Computing
https://doi.org/10.1007/s00607-024-01290-1

REGULAR PAPER

Improved optimal foraging algorithm for global optimization

Chen Ding[1] · **GuangYu Zhu**[2,3]

Received: 28 May 2023 / Accepted: 3 April 2024

Abstract

The optimal foraging algorithm (OFA) is a swarm-based algorithm motivated by animal behavioral ecology theory. When solving complex optimization problems characterized by multiple peaks, OFA is easy to get trapped in local minima and encounters slow convergence. Therefore, this paper presents an improved optimal foraging algorithm with social behavior based on quasi-opposition (QOS-OFA) to address these problems. First, quasi-opposition-based learning (QOBL) is introduced to improve the overall quality of the population in the initialization phase. Second, an efficient cosine-based scale factor is designed to accelerate the exploration of the search space. Third, a new search strategy with social behavior is designed to enhance local exploitation. The cosine-based scale factor is used as a regulator to achieve a balance between global exploration and local exploitation. The proposed QOS-OFA is compared with seven meta-heuristic algorithms on a CEC benchmark test suite and three real-world optimization problems. The experimental results show that QOS-OFA is better than other competitors on most of the test problems.

Keywords Optimal foraging algorithm · Quasi-opposition-based learning · Social behavior · Global exploration · Local exploitation

✉ GuangYu Zhu
zhugy@fzu.edu.cn

Chen Ding
chen_d@tongji.edu.cn

[1] School of Electronics and Information Engineering, Tongji University, Shanghai, People's Republic of China

[2] School of Mechanical Engineering and Automation, Fuzhou University, Fuzhou, People's Republic of China

[3] Qi Shan Campus of Fuzhou University, No.2 Xue Yuan Road, Fuzhou City, Fujian Province, People's Republic of China

Published online: 17 April 2024

Mathematics Subject Classification 68W50

1 Introduction

Examples of global optimization problems (GOPs) exist in a variety of fields, such as business, engineering, geography, and physics. Some GOPs are characterized by non-differentiability and high dimensionality of search space, making them hard to be effectively solved by traditional optimization methods [1]. To solve these problems, a wide range of meta-heuristic algorithms (MAs) have been designed in the past decades. Due to their simple structure and easy implementation, MAs have been widely used in recent years. Generally, MA's optimization process is divided into two stages: global exploration and local exploitation. In the first stage, the solution globally explores the search space and collects as much information as possible about the whole search space. Then, the promising areas found in the first stage are investigated in detail to obtain near-optimal solutions [2, 3].

According to the source of inspiration, MAs can be classified into three types [4]: (1) physics-based algorithms (PAs), (2) evolution-based algorithms (EAs), and (3) swarm-based algorithms (SAs). An overview of these algorithms is shown in Table 1.

The first type is PAs, which are designed by mimicking physical phenomena such as spirals, wind, rain, and light [5]. Some typical PAs are the central force optimization (CFO) algorithm [6], the big-bang big-crunch (BBBC) algorithm [7], the ray optimization (RO) algorithm [8], the gravitational search algorithm (GSA) [9], and the water cycle algorithm (WCA) [10]. In this branch, some other algorithms have been developed in recent years, including the flow regime algorithm (FRA) [11], the field of force (FOF) algorithm [12], the sine cosine algorithm (SCA) [13], and the optics inspired optimization (OIO) algorithm [14].

The second type of EAs takes inspiration from natural evolutionary laws [15, 16]. One of its earliest examples is the genetic algorithm (GA), which uses crossover, mutation, and selection operators to solve optimization problems [17]. Since then, many EAs aimed at speeding up convergence and reducing the number of parameters have been introduced, such as differential evolution (DE) [18], genetic programming (GP) [19], and evolution strategies (ES) [20]. Among them, DE is one of the familiar EAs, which creates new solutions based on a differentiation strategy. In other studies, some variants of DE, such as success history-based adaptive differential evolution (SHADE) [21] and adaptive differential evolution with linear population size reduction evolution (LSHADE) [22], have been presented to further improve the performance of DE.

The third type of SAs focuses on mimicking the collective intelligent foraging behavior of organisms such as bees, ants, wolves, and bats [23–26]. Particle swarm optimization (PSO) [27] is a popular example of algorithms belonging to this type. PSO is developed based on a swarm of particles defined by iteratively updated position and velocity vectors. Due to its simple structure and excellent convergence performance, PSO has received a lot of attention [28, 29]. In

Table 1 An overview of relevant MAs

Type	Algorithm	Inspiration	Research
PAs	Central force optimization (CFO)	The law of gravity	[6]
	Big-bang big-crunch (BBBC)	The big bang and big crunch theory of the universe evolution	[7]
	Ray optimization (RO)	The light refraction law	[8]
	Gravitational search algorithm (GSA)	The law of gravity and the law of motion	[9]
	Water cycle algorithm (WCA)	The rainfall process	[10]
	Flow regime algorithm (FRA)	The classical fluid mechanics and flow regimes	[11]
	Field of force (FOF)	The gravitation, magnetic, or electric fields	[12]
	Sine cosine algorithm (SCA)	The sine and cosine functions	[13]
	Optics inspired optimization (OIO)	The optical characteristics of convex and concave mirrors	[14]
EAs	Genetic algorithm (GA)	Biological evolution	[17]
	Differential evolution (DE)	Differential vector	[18]
	Genetic programming (GP)	Biological evolution	[19]
	Evolution strategies (ES)	Biological evolution	[20]
SAs	Particle swarm optimization (PSO)	Flock foraging behavior	[27]
	Ant colony optimization (ACO)	Ant colony	[30]
	Artificial bee colony (ABC)	Honey bee swarm	[31]
	Chimp optimization algorithm (ChOA)	Intelligent group haunting of chimps	[32]
	Harris hawks optimization (HHO)	Predatory behavior of harris hawks	[33]
	Grey wolf optimizer (GWO)	Hunting behavior of grey wolves	[34]
	Whale optimization algorithm (WOA)	Social behavior of humpback whales	[35]
	Salp swarm algorithm (SSA)	Swarming behavior of salps	[36]
	Moth-flame optimization (MFO)	Transverse orientation of moths	[37]
	Shuffled frog-leaping algorithm (SFLA)	Foraging behavior of frogs	[38]

addition to PSO, many other SAs have been proposed in the past decades, such as ant colony optimization (ACO) [30], artificial bee colony (ABC) [31], chimp optimization algorithm (ChOA) [32], harris hawks optimization (HHO) [33], grey wolf optimizer (GWO) [34], whale optimization algorithm (WOA) [35], salp swarm algorithm (SSA) [36], moth-flame optimization (MFO) [37], shuffled frog-leaping algorithm (SFLA) [38]. More details about these algorithms can be found in [39].

The optimal foraging algorithm (OFA), an innovative SA, draws inspiration from optimal foraging theory (OFT) and animal foraging behavior [40]. OFA simulates the optimization process as an animal foraging process. Unlike other SAs, OFA is a unique algorithm in which better solutions recruit poorer solutions to traverse the search space. When solving GOPs, its performance is more competitive than some classical MAs, such as DE, PSO, and SFLA [40]. Besides that, OFA has been applied to drilling path optimization [41], parameter optimization [42], production scheduling [43], and white blood cell segmentation [44]. However, OFA still has some drawbacks like many other MAs, though it has been successfully applied to solve some practical problems. These drawbacks were found by observing the explorative and exploitive performance of OFA on a set of unimodal and multimodal functions [40]. For some complex optimization tasks characterized by multiple peaks, OFA encounters slow convergence speed and stagnation in local minima. Moreover, it suffers from the deficiency of exploration and exploitation due to the inefficient search strategy.

Therefore, this paper aims to overcome the drawbacks of OFA by designing an improved algorithm, which is called the optimal foraging algorithm with social behavior based on quasi-opposition (QOS-OFA). The improvement is mainly in three aspects. First, quasi-opposition-based learning (QOBL) is employed in the initialization phase to improve the population quality. Second, a modified scale factor based on the cosine function is used in the position update phase. In this phase, the cosine-based scale factor decreases adaptively over iterations, which helps to accelerate the convergence speed in the exploration stage. Third, the search strategy of OFA is redesigned to update the position of the solution, which introduces social information to enhance local exploitation.

The main contributions of this paper are summarized below.

1) An improved meta-heuristic algorithm called QOS-OFA is proposed. The quality of the initial population is improved by applying quasi-opposition-based learning. A new search strategy with social behavior is designed in QOS-OFA to guide the solutions for local exploitation.
2) An efficient cosine-based scale factor is designed in QOS-OFA, which serves as a regulator of recruitment behavior and social behavior to achieve a balance between global exploration and local exploitation.
3) The effectiveness of the proposed algorithm is validated by comparing the performance of QOS-OFA with that of recent MAs on the benchmark test suite CEC. The superiority of QOS-OFA is demonstrated by numerical analysis, statistical analysis, convergence analysis, and typical case analysis.

4) The applicability of the proposed QOS-OFA is validated on three real-world optimization problems, including the gear train problem, the parameter estimation for frequency-modulated problem, and the drilling path optimization problem.

The remainder of the paper is as follows. The basic concept of standard OFA and QOBL is provided in Sect. 2. The structure of QOS-OFA is elaborated in Sect. 3. The experimental results of QOS-OFA and its competitors on CEC 2020 benchmark functions are presented in Sect. 4. In Sect. 5, the proposed QOS-OFA method is applied to solve three real-world optimization problems and the results are reported. Finally, Sect. 6 concludes this paper.

2 Related work

This section briefly reviews the fundamentals of OFA, since our proposed algorithm is based on OFA. In addition, this section outlines QOBL, which aims to enhance the performance of OFA by improving the quality of the initial population.

