Mathematical Biosciences and Engineering

MBE, 20(9): 17242–17271.
DOI: 10.3934/mbe.2023768
Received: 12 June 2023
Revised: 19 July 2023
Accepted: 02 August 2023
Published: 04 September 2023

http://www.aimspress.com/journal/MBE

Research article

A chaos-based adaptive equilibrium optimizer algorithm for solving global optimization problems

Yuting Liu[1], Hongwei Ding[1,*], Zongshan Wang[1,*], Gushen Jin[2], Bo Li[1], Zhijun Yang[1] and Gaurav Dhiman[3,4,5]

[1] School of Information Science and Engineering, Yunnan University, Kunming, China
[2] Glasgow College, University of Electronic Science and Technology of China, Chengdu, China
[3] Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon
[4] University Centre for Research and Development, Department of Computer Science and Engineering, Chandigarh University, Mohali, India
[5] Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, India

* **Correspondence:** Email: hwding@ynu.edu.cn, zswang@mail.ynu.edu.cn.

Abstract: The equilibrium optimizer (EO) algorithm is a newly developed physics-based optimization algorithm, which inspired by a mixed dynamic mass balance equation on a controlled fixed volume. The EO algorithm has a number of strengths, such as simple structure, easy implementation, few parameters and its effectiveness has been demonstrated on numerical optimization problems. However, the canonical EO still presents some drawbacks, such as poor balance between exploration and exploitation operation, tendency to get stuck in local optima and low convergence accuracy. To tackle these limitations, this paper proposes a new EO-based approach with an adaptive gbest-guided search mechanism and a chaos mechanism (called a chaos-based adaptive equilibrium optimizer algorithm (ACEO)). Firstly, an adaptive gbest-guided mechanism is injected to enrich the population diversity and expand the search range. Next, the chaos mechanism is incorporated to enable the algorithm to escape from the local optima. The effectiveness of the developed ACEO is demonstrated on 23 classical benchmark functions, and compared with the canonical EO, EO variants and other frontier metaheuristic approaches. The experimental results reveal that the developed ACEO method remarkably outperforms the canonical EO and other competitors. In addition, ACEO is implemented to solve a mobile robot path planning (MRPP) task, and compared with other typical metaheuristic techniques. The comparison indicates that ACEO beats its competitors, and the ACEO algorithm can provide high-quality feasible solutions for MRPP.

Keywords: metaheuristic algorithms; equilibrium optimizer; chaos; global optimization; robot path planning

1. Introduction

Real-world design and decision problems can be considered as global optimization problems consisting of different types of objective functions [1]. The objective functions can be classified as continuous or discrete, single or multi-objective, constrained or unconstrained, depending on their characteristics. Numerous real-world multi-modal and non-linear optimization problems are complex, such as parameter calibration, structural design and optimization of neural networks [2]. Without having any gradient information of the objective function, it is challenging to find global or near-global best solutions for solving real-world problems [3]. Traditional mathematical optimization methods are time-consuming and ineffective when faced with these complex problems. Consequently, in order to get global best solutions of complex real-world problems, metaheuristic approaches are continuously developed, which are the idea of finding the best solution based on intuitive, empirical or simulation of some natural phenomena and mechanistic constructs [4]. Furthermore, metaheuristic algorithms do not require gradient information, consider only inputs and outputs and have been a great development. Well-performing algorithms provide a tool for researchers to address optimization problems in different fields [5]. Moreover, metaheuristics are highly flexible and do not need specific adaptation of the algorithm according to the type of problem, thus they are becoming increasingly popular and have been successfully adopted for complex optimization problems in various domains [6].

The equilibrium optimizer (EO) algorithm is a novel physics-based metaheuristic technique proposed by Faramarzi et al. [7] in 2020. The algorithm is inspired by a hybrid dynamic mass equilibrium differential equation on a controlled fixed volume, which describes the elementary physical phenomenon of the conservation of a mass during the entrance, departure and generation within a controlled volume. The principle is to consider each particle and its concentration as an independent individual and, after that, update the individuals stochastically in accordance with the concentration of the equilibrium candidate, finally reaching the equilibrium state. Compared with other intelligent optimization algorithms, EO has several merits, including simple framework, ease of implementation, strong adaptability, few parameters and ease of hybridizing with other algorithms. Compared with the genetic algorithm (GA) [8], the particle swarm algorithm (PSO) [9], the grey wolf optimization algorithm (GWO) [10], the gravitational search algorithm (GSA) [11], the salp swarm algorithm (SSA) [12] and the covariance matrix adaptation evolution strategy (CMA-ES) [13], EO has been proven to perform exceptionally well.

Although EO is superior to other popular methods, it still has some defects such as a great tendency to fall into local optima, slow convergence and immature balance between exploration and exploitation. Therefore, many scholars have studied in-depth and proposed some effective ways to improve EO performance. In [14], an opposition-based learning EO algorithm is proposed, called EOOBLE. First, an opposition-based learning mechanism is injected in the initialization and update process of basic EO. Secondly, a levy flight mechanism is employed in the concentration update equation. Finally, an evolutionary population dynamics mechanism is adopted to avoid getting trapped in local optima. The performance of the EOOBLE is verified on 25 benchmark functions with dimensions from 100 to 5000, and compared with the original EO, EO variants and some well-known metaheuristics. The statistical results show that EOOBLE is an advantageous algorithm

for tackling high-dimensional global optimization problems. Furthermore, the effectiveness of EOOBLE is proven in a high-dimensional engineering design problem. In [15], an enhanced EO (EEO) algorithm based on three communication strategies is proposed. The accuracy of the EEO is verified on 28 benchmark functions, and compared with existing optimization methods. The analysis illustrates that the EEO algorithm outperforms its competitors. Additionally, EEO is utilized to address a discrete job shop scheduling problem (JSSP), and compared with the three improvement approaches of EEO. The experimental results reveal that EEO achieves significant improvements in solving JSSP. In [16], a self-adaptive EO (self-EO) is introduced, which integrates four effective exploring stages. The performance of the self-EO algorithm is verified on numerous optimization problems, including ten functions of the CEC'20 benchmark, three engineering optimization problems, two combinatorial optimization problems and three multi-objective problems. Moreover, the proposed self-EO is compared with nine metaheuristic techniques, including the standard EO, and eight well-performing metaheuristic techniques. The analysis shows that the self-EO has a better searching capability, and a faster convergence rate than the other algorithms. In [17], a new multi-objective EO (MOEO) is proposed. The crowding distance mechanism is used to balance the exploitation and exploration during the search process. Furthermore, a non-dominant sorting mechanism is combined with the MOEO algorithm to maintain population diversity. The performance of the MOEO algorithm is evaluated for 33 contextual problems, and compared with other state-of-the-art multi-objective optimization methods. Quantitative and qualitative experiments show that the MOEO algorithm has a high efficiency and exploration capability for multi-objective problems. In [18], a new EO version, called EEO, is proposed, which incorporates a performance-enhancing new levy flight mechanism strategy. The effectiveness of the EEO algorithm is confirmed on the ten functions of the CEC'20 test suite, compared with other high-performance algorithms. Subsequently, EEO is utilized to resolve the optimal power flow (OPF) problem. The results of EEO are compared with standard EO, and other metaheuristics. These simulations show that EEO has better performances than 20 published approaches and the original EO. Moreover, the superiority of EEO is illustrated by six different cases that involve different objective minimization. The comparisons show that EEO can provide viable solutions for various OPF problems. In [19], a new algorithm based on the hybrid of EO and pattern search (PS) techniques, called EO-PS, is introduced. The EO-PS algorithm operates in two stages. The first stage performs EO to explore the search space and achieve the desired region by utilizing the equilibrium pool of elite particles. The second stage incorporates PS to lead the search to better neighborhoods and gain high quality solutions by employing its detection and pattern motion. The proposed EO-PS is utilized to handle the single and multi-objective optimization problems of wind farm layout optimization in different wind speed scenarios. Additionally, EO-PS is studied on irregular land space in the Gulf of Suez-red sea, Egypt. The comprehensive results show that EO-PS can obtain superiority compared with other advanced methods in terms of the quality and reliability of the solution. In [20], a multi-objective equilibrium optimizer slime mould algorithm (MOEOSMA) known as MOEOSMA is proposed. In the MOEOSMA, the elite archiving mechanism is utilized to facilitate convergence of the algorithm and the crowding distance method is employed to keep the distribution of the Pareto frontiers, dynamic coefficients are provided for adjusting exploration and exploitation and the equilibrium pool method is employed to simulate the collaborative foraging behavior of the slime molds to improve the global search ability of the algorithm. The performance of MOEOSMA is investigated on the CEC2020 test suite, eight real multi-objective constrained engineering problems and four large scale truss structure optimization problems. The results reveal that MOEOSMA is significantly better than other comparable algorithms. Meanwhile, the algorithm finds more Pareto optimal solutions and

