed algorithm has excellent robustness in solving these test problems. In particular, on functions F_1, F_2, F_3, F_4, F_9, F_{10}, F_{11}, F_{14}, F_{16}, and F_{17}, CHAOARO does not produce any outliers. On the remaining functions, although there exist individual outliers, the general distribution of CHAOARO regarding the median, minimum, and maximum values is likewise more concentrated than others. Experimental findings demonstrate that the stability of CHAOARO is considerably improved compared to its predecessors AO and ARO, which benefits largely from the key ASM and COBL strategies.

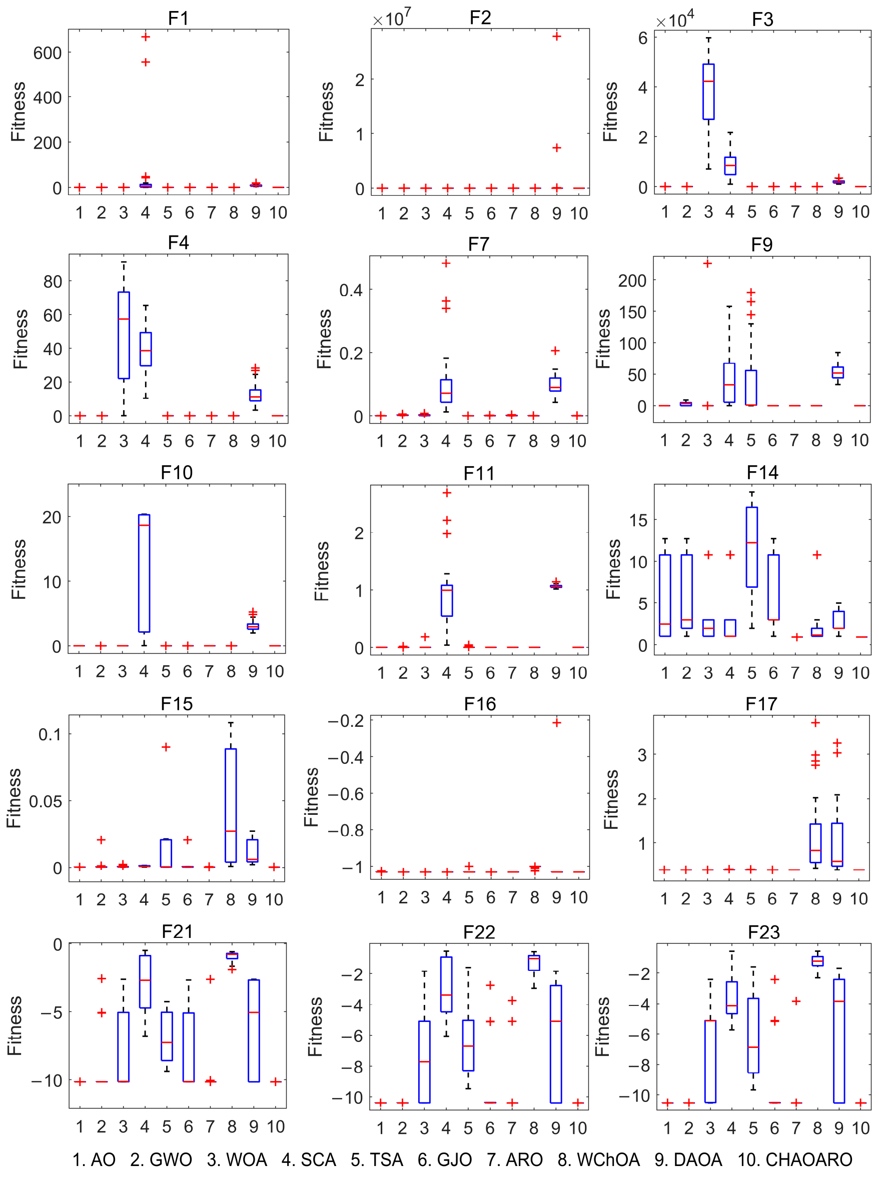

Figure 8. Boxplots of CHAOARO and other comparison algorithms on some benchmark functions.

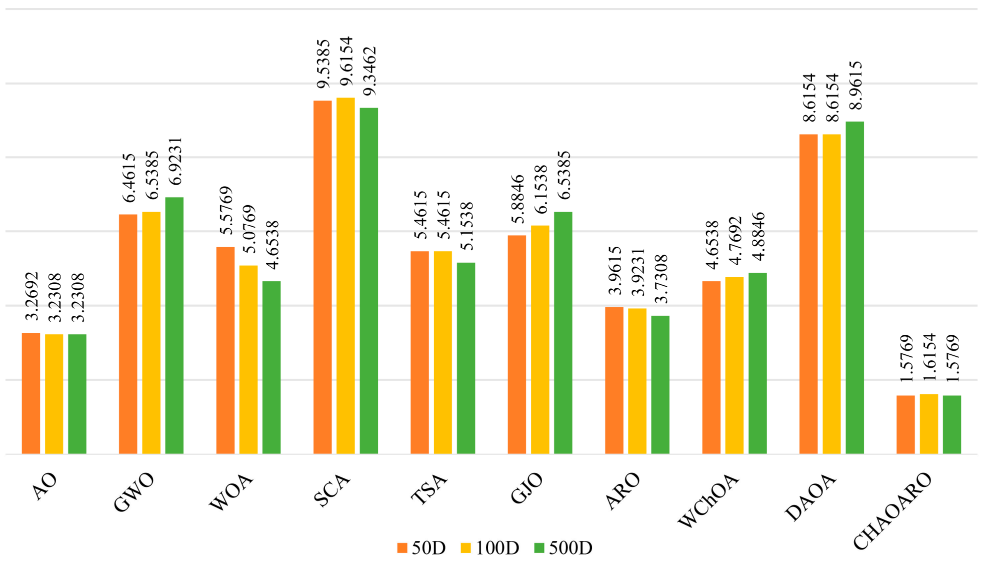

Figure 9. Friedman mean ranking of CHAOARO and its peers in different dimensions.

4.1.5. Wilcoxon Rank-Sum Test

The Wilcoxon rank-sum test is a non-parametric statistical method used to assess the performance difference between two samples at a significance level of 0.05 [68]. To be specific, if the p-value is less than 0.05, it means that there is a significant difference between CHAOARO and the comparison algorithm, i.e., CHAOARO performs better than its opponent ("+"). In contrast, when the p-value is greater than 0.05, it indicates that the difference between CHAOARO and the comparison algorithm is not apparent, i.e., CHAOARO performs worse than its opponent ("−"). In addition, NaN represents that CHAOARO and the comparison algorithm have consistent performance in terms of statistics ("="). Table 7 outlines the p-values between CHAOARO and each comparison optimizer obtained via Wilcoxon rank-sum test on 23 benchmark functions. As can be seen from this table, the proposed method is significantly superior to AO on 20 functions, GWO on 23 functions, WOA on 22 functions, SCA on 23 functions, TSA on 23 functions, GJO on 21 functions, ARO on 16 functions, WChOA on 21 functions, and DAOA on 23 functions. These statistical results provide evidence that CHAOARO shows better significant optimization performance on almost all test functions compared with other comparison algorithms.

4.1.6. Computation Time Analysis

To quantitatively analyze the computational cost of the proposed algorithm, the average runtime of ten algorithms on 23 benchmark functions is reported in Table 8. We calculate the total operation time for each algorithm and give the corresponding rankings as follows: WChOA (27.9750 s) > AO (6.2460 s) > GJO (5.4630 s) > CHAOARO (4.3330 s) > ARO (4.2020 s) > GWO (4.0796 s) > TSA (3.5097 s) > SCA (3.3658 s) > DAOA (2.7242 s) > WOA (2.3238 s). From the results, it can be seen that the time consumption of CHAOARO is slightly higher than that of ARO but significantly lower than that of AO. Although the algorithm designed in this paper has a more complex framework compared with the basic AO and ARO algorithms, mainly due to the hybrid operation, ASM, and COBL mechanisms, its optimization performance is greatly improved. On the other hand, using the added steps to obtain more reliable solutions does not result in too much time cost required to be sacrificed by CHAOARO. On the whole, given that CHAOARO has better exploration and exploitation capabilities than other comparison algorithms, a little computation time is acceptable. The proposed method is expected to be successfully adopted in some real-time applications.

Table 7. Statistical results of Wilcoxon rank-sum test for different algorithms on 23 benchmark functions.

F_n	**CHAOARO VS.**								
	AO	**GWO**	**WOA**	**SCA**	**TSA**	**GJO**	**ARO**	**WChOA**	**DAOA**
F_1	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_2	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_3	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_4	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
F_5	2.12×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.41×10^{-9}	2.95×10^{-11}	3.02×10^{-11}
F_6	2.15×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F_7	2.92×10^{-2}	3.02×10^{-11}	2.92×10^{-9}	3.02×10^{-11}	5.83×10^{-3}	7.04×10^{-7}	1.47×10^{-7}	8.77×10^{-3}	3.02×10^{-11}
F_8	1.07×10^{-7}	3.02×10^{-11}	3.18×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	7.74×10^{-6}	3.02×10^{-11}	4.18×10^{-9}
F_9	NaN	1.17×10^{-12}	$\mathbf{1.61 \times 10^{-1}}$	1.21×10^{-12}	6.25×10^{-10}	NaN	NaN	NaN	1.21×10^{-12}
F_{10}	NaN	1.04×10^{-12}	3.76×10^{-8}	1.21×10^{-12}	2.71×10^{-14}	1.55×10^{-13}	NaN	7.15×10^{-13}	1.21×10^{-12}
F_{11}	NaN	2.16×10^{-2}	3.34×10^{-3}	1.21×10^{-12}	4.19×10^{-2}	NaN	NaN	NaN	1.21×10^{-12}
F_{12}	2.32×10^{-6}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F_{13}	8.68×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.86×10^{-11}	3.02×10^{-11}
F_{14}	1.72×10^{-12}	1.72×10^{-12}	1.72×10^{-12}	1.72×10^{-12}	1.72×10^{-12}	1.72×10^{-12}	3.34×10^{-5}	1.72×10^{-12}	1.72×10^{-12}
F_{15}	3.50×10^{-9}	9.76×10^{-10}	9.92×10^{-11}	3.34×10^{-11}	9.92×10^{-11}	3.82×10^{-10}	3.67×10^{-3}	3.02×10^{-11}	3.02×10^{-11}
F_{16}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}	1.45×10^{-11}	$\mathbf{8.04 \times 10^{-2}}$	1.45×10^{-11}	1.45×10^{-11}
F_{17}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	NaN	1.21×10^{-12}	1.21×10^{-12}
F_{18}	6.47×10^{-12}	6.47×10^{-12}	6.47×10^{-12}	6.47×10^{-12}	6.47×10^{-12}	6.47×10^{-12}	$\mathbf{2.23 \times 10^{-1}}$	6.47×10^{-12}	6.47×10^{-12}
F_{19}	7.57×10^{-12}	7.57×10^{-12}	7.57×10^{-12}	7.57×10^{-12}	7.57×10^{-12}	7.57×10^{-12}	$\mathbf{3.26 \times 10^{-1}}$	7.57×10^{-12}	7.57×10^{-12}
F_{20}	3.99×10^{-4}	1.25×10^{-4}	1.25×10^{-4}	3.02×10^{-11}	1.11×10^{-6}	4.68×10^{-8}	1.78×10^{-5}	3.02×10^{-11}	6.76×10^{-5}
F_{21}	1.17×10^{-11}	3.16×10^{-12}	3.16×10^{-12}	3.16×10^{-12}	3.16×10^{-12}	3.16×10^{-12}	4.22×10^{-10}	3.16×10^{-12}	8.44×10^{-12}
F_{22}	2.36×10^{-12}	2.36×10^{-12}	2.36×10^{-12}	2.36×10^{-12}	2.36×10^{-12}	2.36×10^{-12}	1.89×10^{-8}	2.36×10^{-12}	2.36×10^{-12}
F_{23}	9.04×10^{-12}	9.04×10^{-12}	9.04×10^{-12}	9.04×10^{-12}	9.04×10^{-12}	9.04×10^{-12}	1.04×10^{-5}	9.04×10^{-12}	9.04×10^{-12}
+/=/−	20/3/0	23/0/0	22/0/1	23/0/0	23/0/0	21/2/0	16/4/3	21/2/0	23/0/0

The *p*-value obtained greater than 0.05 have been highlighted in **boldface**.

Table 8. Average computation time of CHAOARO and other algorithms on all benchmark functions (unit: s).