2.1 Optimal foraging algorithm

In general, a GOP is expressed in the following form.

$$f(\boldsymbol{x}^*) = minf(\boldsymbol{x}), \boldsymbol{x} = \{x_1, \ldots, x_i, \ldots, x_D\}, R = \{\boldsymbol{x} | a_i \le x_i \le b_i|\}$$

where R is the search space, $\boldsymbol{x} \in R$ is a D-dimensional vector, which is viewed as a solution in OFA. $f(\boldsymbol{x}^*)$ is the optimal fitness value of $\boldsymbol{x}^*$, a_i and b_i indicate the lower bound and upper bound of the i-th decision vector x_i, respectively. In the OFA's optimization process, all the solutions are randomly generated according to $x_{ji}^{t+1} = a_i + rand(0,1) \times (b_i - a_i)$ in the initialization phase. The solutions are then ranked from the best $\boldsymbol{x}_1$ to the worst $\boldsymbol{x}_N$ according to the fitness value. A solution $\boldsymbol{x}_j$ with worse value is recruited by a solution $\boldsymbol{x}_b$ with better value, and the best solution $\boldsymbol{x}_1$ is recruited by the worst solution $\boldsymbol{x}_N$. The search strategy is formulated as [40]:

$$\begin{cases} \boldsymbol{x}_j^{t+1} = \boldsymbol{x}_j^t + k \times (r_1 - r_2) \times (\boldsymbol{x}_j^t - \boldsymbol{x}_b^t), b < j; \\ \boldsymbol{x}_1^{t+1} = \boldsymbol{x}_1^t + k \times (r_1 - r_2) \times (\boldsymbol{x}_1^t - \boldsymbol{x}_N^t), j = 2, \ldots, N. \end{cases} \tag{1}$$

where $k = t/T$ is a scale factor, T is the maximum number of iterations, and r_1 and $r_2 \in$[0,1] are uniform random numbers. From Eq. (1), it can be observed that although r_1 and r_2 are non-negative numbers, the operator of subtraction indicates that the offspring solution $\boldsymbol{x}_j^{t+1}$ can be obtained at a position around $\boldsymbol{x}_j^t$ depending on the relative size of values.

After updating the position, the following equation is used to select a better one between $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_j^{t+1}$ [40].

$$\frac{\lambda_j^{t+1} \times f_j^{t+1}}{1 + (t+1) \times \lambda_j^{t+1}} < \frac{f_j^t}{t} \tag{2}$$

where λ_j^{t+1} is a uniformly random number in [0,1]. f_j^t and f_j^{t+1} denote the fitness value of $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_j^{t+1}$, respectively. If Eq. (2) is satisfied, $\boldsymbol{x}_j^{t+1}$ is selected; otherwise, $\boldsymbol{x}_j^t$ is retained.

2.2 Quasi-opposition-based learning

Opposition-based learning (OBL), proposed by Tizhoosh, is a popular computational theory in the intelligence community [45]. The main idea behind OBL is to consider the current estimate and its corresponding opposite side to find a better solution. When the objective of an algorithm is to obtain an optimal solution for a given problem, considering the current position and its opposite position simultaneously is beneficial to improving the search ability of the algorithm. In [46], it has been mathematically proven that the opposite solution has more chances to be near the unknown optimal solution than a randomly generated solution. Therefore, OBL has been used to improve the performance of some MAs such as GWO [47], HHO [48], ChOA [49], and SSA [50].

As the research goes on, the concept of OBL is extended. A new version of OBL named quasi-opposition-based learning (QOBL) is presented [51]. Empirical studies demonstrate that quasi-opposite solutions have a higher probability of being close to the optimal solution than opposite solution. For a random solution $\boldsymbol{x} = [x_1, \ldots, x_D]$ in a D-dimensional search space, its quasi-opposite solution $\bar{\boldsymbol{x}}^q = [\bar{x}_1^q, \ldots, \bar{x}_D^q]$ is defined below [51] and shown in Fig. 1.

$$\bar{x}_i^q = rand(c_i, \bar{x}_i^o), i = 1, \ldots, D \tag{3}$$

$$\bar{x}_i^o = a_i + b_i - x_i \tag{4}$$

where $c_i = \frac{a_i + b_i}{2}$ is the center of the boundary interval $[a_i, b_i]$, $\bar{x}_i^o$ is the opposite solution of x_i [45], and $\bar{x}_i^q$ is a uniformly random solution in $[c_i, \bar{x}_i^o]$.

3 Proposed algorithm

There are three improvements in the proposed QOS-OFA. In the initial phase, the population quality is improved by the QOBL method. In the position update phase, an efficient cosine-based scale factor is designed to accelerate the

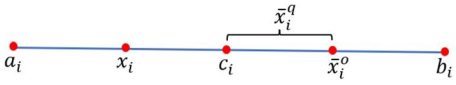


Fig. 1 Representation of the quasi-opposite solution

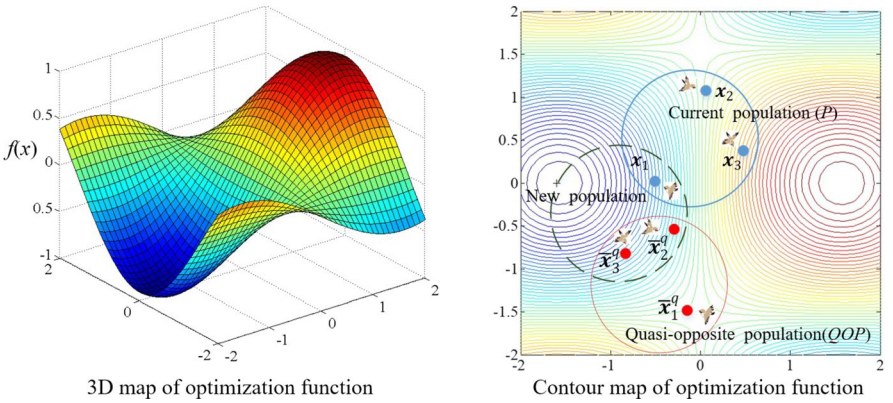
exploration of the search space. Moreover, a new search strategy is designed to enhance local exploitation by introducing social information. With these improvements, a tradeoff between global exploration and local exploitation is achieved in the optimization process.

3.1 Details of the proposed algorithm

3.1.1 The population quality improved by the QOBL method

The optimization process of OFA is to find the best solution in the search space. Because OFA is a stochastic algorithm, the quality of the initial population has a certain impact on the solution accuracy and convergence performance. In the standard OFA, the initial solution is randomly generated, which is casual and simple. The main weakness of this way is that the initial population lacks ergodicity. The solutions may be too concentrated or dispersed in a certain region, which can reduce the convergence speed and affect the search for global optimum to some extent. To solve this problem, we employ QOBL to improve the quality of the initial population. The main advantage of QOBL is that quasi-opposite solutions have more chances of being closer to the unknown optimal solution than randomly generated solutions [46]. The QOBL method provides a new direction of optimization for the current solution. With this new direction, the potential solution can be located with a higher chance, especially when the quasi-opposite solution is exactly close to the global optimum. Thus, the QOBL method can improve the quality of the initial population and enable the algorithm to explore the search space effectively.

An example of applying the QOBL method is shown in Fig. 2. The 3D map and contour map of the optimization function are depicted. The implementation steps of QOBL are as follows.



3D map of optimization function

Contour map of optimization function

Fig. 2 Example of applying the QOBL method

(1) Suppose that the population size N is equal to 3. Randomly generated an initial population P, which consists of three current solutions, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$.
(2) Calculate the fitness values of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ according to the objective function.
(3) Calculate the quasi-opposite solutions by Eq.(3) to obtain a quasi-opposite population QOP. The QOP consists of three quasi-opposite solutions, $\bar{\boldsymbol{x}}_1^q$, $\bar{\boldsymbol{x}}_2^q$, and $\bar{\boldsymbol{x}}_3^q$.
(4) Calculate the fitness value of each quasi-opposite solution according to the objective function.
(5) Select the three fittest solutions from the population set $\{P, QOP\}$ in terms of the fitness value. They are combined into a new population for the next iteration. As shown in Fig. 2, the new population consists of $\boldsymbol{x}_1$, $\bar{\boldsymbol{x}}_2^q$, and $\bar{\boldsymbol{x}}_3^q$.

Based on the above steps, it can be observed that the new population is closer to the global optimum than the population P. The quality of the initial population is improved by the QOBL method.

3.1.2 The cosine-based scale factor

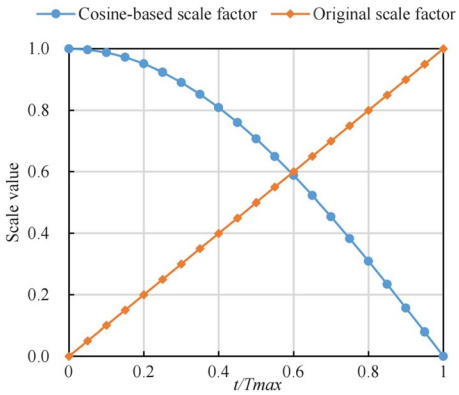
In the position update phase of OFA, all solutions update their positions by Eq. (1), in which the scale factor $k = t/T$ increases from 0 to 1 gradually as the iterative process proceeds. In this case, the recruitment behavior between solutions is weak initially, and the position increment of the solutions is small in the exploration stage. Thus, the solutions are unable to collect as much information as possible about the search space, and the role of the global search is inefficient. Therefore, an improvement is needed to overcome this drawback existed in the OFA.

To accomplish this improvement, an efficient cosine-based scale factor $K(t)$ is designed to replace the original scale factor, which is beneficial to accelerating the exploration of the search space. In many cases, although chaotic, linear, and exponential functions are used to control the exploration strength during the search process, the cosine function is selected in our work after extensive simulations showing its superiority compared to other functions. The mathematical equation of $K(t)$ is given as follows.

$$K(t) = \cos\left[\frac{\pi}{2} \times \frac{Current\ iteration\ number(t)}{Maximum\ iteration\ number(T)}\right] \tag{5}$$

where $K(t)$ monotonically decreases from 1 to 0 as the number of iterations increases. The curve of $K(t)$ is shown in Fig. 3. Compared with the original scale factor, a big value of $K(t)$ gives a height effect on the recruitment behavior of solutions, thus allowing the solution to explore the search space globally in the early stage, whereas a small value of $K(t)$ tends to enable local exploitation, thus facilitating the refinement of the solution in the later stage.