remains well-distributed in the decision space and objective space. In [21], an improved quantum equilibrium optimizer (QEO) algorithm combining quantum coding and quantum rotating gate strategies for linear antenna arrays is introduced. The excitation amplitude of the array elements in the linear antenna array model is optimized by numerical simulation using QEO to minimize the interference of the side lobe levels on the main lobe radiation. Six different metaheuristics are employed to perform optimization of the excitation amplitude of the line antenna array elements for three different arrays. Experimental results demonstrate that QEO is more competitive than other optimization algorithms and is more advantageous in obtaining maximum side lobe level reduction. In [22], an improved version of EO, called equilibrium optimizer slime mould algorithm (EOSMA), is proposed. First, the exploration and exploitation capabilities of slime mould algorithm (SMA) are adjusted. Next, the anisotropic search operator of SMA is replaced by the search operator of EO to guide the search space of SMA. Finally, a stochastic differential mutation operator is incorporated to facilitate SMA to get rid of local optimality and increase the diversity of the population. The performance of EOSMA is validated on CEC2019, CEC2021 test suites, and nine engineering design problems. The results reveal that EOSMA is significantly better than 15 famous comparative algorithms on the CEC2019 benchmark problems. EOSMA performs clearly better than the three comparable algorithms on the CEC2021 benchmark functions. In addition, EOSMA outperforms other state-of-the-art comparison algorithms on all nine engineering problems.

Although the aforementioned EO variants have improved the performance of the basic EO, there are still some shortcomings. For example, in [14], although EOOBLE demonstrates better performance in solving high-dimensional problems, the additional search mechanism increases the computational complexity of the algorithm. In [15], EEO overlooks the balance between exploitation and exploration, leading to low convergence accuracy. In [16], self-EO improves the convergence speed of the basic EO but fails to address the issue of loss of population diversity. In [17], MOEO maintains population diversity and balances exploration and exploitation, but the additional search mechanism prolongs the optimization time. In [18], EEO aims to enhance the exploration and exploitation processes of the algorithm but neglects the diversity of particles and the possibility of premature stagnation during the search. Moreover, other variants of EO only focus on certain deficiencies of the basic EO without providing a comprehensive solution, resulting in remaining flaws such as imbalanced exploitation and exploration operations, low quality of randomly generated initial populations, and limited potential for large-scale jumps during population iterations, leading to poor convergence performance [23]. The main limitations of the EO algorithm are analyzed in detail in the next section. Based on the above motivations, the present work develops an EEO based on chaos, known as a chaos-based adaptive equilibrium optimizer algorithm (ACEO). First, a new chaos-based update rule is proposed to reduce the possibility of falling into a local optimum. Then, an adaptive gbest-guided search mechanism is developed to enrich the population diversity and expand the search area. The main objective of this work is to thoroughly analyze the shortcomings of the EO algorithm and to propose an improved EO variant that will improve the performance and stability of the basic method. The main highlights of this paper are outlined as follows.

- A novel EO variant, called ACEO, is developed. ACEO employs two mechanisms, which are an adaptive gbest-guided search mechanism and chaos mechanism. The effectiveness of the two components is examined using the ablation study in Section 4.4.
- The efficacy of the developed ACEO is verified on 23 classical benchmark functions, and compared with the canonical EO, EO variants and other state-of-the-art metaheuristic approaches. In addition, the experimental results are statistically analyzed using the Friedman mean rank test and the Wilcoxon signed rank test.

- To inspect the feasibility of the ACEO algorithm, it is adopted to resolve a mobile robot path planning (MRPP) task, and compare with some classical metaheuristic methods.

The rest of this paper is presented as follows. In Section 2, the update principle and drawbacks of the canonical EO is described. Section 3 discusses and analyzes the developed ACEO algorithm. Section 4 investigates the performance of the developed ACEO on 23 benchmark functions, and the ablation study of each mechanism employed. Section 5 discusses the ACEO-based path-planning task for mobile robots. Section 6 elaborates comprehensive conclusions.

2. Relate works

2.1. Basic EO algorithm

2.1.1 Initialization

Similar to most metaheuristic algorithms, EO utilizes candidate solutions initialized in the search space to initiate the optimization process. The initial concentration is given as follows.

$$C_i = C_{\min} + randi(C_{\max} - C_{\min}), i = 1, 2, \cdots, N \tag{1}$$

where C_i is the concentration of the ith initial particle, $C_{\max}$ and $C_{\min}$ are the upper and lower limit of each dimension of the search space, N is the number of particles in the population.

2.1.2. Equilibrium pool and candidates (C_{eq})

The four particles with the optimal fitness values and the mean of these four particles are selected, and these five particles are utilized to build the equilibrium pool. The C_{eq_pool} and C_{eq_ave} are expressed as follows, respectively.

$$C_{eq\_pool} = \{C_{eq1}, C_{eq2}, C_{eq3}, C_{eq4}, C_{eq\_ave}\} \tag{2}$$

$$C_{eq\_ave} = \frac{C_{eq1} + C_{eq2} + C_{eq3} + C_{eq4}}{4} \tag{3}$$

2.1.3. Exponential term (F)

The exponential term F is a parameter that maintains the balance between global search and local search during the optimization process, which is described as follows.

$$F = e^{-\lambda(t - t_0)} \tag{4}$$

where λ is a vector of random variables in the interval [0,1], t is a nonlinear function, and is calculated as follows.

$$t = (1 - \frac{Iter}{Max\_iter})^{(a_2 \frac{Iter}{Max\_iter})} \tag{5}$$

where *Iter* and *Max_iter* are the number of current iterations and the maximum number of iterations, respectively, a_2 is equal to 1 and controls the exploitation capacity, where the larger the a_2, the higher the exploitation ability and the lower the exploration ability and t_0 is expressed as follows.

$$t_0 = \frac{1}{\lambda} \ln(-a_1 sign(r - 0.5)[1 - e^{-\lambda t}]) + t \tag{6}$$

where a_1 is equal to 2 and controls exploration capability, $sign(r–0.5)$ managing the direction of development, r is a random vector between 0 and 1. Substituting Eq (6) into Eq (4), the exponential term F can be expressed as follows.

$$F = a_1 sign(r - 0.5)[1 - e^{-\lambda t}] \tag{7}$$

2.1.4. Generation rate (G)

The generation rate G is a parameter that controls the local search capability, and is defined as follows.

$$G = G_0 e^{-\lambda(t - t_0)} = G_0 F \tag{8}$$

where

$$G_0 = GCP(C_{eq} - \lambda C) \tag{9}$$

$$GCP = \begin{cases} 0.5 r_1, r_2 \geq GP \\ 0, r_2 < GP \end{cases} \tag{10}$$

where r_1 and r_2 are random numbers in the interval [0,1], and GCP is the generation rate control parameter, which is determined by the generation probability GP.