F_n	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
F_1	2.08×10^{-1}	1.91×10^{-1}	$\mathbf{5.68 \times 10^{-2}}$	1.30×10^{-1}	1.51×10^{-1}	2.72×10^{-1}	1.40×10^{-1}	1.86×10^{0}	9.08×10^{-2}	1.72×10^{-1}
F_2	1.82×10^{-1}	1.74×10^{-1}	$\mathbf{5.39 \times 10^{-2}}$	1.28×10^{-1}	1.46×10^{-1}	2.31×10^{-1}	1.53×10^{-1}	1.88×10^{0}	8.88×10^{-2}	1.43×10^{-1}
F_3	4.79×10^{-1}	3.11×10^{-1}	$\mathbf{1.94 \times 10^{-1}}$	2.60×10^{-1}	2.76×10^{-1}	3.60×10^{-1}	2.79×10^{-1}	1.96×10^{0}	2.20×10^{-1}	2.82×10^{-1}
F_4	1.82×10^{-1}	1.80×10^{-1}	$\mathbf{5.10 \times 10^{-2}}$	1.40×10^{-1}	1.48×10^{-1}	2.35×10^{-1}	1.38×10^{-1}	1.93×10^{0}	8.28×10^{-2}	1.44×10^{-1}
F_5	2.15×10^{-1}	2.12×10^{-1}	$\mathbf{7.87 \times 10^{-2}}$	1.54×10^{-1}	1.70×10^{-1}	2.54×10^{-1}	1.47×10^{-1}	1.92×10^{0}	1.00×10^{-1}	1.56×10^{-1}
F_6	1.83×10^{-1}	1.98×10^{-1}	$\mathbf{5.66 \times 10^{-2}}$	1.37×10^{-1}	1.57×10^{-1}	2.50×10^{-1}	1.39×10^{-1}	1.82×10^{0}	8.77×10^{-2}	1.35×10^{-1}
F_7	3.07×10^{-1}	2.55×10^{-1}	$\mathbf{1.30 \times 10^{-1}}$	2.03×10^{-1}	1.94×10^{-1}	3.11×10^{-1}	2.15×10^{-1}	1.87×10^{0}	1.47×10^{-1}	2.16×10^{-1}
F_8	2.13×10^{-1}	1.87×10^{-1}	$\mathbf{6.55 \times 10^{-2}}$	1.48×10^{-1}	1.65×10^{-1}	2.46×10^{-1}	1.57×10^{-1}	1.85×10^{0}	1.06×10^{-1}	1.62×10^{-1}
F_9	1.93×10^{-1}	2.00×10^{-1}	$\mathbf{6.20 \times 10^{-2}}$	1.34×10^{-1}	1.40×10^{-1}	2.27×10^{-1}	1.24×10^{-1}	1.87×10^{0}	9.87×10^{-2}	1.43×10^{-1}
F_{10}	1.90×10^{-1}	1.89×10^{-1}	$\mathbf{6.13 \times 10^{-2}}$	1.39×10^{-1}	1.38×10^{-1}	2.28×10^{-1}	1.39×10^{-1}	1.81×10^{0}	9.58×10^{-2}	1.36×10^{-1}
F_{11}	2.26×10^{-1}	2.02×10^{-1}	$\mathbf{7.32 \times 10^{-2}}$	1.60×10^{-1}	1.65×10^{-1}	2.45×10^{-1}	1.51×10^{-1}	1.90×10^{0}	1.15×10^{-1}	1.59×10^{-1}
F_{12}	5.45×10^{-1}	3.44×10^{-1}	$\mathbf{2.26 \times 10^{-1}}$	2.99×10^{-1}	3.18×10^{-1}	4.01×10^{-1}	3.28×10^{-1}	1.96×10^{0}	2.83×10^{-1}	3.41×10^{-1}
F_{13}	5.35×10^{-1}	3.42×10^{-1}	$\mathbf{2.35 \times 10^{-1}}$	3.05×10^{-1}	2.98×10^{-1}	3.75×10^{-1}	3.14×10^{-1}	1.93×10^{0}	2.67×10^{-1}	3.33×10^{-1}
F_{14}	9.38×10^{-1}	4.27×10^{-1}	4.27×10^{-1}	$\mathbf{4.22 \times 10^{-1}}$	4.37×10^{-1}	5.09×10^{-1}	5.08×10^{-1}	5.35×10^{-1}	4.28×10^{-1}	5.35×10^{-1}
F_{15}	1.63×10^{-1}	6.73×10^{-2}	5.17×10^{-2}	6.26×10^{-2}	6.15×10^{-2}	1.43×10^{-1}	1.34×10^{-1}	3.42×10^{-1}	$\mathbf{4.93 \times 10^{-2}}$	1.29×10^{-1}
F_{16}	1.60×10^{-1}	5.64×10^{-2}	5.17×10^{-2}	5.36×10^{-2}	5.51×10^{-2}	1.43×10^{-1}	1.30×10^{-1}	2.39×10^{-1}	$\mathbf{4.60 \times 10^{-2}}$	1.33×10^{-1}
F_{17}	1.53×10^{-1}	5.09×10^{-2}	5.12×10^{-2}	4.69×10^{-2}	4.88×10^{-2}	1.23×10^{-1}	1.27×10^{-1}	2.26×10^{-1}	$\mathbf{4.61 \times 10^{-2}}$	1.34×10^{-1}
F_{18}	1.54×10^{-1}	5.07×10^{-2}	$\mathbf{4.02 \times 10^{-2}}$	4.47×10^{-2}	4.44×10^{-2}	1.27×10^{-1}	1.30×10^{-1}	2.23×10^{-1}	4.18×10^{-2}	1.33×10^{-1}
F_{19}	1.81×10^{-1}	7.72×10^{-2}	6.50×10^{-2}	6.95×10^{-2}	6.60×10^{-2}	1.45×10^{-1}	1.39×10^{-1}	2.84×10^{-1}	$\mathbf{5.41 \times 10^{-2}}$	1.39×10^{-1}
F_{20}	1.69×10^{-1}	8.13×10^{-2}	5.37×10^{-2}	7.06×10^{-2}	7.32×10^{-2}	1.55×10^{-1}	1.47×10^{-1}	4.74×10^{-1}	$\mathbf{5.24 \times 10^{-2}}$	1.37×10^{-1}
F_{21}	2.08×10^{-1}	7.92×10^{-2}	$\mathbf{6.33 \times 10^{-2}}$	7.74×10^{-2}	7.97×10^{-2}	1.55×10^{-1}	1.52×10^{-1}	3.79×10^{-1}	6.93×10^{-2}	1.52×10^{-1}
F_{22}	2.19×10^{-1}	9.86×10^{-2}	8.30×10^{-2}	8.25×10^{-2}	8.29×10^{-2}	1.56×10^{-1}	1.46×10^{-1}	3.42×10^{-1}	$\mathbf{7.09 \times 10^{-2}}$	1.54×10^{-1}
F_{23}	2.43×10^{-1}	1.06×10^{-1}	9.30×10^{-2}	9.90×10^{-2}	9.51×10^{-2}	1.72×10^{-1}	1.65×10^{-1}	3.71×10^{-1}	$\mathbf{8.37 \times 10^{-2}}$	1.65×10^{-1}
Total runtime	6.2460	4.0796	2.3238	3.3658	3.5097	5.4630	4.2020	27.9750	2.7242	4.3330

The best values obtained have been highlighted in **boldface**.

4.1.7. Scalability Analysis

The scalability test can be used to investigate the impact of problems in different dimensions on the optimization performance of the algorithm. To check whether the proposed algorithm suffers from dimension disaster while tackling high-dimensional optimization problems, this subsection will apply CHAOARO to optimize the 13 variable-dimension benchmark functions F_1~F_{13} in Table 4. We increase the test dimensions (D) from 30 to 50, 100, and 500, and the average fitness and standard deviation results obtained from 30 independent runs of each algorithm are filled in Table 9.

As can be found in Table 9, CHAOARO also exhibits superior search capabilities than comparison algorithms on high-dimensional problems. With the expansion of dimensionality, more elements need to be optimized, so the convergence accuracy of most algorithms decreases to some extent, but CHAOARO is able to steadily find the theoretical optimum (0) on functions F_1, F_2, F_3, F_4, F_9, and F_{11} in all dimensions. For functions F_5, F_6, F_{12}, and F_{13}, the Avg and Std of the original AO and ARO gradually deteriorate with the increase of dimension; nevertheless, CHAOARO maintains high solution accuracy. For functions F_7 and F_8, the performance of CHAOARO is slightly inferior to that of TSA and WOA, respectively, ranking second among all algorithms. Figure 9 illustrates the Friedman mean rankings of CHAOARO and the other nine comparison algorithms on these scalable functions. From this figure, it can be seen that CHAOARO has the best overall performance among all algorithms regardless of dimensionality.

Based on the above, it can be concluded that when resolving high-dimensional problems, CHAOARO can maintain well exploration and exploitation trends at the same time.

4.2. Experiment 2: IEEE CEC2019 Test Suite

The above series of comparison experiments based on classical benchmark functions have successfully witnessed the superiority of CHAOARO in solving simple optimization problems. To further validate the effectiveness of our improved algorithm in addressing complex optimization problems, in this section, we will evaluate the performance of CHAOARO by using the IEEE CEC2019 test suite. Table 10 provides the details of the ten benchmark functions in the IEEE CEC2019 test suite. Likewise, CHAOARO and the other nine algorithms run independently 30 times on each function with the maximum iteration and population size set as 500 and 30, respectively. And the experimental results are shown in Table 11.

Table 9. Comparison results of CHAOARO and other algorithms on 13 unimodal and multimodal benchmark functions in different dimensions.