Fig. 3 Scale factor



3.1.3 The search strategy with social behavior

Furthermore, the significance of using social information should not be underestimated. In OFA, the global best solution $\boldsymbol{x}_1^t$ can be considered as the leader of the population. During the search process, its information is crucial for others to explore the potential region. However, in the search strategy of the standard OFA, all solutions except $\boldsymbol{x}_1^t$ are randomly recruited by better solutions to change their position states, which indicates that the information stored in the global best solution $\boldsymbol{x}_1^t$ is not fully used. Also, if the recruit solutions get stuck in local minima, there exists a high probability that the whole population falls into local minima. Therefore, a new search strategy with social behavior is proposed in QOS-OFA and expressed by the following equation.

$$\begin{aligned} \boldsymbol{x}_j^{t+1} &= \boldsymbol{x}_j^t + K(t) \times \overbrace{(r_1 - r_2) \times (\boldsymbol{x}_j^t - \boldsymbol{x}_b^t)}^{\textit{Recruitment behavior}} + (1 - K(t)) \times \overbrace{(\boldsymbol{x}_1^t - \boldsymbol{x}_r^t)}^{\textit{Social behavior}} \ ; \\ \boldsymbol{x}_1^{t+1} &= \boldsymbol{x}_1^t + K(t) \times (r_1 - r_2) \times (\boldsymbol{x}_1^t - \boldsymbol{x}_N^t), j < N, j = 2, \ldots, N. \end{aligned} \tag{6}$$

where $(\boldsymbol{x}_1^t - \boldsymbol{x}_r^t)$ is the vector of social behavior, and $\boldsymbol{x}_r^t$ is a randomly selected solution from the population. This vector provides a direction towards the promising regions of the search space using the global best solution $\boldsymbol{x}_1^t$, and enhances the local minimum avoidance ability by the random solution $\boldsymbol{x}_r^t$. As can be seen from the first line in Eq. (6), $K(t)$ serves as a regulator of recruitment behavior and social behavior. In the exploration stage, solutions explore the search space globally and some potential regions can be always discovered through recruitment behavior. In the exploitation stage, solutions are guided by social information to investigate the potential regions around the global best solution $\boldsymbol{x}_1^t$, thus better solutions can be found over time. Hence, a tradeoff between global exploration and local exploitation is achieved in the optimization process.

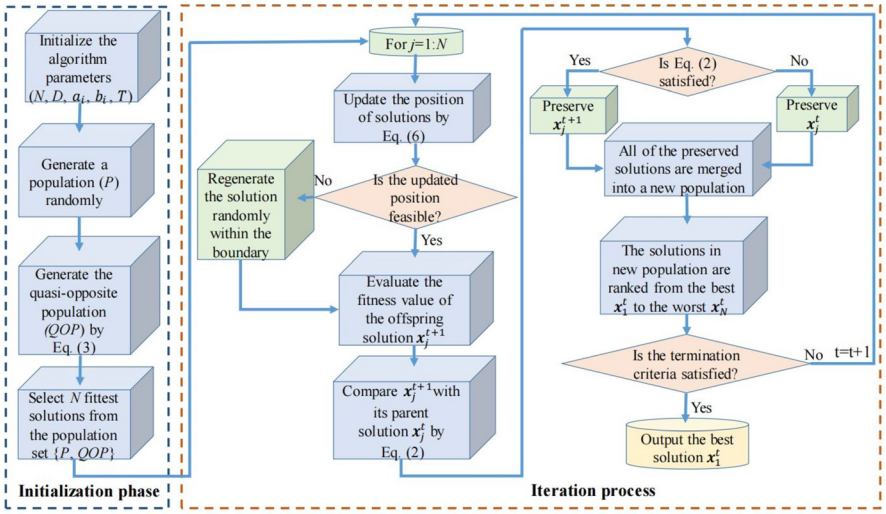


Fig. 4 The flowchart of QOS-OFA

3.2 The main steps of the proposed algorithm

The flowchart of QOS-OFA is shown in Fig. 4. In the initialization phase, a population P is randomly generated by $x_{ji}^{t+1} = a_i + rand(0,1) \times (b_i - a_i)$. At the same time, a quasi-opposite population QOP is generated by calculating the quasi-opposite positions of P. Then, the fitness values of solutions in two populations are calculated, and the N fittest solutions are selected to form a new population. In this manner, the quality of the initial population can be improved, thus allowing the algorithm to explore the search space more efficiently. During the iteration process, a new search strategy with social behavior is applied to iteratively update the positions of the solutions. In this strategy, solutions explore the search space globally in the early stage, while in the later stage, the solutions are guided by social information to exploit the promising regions. As a result, a balance between global exploration and local exploitation can be realized, which is the core of the proposed algorithm. Specifically, the main steps of the proposed algorithm are as follows.

(1) ***Initialization***

(1-1) Initialize the population size N, the maximum number of iterations T, the number of decision variables D, the boundary $[a_i, b_i]$ of the given problem. Set the current iteration number t=0.

(1-2) Based on the procedure described in Section **3.1.1**, select the N fittest solutions from the population set $\{P,QOP\}$ to form a new population for the next iteration.

(2) *Iteration process*

(2-1) Generate the offspring solutions $\boldsymbol{x}^{t+1}$ by Eq. (6) based on the ranking order.
(2-2) Check the feasibility of the offspring solutions, and regenerate the solution by $x_{ji}^{t+1} = a_i + rand(0,1) \times (b_i - a_i)$ if the solution is outside the boundary.
(2-3) Evaluate the fitness value of each offspring solution $\boldsymbol{x}^{t+1}$, and compare it with its parent solution $\boldsymbol{x}^t$ by Eq. (2). If Eq. (2) is satisfied, $\boldsymbol{x}^{t+1}$ is preserved; otherwise, $\boldsymbol{x}^t$ is preserved.
(2-4) All the preserved solutions are merged into the new population for the next iteration. $t = t + 1$.
(2-5) Sort all solutions in the population from best to worst according to the fitness value.
(2–6) Check the termination criteria. If the termination criteria are satisfied, output the optimization results; otherwise, return to step (2–1).

3.3 Computational complexity

The computational complexity of the proposed QOS-OFA depends mainly on the step of the ***Iteration process***. To be specific, it is associated with the number of decision variables D, the maximum number of iterations T, and the population size N. Thus, the stepwise computational complexity of QOS-OFA and standard OFA in terms of big-O notation is calculated as follows. For QOS-OFA, the overall computational complexity is $O(\text{QOS-OFA}) = O(\text{QOBL-based initialization}) + O($ position update based on the new search strategy$) + O(\text{selection of better solutions}) = O(2) + O(TND) + O(TN) = O(2 + TND + TN) = O(TND + TN)$. Similarly, $O(\text{OFA}) = O(1) + O(TND) + O(TN) = O(1 + TND + TN) = O(TND + TN)$. Therefore, for the worst case of computation time, the computational complexity of the proposed QOS-OFA and the standard OFA are the same.

4 Experiments

The proposed QOS-OFA is evaluated on a variety of test functions with different levels of difficulty, including two unimodal, four multimodal, six hybrid, and six composite functions. The details of these functions are listed in Table 2. They are selected from the benchmark suite CEC 2020 since this benchmark suite is suitable for validating the explorative and exploitive performance of the proposed algorithm. Experiments are conducted on these functions to evaluate the algorithm from different perspectives such as numerical analysis, convergence analysis, and statistical analysis. In addition, three typical cases are selected to analyze the exploration and exploitation capabilities of the proposed QOS-OFA.

Table 2 Details of benchmark functions

Function name	Global optimum
Unimodal functions	
Shifted and Rotated Bent Cigar function (CEC1)	100
Shifted and Rotated Zakharov function (CEC3)	300
Multimodal functions	
Shifted and Rotated Rastrigin's function (CEC5)	500
Shifted and Rotated Expanded Scaffer's function (CEC6)	600
Shifted and Rotated Non-continuous Rastrigin function (CEC8)	800
Shifted and Rotated Levy function (CEC9)	900
Hybrid functions	
Hybrid function 1 (N = 3) (CEC11)	1100
Hybrid function 3 (N = 3) (CEC13)	1300
Hybrid function 4 (N = 4) (CEC14)	1400
Hybrid function 6 (N = 4) (CEC16)	1600
Hybrid function 4 (N = 5) (CEC18)	1800
Hybrid function 10 (N = 6) (CEC20)	2000
Composite functions	
Composition function 2 (N = 3) (CEC22)	2200
Composition function 4 (N = 4) (CEC24)	2400
Composition function 6 (N = 5) (CEC26)	2600
Composition function 7 (N = 6) (CEC27)	2700
Composition function 8 (N = 6) (CEC28)	2800
Composition function 10 (N = 3) (CEC30)	3000

4.1 Parameter setting

In the experiments, QOS-OFA is compared with seven algorithms: SCA [13], GWO [34], WOA [35], SSA [36], MFO [37], and LSHADE [22]. Each algorithm evolves with its own iteration steps until the termination criteria are satisfied.

The common parameters for all the algorithms are as follows. The dimension of the function is fixed at $D = 30$ and $D = 50$, the maximum number of function evaluations is set to $NFEs = 10,000 \times D$, the maximum number of iterations is fixed at T=10,000, the population size is equal to $N = NFEs/T$, and the boundary $[a_i, b_i]$ of each decision variable is [-100, 100]. Experiments on each test problem are conducted 30 times with different initial populations. In all experiments, *NFEs* is taken as the termination criteria to guarantee a fair comparison.