Therefore, the concentration update formula of the EO algorithm is as follows.

$$C = C_{eq} + (C - C_{eq}) \cdot F + \frac{G}{\lambda V}(1 - F) \tag{11}$$

2.2. Deficiencies of EO

The canonical EO has some advantages, such as a simple framework, being easy to implement, having few parameters and being easy to mix with other algorithms. Although the EO algorithm has shown competitive performance on global optimization problems, it still has some limitations. This subsection will provide a detailed analysis of the shortcomings of the EO algorithm as follows:

- The information of the equilibrium candidates in the equilibrium pool is not sufficiently utilized. The canonical EO establishes an equilibrium pool, which is the core component of the EO algorithm. The equilibrium candidates in the equilibrium pool provide information about the equilibrium state of the algorithm, but it is selected by ranking according to the size of the fitness

value, which increases the risk of obtaining similar solutions, thus reducing the diversity of particles, and cannot guarantee that the algorithm converges to the optimum. Hence, making full use of the information of the equilibrium candidates can guide the particles to more promising regions and can improve their performance.

- Low exploration ability. Based on the concentration update rule, the particle updates its position only according to the equilibrium state equation. During the iterative updating process, the differences of the positions between the equilibrium candidate particles become smaller and smaller, and at the later stage the positions between the particles are all extremely similar, which can lead to falling into a local optimum and thus stagnating prematurely. However, this aspect is particularly problematic in the canonical EO, reducing the performance of the algorithm.

The above analysis indicates that EO has some limitations that affect expected performance. Existing EO research has worked to alleviate some of these limitations, but not all of them. This is the motivation to propose a new EO variant. Thus, we propose a new EO variant based on the above drawbacks and address the drawbacks that still exist in the existing EO variants by novel modifications. In the following sections, we discuss the new EO variant in detail.

3. The proposed ACEO

In this section, the developed ACEO is explained in detail. At the particle update concentration stage, an adaptive gbest-guided mechanism is introduced to enrich the selection of the equilibrium candidate. During the concentration update process, the chaos mechanism is performed for all particles to avoid getting caught up in the local optimum, thus reaching a proper balance between exploration and exploitation.

3.1. Adaptive gbest-guided search mechanism

Reaching an appropriate balance between global and local search is an important feature of metaheuristic algorithms [24]. For the canonical EO, the exponential term F is a key operator responsible for balancing exploration and exploitation, as shown in Eq (11). There are three terms in Eq (11): the first term is the equilibrium candidate, the second term is responsible for the global search to find the optimal solution and the third term is responsible for exploitation to make the solution more precise. Obviously, in this equation, the latter two terms move the algorithm in the desirable direction. Nevertheless, the equilibrium candidate of the first term is selected in the equilibrium pool by ranking according to the size of the fitness value, which increases the risk of similarity to obtain the solution, and thus does not ensure the convergence to the optimal result. To alleviate the above limitations, we propose an adaptive gbest-guided concentration update equation to replace Eq (11). The proposed new update equation is as follows.

$$C = \omega \times C_{eq} + (C - C_{eq}).F + \frac{G}{\lambda V}(1 - F) \tag{12}$$

where ω is an inertia weight coefficient.

Inspired by PSO, the inertia weight mechanism has been widely applied to metaheuristic algorithms. For example, Wang et al. [25] propose an inertia weight strategy to enhance the leader's position update equation of the SSA, thus taking full advantage of the information provided from the food source to the leader. Pathak and Srivastava [26] involve the Sugeno fuzzy inertia weight in the

velocity updating equation of the bat algorithm (BA). Chen et al. [27] employ the mechanism of dynamically adjusting the inertia weight to enhance the updating method of the wolf swarm position of GWO. Ding et al. [28] introduce an adaptive inertia weight factor to modify the follower position update equation of the SSA. Yin and Mao [29] add inertia weights to modify the position vector of the grey wolf of GWO. Consequently, we propose a novel adaptive inertia weight that is described by the following mathematical formula.

$$\omega = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \times a^{b^{Iter - \frac{Max\_iter}{2}}} \tag{13}$$

where $\omega_{\max}$, $\omega_{\min}$ denote the maximum and minimum values of the weight, respectively. a and b are constants.

Figure 1 plots the nonlinear decline curves of the inertia weight ω. As depicted in Figure 1, the curve is similar to an "inverse S" curve with a range of [0.2,0.9]. During this range, the slopes of the curve start to steepen gradually, then reach the minimum at the lowest point. Subsequently, the slopes of the curve start to slow down gradually, and finally converge to 0. The curve changes relatively smoothly near the center point, and changes more rapidly on both sides. Based on the characteristics of the curve, the values of ω are larger at the initial stage of the iterations, facilitating the particle to explore the search space. However, as the iterations progressed, the values of ω gradually decrease, which favor making the solution more accurate in the regions already searched.

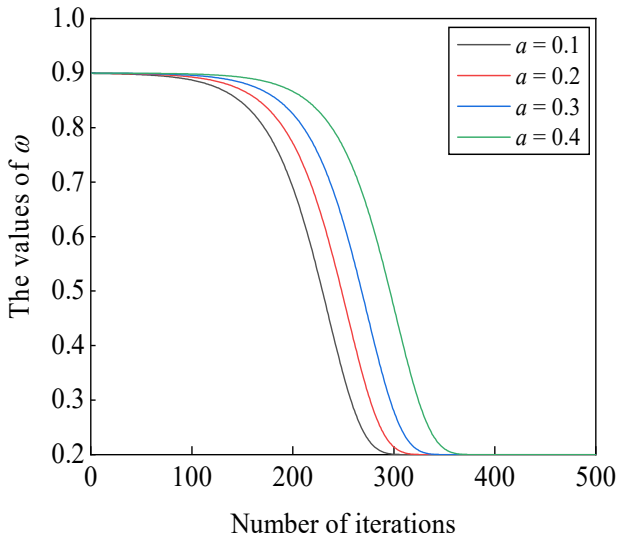

Figure 1. Curve of the proposed inertia weight ω.

3.2. Chaos mechanism

In this subsection, we introduce a chaos mechanism to make EO avoid being trapped in a local optimum. Chaos is a random-like phenomenon that exists in nonlinear and deterministic systems. It is exceedingly sensitive to initial conditions, and is neither periodic nor convergent. Chaotic motion is not repetitive, and can also be searched at a higher rate than random traversal search that relies on probabilities. Its pseudo randomness is reflected in the stochastic search process, which greatly

improves search efficiency. As a result, chaos search has more strengths than random search, it has been broadly applied to metaheuristic techniques and obtained excellent effects in improving optimization algorithms. For example, Gezici and Livatyalı [30] use the harris hawk optimization (HHO) and 10 different chaotic graphs for hybridization to improve the performance of the algorithm. Liang et al. [31] utilize the ergodicity of the chaos factor to make the marine predators algorithm (MPA) easier to jump out of local optima. Feng et al. [32] propose a chaos-based loudness approach to enhance the BA algorithm. Gharehchopogh et al. [33] embed twelve chaotic maps into the farmland fertility algorithm (FFA) to search for the optimal numbers of prospectors and improve the exploitation of the most promising solutions. Joshi [34] embeds chaos-based opposition learning in the GSA to achieve stagnation-free search.

Compared with other chaos models, the logistic chaotic model has particularly complex dynamic behavior, and is easy to implement. Consequently, the present work adopts the logical chaotic model to map the population. The logical chaotic sequence is expressed as follows.

$$X_{i+1} = \mu X_i(1 - X_i), i = 1, 2, \ldots, N-1 \tag{14}$$

where μ is the control parameter, usually $\mu = 4$. When $\mu = 4$, the particles move chaotically. X_i is a random number between (0,1). N is the population size.

The updated model of the concentration update for the particles is as follows.

$$C_i^c = X_i C_i \tag{15}$$

where C_i^c is the ith particle concentration with chaotic disturbance. X_i is the ith chaotic value in the chaotic sequence.