F_n	Dimension	Criteria	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
F_1	50	Avg	4.99×10^{-110}	4.10×10^{-20}	2.25×10^{-73}	8.37×10^{2}	6.82×10^{-186}	3.19×10^{-40}	4.35×10^{-55}	7.27×10^{-272}	5.52×10^{1}	$\mathbf{0.00 \times 10^{0}}$
		Std	2.19×10^{-109}	3.91×10^{-20}	1.22×10^{-72}	9.04×10^{2}	$\mathbf{0.00 \times 10^{0}}$	6.97×10^{-40}	1.60×10^{-54}	$\mathbf{0.00 \times 10^{0}}$	1.55×10^{1}	$\mathbf{0.00 \times 10^{0}}$
	100	Avg	9.85×10^{-102}	1.31×10^{-12}	5.71×10^{-73}	1.06×10^{4}	1.39×10^{-176}	1.51×10^{-27}	2.53×10^{-52}	1.97×10^{-262}	9.03×10^{2}	$\mathbf{0.00 \times 10^{0}}$
		Std	4.63×10^{-101}	9.19×10^{-13}	1.87×10^{-72}	7.41×10^{3}	$\mathbf{0.00 \times 10^{0}}$	3.13×10^{-27}	9.66×10^{-52}	$\mathbf{0.00 \times 10^{0}}$	2.17×10^{2}	$\mathbf{0.00 \times 10^{0}}$
	500	Avg	9.79×10^{-99}	1.54×10^{-3}	4.52×10^{-68}	1.93×10^{5}	1.02×10^{-162}	8.15×10^{-13}	3.13×10^{-49}	3.35×10^{-249}	2.86×10^{5}	$\mathbf{0.00 \times 10^{0}}$
		Std	4.02×10^{-98}	5.27×10^{-4}	2.45×10^{-67}	7.20×10^{4}	$\mathbf{0.00 \times 10^{0}}$	1.25×10^{-12}	1.69×10^{-48}	$\mathbf{0.00 \times 10^{0}}$	2.11×10^{4}	$\mathbf{0.00 \times 10^{0}}$
F_2	50	Avg	9.78×10^{-57}	2.25×10^{-12}	1.95×10^{-49}	6.00×10^{-1}	1.19×10^{-95}	8.35×10^{-25}	7.24×10^{-32}	2.17×10^{-139}	5.00×10^{14}	$\mathbf{0.00 \times 10^{0}}$
		Std	4.50×10^{-56}	1.31×10^{-12}	1.07×10^{-48}	7.89×10^{-1}	3.80×10^{-95}	1.58×10^{-24}	2.45×10^{-31}	3.26×10^{-139}	2.66×10^{15}	$\mathbf{0.00 \times 10^{0}}$
	100	Avg	1.06×10^{-61}	4.01×10^{-8}	2.83×10^{-49}	6.44×10^{0}	3.34×10^{-91}	1.33×10^{-17}	9.08×10^{-31}	5.58×10^{-134}	4.99×10^{40}	$\mathbf{0.00 \times 10^{0}}$
		Std	5.83×10^{-61}	1.31×10^{-8}	9.54×10^{-49}	5.09×10^{0}	4.03×10^{-91}	8.41×10^{-18}	1.89×10^{-30}	8.48×10^{-134}	2.73×10^{41}	$\mathbf{0.00 \times 10^{0}}$
	500	Avg	1.58×10^{-56}	1.06×10^{-2}	1.68×10^{-49}	1.01×10^{2}	5.50×10^{-83}	5.72×10^{-9}	2.22×10^{-29}	5.75×10^{-127}	6.86×10^{246}	$\mathbf{0.00 \times 10^{0}}$
		Std	8.64×10^{-56}	1.67×10^{-3}	6.06×10^{-49}	6.08×10^{1}	2.13×10^{-83}	2.66×10^{-9}	5.21×10^{-29}	4.62×10^{-127}	Inf	$\mathbf{0.00 \times 10^{0}}$
F_3	50	Avg	1.93×10^{-100}	3.79×10^{-1}	1.96×10^{5}	5.34×10^{4}	6.24×10^{-175}	3.04×10^{-9}	2.46×10^{-37}	2.57×10^{-178}	1.73×10^{4}	$\mathbf{0.00 \times 10^{0}}$
		Std	1.02×10^{-99}	9.20×10^{-1}	3.78×10^{4}	1.72×10^{4}	0.00×10^{0}	1.18×10^{-8}	1.35×10^{-36}	0.00×10^{0}	4.30×10^{3}	$\mathbf{0.00 \times 10^{0}}$
	100	Avg	4.93×10^{-98}	7.18×10^{2}	1.19×10^{6}	2.36×10^{5}	1.42×10^{-166}	2.16×10^{1}	1.45×10^{-34}	1.12×10^{-34}	1.07×10^{5}	$\mathbf{0.00 \times 10^{0}}$
		Std	1.75×10^{-97}	5.34×10^{2}	3.34×10^{5}	6.07×10^{4}	0.00×10^{0}	1.18×10^{2}	7.94×10^{-34}	6.16×10^{-34}	9.99×10^{3}	$\mathbf{0.00 \times 10^{0}}$
	500	Avg	4.70×10^{-100}	3.29×10^{5}	2.84×10^{7}	6.83×10^{6}	1.59×10^{-154}	6.14×10^{4}	2.29×10^{-31}	3.03×10^{3}	2.87×10^{6}	$\mathbf{0.00 \times 10^{0}}$
		Std	2.24×10^{-99}	7.04×10^{4}	9.61×10^{6}	1.56×10^{6}	3.07×10^{-154}	6.75×10^{4}	8.60×10^{-31}	1.42×10^{4}	3.62×10^{5}	$\mathbf{0.00 \times 10^{0}}$
F_4	50	Avg	2.24×10^{-51}	6.27×10^{-4}	6.94×10^{1}	6.72×10^{1}	2.52×10^{-88}	6.93×10^{-7}	4.71×10^{-23}	5.16×10^{-126}	4.56×10^{1}	$\mathbf{0.00 \times 10^{0}}$
		Std	1.22×10^{-50}	7.18×10^{-4}	2.75×10^{1}	6.48×10^{0}	5.03×10^{-88}	2.77×10^{-6}	1.33×10^{-22}	2.83×10^{-125}	8.64×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	100	Avg	1.61×10^{-51}	8.37×10^{-1}	7.93×10^{1}	8.95×10^{1}	2.36×10^{-84}	4.16×10^{0}	1.51×10^{-21}	5.43×10^{-96}	7.35×10^{1}	$\mathbf{0.00 \times 10^{0}}$
		Std	7.16×10^{-51}	9.52×10^{-1}	2.07×10^{1}	2.96×10^{0}	3.38×10^{-84}	7.70×10^{0}	6.59×10^{-21}	1.90×10^{-95}	6.21×10^{0}	$\mathbf{0.00 \times 10^{0}}$
	500	Avg	5.05×10^{-55}	6.36×10^{1}	8.08×10^{1}	9.90×10^{1}	2.01×10^{-75}	8.26×10^{1}	2.60×10^{-19}	5.29×10^{-74}	9.71×10^{1}	$\mathbf{0.00 \times 10^{0}}$
		Std	2.03×10^{-54}	6.24×10^{0}	1.81×10^{1}	4.10×10^{-1}	9.25×10^{-75}	4.30×10^{0}	6.57×10^{-19}	2.38×10^{-73}	1.08×10^{0}	$\mathbf{0.00 \times 10^{0}}$
F_5	50	Avg	2.36×10^{-2}	4.73×10^{1}	4.82×10^{1}	6.13×10^{6}	4.86×10^{1}	4.78×10^{1}	5.21×10^{-1}	4.90×10^{1}	5.69×10^{3}	$\mathbf{1.06 \times 10^{-2}}$
		Std	4.98×10^{-2}	7.14×10^{-1}	3.67×10^{-1}	6.39×10^{6}	3.00×10^{-1}	7.98×10^{-1}	5.50×10^{-1}	2.95×10^{-4}	6.80×10^{3}	$\mathbf{2.23 \times 10^{-2}}$
	100	Avg	2.55×10^{-1}	9.79×10^{1}	9.82×10^{1}	1.20×10^{8}	9.85×10^{1}	9.83×10^{1}	1.70×10^{0}	9.90×10^{1}	8.33×10^{4}	$\mathbf{1.36 \times 10^{-2}}$
		Std	8.35×10^{-1}	7.02×10^{-1}	1.96×10^{-1}	6.73×10^{7}	2.60×10^{-1}	4.54×10^{-1}	2.10×10^{0}	1.09×10^{-1}	3.48×10^{4}	$\mathbf{2.41 \times 10^{-2}}$
	500	Avg	1.16×10^{0}	4.98×10^{2}	4.96×10^{2}	2.03×10^{9}	4.98×10^{2}	4.98×10^{2}	7.40×10^{0}	4.99×10^{2}	6.68×10^{8}	$\mathbf{1.03 \times 10^{-1}}$
		Std	1.22×10^{0}	2.63×10^{-1}	3.49×10^{-1}	5.69×10^{8}	2.66×10^{-1}	2.02×10^{-1}	9.29×10^{0}	1.21×10^{-1}	9.84×10^{7}	$\mathbf{1.11 \times 10^{-1}}$
F_6	50	Avg	3.88×10^{-4}	2.77×10^{0}	1.24×10^{0}	9.46×10^{2}	1.12×10^{1}	6.14×10^{0}	3.21×10^{-2}	5.60×10^{0}	5.49×10^{1}	$\mathbf{1.50 \times 10^{-5}}$
		Std	6.94×10^{-4}	4.90×10^{-1}	4.53×10^{-1}	1.13×10^{3}	7.38×10^{-1}	7.29×10^{-1}	2.86×10^{-2}	4.41×10^{-1}	1.31×10^{1}	$\mathbf{2.09 \times 10^{-5}}$
	100	Avg	3.18×10^{-4}	9.79×10^{0}	4.52×10^{0}	1.17×10^{4}	2.41×10^{1}	1.65×10^{1}	4.09×10^{-1}	1.56×10^{1}	8.71×10^{2}	$\mathbf{1.71 \times 10^{-4}}$
		Std	6.57×10^{-4}	9.06×10^{-1}	1.24×10^{0}	7.86×10^{3}	8.07×10^{-1}	8.91×10^{-1}	2.00×10^{-1}	7.13×10^{-1}	1.71×10^{2}	$\mathbf{3.90 \times 10^{-4}}$
	500	Avg	1.45×10^{-3}	9.11×10^{1}	3.08×10^{1}	2.01×10^{5}	1.25×10^{2}	1.10×10^{2}	4.25×10^{0}	1.12×10^{2}	2.82×10^{5}	$\mathbf{7.90 \times 10^{-4}}$
		Std	3.38×10^{-3}	1.34×10^{0}	8.25×10^{0}	7.46×10^{4}	$\mathbf{1.32 \times 10^{-7}}$	1.36×10^{0}	1.68×10^{0}	1.28×10^{0}	1.94×10^{4}	1.33×10^{-3}
F_7	50	Avg	2.37×10^{-4}	3.52×10^{-3}	3.81×10^{-3}	2.68×10^{0}	$\mathbf{9.04 \times 10^{-5}}$	7.04×10^{-4}	8.22×10^{-4}	1.19×10^{-4}	3.88×10^{-1}	9.69×10^{-5}
		Std	1.96×10^{-4}	1.72×10^{-3}	5.06×10^{-3}	2.83×10^{0}	$\mathbf{6.21 \times 10^{-5}}$	4.95×10^{-4}	5.22×10^{-4}	9.93×10^{-5}	7.76×10^{-2}	1.71×10^{-4}

Table 9. *Cont.*

F_n	Dimension	Criteria	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
	100	Avg	2.59×10^{-4}	7.68×10^{-3}	3.50×10^{-3}	1.78×10^{2}	**8.68×10^{-5}**	1.50×10^{-3}	8.56×10^{-4}	1.20×10^{-4}	2.47×10^{0}	1.00×10^{-4}
		Std	2.00×10^{-4}	2.99×10^{-3}	5.11×10^{-3}	8.68×10^{1}	**8.17×10^{-5}**	7.88×10^{-4}	7.74×10^{-4}	9.80×10^{-5}	4.96×10^{-1}	9.68×10^{-5}
	500	Avg	3.18×10^{-4}	4.68×10^{-2}	4.54×10^{-3}	1.47×10^{4}	**8.81×10^{-5}**	6.37×10^{-3}	1.18×10^{-3}	2.08×10^{-4}	4.20×10^{3}	1.34×10^{-4}
		Std	2.49×10^{-4}	9.83×10^{3}	4.80×10^{-3}	3.11×10^{3}	**8.59×10^{-5}**	2.99×10^{-3}	6.76×10^{-4}	2.40×10^{-4}	7.61×10^{2}	1.69×10^{-4}
F_8	50	Avg	-8.40×10^{3}	-9.15×10^{3}	**-1.68×10^{4}**	-4.79×10^{3}	-4.44×10^{3}	-5.49×10^{3}	-1.37×10^{4}	-3.25×10^{3}	-1.13×10^{4}	-1.50×10^{4}
		Std	1.71×10^{3}	1.19×10^{3}	3.10×10^{3}	3.80×10^{2}	7.01×10^{2}	1.86×10^{3}	7.95×10^{2}	**4.45×10^{2}**	1.15×10^{3}	7.76×10^{2}
	100	Avg	-1.11×10^{4}	-1.63×10^{4}	**-3.51×10^{4}**	-6.75×10^{3}	-6.32×10^{3}	-9.77×10^{3}	-2.23×10^{4}	-4.55×10^{3}	-1.97×10^{4}	-2.42×10^{4}
		Std	3.38×10^{3}	1.56×10^{3}	4.87×10^{3}	6.01×10^{2}	9.38×10^{2}	4.01×10^{3}	2.55×10^{3}	**4.37×10^{2}**	1.57×10^{3}	1.21×10^{3}
	500	Avg	-2.86×10^{4}	-5.45×10^{4}	**-1.84×10^{5}**	-1.54×10^{4}	-1.37×10^{4}	-2.72×10^{4}	-6.19×10^{4}	-1.05×10^{4}	-5.63×10^{4}	-6.46×10^{4}
		Std	8.37×10^{3}	1.15×10^{4}	2.77×10^{4}	**1.36×10^{3}**	2.14×10^{3}	1.40×10^{4}	2.40×10^{3}	1.41×10^{3}	5.34×10^{3}	2.27×10^{3}
F_9	50	Avg	**0.00×10^{0}**	5.99×10^{0}	**0.00×10^{0}**	1.35×10^{2}	5.65×10^{-1}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	1.55×10^{2}	**0.00×10^{0}**
		Std	**0.00×10^{0}**	6.77×10^{0}	**0.00×10^{0}**	5.68×10^{1}	6.26×10^{-1}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	1.95×10^{1}	**0.00×10^{0}**
	100	Avg	**0.00×10^{0}**	1.01×10^{1}	**0.00×10^{0}**	2.86×10^{2}	8.35×10^{-1}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	6.03×10^{2}	**0.00×10^{0}**
		Std	**0.00×10^{0}**	7.30×10^{0}	**0.00×10^{0}**	1.05×10^{2}	6.56×10^{-1}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	3.83×10^{1}	**0.00×10^{0}**
	500	Avg	3.03×10^{-14}	6.65×10^{1}	**0.00×10^{0}**	1.20×10^{3}	4.34×10^{-1}	3.40×10^{-12}	**0.00×10^{0}**	**0.00×10^{0}**	5.03×10^{3}	**0.00×10^{0}**
		Std	1.66×10^{-13}	1.81×10^{1}	**0.00×10^{0}**	4.02×10^{2}	5.74×10^{-1}	1.72×10^{-12}	**0.00×10^{0}**	**0.00×10^{0}**	2.61×10^{2}	**0.00×10^{0}**
F_{10}	50	Avg	**8.88×10^{-16}**	4.50×10^{-11}	4.09×10^{-15}	1.63×10^{1}	4.80×10^{-15}	1.34×10^{-14}	**8.88×10^{-16}**	4.44×10^{-15}	5.14×10^{0}	**8.88×10^{-16}**
		Std	**0.00×10^{0}**	2.51×10^{-11}	2.16×10^{-15}	6.91×10^{0}	1.08×10^{-15}	3.33×10^{-15}	**0.00×10^{0}**	0.00×10^{0}	8.40×10^{-1}	**0.00×10^{0}**
	100	Avg	**8.88×10^{-16}**	1.32×10^{-7}	4.80×10^{-15}	1.95×10^{1}	5.98×10^{-15}	5.30×10^{-14}	**8.88×10^{-16}**	4.09×10^{-15}	1.42×10^{1}	**8.88×10^{-16}**
		Std	**0.00×10^{0}**	5.46×10^{-8}	2.85×10^{-15}	2.95×10^{0}	1.79×10^{-15}	1.14×10^{-14}	**0.00×10^{0}**	1.08×10^{-15}	2.81×10^{0}	**0.00×10^{0}**
	500	Avg	**8.88×10^{-16}**	1.73×10^{-3}	4.68×10^{-15}	1.94×10^{1}	7.16×10^{-15}	3.13×10^{-8}	**8.88×10^{-16}**	4.09×10^{-15}	1.94×10^{1}	**8.88×10^{-16}**
		Std	**0.00×10^{0}**	2.61×10^{-4}	2.27×10^{-15}	3.23×10^{0}	1.53×10^{-15}	1.45×10^{-8}	**0.00×10^{0}**	1.08×10^{-15}	4.60×10^{-2}	**0.00×10^{0}**
F_{11}	50	Avg	**0.00×10^{0}**	3.93×10^{-3}	1.41×10^{-2}	9.60×10^{0}	1.60×10^{-3}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	1.55×10^{0}	**0.00×10^{0}**
		Std	**0.00×10^{0}**	7.52×10^{-3}	5.43×10^{-2}	8.83×10^{0}	3.67×10^{-3}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	1.68×10^{-1}	**0.00×10^{0}**
	100	Avg	**0.00×10^{0}**	5.44×10^{-3}	**0.00×10^{0}**	8.96×10^{1}	9.84×10^{-4}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	9.14×10^{0}	**0.00×10^{0}**
		Std	**0.00×10^{0}**	1.12×10^{-2}	**0.00×10^{0}**	5.30×10^{1}	3.01×10^{-3}	**0.00×10^{0}**	**0.00×10^{0}**	**0.00×10^{0}**	1.64×10^{0}	**0.00×10^{0}**
	500	Avg	**0.00×10^{0}**	2.16×10^{-2}	**0.00×10^{0}**	1.68×10^{3}	4.69×10^{-4}	1.12×10^{-3}	**0.00×10^{0}**	**0.00×10^{0}**	2.58×10^{3}	**0.00×10^{0}**
		Std	**0.00×10^{0}**	4.03×10^{-2}	**0.00×10^{0}**	7.29×10^{2}	2.57×10^{-3}	6.14×10^{-3}	**0.00×10^{0}**	**0.00×10^{0}**	1.96×10^{2}	**0.00×10^{0}**
F_{12}	50	Avg	2.33×10^{-6}	1.16×10^{-1}	2.18×10^{-2}	1.53×10^{7}	1.09×10^{0}	3.81×10^{-1}	3.12×10^{-3}	2.63×10^{-1}	1.54×10^{1}	**7.69×10^{-7}**
		Std	4.17×10^{-6}	7.66×10^{-2}	9.32×10^{-3}	1.60×10^{7}	2.18×10^{-1}	9.88×10^{-2}	1.23×10^{-2}	3.75×10^{-2}	9.64×10^{0}	**1.51×10^{-6}**
	100	Avg	1.74×10^{-6}	2.95×10^{-1}	5.04×10^{-2}	3.26×10^{8}	1.18×10^{0}	5.85×10^{-1}	4.94×10^{-3}	4.86×10^{-1}	2.76×10^{2}	**8.11×10^{-7}**
		Std	4.41×10^{-6}	7.17×10^{-2}	3.13×10^{-2}	1.68×10^{8}	9.35×10^{-2}	8.16×10^{-2}	2.75×10^{-3}	6.09×10^{-2}	5.62×10^{2}	**1.17×10^{-6}**
	500	Avg	1.62×10^{-6}	7.47×10^{-1}	9.08×10^{-2}	5.93×10^{9}	1.20×10^{0}	9.35×10^{-1}	1.37×10^{-2}	9.91×10^{-1}	1.14×10^{9}	**5.41×10^{-7}**
		Std	3.38×10^{-6}	4.58×10^{-2}	4.73×10^{-2}	1.31×10^{9}	9.37×10^{-3}	2.39×10^{-2}	4.41×10^{-3}	2.74×10^{-2}	2.51×10^{8}	**6.73×10^{-7}**
F_{13}	50	Avg	1.36×10^{-5}	2.18×10^{0}	1.13×10^{0}	3.16×10^{7}	4.73×10^{0}	3.52×10^{0}	2.83×10^{-2}	5.00×10^{0}	7.85×10^{1}	**6.92×10^{-6}**
		Std	1.29×10^{-5}	3.65×10^{-1}	4.54×10^{-1}	3.59×10^{7}	2.08×10^{-1}	2.45×10^{-1}	2.74×10^{-2}	**4.93×10^{-8}**	2.85×10^{1}	1.11×10^{-5}
	100	Avg	2.10×10^{-5}	6.90×10^{0}	2.84×10^{0}	4.80×10^{8}	9.69×10^{0}	8.46×10^{0}	2.76×10^{-1}	1.00×10^{1}	8.03×10^{3}	**1.40×10^{-5}**
		Std	2.94×10^{-5}	4.47×10^{-1}	1.01×10^{0}	2.79×10^{8}	3.18×10^{-1}	3.39×10^{-1}	2.19×10^{-1}	**1.23×10^{-7}**	9.28×10^{3}	2.10×10^{-5}
	500	Avg	7.84×10^{-4}	5.11×10^{1}	1.90×10^{1}	9.97×10^{9}	4.98×10^{1}	4.79×10^{1}	2.79×10^{0}	5.00×10^{1}	2.42×10^{9}	**1.83×10^{-4}**
		Std	1.33×10^{-3}	1.49×10^{0}	4.90×10^{0}	1.80×10^{9}	6.15×10^{-2}	3.66×10^{-1}	2.11×10^{0}	**6.00×10^{-7}**	4.90×10^{8}	3.21×10^{-4}