The parameters of the algorithms, except for QOS-OFA, are as follows. For SCA, the constant value a is equal to 2. For SSA, the discrepancy between iterations is defined as 1. For WOA, the parameter for defining the shape of the logarithmic spiral is set to 1. For OFA, the scale factor k is equal to t/T. For LSHADE, the memory size H, the p^m value for mutation, and r^{arc} related to external archive size are set to 5,

Table 3 Comparison among eight algorithms on the 30D benchmark functions

Func	Criteria	GWO	MFO	OFA	SSA	WOA	SCA	LSHADE	QOS-OFA
CEC1	Mean	4.91E+09	1.75E+10	3.01E+03	2.62E+03	3.61E+07	4.32E+10	**1.00E+02**	5.57E+02
	SD	2.69E+09	2.60E+09	2.77E+03	2.66E+03	6.59E+07	4.13E+09	**0.00E+00**	4.09E+02
	Time	1.53E+00	1.14E+00	1.02E+00	1.06E+00	7.20E-01	1.19E+00	1.84E+00	1.23E+00
CEC3	Mean	3.53E+04	9.05E+04	7.79E+03	**3.00E+02**	5.82E+04	1.46E+05	2.65E+04	**3.00E+02**
	SD	1.03E+04	1.06E+04	2.11E+03	**0.00E+00**	1.69E+04	2.08E+04	4.13E+04	1.07E+00
	Time	1.55E+00	1.17E+00	1.05E+00	1.08E+00	7.40E-01	1.22E+00	1.91E+00	1.30E+00
CEC5	Mean	6.41E+02	8.11E+02	6.88E+02	6.24E+02	6.58E+02	8.96E+02	**5.32E+02**	5.44E+02
	SD	3.31E+01	2.06E+01	1.21E+01	4.18E+01	3.30E+01	1.98E+01	**7.39E+00**	1.11E+01
	Time	1.84E+00	1.45E+00	1.33E+00	1.38E+00	1.03E+00	1.51E+00	2.11E+00	1.55E+00
CEC6	Mean	6.12E+02	6.53E+02	6.01E+02	6.36E+02	6.53E+02	6.84E+02	6.00E+02	**6.00E+02**
	SD	4.89E+00	5.11E+00	1.80E-01	1.34E+01	5.93E+00	5.60E+00	9.70E-01	**1.60E-01**
	Time	3.01E+00	2.50E+00	2.46E+00	7.97E+01	2.02E+00	2.63E+00	2.88E+00	2.81E+00
CEC8	Mean	9.16E+02	1.10E+03	9.75E+02	9.31E+02	9.87E+02	1.16E+03	**8.33E+02**	8.45E+02
	SD	2.40E+01	1.63E+01	1.19E+01	2.96E+01	2.87E+01	2.09E+01	**7.12E+00**	1.27E+01
	Time	2.23E+00	1.79E+00	1.70E+00	1.68E+00	1.27E+00	1.86E+00	2.72E+00	1.98E+00
CEC9	Mean	2.17E+03	7.71E+03	9.22E+02	3.19E+03	5.16E+03	1.18E+04	9.57E+02	**9.00E+02**
	SD	6.80E+02	9.02E+02	2.52E+01	1.27E+03	1.72E+03	1.07E+03	4.84E+01	**3.10E-01**
	Time	2.34E+00	1.87E+00	1.71E+00	1.73E+00	1.32E+00	1.94E+00	2.70E+00	2.03E+00
CEC11	Mean	2.35E+03	4.86E+03	1.17E+03	1.23E+03	2.10E+03	7.04E+03	1.25E+03	**1.13E+03**
	SD	9.32E+02	1.06E+03	2.20E+01	4.11E+01	8.81E+02	9.29E+02	5.28E+01	**2.09E+01**
	Time	2.06E+00	1.61E+00	1.53E+00	1.50E+00	1.07E+00	1.74E+00	2.47E+00	1.78E+00
CEC13	Mean	3.85E+07	1.12E+07	2.75E+04	6.07E+04	3.12E+07	3.52E+09	**1.91E+03**	1.63E+04
	SD	1.17E+08	1.78E+07	5.31E+03	2.26E+04	9.54E+07	9.54E+08	**7.30E+02**	7.67E+03
	Time	2.38E+00	1.92E+00	1.68E+00	1.85E+00	1.29E+00	1.97E+00	2.98E+00	2.02E+00
CEC14	Mean	1.02E+05	2.38E+05	5.54E+03	2.60E+03	7.94E+05	1.87E+06	1.58E+03	**1.52E+03**
	SD	2.44E+05	1.47E+05	2.35E+03	7.91E+02	7.38E+05	7.11E+05	5.36E+01	**2.99E+01**
	Time	3.29E+00	2.73E+00	2.55E+00	2.59E+00	2.06E+00	2.79E+00	3.80E+00	2.94E+00
CEC16	Mean	2.56E+03	3.74E+03	3.01E+03	2.57E+03	3.33E+03	4.67E+03	2.11E+03	**1.96E+03**
	SD	3.15E+02	1.86E+02	**1.62E+02**	2.74E+02	3.21E+02	2.45E+02	1.92E+02	1.90E+02
	Time	2.59E+00	1.99E+00	1.88E+00	1.92E+00	1.37E+00	2.11E+00	3.11E+00	2.26E+00
CEC18	Mean	7.64E+05	4.91E+06	3.41E+05	1.11E+05	2.26E+06	2.42E+07	**2.86E+03**	1.52E+04
	SD	2.02E+06	1.80E+06	9.02E+04	8.42E+04	3.22E+06	9.70E+06	**1.80E+03**	1.05E+04
	Time	2.27E+00	1.82E+00	1.69E+00	1.71E+00	1.27E+00	1.87E+00	2.89E+00	1.99E+00
CEC20	Mean	2.36E+03	2.76E+03	2.38E+03	2.39E+03	2.67E+03	3.04E+03	2.22E+03	**2.16E+03**
	SD	9.57E+01	8.10E+01	7.66E+01	8.58E+01	1.81E+02	1.09E+02	8.91E+01	**5.21E+01**
	Time	2.90E+00	2.51E+00	2.41E+00	2.39E+00	2.03E+00	2.58E+00	3.11E+00	2.68E+00
CEC22	Mean	5.43E+03	4.26E+03	**2.30E+03**	**2.30E+03**	5.72E+03	9.93E+03	2.88E+03	**2.30E+03**
	SD	2.19E+03	2.38E+02	1.55E+00	1.42E+00	2.14E+03	2.59E+02	1.11E+03	**9.80E-01**
	Time	3.42E+00	3.04E+00	2.92E+00	2.93E+00	2.57E+00	3.12E+00	3.19E+00	3.15E+00
CEC24	Mean	3.37E+03	3.21E+03	2.87E+03	2.92E+03	3.31E+03	3.57E+03	2.89E+03	**2.87E+03**
	SD	1.64E+02	2.45E+01	1.04E+02	2.81E+01	9.14E+01	4.90E+01	1.77E+01	**1.73E+01**
	Time	4.13E+00	3.74E+00	3.68E+00	3.71E+00	3.30E+00	3.85E+00	3.88E+00	3.93E+00
CEC26	Mean	5.95E+03	7.64E+03	**2.81E+03**	3.73E+03	7.63E+03	9.55E+03	4.11E+03	3.09E+03
	SD	6.37E+02	2.58E+02	**2.61E+01**	1.12E+03	1.05E+03	4.96E+02	2.14E+02	4.56E+02
	Time	4.41E+00	4.02E+00	3.93E+00	3.90E+00	3.55E+00	4.11E+00	3.92E+00	4.14E+00

Table 3 (continued)

Func	Criteria	GWO	MFO	OFA	SSA	WOA	SCA	LSHADE	QOS-OFA
CEC27	Mean	**3.20E+03**	3.56E+03	**3.20E+03**	3.24E+03	3.70E+03	**3.20E+03**	3.22E+03	**3.20E+03**
	SD	**0.00E+00**	4.61E+01	7.25E+00	1.96E+01	1.73E+02	**0.00E+00**	1.38E+01	1.20E+01
	Time	5.00E+00	4.80E+00	4.55E+00	4.56E+00	4.19E+00	4.83E+00	4.47E+00	4.79E+00
CEC28	Mean	3.30E+03	4.23E+03	3.16E+03	3.17E+03	3.60E+03	3.30E+03	3.17E+03	**3.14E+03**
	SD	**8.00E-02**	1.53E+02	3.53E+01	5.12E+01	2.62E+02	9.00E-02	7.27E+01	4.55E+01
	Time	4.45E+00	4.04E+00	3.95E+00	3.96E+00	3.58E+00	4.08E+00	4.05E+00	4.19E+00
CEC30	Mean	3.64E+06	1.12E+07	3.22E+04	9.30E+05	2.18E+07	3.25E+08	**6.27E+03**	6.98E+03
	SD	2.50E+06	6.76E+06	2.04E+04	5.32E+05	2.50E+07	1.22E+08	2.79E+03	**1.44E+03**
	Time	7.68E+00	7.27E+00	7.16E+00	7.14E+00	6.80E+00	7.38E+00	6.22E+00	7.39E+00

0.11, and 1.4, respectively. However, for GWO and MFO, no additional parameters need to be specified.

4.2 Experimental results and analysis

4.2.1 Numerical analysis

In the experiments, the average function value (Mean), the standard deviation of the function value (SD), and the execution time (Time) are selected as the performance metrics. For the 30D functions, the results obtained by QOS-OFA and seven others are listed in Table 3. For the 50D functions, the results are shown in Table 4. In both tables, the best results of Mean and SD are highlighted in bold.

The unimodal functions (CEC1, 3) have only one global minimum, which are suitable for evaluating the local exploitation capability of the algorithm. From Table 3, QOS-OFA obtains the second-best performance in CEC1 and the global optimal solution in CEC3. From Table 4, in terms of Mean and SD metrics, QOS-OFA obtains the third-best results in CEC1, and is second only to SSA in CEC3. Compared to other algorithms, QOS-OFA exhibits better or at least competitive performance in exploiting the search region.

Since the multimodal functions (CEC 5, 6, 8, 9) have multiple local minima, the exploration capability of algorithms can be assessed. For the 30D and 50D functions, QOS-OFA outperforms other competitors in CEC 6 and 9, and is second only to LSHADE in CEC 5 and 8 in terms of Mean and SD metrics. Based on the comparative results, QOS-OFA's performance in exploring promising regions is validated.