The population positions of the canonical EO are always determined by the equilibrium state equation, resulting in the phenomenon that the algorithm suffers from falling into a local optimum and stagnating. Consequently, our idea is to first enrich the diversity of equilibrium candidates using the update equation of Eq (12), to enlarge the search area, and then use Eq (15) to perform chaotic search for particles in the population, to improve the search efficiency and avoid falling into the local optimum. The specific implementation steps are that a large number of number sequences will be generated by the initial value of the logical chaotic mapping. The ergodic character of the generated chaotic numbers makes the algorithm easier in the iterative search process, and under this condition, all possible iterative future states are evaluated non-repetitively. This non-repetitiveness of chaotic numbers not only enhances the convergence ability but also improves the search efficiency of the algorithm. Based on the above analysis about chaos, it is very advantageous to use chaotic graphs instead of randomness. These properties of chaos provide a significant improvement in the performance of the canonical EO.

3.3. The framework of ACEO

The developed ACEO algorithm introduces two mechanisms, and its implementation does not affect the canonical EO framework. Because the candidate solutions are selected within the equilibrium pool by ranking according to the size of the fitness value, increasing the similarity risk of the obtaining solutions cannot ensure convergence to the optimum. Therefore, the adaptive gbest-guided search mechanism is introduced to overcome this shortcoming. During the update

concentration stage, the adaptive gbest-guided search mechanism enhances the navigational ability of the equilibrium candidate through inertia weights, thus increasing its diversity and expanding the search area. In the early stages of searching, the inertia weight has larger values, and the randomness of the selection of the equilibrium candidate is also larger. Thus, the solution space can be sufficiently explored. As the number of iterations increases, in the later stages the smaller inertia weight values sufficiently exploit the already searched range, and the equilibrium candidate is chosen to be more favorable to equilibrium concentrations. Throughout the search process, it is hoped that the equilibrium candidate will move to a more prospective region with the constant adjustment of the inertia weight values. Nevertheless, during the updating process, the particles in the population undergo random exploration, and the positions of the particles are always determined by the equilibrium state equation. This may cause the algorithm to easily fall into a local optimum and result in premature stagnation of the search. The chaos mechanism solves this problem by utilizing the ergodic properties and non-repeatability of produced chaos numerical sequences, which accelerates the convergence speed and effectively causes the particles to jump out of the local optimum. Consequently, based on the constantly changing inertia weight values and the characteristics of chaos, a proper balance is achieved between exploration and exploitation of particle concentration. The pseudo-code of ACEO is given in Algorithm 1. Figure 2 displays the flowchart of ACEO.

Algorithm 1 Pseudo-code of ACEO

population size N, dimension d, upper and lower bounds of the search space C_{max}, C_{min}
Initialize the population by Eq (1)
$Iter = 1$
while ($Iter \leq Max\_iter$)
    Reinitialize the out-of-bounds individual
    Calculate individual fitness values and select C_{eq1}, C_{eq2}, C_{eq3}, C_{eq4}
    if $Iter > 1$
        Perform memory saving on the population to retain high-quality individuals
    end if
    Calculate the average particle C_{eq_ave} using Eq (3)
    Construct the equilibrium pool C_{eq_pool} using Eq (2)
    Update the adaptive parameter t using Eq (5)
    for $i = 1{:}N$
        Generate random number matrices r and λ
        Select C_{eq} randomly from the equilibrium pool C_{eq_pool}
        Construct F using Eq (7)
        Calculate GCP using Eq (10)
        Construct G_0 using Eq (9)
        Construct G using Eq (8)
        Update the individual using Eq (12)
    end for
    for $i = 1{:}N$
        Update the concentrations of the particles using Eq (15)
    end for
$Iter = Iter + 1$
end while
Return the elite candidate solution C_{eq1}

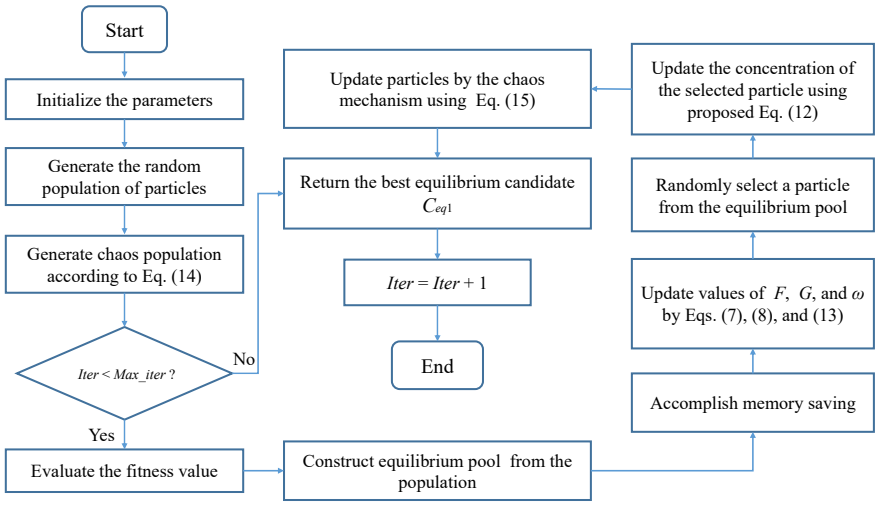

Figure 2. The flowchart of the developed ACEO algorithm.

3.4. Computational complexity

Computational complexity is an important criterion for analyzing the performance of an algorithm. In this paper, Big-O notation is employed to represent the complexity. The computational complexity of the proposed ACEO consists of initialization, fitness evaluation, memory saving, concentration update and chaos update mechanism. The computational complexity of initializing N individuals is O($N \times d$), where N is the number of particles and d is the number of dimensions. The computational complexity of the fitness evaluation is O($t \times f \times d$), where t is the number of iterations and f is the fitness evaluation. The computational complexity of the memory saving is O($t \times N$). The computational complexity of the concentration update is O($t \times N \times d$). The computational complexity of the chaos update mechanism is O($t \times N \times d$). Consequently, the computational complexity of ACEO is O($N \times d + t \times f \times d + t \times N + t \times N \times d + t \times N \times d$) $\cong$ O($t \times f \times d + t \times N \times d$). The computational complexity of the canonical EO is O($t \times f \times d + t \times N \times d$).

In conclusion, the developed ACEO has the same computational complexity as the canonical EO.

4. Simulation and discussion

To evaluate the performance of the ACEO algorithm, a series of experiments are done to deal with different benchmark functions.

4.1. Benchmark test functions

To facilitate comparison, we test the developed algorithm on 23 classical benchmark functions. The details of these benchmark functions are outlined in Table 1.

In Table 1, 23 classical benchmark functions are subdivided into unimodal functions and multimodal functions. The unimodal functions are appropriate for testing the exploitation capability of the algorithm because of it only has one global best solution. In contrast, for the multimodal functions that have multiple local best solutions, suitable for testing the global search ability of the optimizer [35].

Table 1. The characteristics of the classical benchmark functions.