The best values obtained have been highlighted in **boldface**.

Table 10. IEEE CEC2019 test suite.

Function	Name	Dimension (*D*)	Range	$F_{\min}$
CEC01	Storn's Chebyshev Polynomial Fitting Problem	9	[−8192, 8192]	1
CEC02	Inverse Hilbert Matrix Problem	16	[−16384, 16384]	1
CEC03	Lennard-Jones Minimum Energy Cluster	18	[−4, 4]	1
CEC04	Rastrigin's Function	10	[−100, 100]	1
CEC05	Griewangk's Function	10	[−100, 100]	1
CEC06	Weierstrass Function	10	[−100, 100]	1
CEC07	Modified Schwefel's Function	10	[−100, 100]	1
CEC08	Expanded Schaffer's F6 Function	10	[−100, 100]	1
CEC09	Happy Cat Function	10	[−100, 100]	1
CEC10	Ackley Function	10	[−100, 100]	1

From Table 11, it is evident that CHAOARO can show better optimization performance than its peers on almost all test functions. On functions CEC01 and CEC04~CEC10, the solution obtained by CHAOARO is much closer to the theoretical optimum than other comparison algorithms. On functions CEC02 and CEC03, though certain optimizers can achieve the same average fitness value as CHAOARO, the latter has a smaller standard deviation, which again proves the excellent robustness of the proposed work. Furthermore, the Wilcoxon rank-sum test and Friedman ranking test used for statistical analysis are also conducted to check whether the performance of CHAOARO has been significantly improved on the IEEE CEC2019 test set. According to the statistical results, CHAOARO outperforms AO, GWO, WOA, SCA, TSA, GJO, WChOA, and DAOA on 10 test functions and outperforms ARO on 9 functions. Besides, CHAOARO gains a Friedman mean ranking value of 1.0, which ranks first in the competition. These findings imply that the proposed algorithm not only can provide higher-quality solutions for simple optimization problems but also is very competitive in solving complex numerical optimization problems.

Figure 10 plots the convergence curves of CHAOARO and other comparison algorithms on 10 CEC2019 test functions. It can be visually seen from this figure that CHAOARO scores a promising convergence pattern, which shows great improvements over the basic AO and ARO. On functions CEC01, CEC02, CEC07, and CEC08, CHAOARO has a large decay rate in the early search phase, which enables it to obtain the most satisfactory outcomes with the least number of iterations among all algorithms. This is primarily attributed to the increased population diversity by COBL strategy. On functions CEC04, CEC05, CEC06, and CEC10, the superior exploitation capability of CHAOARO is well demonstrated. CHAOARO follows the same trend as some of its competitors in the early iterations, but in the later iterations, as most algorithms fall into the local optima, the proposed method is still approaching the global optimum and thus achieves better final convergence accuracy. On functions CEC03 and CEC09, CHAOARO also performs very competitively in terms of convergence accuracy and speed.

Table 11. Comparison results of CHAOARO and other algorithms on 10 CEC2019 test functions.

F_n	Criteria	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
CEC01	Avg	1.31×10^{5}	1.80×10^{8}	4.28×10^{10}	9.17×10^{9}	6.38×10^{4}	2.56×10^{8}	4.16×10^{4}	9.65×10^{4}	7.51×10^{10}	**4.05×10^{4}**
	Std	3.59×10^{4}	2.43×10^{8}	6.20×10^{10}	1.11×10^{10}	1.38×10^{4}	6.33×10^{8}	3.33×10^{3}	9.87×10^{3}	5.82×10^{10}	**2.74×10^{3}**
	p-value	3.02×10^{-11}	6.70×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	6.70×10^{-11}	3.65×10^{-8}	1.62×10^{-5}	3.02×10^{-11}	3.02×10^{-11}	—
	Rank	5	6	9	8	3	7	2	4	10	1
CEC02	Avg	1.79×10^{1}	1.74×10^{1}	1.74×10^{1}	1.75×10^{1}	1.84×10^{1}	1.74×10^{1}	**1.73×10^{1}**	1.75×10^{1}	6.08×10^{1}	**1.73×10^{1}**
	Std	2.88×10^{-1}	2.91×10^{-2}	6.07×10^{-2}	7.72×10^{-2}	6.66×10^{-1}	7.74×10^{-2}	2.82×10^{-5}	1.87×10^{-1}	2.32×10^{1}	**6.16×10^{-6}**
	p-value	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.43×10^{-3}	3.02×10^{-11}	3.02×10^{-11}	—
	Rank	8	3	4	6	9	5	2	7	10	1
CEC03	Avg	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	**1.27×10^{1}**	1.27×10^{1}
	Std	6.18×10^{-4}	4.52×10^{-4}	8.09×10^{-7}	5.85×10^{-4}	1.67×10^{-3}	3.66×10^{-4}	1.45×10^{-8}	6.33×10^{-4}	1.14×10^{-7}	**1.38×10^{-10}**
	p-value	3.02×10^{-11}	4.08×10^{-11}	1.61×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	7.51×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	—
	Rank	8	6	4	7	10	5	2	9	3	1
CEC04	Avg	7.95×10^{3}	5.54×10^{1}	3.51×10^{2}	1.53×10^{3}	7.01×10^{3}	1.16×10^{3}	3.98×10^{1}	2.17×10^{4}	7.09×10^{1}	**3.92×10^{1}**
	Std	2.46×10^{3}	2.18×10^{1}	2.57×10^{2}	5.31×10^{2}	3.32×10^{3}	1.32×10^{3}	2.11×10^{1}	7.97×10^{3}	2.00×10^{1}	**1.60×10^{1}**
	p-value	3.02×10^{-11}	2.16×10^{-3}	3.69×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	1.33×10^{-10}	1.23×10^{-3}	$3.02 \times 10—^{11}$	1.87×10^{-7}	—
	Rank	9	3	5	7	8	6	2	10	4	1
CEC05	Avg	4.18×10^{0}	1.39×10^{0}	1.93×10^{0}	2.24×10^{0}	3.23×10^{0}	1.69×10^{0}	1.14×10^{0}	5.72×10^{0}	1.31×10^{0}	**1.10×10^{0}**
	Std	8.36×10^{-1}	2.35×10^{-1}	5.18×10^{-1}	1.31×10^{-1}	9.84×10^{-1}	4.38×10^{-1}	9.58×10^{-2}	7.81×10^{-1}	1.31×10^{-1}	**8.78×10^{-2}**
	p-value	3.02×10^{-11}	1.25×10^{-7}	3.34×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.37×10^{-10}	7.51×10^{-1}	3.02×10^{-11}	1.16×10^{-7}	—
	Rank	9	4	6	7	8	5	2	10	3	1
CEC06	Avg	9.89×10^{0}	1.10×10^{1}	9.34×10^{0}	1.11×10^{1}	1.11×10^{1}	1.11×10^{1}	5.47×10^{0}	1.09×10^{1}	1.09×10^{1}	**5.35×10^{0}**
	Std	1.17×10^{0}	8.05×10^{-1}	1.22×10^{0}	7.01×10^{-1}	8.57×10^{-1}	6.59×10^{-1}	1.17×10^{0}	**6.23×10^{-1}**	6.46×10^{-1}	1.09×10^{0}
	p-value	3.69×10^{-11}	3.02×10^{-11}	6.70×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	8.07×10^{-4}	3.02×10^{-11}	3.02×10^{-11}	—
	Rank	4	7	3	9	10	8	2	5	6	1
CEC07	Avg	8.05×10^{2}	4.79×10^{2}	6.14×10^{2}	7.97×10^{2}	9.52×10^{2}	6.76×10^{2}	1.29×10^{2}	1.00×10^{3}	3.48×10^{2}	**1.02×10^{2}**
	Std	2.38×10^{2}	2.85×10^{2}	2.33×10^{2}	1.71×10^{2}	3.03×10^{2}	2.86×10^{2}	1.10×10^{2}	1.90×10^{2}	2.33×10^{2}	**1.05×10^{2}**
	p-value	4.08×10^{-11}	2.78×10^{-7}	3.82×10^{-10}	4.08×10^{-11}	3.69×10^{-11}	5.00×10^{-9}	2.82×10^{-8}	3.02×10^{-11}	2.68×10^{-4}	—
	Rank	8	4	5	7	9	6	2	10	3	1
CEC08	Avg	5.88×10^{0}	5.27×10^{0}	6.02×10^{0}	6.03×10^{0}	6.40×10^{0}	5.59×10^{0}	4.46×10^{0}	6.61×10^{0}	5.63×10^{0}	**4.17×10^{0}**
	Std	5.82×10^{-1}	7.25×10^{-1}	6.31×10^{-1}	4.87×10^{-1}	6.33×10^{-1}	8.57×10^{-1}	6.99×10^{-1}	6.90×10^{-1}	7.29×10^{-1}	**3.61×10^{-1}**
	p-value	4.20×10^{-10}	1.39×10^{-6}	1.33×10^{-10}	6.70×10^{-11}	3.02×10^{-11}	1.60×10^{-7}	1.19×10^{-5}	3.02×10^{-11}	1.85×10^{-8}	—
	Rank	6	3	7	8	9	4	2	10	5	1

Table 11. *Cont.*

F_n	Criteria	AO	GWO	WOA	SCA	TSA	GJO	ARO	WChOA	DAOA	CHAOARO
CEC09	Avg	1.32×10^{3}	4.37×10^{0}	4.74×10^{0}	1.23×10^{2}	6.02×10^{2}	1.04×10^{2}	2.75×10^{0}	6.80×10^{2}	2.96×10^{0}	$\mathbf{2.72 \times 10^{0}}$
	Std	4.25×10^{2}	9.39×10^{-1}	7.64×10^{-1}	9.64×10^{1}	6.08×10^{2}	3.04×10^{2}	2.76×10^{-1}	1.26×10^{1}	4.30×10^{-1}	$\mathbf{2.46 \times 10^{-1}}$
	p-value	3.02×10^{-11}	8.89×10^{-10}	4.08×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	8.53×10^{-3}	3.02×10^{-11}	3.67×10^{-3}	—
	Rank	10	4	5	7	8	6	2	9	3	1
CEC10	Avg	2.03×10^{1}	2.05×10^{1}	2.03×10^{1}	2.05×10^{1}	2.05×10^{1}	2.03×10^{1}	2.00×10^{1}	2.05×10^{1}	2.04×10^{1}	$\mathbf{1.82 \times 10^{1}}$
	Std	1.25×10^{-1}	7.57×10^{-2}	1.37×10^{-1}	9.19×10^{-2}	9.50×10^{-2}	1.10×10^{-1}	6.03×10^{-2}	9.61×10^{-2}	7.20×10^{-2}	$\mathbf{5.75 \times 10^{-2}}$
	p-value	2.15×10^{-10}	3.02×10^{-11}	6.07×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	4.20×10^{-10}	1.51×10^{-10}	3.02×10^{-11}	3.02×10^{-11}	—
	Rank	4	7	5	8	9	3	2	10	6	1
+/=/−		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	10/0/0	9/0/1	10/0/0	10/0/0	—
Friedman Mean Rank		7.1	4.7	5.3	7.4	8.3	5.5	2.0	8.4	5.3	1.0
Final Ranking		7	3	4	8	9	6	2	10	4	1

The best values obtained have been highlighted in **boldface**.