Six hybrid functions (CEC 11, 13, 14, 16, 18, 20) and six composite functions (CEC 22, 24, 26, 27, 28, 30) are selected. The common characteristic of these functions is that each of them has many local minima. The ability to balance global exploration and local exploitation and the ability to avoid local minima can be measured by these functions. As can be seen from Tables 3 and 4, when D=30, the performance of the QOS-OFA is better than other competitors in CEC

Table 4 Comparison among eight algorithms on the 50D benchmark functions

Func	Criteria	GWO	MFO	OFA	SSA	WOA	SCA	LSHADE	QOS-OFA
CEC1	Mean	1.91E+10	6.15E+10	6.39E+02	3.12E+03	4.40E+07	1.51E+11	**1.00E+02**	1.80E+03
	SD	6.02E+09	7.21E+09	5.40E+02	3.27E+03	2.50E+07	1.25E+10	**0.00E+00**	2.12E+03
	Time	5.01E+00	3.89E+00	3.26E+00	3.97E+00	2.73E+00	4.13E+00	3.82E+00	3.63E+00
CEC3	Mean	8.81E+04	1.94E+05	8.06E+04	**3.00E+02**	7.14E+04	3.46E+05	3.38E+04	1.13E+04
	SD	1.75E+04	2.04E+04	8.42E+03	**0.00E+00**	2.33E+04	3.92E+04	7.72E+04	2.91E+03
	Time	5.09E+00	3.98E+00	3.34E+00	3.99E+00	2.79E+00	4.17E+00	3.84E+00	3.73E+00
CEC5	Mean	7.69E+02	1.13E+03	9.07E+02	7.83E+02	8.66E+02	1.31E+03	**5.57E+02**	6.06E+02
	SD	5.31E+01	2.78E+01	2.07E+01	5.20E+01	6.33E+01	3.41E+01	**1.44E+01**	2.29E+01
	Time	5.85E+00	4.71E+00	4.09E+00	4.79E+00	3.54E+00	4.91E+00	4.16E+00	4.49E+00
CEC6	Mean	6.27E+02	6.74E+02	6.03E+02	6.47E+02	6.64E+02	7.06E+02	**6.02E+02**	**6.02E+02**
	SD	5.59E+00	3.96E+00	5.80E-01	1.19E+01	5.00E+00	4.78E+00	8.60E-01	**4.10E-01**
	Time	8.52E+00	7.18E+00	6.81E+00	7.26E+00	6.09E+00	7.41E+00	5.98E+00	7.44E+00
CEC8	Mean	1.09E+03	1.44E+03	1.20E+03	1.06E+03	1.23E+03	1.64E+03	**8.54E+02**	9.07E+02
	SD	3.54E+01	2.13E+01	1.65E+01	5.94E+01	6.60E+01	2.89E+01	**1.07E+01**	1.72E+01
	Time	6.28E+00	5.02E+00	4.46E+00	5.17E+00	3.82E+00	5.36E+00	4.64E+00	4.79E+00
CEC9	Mean	1.06E+04	2.85E+04	2.91E+03	9.69E+03	1.72E+04	4.89E+04	1.20E+03	**9.07E+02**
	SD	3.90E+03	3.48E+03	8.40E+02	2.82E+03	2.88E+03	4.65E+03	1.73E+02	**4.42E+00**
	Time	6.00E+00	4.87E+00	4.21E+00	4.88E+00	3.61E+00	5.06E+00	4.30E+00	4.60E+00
CEC11	Mean	5.93E+03	1.60E+04	1.33E+03	1.31E+03	2.07E+03	3.26E+04	1.31E+03	**1.15E+03**
	SD	2.80E+03	2.77E+03	**1.57E+01**	4.89E+01	7.79E+02	3.68E+03	4.76E+01	1.89E+01
	Time	5.42E+00	4.24E+00	3.63E+00	4.36E+00	3.07E+00	4.48E+00	3.82E+00	4.03E+00
CEC13	Mean	5.69E+08	1.06E+09	2.29E+03	1.14E+05	3.98E+05	2.92E+10	2.96E+03	**2.15E+03**
	SD	1.31E+09	4.22E+08	8.22E+02	6.18E+04	6.20E+05	2.29E+09	2.94E+03	**8.21E+02**
	Time	5.53E+00	4.30E+00	3.73E+00	4.47E+00	3.16E+00	4.53E+00	4.19E+00	4.07E+00
CEC14	Mean	1.20E+06	3.35E+06	5.26E+04	3.29E+04	1.50E+06	9.38E+06	**1.70E+03**	2.06E+03
	SD	3.73E+06	1.14E+06	1.80E+04	2.23E+04	1.47E+06	3.23E+06	**7.52E+01**	6.35E+02
	Time	6.82E+00	5.55E+00	4.85E+00	5.62E+00	4.31E+00	5.79E+00	4.86E+00	5.29E+00
CEC16	Mean	3.31E+03	5.90E+03	4.35E+03	3.09E+03	5.07E+03	8.18E+03	2.52E+03	**2.40E+03**
	SD	5.47E+02	2.64E+02	2.17E+02	3.94E+02	7.88E+02	3.33E+02	2.25E+02	**2.04E+02**
	Time	6.14E+00	4.92E+00	4.32E+00	5.09E+00	3.66E+00	5.24E+00	4.54E+00	4.77E+00
CEC18	Mean	5.53E+06	3.08E+07	1.08E+06	1.98E+05	1.24E+07	5.22E+07	**3.03E+03**	3.64E+04
	SD	8.19E+06	1.07E+07	4.00E+05	8.14E+04	1.21E+07	1.56E+07	**1.31E+03**	1.32E+04
	Time	5.60E+00	4.44E+00	3.85E+00	4.52E+00	3.31E+00	4.65E+00	4.14E+00	4.23E+00
CEC20	Mean	2.76E+03	3.89E+03	3.32E+03	3.00E+03	3.27E+03	3.99E+03	**2.38E+03**	2.55E+03
	SD	2.21E+02	1.36E+02	1.53E+02	3.02E+02	3.26E+02	1.96E+02	**1.36E+02**	1.90E+02
	Time	8.29E+00	7.23E+00	6.49E+00	7.25E+00	5.96E+00	7.42E+00	5.99E+00	6.97E+00
CEC22	Mean	1.13E+04	1.55E+04	2.68E+03	8.78E+03	1.17E+04	1.65E+04	6.58E+03	**2.30E+03**
	SD	3.06E+03	1.21E+03	1.38E+03	2.37E+03	1.24E+03	4.12E+02	8.64E+02	**1.53E+00**
	Time	1.33E+01	1.22E+01	1.14E+01	1.21E+01	1.08E+01	1.26E+01	9.40E+00	1.17E+01
CEC24	Mean	4.21E+03	3.81E+03	3.23E+03	3.15E+03	4.04E+03	4.42E+03	3.06E+03	**3.01E+03**
	SD	1.13E+02	6.07E+01	1.66E+01	5.90E+01	1.93E+02	6.78E+01	**1.21E+01**	2.27E+01
	Time	1.62E+01	1.49E+01	1.51E+01	1.49E+01	1.35E+01	1.51E+01	1.19E+01	1.49E+01
CEC26	Mean	9.95E+03	1.27E+04	**2.98E+03**	3.50E+03	1.26E+04	1.59E+04	4.45E+03	4.78E+03
	SD	1.50E+03	6.29E+02	2.17E+02	1.56E+03	1.05E+03	1.39E+03	**1.92E+02**	**1.92E+02**
	Time	2.01E+01	1.95E+01	1.79E+01	1.88E+01	1.83E+01	1.98E+01	1.44E+01	1.86E+01

Table 4 (continued)

Func	Criteria	GWO	MFO	OFA	SSA	WOA	SCA	LSHADE	QOS-OFA
CEC27	Mean	**3.20E+03**	4.87E+03	3.23E+03	3.43E+03	5.38E+03	**3.20E+03**	3.36E+03	3.23E+03
	SD	**0.00E+00**	1.65E+02	2.60E+01	6.72E+01	4.83E+02	**0.00E+00**	6.70E+01	1.94E+01
	Time	2.44E+01	2.25E+01	2.21E+01	5.57E+01	2.13E+01	2.23E+01	1.71E+01	2.25E+01
CEC28	Mean	3.30E+03	8.58E+03	3.28E+03	3.31E+03	4.38E+03	3.30E+03	3.29E+03	**3.27E+03**
	SD	4.00E-02	5.64E+02	1.64E+01	2.55E+01	4.25E+02	**2.00E-02**	2.38E+01	1.86E+01
	Time	1.60E+01	1.48E+01	1.42E+01	1.50E+01	1.37E+01	1.50E+01	1.10E+01	1.46E+01
CEC30	Mean	4.71E+07	4.33E+08	9.07E+05	2.15E+07	2.65E+08	4.30E+09	7.64E+05	**7.21E+05**
	SD	2.22E+07	1.11E+08	2.27E+05	3.30E+06	7.59E+07	9.79E+08	1.25E+05	**4.72E+04**
	Time	2.27E+01	2.16E+01	2.10E+01	2.17E+01	2.04E+01	2.20E+01	1.55E+01	2.15E+01

11, 14, 16, 20, 22, 24, and 28. When D=50, QOS-OFA outperforms other methods in terms of Mean metrics in CEC 11, 13, 16, 22, 24, 28, and 30. Thus, the explorative and exploitative performance of QOS-OFA is validated by comparing it with other MAs.

Among the compared algorithms, QOS-OFA has the third longest execution time on most of the 30D functions and the fifth longest on most of the 50D functions. The application of the new strategy requires a little more time than the standard OFA. Combining the Mean and SD metrics, QOS-OFA achieves a tradeoff in terms of solution accuracy and computational complexity.

4.2.2 Statistical analysis

To make the optimization results more convincing, two well-known nonparametric statistical methods-Friedman test and Wilcoxon signed rank test- are adopted to check whether there exist significant differences among the algorithms.

The average function value obtained by different algorithms is considered for the Friedman test. In this method, the ranking order of the algorithm for a specified function is found, and then the average ranking value is calculated to obtain the final ranking of each algorithm for the test suite. The results of the Friedman test are shown in Fig. 5. The QOS-OFA is ranked first among the eight algorithms for the 30D and 50D functions.