Function type	Function formulation	Search range	$f_{\min}$
Unimodal	$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-100,100]$	0
	$f_2(x)=\sum_{i=1}^{D} ix_i^2$	$[-10,10]$	0
	$f_3(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	$[-100,100]$	0
	$f_4(x)=\max_i\{\lvert x_i\rvert, 1\le x_i\le D\}$	$[-100,100]$	0
	$f_5(x)=\sum_{i=1}^{D}\left(\lfloor x_i+0.5\rfloor\right)^2$	$[-100,100]$	0
	$f_6(x)=\sum_{i=1}^{D} ix_i^4$	$[-1.28,1.28]$	0
	$f_7(x)=\sum_{i=1}^{D} ix_i^4+random[0,1)$	$[-1.28,1.28]$	0
	$f_8(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	$[-1,1]$	0
	$f_9(x)=\sum_{i=1}^{D}(10^6)^{(i-1)/(D-1)}x_i^2$	$[-100,100]$	0
	$f_{10}(x)=x_1^2+10^6\cdot\sum_{i=2}^{D}x_i^6$	$[-100,100]$	0
	$f_{11}(x)=10\cdot x_1^2+\sum_{i=2}^{D}x_i^6$	$[-1,1]$	0
Multimodal	$f_{12}(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]$	0
	$f_{13}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	$[-32,32]$	0
	$f_{14}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$	0
	$f_{15}(x)=\sum_{i=1}^{D}\lvert x_i\cdot\sin(x_i)+0.1x_i\rvert$	$[-10,10]$	0
	$f_{16}(x)=\sin^2(\pi x_1)+\sum_{i=1}^{D-1}[x_i^2\cdot(1+10\sin^2(\pi x_1))+(x_i-1)^2-\sin^2(2\pi x_i)]$	$[-10,10]$	0
	$f_{17}(x)=0.1D-\left(0.1\sum_{i=1}^{D}\cos(5\pi x_i)-\sum_{i=1}^{D}x_i^2\right)$	$[-1,1]$	0
	$f_{18}(x)=\sum_{i=1}^{D}x_i^2+\left(\sum_{i=1}^{D}0.5x_i\right)^2+\left(\sum_{i=1}^{D}0.5x_i\right)^4$	$[-5,10]$	0
	$f_{19}(x)=\sum_{i=1}^{D}(0.2x_i^2+0.1x_i^2\cdot\sin(2x_i))$	$[-10,10]$	0
	$f_{20}(x)=\left[\frac{1}{D-1}\sum_{i=1}^{D}\left(\sqrt{x_i}(\sin(50.0x_i^{0.2})+1)\right)\right]^2$	$[-100,100]$	0
	$f_{21}(x)=\sum_{i=1}^{D-1}[x_i^2+2x_{i+1}^2-0.3\cos(3\pi x_i)-0.4\cos(4\pi x_{i+1})+0.7]$	$[-15,15]$	0
	$f_{21}(x)=\sum_{i=1}^{D-1}(x_i^2+2x_{i+1}^2)^{0.25}\cdot((\sin 50(x_i^2+x_{i+1}^2)^{0.1})^2+1)$	$[-10,10]$	0
	$f_{23}(x)=\sum_{i=1}^{D-1}x_i^6\cdot\left(2+\sin\frac{1}{x_i}\right)$	$[-1,1]$	0

4.2. ACEO in comparison with EO, EO variants, and other metaheuristics

In order to demonstrate the effectiveness of the developed ACEO, the 23 benchmark functions in Table 1 are adopted. The dimension of these functions is set to 100. In this paper, we classify the comparative algorithms into two categories: the EO variants and the state-of-the-art metaheuristic algorithms. Using basic EO and recently reported EO variants as the comparative algorithms allows us to highlight the superiority of our proposed algorithm over EO-based methods, thereby demonstrating that the proposed EO variant further alleviates the limitations of basic EO and improves the algorithm's performance. On the other hand, the inclusion of other state-of-the-art metaheuristic algorithms aims to showcase the superiority of our proposed method over different types of representative algorithms in the optimization community. These advanced metaheuristic algorithms cover various optimization paradigms.

By comparing our method with these algorithms, we can assess the effectiveness and competitiveness of our approach. Based on the above analysis, the comparison algorithms include the canonical EO algorithm, the efficient EO with mutation strategy (mEO) [36], the EO algorithm based on levy flight, the whale optimization algorithm's spiral encirclement strategy, the adaptive proportional mutation mechanism (LWMEO) [37], information utilization strengthened EO (IS-EO) [38], the improved EO with a decreasing equilibrium pool (IEO) [39], the SSA algorithm based on random replacement tactic and double adaptive weighting mechanism (RDSSA) [40], the hybrid enhanced whale optimization SSA algorithm (IWOSSA) [41], selective opposition based grey wolf optimizer (SOGWO) [42], opposition-based learning grey wolf optimizer (OGWO) [43] and moth-flame optimization algorithm with diversity and mutation mechanism (DMMFO) [44]. For objective and fair comparison, the size of the particles and the maximum number of iterations are 30 and 500, respectively. Each optimizer is run 30 times independently in same environment. The other important parameters of the comparative approaches are extracted from the corresponding original literature. In the developed ACEO, $\omega_{max} = 0.9$, $\omega_{min} = 0.2$, $X_1 = 0.7$, $a = 0.4$, $b = 1.05$. The mean (mean) and standard deviation (Std) values are considered as indicators to assess the performance of the algorithms. The mean and the Std of the optimal fitness of 11 optimizers are reported in Table 2. Moreover, the results acquired by ACEO and its competitors are analyzed using the Friedman mean rank test. Moreover, the results of the Friedman mean rank test are recorded in the final column in Table 2.

According to the results in Table 2, it can be observed that ACEO converges to theoretical global optima solutions on all functions other than f_4, f_7, f_{13}, and f_{15}. For these four functions, ACEO achieves a better efficiency than other algorithms. For the EO variants, compared with canonical EO, ACEO is superior on 18 out of 23 cases, except for $f_5, f_{12}, f_{14}, f_{17}, f_{21}$ where it indicates similiar performance. ACEO outperforms LWMEO and IS-EO in tackling all functions. Additionally, the comparative analysis suggested that ACEO is better than mEO and IEO on a considerable number of functions, but similar to them on some functions. For other advanced metaheuristic algorithms, when comparing with RDSSA, ACEO performs better across 18 functions, except for 5 functions in which RDSSA shows similar results. ACEO provides better performance compared with the IWOSSA and DMMFO across all the statistics. With the exception of f_5, ACEO surpasses SOGWO and OGWO on all evaluated functions, demonstrating its competitiveness in addressing 100-dimensional optimization problems. For f_5, ACEO, EO, mEO, IEO, RDSSA, SOGWO and OGWO all find the optimal values. For $f_{12}, f_{14}, f_{17}, f_{21}$, ACEO, EO, mEO, IEO and RDSSA achieve theoretical results, which suggests that the advantages of the introduced strategies in ACEO are not significant in these multimodal cases. For f_{13}, ACEO, EO, mEO, IEO and RDSSA have extremely similar results, which suggests that the performance of the developed ACEO is unremarkable in tackling this multimodal case. Furthermore, the mean and Std of the majority of functions evaluated by ACEO are zero or close to zero, which indicates the stability of ACEO. Figure 3 visually plots the results of the Friedman mean rank test. According to Figure 3, ACEO is the highest rank, followed by RDSSA, mEO, IEO, EO, OGWO, SOGWO, IWOSSA, LWMEO, IS-EO and DMMFO, which illustrates that ACEO provides the best performance among all implemented algorithms. These data demonstrate the superiority of the proposed ACEO algorithm. This is because the adaptive gbest-guided search mechanism introduced using inertia weight values by ACEO enriches the diversity of equilibrium candidate and expands the search region, and the chaos mechanism uses generated chaos number sequences to replace the original stochastic search to accelerate the convergence and avoid falling into a local optimum. The combined effect of these mechanisms allows the developed algorithms to converge to the best results for most of the single-peak and multi-peak functions, as verified in the above experiments. These findings also highlight that the ACEO algorithm exhibits superior performance in terms of convergence and solution quality, compared with EO, EO variants and variants of advanced

metaheuristic algorithms.

Table 2. Comparisons of eleven algorithms on 23 benchmark functions with 100 dimensions.