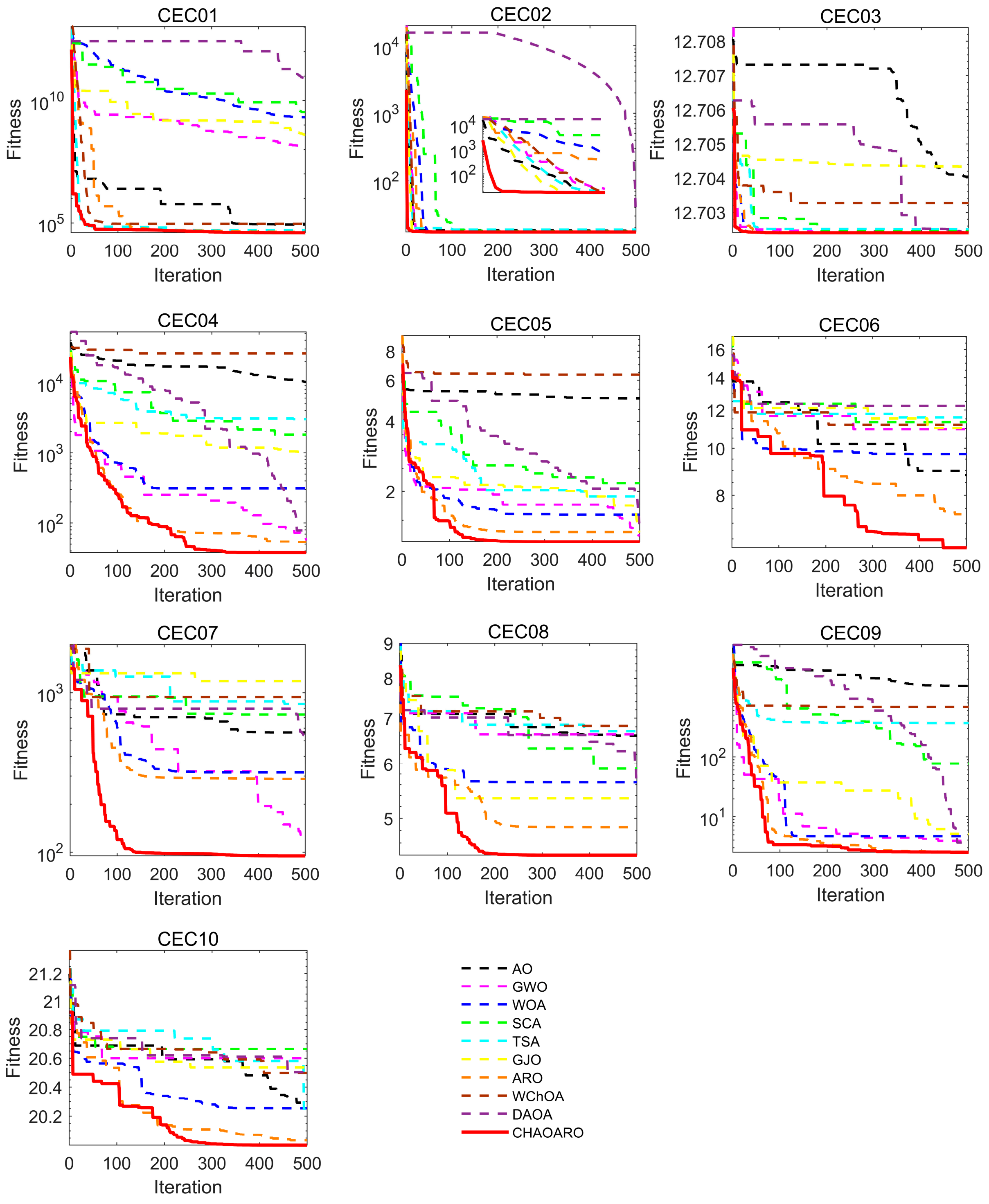

Figure 10. Convergence curves of CHAOARO and other comparison algorithms on 10 CEC2019 test functions.

In conclusion, no matter whether handling simple or complex numerical problems, CHAOARO can be trusted to offer reliable optimization performance in most scenarios. CHAOARO succeeds the strengths of the original AO and ARO and employs ASM and COBL strategies to compensate for defects like the tendency to fall into the local optima and the imbalance between exploration and exploitation, thus achieving better solution accuracy, convergence speed, and robustness. Next, we are going to validate the practicality of the proposed hybrid technique for real-world optimization tasks.

5. CHAOARO for Solving Engineering Design Problems

In this section, five complex engineering design problems, including pressure vessel design problem, cantilever beam design problem, tubular column design problem, speed reducer design problem, and rolling element bearing design problem, are used to sufficiently verify the effectiveness of the proposed CHAOARO at the real-world application level. In contrast to the benchmark functions, these engineering test cases contain several equality and inequality constraints, which present a major challenge for MAs. To deal with these inequality constraints in the problem, here we introduce the penalty function [69] to modify the original objective function. Similarly, CHAOARO runs independently 30 times on each problem with the population size (N) and the maximum number of iterations (T) fixed to 30 and 500, respectively, and the optimal results obtained are compared with different famous optimization methods released in previous studies.

5.1. Design of Pressure Vessel

The design of pressure vessels is a common engineering test case appeared in previous studies for evaluating the performance of optimization techniques. In this design, the objective is to minimize the overall fabrication cost of a cylindrical vessel capped at both ends by hemispherical heads. As illustrated in Figure 11, the thickness of the shell ($T_s = x_1$), the thickness of the head ($T_h = x_2$), the vessel's inner radius ($R = x_3$), and the length of the vessel without heads ($L = x_4$) are the four decision parameters for optimization. This problem is mathematically stated as follows:

Consider $\vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L]$

Minimize $f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

Subject to

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \le 0, g_2(\vec{x}) = -x_3 + 0.00954x_3 \le 0,$$
$$g_3(\vec{x}) = -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1296000 \le 0, g_4(\vec{x}) = x_4 - 240 \le 0.$$

Variable range

$$0 \le x_1, x_2 \le 99, 10 \le x_3, x_4 \le 200.$$

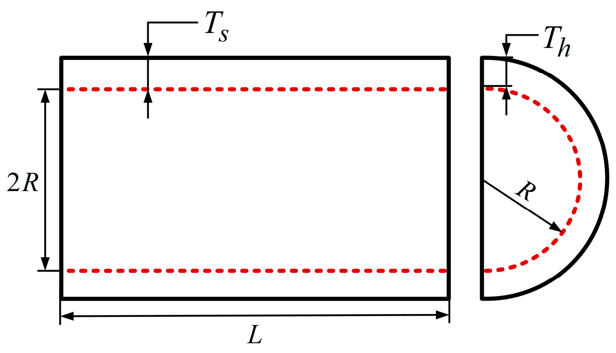

Figure 11. Schematic illustration of pressure vessel design problem.

Table 12 reports the results of CHAOARO and other optimization techniques for the pressure vessel design problem. As can be seen from this table, the minimum cost of 5885.5834 is attained by CHAOARO when the four variables T_s, T_h, R, and L are set as 0.7783, 0.3847, 40.3254, and 199.9213, respectively. Compared with AO, MVO, WOA, GWO, MFO, HHO, SMA, GJO, ARO, and AOASC, the proposed method provides the best design outcome. Therefore, it is reasonable to believe that CHAOARO has remarkable advantages in resolving such problem.

Table 12. Comparison results for pressure vessel design problem.

Algorithms	Optimal Values for Variables				Minimum Cost
	$T_s(x_1)$	$T_h(x_2)$	$R(x_3)$	$L(x_4)$	
AO [46]	1.0540	0.1828	59.6219	38.8050	5949.2258
MVO [20]	0.8125	0.4375	42.0907	176.7387	6060.8066
WOA [30]	0.8125	0.4375	42.0987	176.6390	6059.7410
GWO [27]	0.8125	0.4345	42.0892	176.7587	6051.5639
MFO [29]	0.8125	0.4375	42.0984	176.6366	6059.7143
HHO [32]	0.8176	0.4072	42.0917	176.7196	6000.4626
SMA [34]	0.7931	0.3932	40.6711	196.2178	5994.1857
GJO [36]	0.7783	0.3848	40.3219	200.0000	5887.0711
ARO [55]	0.7782	0.3848	40.3234	199.9479	5885.6679
AOASC [70]	0.8254	0.4262	42.7605	169.3396	6048.6812
CHAOARO	0.7783	0.3847	40.3254	199.9213	**5885.5834**

The best values obtained have been highlighted in **boldface**.

5.2. Design of Cantilever Beam

The cantilever beam design problem originated by [71] is one of the most representative issues in the area of mechanics and civil engineering. This problem aims to minimize the total weight of a cantilever beam with a square section while satisfying the load-carrying conditions. As illustrated in Figure 12, the height or width of the five square hollow elements are the decision variables that need to be taken into account in the minimization process, and the thickness of each element is constant. The mathematical formula of this problem can be expressed as follows:

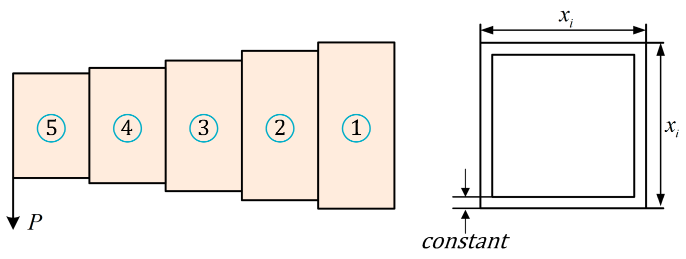

Figure 12. Schematic illustration of cantilever beam design problem.

Consider $\vec{x} = [x_1, x_2, x_3, x_4, x_5]$
Minimize $f(\vec{x}) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5)$
Subject to

$$g(\vec{x}) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$$

Variable range

$$0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100$$

In Table 13, the optimum variables and solutions of different algorithms for the cantilever beam design problem are listed. As it presents, CHAOARO can reveal a much smaller weight than the majority of other competitors, which is 1.339956. In addition, the performance of RUN on this application is equally competitive. These experimental data demonstrate the promising potential of CHAOARO in terms of reducing the total weight of cantilever beams.

Table 13. Comparison results for cantilever beam design problem.

Algorithms	Optimal Values for Variables					Minimum Weight
	x_1	x_2	x_3	x_4	x_5	
CS [26]	6.0089	5.3049	4.5023	3.5077	2.1504	1.339990
MVO [20]	6.0239	5.3060	4.4950	3.4960	2.1527	1.339960
MFO [29]	5.9849	5.3167	4.4973	3.5136	2.1616	1.339988
SMA [34]	6.0178	5.3109	4.4938	3.5011	2.1502	1.339957
ARO [55]	6.0068	5.3114	4.4935	3.5029	2.1590	1.339960
SOS [72]	6.0188	5.3034	4.4959	3.4990	2.1556	1.339960
AHA [58]	6.0138	5.3024	4.4963	3.5084	2.1527	1.339957
RUN [73]	6.0049	5.3190	4.4868	3.5033	2.1595	**1.339956**
HAGSA [3]	5.9271	5.3962	4.5081	3.4760	2.1726	1.340400
ERHHO [74]	6.0509	5.2639	4.5140	3.4605	2.1878	1.340200
CHAOARO	6.0163	5.3099	4.4951	3.5007	2.1517	**1.339956**

The best values obtained have been highlighted in **boldface**.

5.3. Design of Tubular Column

In this optimization, the task is to design a uniform column of the tubular section with the length $L = 250$ cm at minimum cost so as to withstand the compressive load $P = 2500$ kgf. As illustrated in Figure 13, this problem involves two design variables, namely the column's mean diameter ($d = x_1$) and the thickness of tube ($t = x_2$). Besides, the yield stress (σ_y), modulus of elasticity (E), and density (ρ) of the material used to construct the column are 500 kgf/cm^2, 0.85×10^6 kgf/cm^2, and 0.0025 kgf/cm^3, respectively. For it, the mathematical model is as follows:

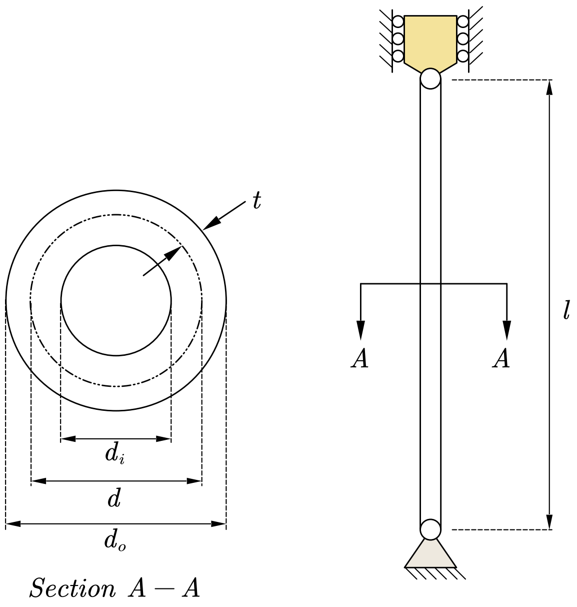

Figure 13. Schematic illustration of tubular column design problem.