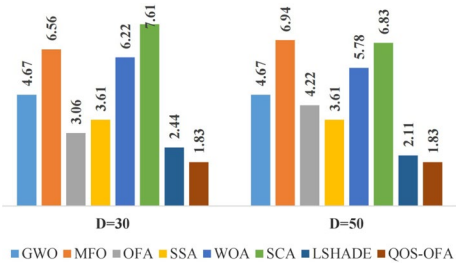


Fig. 5 Histogram of Friedman test under D=30/50

Table 5 The results of Wilcoxon signed rank test at a 0.05 significance level (D=30)

QOS-OFA vs	GWO	MFO	OFA	SSA	WOA	SCA	LSHADE
Func	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)
CEC1	3.02E-11(+)	3.02E-11(+)	4.64E-05(+)	5.56E-04(+)	5.56E-04(+)	3.02E-11(+)	2.67E-11(-)
CEC3	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(-)	3.02E-11(+)	3.02E-11(+)	7.95E-03(+)
CEC5	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	6.70E-11(+)	3.02E-11(+)	3.02E-11(+)	2.28E-05(-)
CEC6	3.02E-11(+)	3.02E-11(+)	4.50E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	2.62E-03(+)
CEC8	6.07E-11(+)	3.02E-11(+)	3.02E-11(+)	3.34E-11(+)	3.02E-11(+)	3.02E-11(+)	4.08E-05(-)
CEC9	3.02E-11(+)	3.02E-11(+)	2.87E-10(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)
CEC11	3.02E-11(+)	3.02E-11(+)	1.87E-07(+)	1.96E-10(+)	3.02E-11(+)	3.02E-11(+)	8.15E-11(+)
CEC13	1.09E-10(+)	3.02E-11(+)	7.60E-07(+)	9.92E-11(+)	4.98E-11(+)	3.02E-11(+)	4.08E-11(-)
CEC14	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	2.15E-06(+)
CEC16	1.41E-09(+)	3.02E-11(+)	3.02E-11(+)	1.69E-09(+)	3.02E-11(+)	3.02E-11(+)	5.83E-03(+)
CEC18	8.15E-11(+)	3.02E-11(+)	3.02E-11(+)	4.98E-11(+)	3.02E-11(+)	3.02E-11(+)	1.46E-10(-)
CEC20	1.33E-10(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	4.86E-03(+)
CEC22	3.02E-11(+)	3.02E-11(+)	2.78E-07(+)	1.68E-04(+)	3.02E-11(+)	3.02E-11(+)	6.63E-01(=)
CEC24	3.02E-11(+)	3.02E-11(+)	1.84E-02(+)	1.55E-09(+)	3.02E-11(+)	3.02E-11(+)	2.13E-04(+)
CEC26	3.02E-11(+)	3.02E-11(+)	5.59E-02(-)	1.76E-01(=)	3.02E-11(+)	3.02E-11(+)	2.44E-09(+)
CEC27	6.63E-01(=)	3.02E-11(+)	4.12E-01(=)	4.07E-11(+)	3.02E-11(+)	6.63E-01(=)	2.02E-08(+)
CEC28	3.02E-11(+)	3.02E-11(+)	1.22E-02(+)	2.07E-02(+)	3.02E-11(+)	3.02E-11(+)	6.20E-01(=)
CEC30	3.02E-11(+)	3.02E-11(+)	2.37E-10(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	2.43E-05(-)

Table 6 The results of Wilcoxon signed rank test at a 0.05 significance level (D=50)

QOS-OFA vs	GWO	MFO	OFA	SSA	WOA	SCA	LSHADE
Func	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)	*p*-value(sign)
CEC1	3.02E-11(+)	3.02E-11(+)	2.46E-01(=)	5.75E-02(=)	3.02E-11(+)	3.02E-11(+)	3.02E-11(-)
CEC3	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(-)	3.02E-11(+)	3.02E-11(+)	9.51E-06(+)
CEC5	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	4.62E-10(-)
CEC6	3.02E-11(+)	3.02E-11(+)	1.86E-09(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	7.22E-06(+)
CEC8	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.34E-11(-)
CEC9	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)
CEC11	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)
CEC13	3.02E-11(+)	3.02E-11(+)	2.17E-02(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.27E-02(+)
CEC14	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	5.19E-07(-)
CEC16	2.23E-09(+)	3.02E-11(+)	3.02E-11(+)	9.26E-09(+)	3.02E-11(+)	3.02E-11(+)	1.02E-01(=)
CEC18	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.69E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(-)
CEC20	3.37E-04(+)	3.02E-11(+)	3.34E-11(+)	3.52E-07(+)	1.86E-09(+)	3.02E-11(+)	6.91E-04(-)
CEC22	3.02E-11(+)	3.02E-11(+)	3.08E-08(+)	1.29E-09(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)
CEC24	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	9.92E-11(+)
CEC26	3.02E-11(+)	3.02E-11(+)	3.02E-11(-)	1.11E-06(-)	3.02E-11(+)	3.02E-11(+)	4.31E-08(-)
CEC27	3.02E-11(-)	3.02E-11(+)	5.89E-01(=)	3.02E-11(+)	3.02E-11(+)	3.02E-11(-)	9.92E-11(+)
CEC28	1.11E-06(+)	3.02E-11(+)	2.84E-04(+)	2.57E-07(+)	3.02E-11(+)	1.11E-06(+)	2.71E-01(=)
CEC30	3.02E-11(+)	3.02E-11(+)	2.01E-01(=)	3.02E-11(+)	3.02E-11(+)	3.02E-11(+)	2.27E-03(+)

Moreover, the Wilcoxon signed rank test is applied to compare the performance of QOS-OFA with its competitors. For each algorithm, the optimal function value obtained from 30 runs of each function is handled as a group. Wilcoxon signed rank test is conducted to detect significant differences between two sample groups at the 0.05 significance level. The results are reported in Tables 5 and 6. The criteria for drawing conclusions are as follows: 1) if the p-value is less than 0.05, the observed difference is significant; 2) if the p-value equals 0.05, both the algorithms are statistically the same; 3) if the p-value is bigger than 0.05, the observed difference is not significant.

Based on the above criteria, the signs '+', '=', '-' in Tables 5 and 6 represent that the QOS-OFA is better than, similar to, and worse than its competitor, respectively. It is observed that QOS-OFA is significantly superior to the OFA in 16 out of 18 functions for the 30D functions, and 14 out of 18 functions for the 50D functions. Compared with other algorithms, QOS-OFA shows significant or at least competitive performance in most of the test functions.

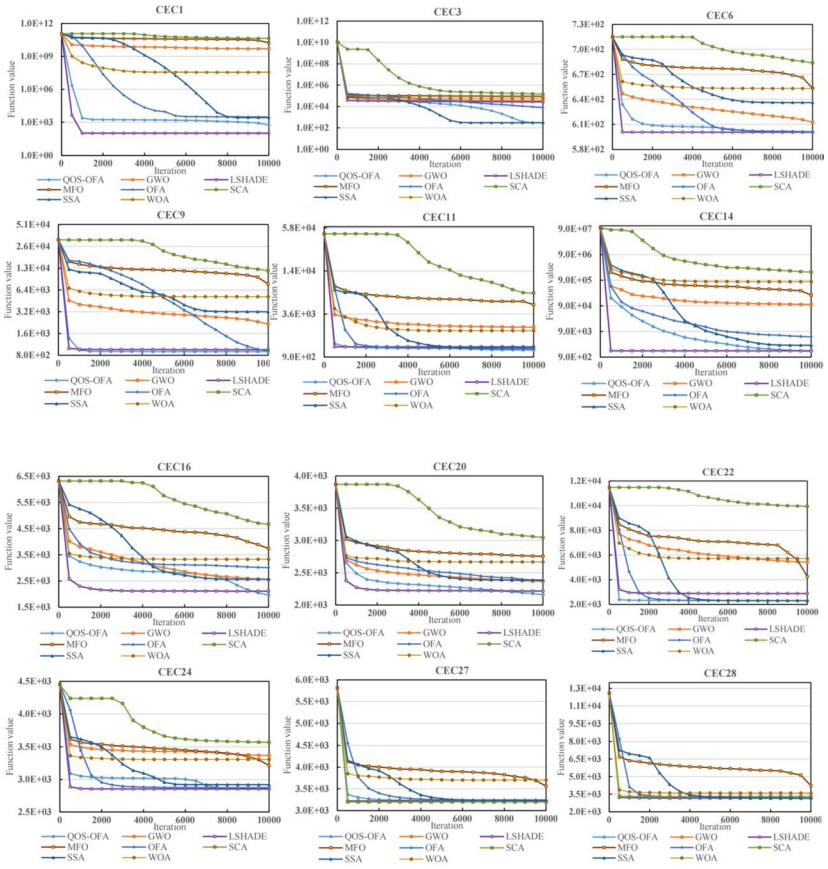


Fig. 6 Convergence graphs of 30D benchmark functions

4.2.3 Convergence analysis

For different algorithms, the average function values of 30 runs in different iterations for the 30D functions are depicted in Fig. 6. In these graphs, the vertical axis represents the objective function value and the horizontal axis indicates the number of iterations. From Fig. 6, the proposed QOS-OFA shows a faster convergence speed than the standard OFA in most cases. This is because the quality of the initial population is increased by the QOBL method and a larger cosine-based scale factor accelerates the convergence speed in the exploration stage. It is concluded that the proposed QOS-OFA exhibits better or at least competitive convergence performance among the compared algorithms.

4.2.4 Typical case analysis

Typical case analysis is an essential part of analyzing the QOS-OFA's social behavior. As shown in Table 3, when D=30, QOS-OFA achieves the global optimal solution in CEC3, 6, and 9. The 3D map and contour map of these functions are illustrated in Fig. 7, where the global optima are symbolized by the sign '+'. The search histories of QOS-OFA and OFA are depicted in the same figure, where the first variable and the second variable are extracted from the 30D solutions. It can be observed that QOS-OFA's solutions converge faster than the standard OFA in the early stage. This is because the designed scale factor $K(t)$ accelerates the exploration of the search space. In the later stage, the solution generated by OFA cannot achieve the global optimum, while the solution generated by the proposed QOS-OFA algorithm can achieve the global optimum. This is because social information is used to guide other solutions to exploit the promising regions around the current best solution. Thus, the global optimal solution can be found by the QOS-OFA algorithm.

5 Applications of QOS-OFA

To analyze the applicability of the proposed algorithm, three unconstrained real-world optimization problems are solved, including the gear train problem, the parameter estimation for frequency-modulated problem, and the drilling path optimization problem.