Function	Results	EO	mEO	LWMEO	IS-EO	IEO	RDSSA	IWOSSA	SOGWO	OGWO	DMMFO	ACEO
f_1	Mean	4.42×10^{-29}	1.14×10^{-37}	0.0025	10.0443	1.13×10^{-29}	2.44×10^{-35}	1.32×10^{-06}	2.33×10^{-12}	4.04×10^{-15}	$3.18 \times 10^{+04}$	0
	Std	9.42×10^{-29}	3.17×10^{-37}	0.0012	37.1003	1.55×10^{-29}	1.34×10^{-34}	7.44×10^{-07}	1.84×10^{-12}	7.39×10^{-15}	$5.74 \times 10^{+03}$	0
	f-rank	5	2	9	10	4	3	8	7	6	11	1
f_2	Mean	1.21×10^{-29}	4.62×10^{-39}	0.0448	0.9648	8.96×10^{-30}	1.52×10^{-41}	6.03×10^{-07}	4.29×10^{-13}	1.78×10^{-15}	$1.57 \times 10^{+04}$	0
	Std	1.78×10^{-29}	1.07×10^{-38}	0.0182	2.2581	2.58×10^{-29}	8.32×10^{-41}	4.12×10^{-07}	2.79×10^{-13}	3.08×10^{-15}	$2.87 \times 10^{+03}$	0
	f-rank	5	3	9	10	4	2	8	7	6	11	1
f_3	Mean	20.3786	28.7498	$3.53 \times 10^{+04}$	$3.65 \times 10^{+05}$	102.6401	1.73×10^{-28}	$9.64 \times 10^{+04}$	$1.88 \times 10^{+03}$	655.1702	$2.45 \times 10^{+05}$	0
	Std	74.1112	120.0153	$2.72 \times 10^{+04}$	$1.06 \times 10^{+05}$	121.5038	9.45×10^{-28}	$1.96 \times 10^{+04}$	$1.68 \times 10^{+03}$	$1.03 \times 10^{+03}$	$3.59 \times 10^{+04}$	0
	f-rank	3	4	8	11	5	2	9	7	6	10	1
f_4	Mean	2.5457	4.09×10^{-06}	49.7062	70.3003	0.0055	3.06×10^{-30}	43.4727	1.0802	1.2644	88.4752	4.10×10^{-215}
	Std	13.9351	9.29×10^{-06}	14.1743	14.0957	0.0113	1.13×10^{-29}	7.2497	0.8604	1.2747	2.6723	0
	f-rank	7	3	9	10	4	2	8	5	6	11	1
f_5	Mean	0	0	39.0333	346.6667	0	0	2.1333	0	0	$3.59 \times 10^{+04}$	0
	Std	0	0	8.9615	$1.21 \times 10^{+03}$	0	0	4.6515	0	0	$7.53 \times 10^{+03}$	0
	f-rank	1	1	9	10	1	1	8	1	1	11	1
f_6	Mean	8.62×10^{-51}	3.31×10^{-57}	1.59×10^{-06}	0.0136	2.24×10^{-50}	4.74×10^{-85}	6.79×10^{-12}	3.35×10^{-25}	4.63×10^{-29}	102.3963	0
	Std	1.55×10^{-50}	1.78×10^{-56}	1.36×10^{-06}	0.0701	4.47×10^{-50}	2.60×10^{-84}	1.59×10^{-11}	4.86×10^{-25}	9.29×10^{-29}	25.3133	0
	f-rank	4	3	9	10	5	2	8	7	6	11	1
f_7	Mean	0.0024	0.0036	0.6300	0.0403	0.0155	7.28×10^{-04}	0.1066	0.0082	0.0025	104.2500	1.22×10^{-05}
	Std	0.0012	0.0020	0.1163	0.0165	0.0056	4.99×10^{-04}	0.0737	0.0033	0.0027	31.2914	9.33×10^{-05}
	f-rank	3	5	10	8	7	2	9	6	4	11	1
f_8	Mean	5.03×10^{-127}	1.65×10^{-219}	5.72×10^{-11}	6.43×10^{-04}	1.38×10^{-127}	1.31×10^{-80}	1.42×10^{-17}	6.77×10^{-65}	1.34×10^{-45}	0.0019	0
	Std	2.65×10^{-126}	0	1.95×10^{-10}	0.0014	7.57×10^{-127}	7.15×10^{-80}	2.90×10^{-17}	3.42×10^{-64}	7.35×10^{-45}	0.0031	0
	f-rank	4	2	9	10	3	5	8	6	7	11	1
f_9	Mean	1.46×10^{-25}	1.11×10^{-35}	$1.09 \times 10^{+03}$	$5.08 \times 10^{+08}$	1.29×10^{-25}	3.00×10^{-55}	0.0086	4.74×10^{-09}	4.77×10^{-12}	$1.95 \times 10^{+08}$	0
	Std	2.80×10^{-25}	2.13×10^{-35}	626.8200	$2.76 \times 10^{+08}$	2.27×10^{-25}	1.64×10^{-54}	0.0050	3.65×10^{-09}	5.89×10^{-12}	$8.43 \times 10^{+07}$	0
	f-rank	5	3	9	11	4	2	8	7	6	10	1
f_{10}	Mean	1.90×10^{-48}	4.82×10^{-56}	$3.03 \times 10^{+05}$	$6.97 \times 10^{+12}$	1.13×10^{-44}	6.91×10^{-74}	$1.88 \times 10^{+05}$	1.13×10^{-15}	4.63×10^{-20}	$3.20 \times 10^{+17}$	0
	Std	5.96×10^{-48}	2.29×10^{-55}	$1.43 \times 10^{+06}$	$3.12 \times 10^{+13}$	5.64×10^{-44}	3.54×10^{-73}	$5.74 \times 10^{+05}$	2.96×10^{-15}	2.43×10^{-19}	$1.44 \times 10^{+17}$	0
	f-rank	4	3	9	10	5	2	8	7	6	11	1
f_{11}	Mean	7.44×10^{-66}	7.82×10^{-77}	1.45×10^{-05}	0.3786	1.55×10^{-60}	2.01×10^{-81}	1.60×10^{-08}	4.59×10^{-34}	2.91×10^{-38}	0.9278	0
	Std	2.58×10^{-65}	2.43×10^{-76}	1.32×10^{-05}	0.2939	8.26×10^{-60}	1.10×10^{-80}	7.34×10^{-08}	2.18×10^{-33}	1.34×10^{-37}	0.3420	0
	f-rank	4	3	9	10	5	2	8	7	6	11	1
f_{12}	Mean	0	0	276.8536	42.0796	0	0	272.4656	10.8178	0.9362	835.7346	0
	Std	0	0	163.9720	60.7463	0	0	124.4195	7.1563	2.5936	55.8474	0
	f-rank	1	1	10	8	1	1	9	7	6	11	1
f_{13}	Mean	3.58×10^{-14}	1.17×10^{-14}	6.6206	2.0333	2.84×10^{-14}	4.32×10^{-15}	2.15×10^{-04}	1.30×10^{-07}	4.34×10^{-09}	19.7062	8.88×10^{-16}
	Std	7.87×10^{-15}	4.01×10^{-15}	6.8457	2.2432	3.61×10^{-15}	6.49×10^{-16}	4.50×10^{-04}	4.24×10^{-08}	2.24×10^{-09}	0.1949	0
	f-rank	5	3	10	9	4	2	8	7	6	11	1
f_{14}	Mean	0	0	0.0068	2.9484	0	0	0.0055	0.0056	0.0032	303.3664	0
	Std	0	0	0.0093	11.8781	0	0	0.0117	0.0132	0.0099	62.0000	0
	f-rank	1	1	9	10	1	1	7	8	6	11	1