Consider $\vec{x} = [x_1, x_2] = [d, t]$
Minimize $f(\vec{x}) = 9.8x_1x_2 + 2x_1$
Subject to

$$g_1(\vec{x}) = \frac{P}{\pi x_1 x_2 \sigma_y} - 1 \leq 0, g_2(\vec{x}) = \frac{8PL^2}{\pi^3 E x_1 x_2 (x_1^2 + x_2^2)} - 1 \leq 0,$$
$$g_3(\vec{x}) = \frac{2.0}{x_1} - 1 \leq 0, g_4(\vec{x}) = \frac{x_1}{14} - 1 \leq 0,$$
$$g_5(\vec{x}) = \frac{0.2}{x_2} - 1 \leq 0, g_6(\vec{x}) = \frac{x_2}{8} - 1 \leq 0.$$

Variable range

$$2 \leq x_1 \leq 14, 0.2 \leq x_2 \leq 0.8.$$

Table 14 records the comparison results between the proposed CHAOARO and the remaining algorithms for tackling the tubular column design problem. It is evident that CHAOARO obtains the lowest optimum cost of 26.48636 among these algorithms when the two variables d and t are set as 5.45218 and 0.29163. Nevertheless, the performances of the basic ARO and AO accordingly rank 3rd and 10th. It is proved that CHAOARO has better effects regarding this design.

Table 14. Comparison results for tubular column design problem.

Algorithms	**Optimal Values for Variables**		**Minimum Cost**
	$d(x_1)$	$t(x_2)$	
AO	5.41639	0.29826	26.66455
CS [26]	5.45139	0.29196	26.53217
SNS [75]	5.45116	0.29197	26.49950
GWO	5.45643	0.29142	26.49618
WOA	5.45658	0.29139	26.49516
SCA	5.37810	0.30510	26.83667
GJO	5.45086	0.29194	26.49682
ARO	5.45665	0.29139	26.49559
ChOA	5.46878	0.29327	26.65508
GSA-GA [76]	5.45116	0.29197	26.53133
CHAOARO	5.45218	0.29163	**26.48636**

The best values obtained have been highlighted in **boldface**.

5.4. Design of Speed Reducer

The speed reducer is one of the most critical components in the gearbox system [77]. The goal of this optimization problem is to reduce the weight of a speed reducer as much as possible under different constraints on surface stress, bending stress, stress in the shafts, and transverse deflection of the shafts. As depicted in Figure 14, there are seven decision variables to be considered in this optimal design, including the face width (x_1), the module of teeth (x_2), the number of teeth in the pinion (x_3), the length of the first shaft between bearings (x_4), the length of the second shaft between bearings (x_5), and the diameter of the shafts (x_6, x_7). The mathematical formulation of this problem is given as follows:

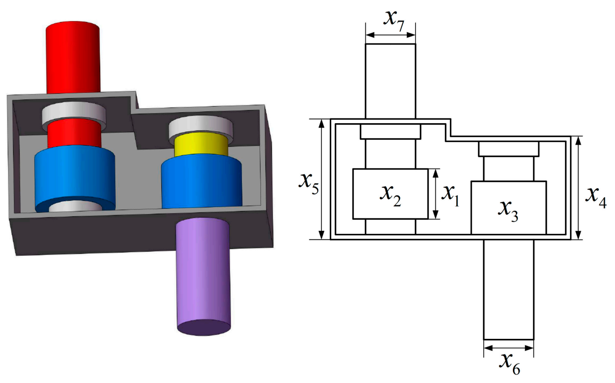

Figure 14. Schematic illustration of speed reducer design problem.

Consider $\vec{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$
Minimize

$$f(\vec{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Subject to

$$
\begin{aligned}
&g_1(\vec{x}) = \tfrac{27}{x_1 x_2^2 x_3} - 1 \le 0, g_2(\vec{x}) = \tfrac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0, \\
&g_3(\vec{x}) = \tfrac{1.93x_4^2}{x_2 x_6^4 x_3} - 1 \le 0, g_4(\vec{x}) = \tfrac{1.93x_5^2}{x_2 x_7^4 x_3} - 1 \le 0, \\
&g_5(\vec{x}) = \tfrac{\sqrt{(\frac{745x_5}{x_2 x_3})^2 + 16.9\times 10^6}}{110x_6^3} - 1 \le 0, g_6(\vec{x}) = \tfrac{\sqrt{(\frac{745x_5}{x_2 x_3})^2 + 157.5\times 10^6}}{85x_7^3} - 1 \le 0, \\
&g_7(\vec{x}) = \tfrac{x_2 x_3}{40} - 1 \le 0, g_8(\vec{x}) = \tfrac{5x_2}{x_1} - 1 \le 0, g_9(\vec{x}) = \tfrac{x_1}{12x_2} - 1 \le 0, \\
&g_{10}(\vec{x}) = \tfrac{1.5x_6 + 1.9}{x_4} - 1 \le 0, g_{11}(\vec{x}) = \tfrac{1.1x_7 + 1.9}{x_5} - 1 \le 0.
\end{aligned}
$$

Variable range

$$
\begin{aligned}
&2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9, \\
&5.0 \le x_7 \le 5.5.
\end{aligned}
$$

CHAOARO is employed to optimize this problem and compared with seven other algorithms respectively. The obtained results are summarized in Table 15. From this table, it is not difficult to observe that the proposed CHAOARO outperforms all other comparison methods published in the literature and achieves the minimum weight of 2994.4488. This effectively indicates that CHAOARO possesses an excellent global optimization capability in the design of the speed reducer.

Table 15. Comparison results for speed reducer design problem.

Algorithms	Optimal Values for Variables							Minimum Weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
AO [46]	3.5021	0.7000	17.0000	7.3099	7.7476	3.3641	5.2994	3007.7328
AOA [23]	3.50384	0.7	17	7.3	7.72933	3.35649	5.2867	2997.9157
SSA [69]	3.500059	0.7	17	7.3	7.8	3.351209	5.286813	2996.7077
SHO [78]	3.50159	0.7	17	7.3	7.8	3.35127	5.28874	2998.5507
AFA [79]	3.500000	0.7000	17	7.302489	7.800067	3.350219	5.286683	2996.3727
STOA [80]	3.50124	0.7	17	7.3	7.8	3.33425	5.26538	2995.9578
SC-GWO [81]	3.50064	0.7	17	7.30643	7.80617	3.35034	5.28694	2996.9859
CHAOARO	3.50001	0.7000	17	7.30002	7.71535	3.35057	5.28666	**2994.4488**

The best values obtained have been highlighted in **boldface**.

5.5. Design of Rolling Element Bearing

The last test case, rolling element bearing design problem, contains ten decision variables and nine constraints for modeling and geometry-based limitations. Its main purpose is to maximize the dynamic load-carrying capacity of a rolling element bearing illustrated in Figure 15. The geometric design parameters are pitch diameter (D_m), ball diameter (D_b), number of balls (Z), inner (f_i) and outer (f_o) raceway curvature radius coefficient, K_{dmin}, K_{dmax}, δ, e, and ζ. Mathematically, this problem is described below:
Consider $\vec{x} = [D_m, D_b, Z, f_i, f_o, K_{d\text{min}}, K_{d\text{max}}, \delta, e, \zeta]$
Maximize

$$
f(\vec{x}) = \begin{cases} f_c Z^{2/3} D_b^{1.8}, if\ D_b \le 25.4\text{mm} \\ 3.647 f_c Z^{2/3} D_b^{1.4}, otherwise \end{cases}
$$

Subject to

$$
\begin{aligned}
&g_1(\vec{x}) = \tfrac{\phi_0}{2\sin^{-1}(D_b/D_m)} - Z + 1 \le 0, g_2(\vec{x}) = 2D_b - K_{d\text{min}}(D-d) > 0, \\
&g_3(\vec{x}) = K_{d\text{max}}(D-d) - 2D_b \ge 0, g_4(\vec{x}) = \zeta B_w - D_b \le 0, \\
&g_5(\vec{x}) = D_m - 0.5(D+d) \ge 0, g_6(\vec{x}) = (0.5+e)(D+d) - D_m \ge 0, \\
&g_7(\vec{x}) = 0.5(D - D_m - D_b) - \delta D_b \ge 0, g_8(\vec{x}) = f_i \ge 0.515, g_9(\vec{x}) = f_o \ge 0.515.
\end{aligned}
$$

where

$$f_c = 37.91\left[1+\left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41},$$
$$\phi_0 = 2\pi - \cos^{-1}\frac{\{(D-d)/2-3(T/4)\}^2+\{D/2-T/4-D_b\}^2-\{d/2+T/4\}^2}{2\{(D-d)/2-3(T/4)\}\{D/2-T/4-D_b\}},$$
$$\gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_o = \frac{r_o}{D_b}, T = D-d-2D_b, D = 160, d = 90, B_w = 30, r_i = r_o = 11.033,$$
$$0.5(D+d) \le D_m \le 0.6(D+d), 0.15(D-d) \le D_b \le 0.45(D-d), 0.515 \le f_i,\ f_o \le 0.6,$$
$$4 \le Z \le 50, 0.4 \le K_{d\min} \le 0.5, 0.6 \le K_{d\max} \le 0.7, 0.3 \le \delta \le 0.4, 0.02 \le e \le 0.1, 0.6 \le \zeta \le 0.85.$$

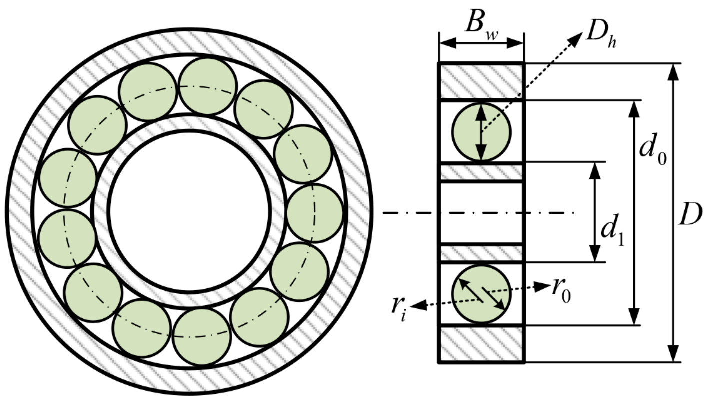

Figure 15. Schematic illustration of rolling element bearing design problem.

The detailed results of CHAOARO for this problem are compared with other metaheuristics in Table 16. It can be seen that the developed technique is able to find a more reliable solution compared with its peers. The load-carrying capacity of CHAOARO optimized design is 85548.8272, showing a significant improvement. This instance once again validates the merits of CHAOARO from the practical application aspect.

Table 16. Comparison results for rolling element bearing design problem.

Algorithms	**HHO** [32]	**RSA** [35]	**TLBO** [37]	**RUN** [73]	**IGTO** [82]	**CHAOARO**
D_m	125	125.1722	125.7191	125.2142	125	125.719
D_b	21.00000	21.29734	21.42559	21.59796	21.41885	21.42554
Z	11.09207	10.88521	11.00000	11.40240	10.94110	10.65574
f_i	0.51500	0.515253	0.51500	0.51500	0.51500	0.51500
f_o	0.51500	0.517764	0.51500	0.51500	0.51500	0.5151428
K_{dmin}	0.40000	0.41245	0.424266	0.40059	0.40000	0.4574078
K_{dmax}	0.60000	0.632338	0.633948	0.61467	0.70000	0.6544766
δ	0.30000	0.301911	0.30000	0.30530	0.30000	0.3000026
e	0.05047	0.024395	0.068858	0.02000	0.02000	0.05889122
ζ	0.60000	0.6024	0.799498	0.63665	0.60000	0.6698756
Optimal load-carrying capacity	83,011.883	83,486.64	81,859.74	83,680.47	85,067.962	**85,548.8272**

The best values obtained have been highlighted in **boldface**.

In summary, this section showcases the effectiveness of the proposed CHAOARO in dealing with real-world engineering test problems subject to different constraints. CHAOARO could perform better than the basic AO, ARO, as well as other existing optimizers with high-quality solutions, largely attributed to the hybrid operation, adaptive switching coefficient F, and COBL that well balance and boost the algorithm exploration and exploitation to varying degrees. In the next section, the superiority of CHAOARO will be further illustrated in another practical case study—parameter identification of PV model.

6. CHAOARO for Parameter Identification of Photovoltaic Model

To cope with the crisis of climate change, environmental pollution, and the depletion of conventional fossil fuels, an increased emphasis has been placed on the search for high-quality renewable energy sources in recent years [83]. Among different renewable energy sources, solar energy is regarded as one of the most promising renewable energy sources since it is clean, abundant, and pollution-free. In most parts of the world, PV systems are widely used to convert solar energy into electrical energy for power generation. The performance of a PV system relies on the chosen PV model and the unknown parameters in the model [84]. Currently, several PV models have been designed, such as single diode model (SDM), double diode model (DDM), and triple diode model (TDM). However, SDM is still extensively utilized in practice attributed to its simplicity and accuracy. Because PV systems usually operate in harsh outdoor environments, a variety of uncertainties may directly effect changes in model parameters, thus reducing the utilization efficiency of solar energy. Hence, it is of great practical significance to develop an accurate and robust method for identifying the unknown parameters of the PV model.