5.1 Gear train

As shown in Fig. 8, the gear train problem is considered as a discrete optimization problem because all its decision variables are constrained to be integers. The four variables associated with this problem denote the number of teeth of the four gears ($n_A = x_1, n_B = x_2, n_D = x_3, n_F = x_4$). The objective of this problem is to

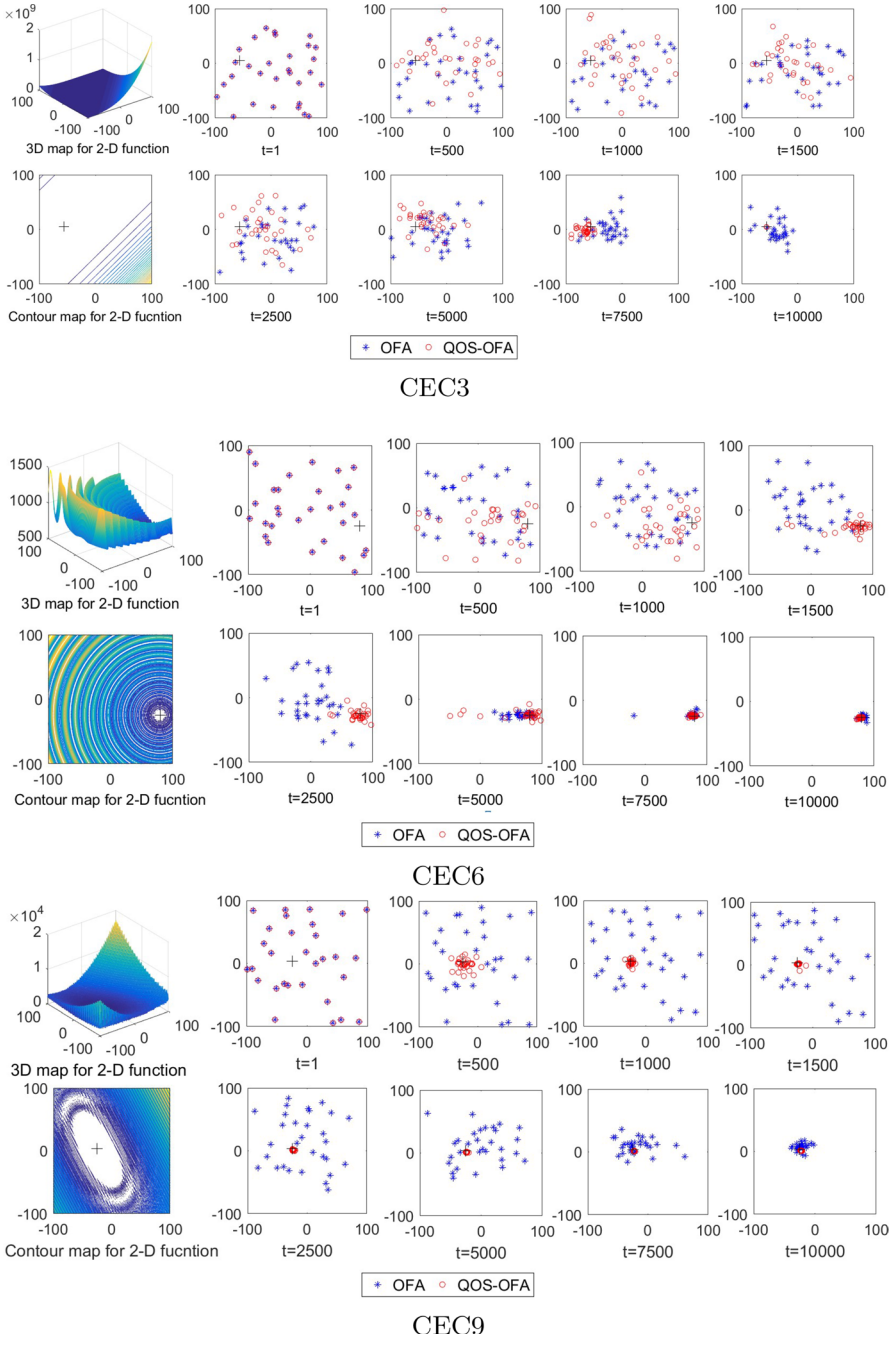


Fig. 7 Search history of QOS-OFA and OFA

Fig. 8 Gear train

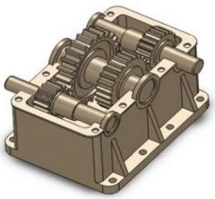


Table 7 Comparison among seven algorithms on gear train problem

Algorithm	Optimal decision variables				Fitness value(min)	*NFEs*
	x_1	x_2	x_3	x_4		
GSA-GA	43	16	19	49	**2.7009E-12**	NA
ITGO	43	16	19	49	**2.7009E-12**	773
RW-GWO	43	16	19	49	**2.7009E-12**	**200**
m-SCA	43	16	19	49	**2.7009E-12**	1270
IAPSO	43	16	19	49	**2.7009E-12**	400
OFA	46	12	26	47	9.9216E-10	360
QOS-OFA	43	16	19	49	**2.7009E-12**	360

Better results are highlighted in bold, NA indicates that the data is not available

minimize the cost of the gear ratio, namely, $(n_B n_D)/(n_F n_A)$. Mathematically, the explicit form of this problem is formulated as follows.

$$Min : f_1(x) = \left(\frac{1}{6.931} - \frac{x_2 x_3}{x_1 x_4} \right)^2, s.t. 12 \leq x_i \leq 60, i = 1, \ldots, 4.$$

In the literature, a hybrid of GSA and GA (GSA-GA) [52], iterative topographical global optimization (ITGO) [53], random walk GWO (RW-GWO) [54], modified SCA (m-SCA) [4], and improved acceleration PSO (IAPSO) [55], have been designed to solve this problem. The best solution obtained by QOS-OFA and other methods are reported in Table 7. As shown, all algorithms except OFA achieve the same optimal solution. In terms of *NFEs*, QOS-OFA requires less computational cost than other competitors except RW-GWO. Therefore, QOS-OFA shows better or at least competitive performance in solving this problem.

5.2 Parameter estimation for frequency-modulated (FM)

FM sound wave synthesis occupies a crucial place in the modern music system. The objective of this problem is to estimate parameters for FM synthesizer and generate a sound signal that is similar to the target sound wave. It is a highly complex multimodal problem with strong epistasis. The parameters that need to be optimized are

Table 8 Comparison among seven algorithms on parameter estimation for frequency-modulated

Algorithm	Mean	Max	Min	SD	NFEs
m-SCA	1.541E+01	2.065E+01	3.86E+00	3.38E+00	NA
RW-GWO	8.87E+00	1.88E+01	6.00E-03	6.95E+00	30,000
ICSA	1.84E+01	2.37E+01	1.13E+01	4.07E+00	NA
ImCSA	2.33E+01	2.64E+01	1.49E+01	2.91E+00	NA
HHO	2.01E+01	2.51E+01	1.17E+01	4.54E+00	NA
OFA	1.61E+01	2.22E+01	1.14E+01	3.06E+00	30,000
QOS-OFA	**3.14E+00**	**1.33E+01**	**4.43E-17**	**5.18E+00**	30,000

Better results are highlighted in bold, NA indicates that the data is not available

denoted as $x = (a_1, w_1, a_2, w_2, a_3, w_3)$. The mathematical expression of this problem is given as follows.

$$Min : f_2(t) = \sum_{t=0}^{100} (y(t) - y_0(t))^2, s.t. -6.4 \le a_1, w_1, a_2, w_2, a_3, w_3 \le 6.35$$

where the estimated sound $y(t)$ and target sound $y_0(t)$ is represented by the following form.

$$y(t) = a_1 \times \sin(w_1 \times \theta \times t + a_2 \sin(w_2 \times \theta \times t + a_3 \sin(w_3 \times \theta \times t))), \theta = 2\pi/100$$
$$y_0(t) = \sin(5 \times \theta \times t - 1.5 \sin(4.8 \times \theta \times t + 2sin(4.9 \times \theta \times t)))$$

The optimization results obtained by QOS-OFA and OFA are listed in Table 8. This problem has been optimized with an intelligent crow search algorithm (ICSA) [56], an improved crow search algorithm (ImCSA) [57], HHO [33], m-SCA [4], and RW-GWO [54]. The results obtained by these algorithms are presented in the same table. It is obvious from Table 8 that QOS-OFA outperforms other competitors in terms of numerical results. The superiority of the QOS-OFA is verified based on the comparison results.

5.3 Drilling path optimization

Drilling path optimization plays an important role in modern manufacturing. It can be regarded as a discrete optimization problem. In this problem, a considerable part

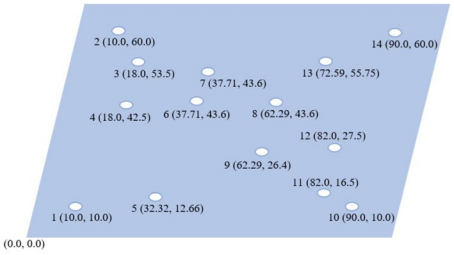


Fig. 9 14-holes workpiece

Table 9 Comparison among six algorithms on drilling path optimization problem

Algorithm	Mean	Max	Min	SD	NFEs
ACO	284.22	**285.9**	284	**0.58**	400,000
GAs	306.1	NA	**280.00**	29.5	400,000
PSO	511.42	573.69	450.90	33.75	30,000
DE	296.89	311.22	281.04	6.95	30,000
OFA	300.79	324.27	281.04	11.03	30000
QOS-OFA	**281.77**	290.38	**280.00**	2.55	30000

Better results are highlighted in bold, NA indicates that the data is not available

of the machining time is spent on moving the worktable from one position to another due to the point-to-point tool movement. Hence, the objective of this problem is to find an optimal sequence $\boldsymbol{sq}$ that minimizes the total machining cost. A typical 14-hole workpiece is depicted in Fig. 9. The best path for the points to be drilled should be identified. The objective function of this problem is given as follows.

$$Min : f_3 = \sum_{i=1}^{13}(|X(\boldsymbol{sq}(i)) - X(\boldsymbol{sq}(i+1)| + |Y(\boldsymbol{sq}(i)) - Y(\boldsymbol{sq}(i+1))|)$$

where X and Y are the x-ordinate value and y-ordinate value of the points shown in Fig. 9, respectively.