Continued on next page

Function	Results	EO	mEO	LWMEO	IS-EO	IEO	RDSSA	IWOSSA	SOGWO	OGWO	DMMFO	ACEO
f_{15}	Mean	5.03×10^{-18}	3.81×10^{-32}	16.9968	0.1297	1.92×10^{-18}	2.31×10^{-26}	18.6665	0.0035	1.54×10^{-04}	60.5130	1.14×10^{-223}
	Std	5.82×10^{-18}	6.38×10^{-32}	7.4658	0.3458	1.98×10^{-18}	1.25×10^{-25}	18.1816	0.0021	4.32×10^{-04}	8.5608	0
	f-rank	5	2	9	8	4	3	10	7	6	11	1
f_{16}	Mean	3.90×10^{-25}	5.56×10^{-36}	$1.83 \times 10^{+03}$	37.9767	5.07×10^{-25}	2.87×10^{-47}	7.7058	0.1661	1.81×10^{-24}	$1.61 \times 10^{+03}$	0
	Std	1.89×10^{-24}	1.61×10^{-35}	890.2121	54.5722	1.80×10^{-24}	1.51×10^{-46}	16.2323	0.4905	4.38×10^{-24}	307.4301	0
	f-rank	4	3	11	9	5	2	8	7	6	10	1
f_{17}	Mean	0	0	6.8036	0.0535	0	0	1.46×10^{-09}	2.91×10^{-14}	3.85×10^{-15}	12.2093	0
	Std	0	0	1.6896	0.1135	0	0	8.80×10^{-10}	7.99×10^{-15}	3.43×10^{-15}	1.6604	0
	f-rank	1	1	10	9	1	1	8	7	6	11	1
f_{18}	Mean	1.27×10^{-27}	1.80×10^{-35}	0.0043	0.8062	3.29×10^{-28}	7.40×10^{-64}	0.0183	2.35×10^{-13}	1.05×10^{-15}	501.1165	0
	Std	4.57×10^{-27}	8.65×10^{-35}	0.0016	1.4083	4.03×10^{-28}	4.06×10^{-63}	0.0238	2.02×10^{-13}	1.03×10^{-15}	100.0237	0
	f-rank	5	3	8	10	4	2	9	7	6	11	1
f_{19}	Mean	8.83×10^{-32}	1.84×10^{-40}	4.7270	0.0044	4.39×10^{-32}	4.18×10^{-51}	2.89×10^{-09}	6.34×10^{-15}	2.43×10^{-26}	177.6119	0
	Std	1.42×10^{-31}	5.86×10^{-40}	14.2336	0.0133	8.72×10^{-32}	2.29×10^{-50}	1.97×10^{-09}	4.68×10^{-15}	1.27×10^{-25}	29.9811	0
	f-rank	5	3	10	9	4	2	8	7	6	11	1
f_{20}	Mean	1.46×10^{-09}	4.28×10^{-19}	5.8175	0.6514	8.29×10^{-10}	5.28×10^{-14}	0.0485	0.0116	8.39×10^{-05}	8.4985	0
	Std	6.53×10^{-10}	3.77×10^{-19}	0.8739	0.4997	4.99×10^{-10}	2.00×10^{-13}	0.0478	0.0028	4.84×10^{-05}	0.2707	0
	f-rank	5	2	10	9	4	3	8	7	6	11	1
f_{21}	Mean	0	0	53.2611	8.5430	0	0	8.81×10^{-07}	1.22×10^{-12}	2.62×10^{-15}	$2.23 \times 10^{+03}$	0
	Std	0	0	6.3040	21.1564	0	0	6.15×10^{-07}	1.10×10^{-12}	2.58×10^{-15}	481.9389	0
	f-rank	1	1	10	9	1	1	8	7	6	11	1
f_{22}	Mean	0.2910	0.1208	6.8114	3.1963	0.2662	0.0216	3.6238	1.5778	0.8526	7.4673	0
	Std	0.0630	0.0555	0.9834	0.7882	0.0721	0.0217	1.3237	0.3033	0.1964	0.2637	0
	f-rank	5	3	10	8	4	2	9	7	6	11	1
f_{23}	Mean	9.62×10^{-65}	2.30×10^{-75}	3.44×10^{-06}	3.51×10^{-07}	1.18×10^{-60}	6.18×10^{-206}	2.19×10^{-09}	1.99×10^{-32}	4.17×10^{-60}	0.6036	0
	Std	4.21×10^{-64}	6.23×10^{-75}	1.73×10^{-06}	9.77×10^{-07}	5.04×10^{-60}	0	5.81×10^{-09}	6.60×10^{-32}	2.29×10^{-59}	0.1912	0
	f-rank	4	3	10	9	5	2	8	7	6	11	1
	Average f-rank	3.7826	2.5217	9.3913	9.4348	3.6957	2.0435	8.2609	6.6087	5.7391	10.8696	1
	Overall f-rank	5	3	9	10	4	2	8	7	6	11	1

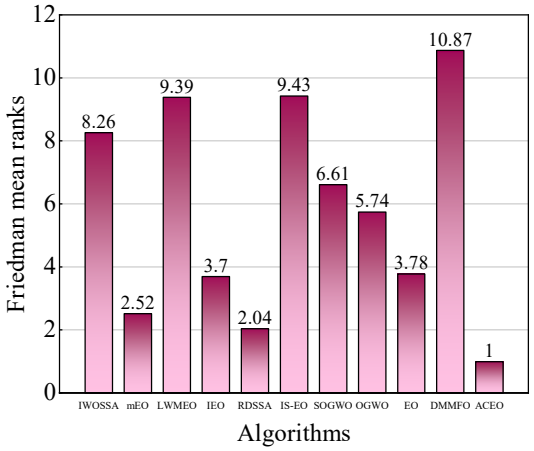

Figure 3. Chart of the Friedman's rank test results obtained by eleven algorithms.

Overall, the comprehensive experimental analysis illustrates that ACEO is a reliable and efficient algorithm that can provide prospective solutions for complex optimization problems.

To show statistical differences between the ACEO algorithm and other comparative approaches, the Wilcoxon signed rank test with a significance level of 5% is employed. The p-values of the Wilcoxon signed rank test of the ACEO, EO, EO variants and other metaheuristic algorithms are given in Table 3. The symbols "+/−/=" in Table 3 imply that the ACEO algorithm is better than, worse than, or equal to the other approaches. From Table 3, the p-values are almost always less than 0.05 on the majority of test functions, which illustrates that the performance of the ACEO is highly superior to other comparative algorithms.

Table 3. Statistical results of the Wilcoxon's test on 100-dimensional benchmark problems.

Function	EO	mEO	LWMEO	IS-EO	IEO	RDSSA	IWOSSA	SOGWO	OGWO	DMMFO
	p-value	*p*-value	*p*-value	*p*-value	*p*-value	*p*-value	*p*-value	*p*-value	*p*-value	*p*-value
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}	1.21×10^{-12}
f_2	4.57×10^{-12}	6.25×10^{-10}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	4.57×10^{-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}	1.21×10^{-12}
f_4	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}	3.02×10^{-11}	3.02×10^{-11}
f_5	N/A	N/A	1.19×10^{-12}	0.0028	N/A	N/A	5.15×10^{-06}	N/A	N/A	1.21×10^{-12}
f_6	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}	1.21×10^{-12}
f_7	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	7.77×10^{-09}	3.02×10^{-11}	3.02×10^{-11}	8.15×10^{-11}	3.02×10^{-11}
f_8	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}	1.21×10^{-12}
f_9	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}	1.21×10^{-12}
f_{10}	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}	1.21×10^{-12}
f_{11}	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}	1.21×10^{-12}
f_{12}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_{13}	9.34×10^{-13}	5.57×10^{-13}	1.21×10^{-12}	1.21×10^{-12}	6.47×10^{-13}	1.17×10^{-13}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}
f_{14}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	1.20×10^{-12}	1.21×10^{-12}
f_{15}	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}	3.02×10^{-11}	3.02×10^{-11}
f_{16}	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}	1.21×10^{-12}
f_{17}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	N/A	N/A	1.21×10^{-12}	1.14×10^{-12}	1.21×10^{-07}	1.21×10^{-12}
f_{18}	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}	1.21×10^{-12}
f_{19}	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}	1.21×10^{-12}
f_{20}	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}	1.21×10^{-12}
f_{21}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	N/A	N/A	1.21×10^{-12}	1.21×10^{-12}	4.55×10^{-12}	1.21×10^{-12}
f_{22}	1.19×10^{-12}	1.20×10^{-12}	9.34×10^{-13}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	1.21×10^{-12}	6.08×10^{-13}
f_{23}	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}	1.21×10^{-12}
+/=/−	18/5/0	18/5/0	23/23/0	23/23/0	18/5/0	18/5/0	23/23/0	22/1/0	22/1/0	23/23/0

4.3. Convergence analysis

To explore intuitively the performance of the developed ACEO, in this section, we plot the convergence curves of all the tested methods and analyze them. The convergence curves of all the tested algorithms of some of the benchmark functions are generated in Figure 4.