In this section, CHAOARO is applied to solve the parameter identification problem of SDM to further verify the superiority of the proposed method. As the most prevalent model to characterize the properties of PV power generation, the SDM consists of a photo-generated current source I_{ph}, a parallel diode D, a parallel resistance R_{sh}, and an equivalent series resistance R_{s}, shown in Figure 16. In this model, the output current I_{o} according to Kirchhoff's current law can be expressed as follows:

$$\begin{aligned} I_{\mathrm{o}} &= I_{\mathrm{ph}} - I_{\mathrm{d}} - I_{\mathrm{sh}} \\ &= I_{\mathrm{ph}} - I_{\mathrm{sd}}\left[\exp\left(\frac{q(V_{\mathrm{o}}+R_{\mathrm{s}}I_{\mathrm{o}})}{nkT}\right) - 1\right] - \frac{V_{\mathrm{o}}+I_{\mathrm{o}}R_{\mathrm{s}}}{R_{\mathrm{sh}}} \end{aligned} \tag{28}$$

where I_{ph} denotes the photo-generated current, I_{d} denotes the diode current, I_{sh} indicates the shunt resistance current, I_{sd} represents the reverse saturation current of the diode D, q is the electron charge equal to $1.60217646 \times 10^{-19}$ C, V_{o} is the output voltage, R_{s} and R_{sh} are the series and parallel resistances, respectively, n is the diode ideality coefficient, k is the Boltzmann constant equal to $1.3806503 \times 10^{-23}$ J/K, and T stands for the absolute temperature in Kelvin.

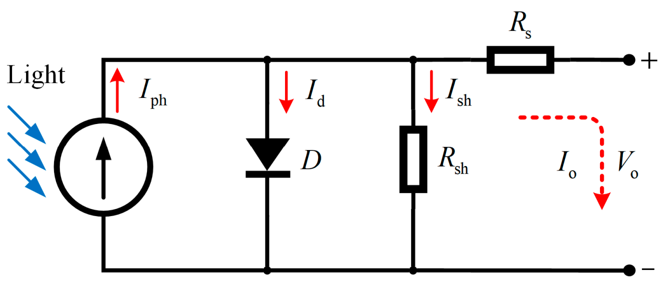

Figure 16. Structure of single diode model.

From Equation (28), it can be observed that there are a total of five unknown parameters that need to be estimated for SDM, namely I_{ph}, I_{sd}, R_{s}, R_{sh}, and n.

To identify the PV parameters using the meta-heuristic algorithm, it is necessary to define an objective function for this optimization problem first. Here, the root mean square error (RMSE) [85], which can reflect the degree of error between the actual measured data and the data estimated by CHAOARO, is introduced as the objective function:

$$\min F(\mathrm{X}) = \mathrm{RMSE}(\mathrm{X}) = \sqrt{\frac{1}{N}\sum_{k=1}^{N} f_k(V_{\mathrm{o}}, I_{\mathrm{o}}, \mathrm{X})^2} \tag{29}$$

where N denotes the number of experimental data. The smaller the RMSE value achieved, the more accurate the identified parameters are.

For SDM, $f_k(V_o, I_o, \mathbf{X})$ and $\mathbf{X}$ in Equation (29) are as follows:

$$\begin{cases} f_k(V_o, I_o, \mathbf{X}) = I_{ph} - I_{sd}\left[\exp\left(\frac{q(V_o + R_s I_o)}{nkT}\right) - 1\right] - \frac{V_o + I_o R_s}{R_{sh}} - I_o \\ \mathbf{X} = \left\{I_{ph}, I_{sd}, R_s, R_{sh}, n\right\} \end{cases}, \tag{30}$$

where $0 \le I_{ph} \le 1, 0 \le I_{sd} \le 1, 0 \le R_s \le 0.5, 0 \le R_{sh} \le 100, 1 \le n \le 2$.

Based on the actual measured current-voltage data in reference [86], where commercial silicon R.T.C French solar cells with a diameter of 57 mm were operated under 1000 W/m^2 at 33 °C, we utilize the proposed method to identify the five unknown parameters of SDM. CHAOARO runs independently 30 times on this test problem with the population size (N) and the maximum number of iterations (T) set to 30 and 500 respectively, and the obtained optimal parameters and RMSE-value are reported in Table 17.

Table 17. Comparison results for the parameter identification of SDM.

Algorithms	I_{ph} (A)	I_{sd} (μA)	R_s (Ω)	R_{sh} (Ω)	n	RMSE
ABC [85]	0.760784	0.321523	0.036398	53.639	1.480601	9.8169×10^{-4}
SMA [87]	0.76076	0.32314	0.03637	53.71489	1.48114	9.8482×10^{-4}
IHBA [88]	0.76101	0.39445	0.034789	55.538	1.4951	1.0272×10^{-3}
IBES [89]	0.760776	0.323	0.036377	53.71853	1.4768	9.8600×10^{-4}
CPSO [90]	0.7607	0.4	0.0354	59.012	1.5033	1.3900×10^{-3}
OBDSSA [91]	0.7608	0.36596	0.0359	56.1662	1.4939	1.0161×10^{-3}
GOTLBO [92]	0.760780	0.331552	0.036265	54.115426	1.483820	9.8744×10^{-4}
CHAOARO	0.760776	0.314950	0.036485	53.19598	1.478640	$\mathbf{7.7330 \times 10^{-4}}$

The best values obtained have been highlighted in **boldface**.

As can be seen from Table 17, CHAOARO obtains the smallest RMSE value of 7.7330×10^{-4} compared to other state-of-the-art peer competitors, namely ABC [85], SMA [87], IHBA [88], IBES [89], CPSO [90], OBDSSA [91], and GOTLBO [92], indicating that CHAOARO has the highest accuracy for parameter identification. Furthermore, the best-extracted parameters attained by CHAOARO are used to generate the current-voltage (*I*-*V*) and power-voltage (*P*-*V*) characteristic curves, as shown in Figure 17. From this figure, it is clear that the estimated values of CHAOARO can fit the actual measured data well, which again demonstrates that the proposed method has excellent prospects and robustness in solving the parameter identification problem of SDM for PV systems.

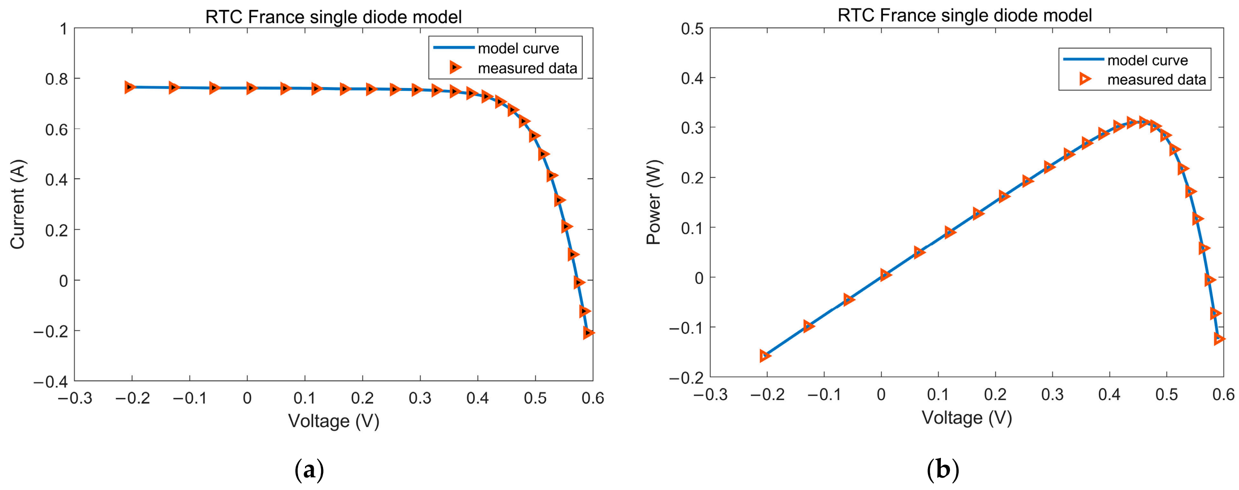

Figure 17. Fitting curves between the measured data and estimated data obtained by CHAOARO on the SDM. (**a**) *I*-*V* characteristics; (**b**) *P*-*V* characteristics.

7. Conclusions and Future Research

In this study, for the characteristics of AO and ARO, we skillfully combine these two algorithms and propose a new hybrid meta-heuristic optimization paradigm, called

CHAOARO, to provide more reliable solutions for complex global optimization problems. The proposed method aims to overcome the limitation of the original algorithm's insufficient search strategies, enrich the diversity of populations, and avoid local optimal stagnation. In CHAOARO, firstly, the global exploration stage of AO and the local exploitation stage of ARO are integrated together to obtain superior overall performance and convergence speed. Secondly, based on the starvation ratio F in African Vultures Optimization Algorithm, an adaptive switching mechanism is designed to better balance the exploration and exploitation patterns of the algorithm. Moreover, the chaotic opposition-based learning tactic is utilized to assist the individual in exploring more unknown search domains and increase the possibility of getting rid of the local optima. To thoroughly evaluate the performance of CHAOARO, we use 23 classical benchmark functions, including thirteen unimodal and multimodal benchmark functions under different dimensions and ten fix-dimension multimodal benchmark functions, as well as the famous IEEE CEC2019 test suite. The significant differences between different competitor algorithms in a statistical sense are verified by using the Friedman ranking test and Wilcoxon rank-sum test. Compared with AO, ARO, and seven other state-of-the-art metaheuristics, the experimental results credibly demonstrate that CHAOARO has superior competitiveness in terms of convergence speed, solution accuracy, local optima avoidance, and stability no matter when solving simple or challenging numerical problems. To prove the effectiveness of the proposed method in practical applications, CHAOARO is further applied to tackle five engineering design problems and the parameter extraction problem of the PV model. Our findings indicate that CHAOARO is a promising auxiliary tool for addressing real-world optimization tasks.

Although CHAOARO can effectively outperform the original AO and ARO, its optimization performance still has room for further improvement. As can be seen from Table 6, the results of CHAOARO on functions F_7 and F_8 are not the most perfect. In the next work, we will try to further enhance the exploration and exploitation capabilities of CHAOARO for better solution accuracy via introducing other modification techniques, such as adaptive β-hill climbing, Lévy flight, and evolutionary population dynamics. And the more challenging IEEE CEC2022 test suite will hopefully be employed to evaluate the performance differences between CHAOARO and some improved variants of AO. In addition, CHAOARO is ready to be applied to solve real-world optimization problems in more fields, for instance, multi-level threshold image segmentation, node localization of wireless sensor network, path planning for drones in a three-dimensional environment, parameter self-tuning of speed proportional integral differential (PID) controller for brushless direct current motors, and fault diagnosis of rolling bearing. It would also make sense to develop a multi-objective version of the CHAOARO algorithm for complex multi-objective projects.

Author Contributions: Conceptualization, Y.W. and Y.X.; methodology, Y.W.; software, Y.X.; validation, Y.W. and Y.X.; formal analysis, J.L.; writing—original draft preparation, Y.W. and J.L.; writing—review and editing, Y.X.; visualization, Y.W.; supervision, J.L. and Y.G.; funding acquisition, Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by the National Natural Science Foundation of China under Grant 52075090, Key Research and Development Program Projects of Heilongjiang Province under Grant GA21A403, Fundamental Research Funds for the Central Universities under Grant 2572021BF01, and Natural Science Foundation of Heilongjiang Province under Grant YQ2021E002.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors are grateful to the editor and reviewers for their constructive comments and suggestions, which have improved the presentation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xiao, Y.; Sun, X.; Guo, Y.; Cui, H.; Wang, Y.; Li, J.; Li, S. An enhanced honey badger algorithm based on Lévy flight and refraction opposition-based learning for engineering design problems. *J. Intell. Fuzzy Syst.* **2022**, *43*, 4517–4540. [CrossRef]
2. Jia, H.; Zhang, W.; Zheng, R.; Wang, S.; Leng, X.; Cao, N. Ensemble mutation slime mould algorithm with restart mechanism for feature selection. *Int. J. Intell. Syst.* **2021**, *37*, 2335–2370. [CrossRef]
3. Liu, Q.; Li, N.; Jia, H.; Qi, Q.; Abualigah, L.; Liu, Y. A hybrid arithmetic optimization and golden sine algorithm for solving industrial engineering design problems. *Mathematics* **2022**, *10*, 1567. [CrossRef]
4. Abd Elaziz, M.; Abualigah, L.; Attiya, I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Future Gener. Comput. Syst.* **2021**, *124*, 142–154. [CrossRef]
5. Guo, W.; Xu, P.; Dai, F.; Hou, Z. Harris hawks optimization algorithm based on elite fractional mutation for data clustering. *Appl. Intell.* **2022**, *52*, 11407–11433. [CrossRef]
6. Shi, K.; Liu, C.; Sun, Z.; Yue, X. Coupled orbit-attitude dynamics and trajectory tracking control for spacecraft electromagnetic docking. *Appl. Math. Model.* **2022**, *101*, 553–572. [CrossRef]
7. Liu, C.; Yue, X.; Zhang, J.; Shi, K. Active disturbance rejection control for delayed electromagnetic docking of spacecraft in elliptical orbits. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 2257–2268. [CrossRef]
8. Hu, G.; Zhong, J.; Du, B.; Wei, G. An enhanced hybrid arithmetic optimization algorithm for engineering applications. *Comput. Meth. Appl. Mech. Eng.* **2022**, *394*, 114901. [CrossRef]
9. Yang, J.; Liu, Z.; Zhang, X.; Hu, G. Elite chaotic manta ray algorithm integrated with chaotic initialization and opposition-based learning. *Mathematics* **2022**, *10*, 2960. [CrossRef]
10. Xiao, Y.; Guo, Y.; Cui, H.; Wang, Y.; Li, J.; Zhang, Y. IHAOAVOA: An improved hybrid aquila optimizer and African vultures optimization algorithm for global optimization problems. *Math. Biosci. Eng.* **2022**, *19*, 10963–11017. [CrossRef]
11. Wen, C.; Jia, H.; Wu, D.; Rao, H.; Li, S.; Liu, Q.; Abualigah, L. Modified remora optimization algorithm with multistrategies for global optimization problem. *Mathematics* **2022**, *10*, 3604. [CrossRef]
12. Jia, H.; Sun, K.; Zhang, W.; Leng, X. An enhanced chimp optimization algorithm for continuous optimization domains. *Complex Intell. Syst.* **2021**, *8*, 65–82. [CrossRef]
13. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–72. [CrossRef]
14. Storn, R.; Price, K. Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
15. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
16. Cheraghalipour, A.; Hajiaghaei-Keshteli, M.; Paydar, M.M. Tree Growth Algorithm (TGA): A novel approach for solving optimization problems. *Eng. Appl. Artif. Intell.* **2018**, *72*, 393–414. [CrossRef]
17. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
18. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
19. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
20. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-Verse Optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2015**, *27*, 495–513. [CrossRef]
21. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
22. Hashim, F.A.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W.; Mirjalili, S. Henry gas solubility optimization: A novel physics-based algorithm. *Future Gener. Comput. Syst.* **2019**, *101*, 646–667. [CrossRef]
23. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Meth. Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
24. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948. [CrossRef]
25. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
26. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2011**, *29*, 17–35. [CrossRef]
27. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
28. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
29. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
30. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
31. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
32. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
33. Kaur, S.; Awasthi, L.K.; Sangal, A.L.; Dhiman, G. Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103541. [CrossRef]
34. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]