The drilling path optimization problem has been studied in the literature using various algorithms including ACO [58], GAs [59], PSO [60], and DE [61]. The optimization results of these methods are presented in Table 9. It can be observed that QOS-OFA obtains the smallest mean function value among all the methods. In addition, the proposed QOS-OFA achieves the same minimum function value as GAs, but with lower computational cost. It is concluded that QOS-OFA shows better performance than other competitors in solving the drilling path optimization problem.

6 Conclusion

In this paper, QOS-OFA is proposed for solving GOPs. In QOS-OFA, QOBL is introduced to find potential solutions to improve the overall quality of the initial population. In the position update phase, an efficient cosine-based scale factor is used to accelerate the convergence speed. The search strategy is designed by introducing social behavior to enhance local exploitation. Various types of GOPs have been taken to validate the effectiveness of QOS-OFA. Seven algorithms, OFA, GWO, MFO, SSA, SCA, WOA, and LSHADE, are used in the experiments to compare their performance with QOS-OFA. The experimental results show that QOS-OFA is very competitive with other algorithms in terms of numerical analysis, statistical analysis, convergence analysis, and typical case analysis. Moreover, QOS-OFA is tested for its efficiency on real-world optimization problems by solving the gear

train problem, the parameter estimation for frequency-modulated problem, and the drilling path optimization problem. QOS-OFA can provide better results for these problems compared to other algorithms.

However, some limitations must be acknowledged. For example, global exploration and local exploitation are controlled entirely by a cosine-based scale factor, which lacks flexibility. If we can propose a criterion that adaptively determines whether to explore or exploit based on the current population quality during the search process, the efficiency of QOS-OFA can be further improved. In future work, QOS-OFA will be improved to ensure its effectiveness in solving other problems such as constrained optimization problems, many-objective optimization problems, and scheduling optimization problems.

Author Contributions All authors contributed to this study. Conceptualization, Software, Investigation, Writing - original draft, Formal analysis, Resources were performed by Chen Ding. Funding acquisition, Project administration, Writing - review and editing were performed by Guangyu Zhu. All authors read and approved the final manuscript.

Funding This work was supported by the Intelligent Manufacturing Integrated Standardization and New Model Application Project in 2016 of MIIT, under Grant (2016) 213 and the Natural Science Foundation of Fujian Province, under Grant 2023J01256.

Data Availability All data generated or analysed during this study are included in this article.

Declarations

Conflict of interest The authors have no relevant financial or non-financial Conflict of interest to disclose.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Li C, Liang K, Chen Y, Pan M (2023) An exploitation-boosted sine cosine algorithm for global optimization. Eng Appl Artif Intel 117:105620
2. Juan C, Norberto H, Fredy S, Valeria V, Joselito M, Pedro L (2022) A majority-minority cellular automata algorithm for global optimization. Expert Syst Appl 203:117379
3. Sun G, Han R, Deng L, Li C, Yang G (2023) Hierarchical structure-based joint operations algorithm for global optimization. Swarm Evol Comput 79:101311
4. Gupta S, Deep K (2019) A hybrid self-adaptive sine cosine algorithm with opposition based learning. Expert Syst Appl 119:210–230
5. Beirami A, Vahidinasab V, Shafie-khah M, Catalão J (2020) Multiobjective ray optimization algorithm as a solution strategy for solving non-convex problems: a power generation scheduling case study. Int J Electric Power 119:105967
6. Formato RA (2007) Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog Electromag Res 77:425–491
7. Erol OK, Eksin I (2006) A new optimization method: Big bang-big crunch. Adv Eng Softw 37(2):106–111
8. Kaveh A, Khayatazad M (2012) A new meta-heuristic method: ray optimization. Comput Struct 112–113:283–294
9. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248

10. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110:151–166
11. Tahani M, Babayan N (2019) Flow regime algorithm (FRA): a physics-based metaheuristics algorithm. Knowl Inf Syst 60(2):1001–1038
12. Kaveh A (2014) Advances in metaheuristic algorithms for optimal design of structures. Springer, Cham
13. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
14. Kashan AH (2015) A new metaheuristic for optimization: optics inspired optimization (OIO). Comput Oper Res 55:99–125
15. Guan B, Zhao Y, Yin Y, Li Y (2021) A differential evolution based feature combination selection algorithm for high-dimensional data. Inform Sci 547:870–886
16. Li T, Yin Y, Yang B, Hou J, Zhou K (2022) A self-learning bee colony and genetic algorithm hybrid for cloud manufacturing services. Computing 104(9):1–27
17. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3:95–99
18. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
19. Koza JR (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4:87–112
20. Michalewicz Z (1996) Evolution strategies and other methods. Genetic Algorithms+Data Structures= Evolution Programs. Springer, Cham, pp 159–177
21. Tanabe R, Fukunaga A (2013) Success-history based parameter adaptation for differential evolution. In: IEEE Congress on Evolutionary Computation (CEC) (pp. 71–78)
22. Tanabe R, Fukunaga A (2014) Improving the search performance of shade using linear population size reduction. In: IEEE Congress on Evolutionary Computation (CEC), (pp. 1658–1665)
23. Wei B, Wang X, Xia X, Jiang M, Ding Z, Huang Y (2021) Novel self-adjusted particle swarm optimization algorithm for feature selection. Computing 103(8):1569–1597
24. Sanjoy C, Sushmita S, Apu K, Sandip C (2021) SHADE-WOA: a metaheuristic algorithm for global optimization. Appl Soft Comput 113:107866
25. Ma C, Huang H, Fan Q, Wei J, Du Y, Gao W (2022) Grey wolf optimizer based on Aquila exploration method. Expert Syst Appl 205:117629
26. Zeedan M, Attiya G, El-Fishawy N (2023) Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing. Computing 105(1):217–247
27. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. Micro Machine and Human Science, In: Proceedings of the Sixth International Symposium, (pp. 39–43)
28. Han H, Bai X, Hou Y, Qiao J (2023) Multitask particle swarm optimization with heterogeneous domain adaptation. IEEE Trans Evol Comput 28:178–192
29. Li W (2023) A cooperative particle swarm optimization with difference learning. Inform Sci 643:119238
30. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
31. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471
32. Khishe M, Mosavi M (2020) Chimp optimization algorithm. Expert Syst Appl 149:113338
33. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comp Syst 97:849–872
34. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
35. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
36. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
37. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249
38. Eusuff M, Lansey K, Pasha F (2006) Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Eng Optim 38(2):129–154
39. Brabazon A, McGarraghy S (2018) Foraging-inspired optimisation algorithms. Springer, Berlin
40. Zhu G, Zhang W (2017) Optimal foraging algorithm for global optimization. Appl Soft Comput 51:294–313

41. Zhang W, Zhu G (2018) Drilling path optimization by optimal foraging algorithm. IEEE Trans Ind Inform 14(7):2847–2856
42. Sayed G, Soliman M, Hassanien A (2018) Modified optimal foraging algorithm for parameters optimization of support vector machine. In: The International Conference on Advanced Machine Learning Technologies and Applications,, vol. 723, (pp. 23–32)
43. Wang H, Zhu G (2023) Multiobjective optimization for FJSP under immediate predecessor constraints based OFA and pythagorean fuzzy set. IEEE Trans Fuzzy Syst 31:3108–30120
44. Ismail S, Mona S, Ella H (2019) A novel chaotic optimal foraging algorithm for unconstrained and constrained problems and its application in white blood cell segmentation. Neural Comput Appl 31(11):1–32
45. Tizhoosh H (2005) Opposition-based learning: a new scheme for machine intelligence. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06) Vienna, Austria
46. Ventresca M, Rahnamayan S, Tizhoosh H (2008) Opposition versus randomness in soft computing techniques. Appl Soft Comput 10(3):956–957
47. Yu X, Xu W, Li C (2021) Opposition-based learning grey wolf optimizer for global optimization. Knowl-Based Syst 226:107139
48. Gupta S, Deep K, Heidari AA, Moayedi H, Wang M (2020) Opposition-based learning Harris hawks optimization with advanced transition rules: principles and analysis. Expert Syst Appl 158:113510
49. Mohammad K (2023) Greedy opposition-based learning for chimp optimization algorithm. Artif Intell Rev 56(8):7633–7663
50. Bilal HA, David P, Rafat H (2022) Improved Salp swarm algorithm for solving single-objective continuous optimization problems. Appl Intell 52(15):17217–17236
51. Rahnamayan S, Tizhoosh H, Salama M (2007) Quasi-oppositional differential evolution. In: IEEE Congress on Evolutionary Computation, Singapore
52. Garg H (2019) A hybrid GSA-GA algorithm for constrained optimization problems. Inform Sci 478:499–523
53. Ferreira M, Rocha M, Silva Neto A, Sacco W (2018) A constrained ITGO heuristic applied to engineering optimization. Expert Syst Appl 110:106–124
54. Gupta S, Deep K (2018) A novel random walk grey wolf optimizer. Swarm Evol Comput 44:101–112
55. Guedria N (2016) Improved accelerated PSO algorithm for mechanical engineering optimization problems. Appl Soft Comput 40:455–467
56. Shalini S, Akash S (2020) Development and applications of an intelligent crow search algorithm based on opposition based learning. ISA Trans 99:210–230
57. Díaz P, Pérez C, Erik C, Omar A, Gálvez J, Salvador H (2018) An improved crow search algorithm applied to energy problems. Energies 11(3):571
58. Yarpiz (2023) Ant Colony Optimization (ACO) Available: https://www.mathworks.com/matlabcentral/fileexchange/52859-ant-colony-optimization-aco
59. Roberge V, Tarbouchi M, Labonte G (2013) Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Trans Ind Inform 9(1):132–141
60. Dong W, Zhou M (2017) A supervised learning and control method to improve particle swarm optimization algorithms. IEEE Trans Syst, Man Cybern Syst 99:1–14
61. Elsayed S, Sarker RA, Essam DL (2013) An improved self-adaptive differential evolution algorithm for optimization problems. IEEE Trans Ind Inform 9(1):89–99