35. Abualigah, L.; Elaziz, M.A.; Sumari, P.; Geem, Z.W.; Gandomi, A.H. Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Syst. Appl.* **2022**, *191*, 116158. [CrossRef]
36. Chopra, N.; Mohsin Ansari, M. Golden jackal optimization: A novel nature-inspired optimizer for engineering applications. *Expert Syst. Appl.* **2022**, *198*, 116924. [CrossRef]
37. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [CrossRef]
38. Manjarres, D.; Landa-Torres, I.; Gil-Lopez, S.; Del Ser, J.; Bilbao, M.N.; Salcedo-Sanz, S.; Geem, Z.W. A survey on applications of the harmony search algorithm. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1818–1831. [CrossRef]
39. Zhang, Q.; Wang, R.; Yang, J.; Ding, K.; Li, Y.; Hu, J. Collective decision optimization algorithm: A new heuristic optimization method. *Neurocomputing* **2017**, *221*, 123–137. [CrossRef]
40. Askari, Q.; Younas, I.; Saeed, M. Political optimizer: A novel socio-inspired meta-heuristic for global optimization. *Knowl.-Based Syst.* **2020**, *195*, 105703. [CrossRef]
41. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
42. Zheng, R.; Jia, H.; Abualigah, L.; Liu, Q.; Wang, S. Deep ensemble of slime mold algorithm and arithmetic optimization algorithm for global optimization. *Processes* **2021**, *9*, 1774. [CrossRef]
43. Zhang, Y.J.; Yan, Y.X.; Zhao, J.; Gao, Z.M. CSCAHHO: Chaotic hybridization algorithm of the Sine Cosine with Harris Hawk optimization algorithms for solving global optimization problems. *PLoS ONE* **2022**, *17*, e0263387. [CrossRef] [PubMed]
44. Cheng, X.; Li, J.; Zheng, C.; Zhang, J.; Zhao, M. An improved PSO-GWO algorithm with chaos and adaptive inertial weight for robot path planning. *Front. Neurorobot.* **2021**, *15*, 770361. [CrossRef] [PubMed]
45. Kundu, T.; Garg, H. LSMA-TLBO: A hybrid SMA-TLBO algorithm with lévy flight based mutation for numerical optimization and engineering design problems. *Adv. Eng. Softw.* **2022**, *172*, 103185. [CrossRef]
46. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
47. Guo, Z.; Yang, B.; Han, Y.; He, T.; He, P.; Meng, X.; He, X. Optimal PID tuning of PLL for PV inverter based on aquila optimizer. *Front. Energy Res.* **2022**, *9*, 812467. [CrossRef]
48. Fatani, A.; Dahou, A.; Al-Qaness, M.A.A.; Lu, S.; Abd Elaziz, M. Advanced feature extraction and selection approach using deep learning and Aquila optimizer for IoT intrusion detection system. *Sensors* **2021**, *22*, 140. [CrossRef]
49. Zhao, J.; Gao, Z.-M.; Chen, H.-F. The simplified aquila optimization algorithm. *IEEE Access* **2022**, *10*, 22487–22515. [CrossRef]
50. Wang, S.; Jia, H.; Abualigah, L.; Liu, Q.; Zheng, R. An improved hybrid aquila optimizer and harris hawks algorithm for solving industrial engineering optimization problems. *Processes* **2021**, *9*, 1551. [CrossRef]
51. Yu, H.; Jia, H.; Zhou, J.; Hussien, A.G. Enhanced Aquila optimizer algorithm for global optimization and constrained engineering problems. *Math. Biosci. Eng.* **2022**, *19*, 14173–14211. [CrossRef]
52. Gao, B.; Shi, Y.; Xu, F.; Xu, X. An improved Aquila optimizer based on search control factor and mutations. *Processes* **2022**, *10*, 1451. [CrossRef]
53. Verma, M.; Sreejeth, M.; Singh, M. Application of hybrid metaheuristic technique to study influence of core material and core trench on performance of surface inset PMSM. *Arab. J. Sci. Eng.* **2021**, *47*, 3037–3053. [CrossRef]
54. Zhang, Y.-J.; Yan, Y.-X.; Zhao, J.; Gao, Z.-M. AOAAO: The hybrid algorithm of arithmetic optimization algorithm with aquila optimizer. *IEEE Access* **2022**, *10*, 10907–10933. [CrossRef]
55. Wang, L.; Cao, Q.; Zhang, Z.; Mirjalili, S.; Zhao, W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105082. [CrossRef]
56. Wang, Y.; Huang, L.; Zhong, J.; Hu, G. LARO: Opposition-based learning boosted artificial rabbits-inspired optimization algorithm with Lévy flight. *Symmetry* **2022**, *14*, 2282. [CrossRef]
57. Zhuoran, Z.; Changqiang, H.; Hanqiao, H.; Shangqin, T.; Kangsheng, D. An optimization method: Hummingbirds optimization algorithm. *J. Syst. Eng. Electron.* **2018**, *29*, 386–404. [CrossRef]
58. Zhao, W.; Wang, L.; Mirjalili, S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Comput. Meth. Appl. Mech. Eng.* **2022**, *388*, 114194. [CrossRef]
59. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
60. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, 28–30 November 2005; pp. 695–701. [CrossRef]
61. Nguyen, T.-T.; Wang, H.-J.; Dao, T.-K.; Pan, J.-S.; Liu, J.-H.; Weng, S. An improved slime mold algorithm and its application for optimal operation of cascade hydropower stations. *IEEE Access* **2020**, *8*, 226754–226772. [CrossRef]
62. Wang, S.; Jia, H.; Liu, Q.; Zheng, R. An improved hybrid Aquila Optimizer and Harris Hawks Optimization for global optimization. *Math. Biosci. Eng.* **2021**, *18*, 7076–7109. [CrossRef]
63. Long, W.; Jiao, J.; Liang, X.; Cai, S.; Xu, M. A random opposition-based learning grey wolf optimizer. *IEEE Access* **2019**, *7*, 113810–113825. [CrossRef]
64. Xiao, Y.; Sun, X.; Zhang, Y.; Guo, Y.; Wang, Y.; Li, J. An improved slime mould algorithm based on tent chaotic mapping and nonlinear inertia weight. *Int. J. Innov. Comput Inf. Control* **2021**, *17*, 2151–2176. [CrossRef]

65. Khishe, M.; Nezhadshahbodaghi, M.; Mosavi, M.R.; Martin, D. A weighted chimp optimization algorithm. *IEEE Access* **2021**, *9*, 158508–158539. [CrossRef]
66. Khodadadi, N.; Snasel, V.; Mirjalili, S. Dynamic arithmetic optimization algorithm for truss optimization under natural frequency constraints. *IEEE Access* **2022**, *10*, 16188–16208. [CrossRef]
67. Theodorsson-Norheim, E. Friedman and Quade tests: Basic computer program to perform nonparametric two-way analysis of variance and multiple comparisons on ranks of several related samples. *Comput. Biol. Med.* **1987**, *17*, 85–99. [CrossRef]
68. García, S.; Fernández, A.; Luengo, J.; Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.* **2010**, *180*, 2044–2064. [CrossRef]
69. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [CrossRef]
70. Abualigah, L.; Ewees, A.A.; Al-qaness, M.A.A.; Elaziz, M.A.; Yousri, D.; Ibrahim, R.A.; Altalhi, M. Boosting arithmetic optimization algorithm by sine cosine algorithm and levy flight distribution for solving engineering optimization problems. *Neural Comput. Appl.* **2022**, *34*, 8823–8852. [CrossRef]
71. Chickermane, H.; Gea, H.C. Structural optimization using a new local approximation method. *Int. J. Numer. Methods Eng.* **1996**, *39*, 829–846. [CrossRef]
72. Cheng, M.-Y.; Prayogo, D. Symbiotic organisms search: A new metaheuristic optimization algorithm. *Comput. Struct.* **2014**, *139*, 98–112. [CrossRef]
73. Ahmadianfar, I.; Heidari, A.A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. *Expert Syst. Appl.* **2021**, *181*, 115079. [CrossRef]
74. Song, M.; Jia, H.; Abualigah, L.; Liu, Q.; Lin, Z.; Wu, D.; Altalhi, M. Modified harris hawks optimization algorithm with exploration factor and random walk strategy. *Comput. Intell. Neurosci.* **2022**, *2022*, 4673665. [CrossRef] [PubMed]
75. Bayzidi, H.; Talatahari, S.; Saraee, M.; Lamarche, C.P. Social network search for solving engineering optimization problems. *Comput. Intell. Neurosci.* **2021**, *2021*, 8548639. [CrossRef]
76. Garg, H. A hybrid GSA-GA algorithm for constrained optimization problems. *Inf. Sci.* **2019**, *478*, 499–523. [CrossRef]
77. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Dwarf mongoose optimization algorithm. *Comput. Meth. Appl. Mech. Eng.* **2022**, *391*, 114570. [CrossRef]
78. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [CrossRef]
79. Baykasoğlu, A.; Ozsoydan, F.B. Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Appl. Soft Comput.* **2015**, *36*, 152–164. [CrossRef]
80. Dhiman, G.; Kaur, A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. *Eng. Appl. Artif. Intell.* **2019**, *82*, 148–174. [CrossRef]
81. Gupta, S.; Deep, K.; Moayedi, H.; Foong, L.K.; Assad, A. Sine cosine grey wolf optimizer to solve engineering design problems. *Eng. Comput.* **2021**, *37*, 3123–3149. [CrossRef]
82. Xiao, Y.; Sun, X.; Guo, Y.; Li, S.; Zhang, Y.; Wang, Y. An improved gorilla troops optimizer based on lens opposition-based learning and adaptive β-Hill climbing for global optimization. *CMES-Comput. Model. Eng. Sci.* **2022**, *131*, 815–850. [CrossRef]
83. Chen, X.; Yu, K. Hybridizing cuckoo search algorithm with biogeography-based optimization for estimating photovoltaic model parameters. *Sol. Energy* **2019**, *180*, 192–206. [CrossRef]
84. Zhao, J.; Zhang, Y.; Li, S.; Wang, Y.; Yan, Y.; Gao, Z. A chaotic self-adaptive JAYA algorithm for parameter extraction of photovoltaic models. *Math. Biosci. Eng.* **2022**, *19*, 5638–5670. [CrossRef] [PubMed]
85. Oliva, D.; Cuevas, E.; Pajares, G. Parameter identification of solar cells using artificial bee colony optimization. *Energy* **2014**, *72*, 93–102. [CrossRef]
86. Easwarakhanthan, T.; Bottin, J.; Bouhouch, I.; Boutrit, C. Nonlinear minimization algorithm for determining the solar cell parameters with microcomputers. *Int. J. Sol. Energy* **1986**, *4*, 1–12. [CrossRef]
87. Kumar, C.; Raj, T.D.; Premkumar, M.; Raj, T.D. A new stochastic slime mould optimization algorithm for the estimation of solar photovoltaic cell parameters. *Optik* **2020**, *223*, 165277. [CrossRef]
88. Lei, W.; He, Q.; Yang, L.; Jiao, H. Solar photovoltaic cell parameter identification based on improved honey badger algorithm. *Sustainability* **2022**, *14*, 8897. [CrossRef]
89. Ramadan, A.; Kamel, S.; Hassan, M.H.; Khurshaid, T.; Rahmann, C. An improved bald eagle search algorithm for parameter estimation of different photovoltaic models. *Processes* **2021**, *9*, 1127. [CrossRef]
90. Huang, W.; Jiang, C.; Xue, L.; Song, D. Extracting solar cell model parameters based on chaos particle swarm algorithm. In Proceedings of the 2011 International Conference on Electric Information and Control Engineering, Wuhan, China, 15–17 April 2011; pp. 398–402. [CrossRef]
91. Wang, Z.; Ding, H.; Yang, J.; Wang, J.; Li, B.; Yang, Z.; Hou, P. Advanced orthogonal opposition-based learning-driven dynamic salp swarm algorithm: Framework and case studies. *IET Control Theory Appl.* **2022**, *16*, 945–971. [CrossRef]
92. Chen, X.; Yu, K.; Du, W.; Zhao, W.; Liu, G. Parameters identification of solar cell models using generalized oppositional teaching learning based optimization. *Energy* **2016**, *99*, 170–180. [CrossRef]