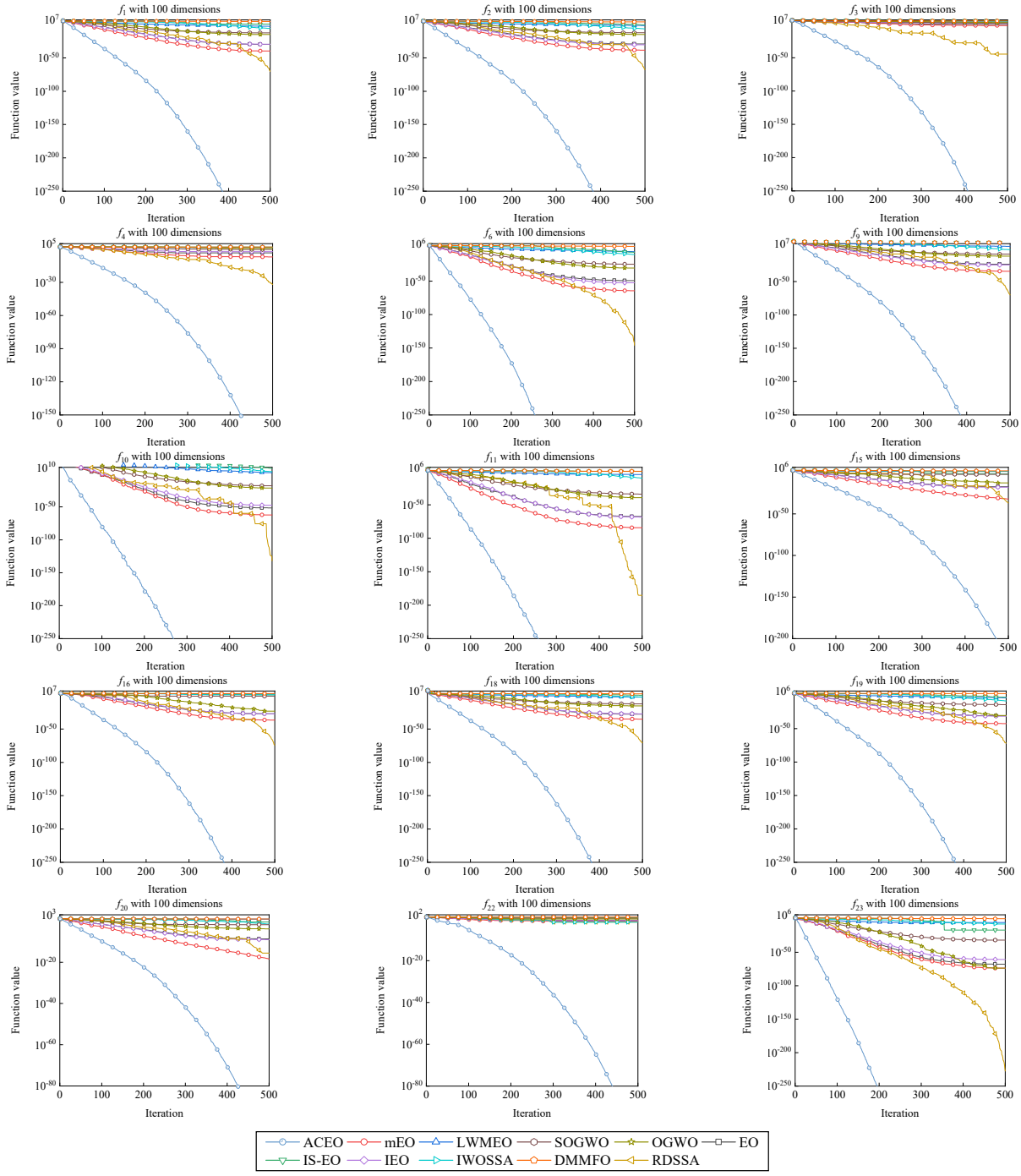

Figure 4. Convergence curves of eleven algorithms on some selected 100-dimensional functions.

Based on Figure 4, it can be observed that the developed ACEO exhibits higher competitiveness compared to other methods, which demonstrates its potential to optimize efficiency. The convergence curves of ACEO present an abrupt change at the beginning of the iterations, indicating its ability to explore the search space extensively. This can be attributed to the larger inertia weights in the adaptive gbest-guided search introduced at the beginning of the iteration that enriches the diversity and expands the exploration area. As the iteration progresses, the nonlinear gradual decrease of the inertia weight values causes abrupt changes in the convergence process. Then, the movement changes of ACEO gradually decrease and converge to the optimal solution. This trend is particularly

evident for most of the functions. The reduction in changes can be attributed to the gradual reduction of the inertia weights and the accelerated convergence of the generated chaos number sequences, which well prevents falling into the local optimum. The fitness value of the proposed ACEO ultimately converged to zero after several stages, whereas the other algorithms did not achieve zero fitness and displayed stagnation. Notably, the convergence performance of DMMFO was found to be the worst among all the implemented methods. Additionally, ACEO exhibits a faster decay rate on all the functions. Overall, ACEO demonstrates a stronger ability to locate the optimal area and has a faster convergence rate to the best solution than the other methods.

4.4. Component analysis

ACEO adds two mechanisms to modify the overall performance based on the canonical EO, namely the adaptive gbest-guided search mechanism and the chaotic map. In this subsection, the validity of these two operators is analyzed. We perform a series of experiments on 23 100-dimensional benchmark problems, as shown in Table 1. To distinguish between the two mechanisms, a single strategy is named. First only the gbest-guided search mechanism is employed, called AEO. Second only the chaotic map strategy is introduced, called LEO. The general parameters are kept the same as in Section 4.2. The performance of the methods involved is evaluated by the means and Stds, as shown in Table 4. Furthermore, the Friedman's rank test results are also used to examine performance and are recorded at the bottom. To intuitively display the results of the Friedman's rank test, we draw them in Figure 5. The convergence curves of the relevant methods of some of the selected functions are plotted in Figure 6.

From Table 4, it can be shown that ACEO outperforms AEO, LEO and EO on the vast majority of functions, which indicates that each mechanism on its own significantly enhances canonical EO. From Figure 5, ACEO ranks first, followed by AEO, LEO and EO, which further demonstrates that the two mechanisms implemented are effective and the developed ACEO provides a superior performance. For f_7 and f_{13}, ACEO achieves similar outcomes to AEO and LEO, which show that the synergistic effect of the two mechanisms is less evident in solving these two functions. ACEO, AEO, LEO, and EO all find theoretical optima on f_5, f_{12}, f_{14}, f_{17} and f_{21}. The advantage of introducing two mechanisms on these functions is not apparent because the canonical EO has converged to the optimum when solving these functions. On the remaining functions, each of the two mechanisms introduced can independently enhance the canonical EO. Moreover, the two mechanisms can reach the best in synergy as can be intuitively observed from the convergence curves in Figure 6. Additionally, Figure 6 highlights that AEO converges faster than LEO, while canonical EO performs the worst, which suggests that both introduced mechanisms can improve the performance of canonical EO, in which the adaptive gbest-guided search mechanism contributes more than the chaos mechanism. This is because the inertia weight in the adaptive gbest-guided search mechanism plays an important role throughout the iteration process. Ultimately, the two implemented mechanisms are extremely advantageous and enhance the overall performance remarkably.

Table 4. Comparisons of EO, AEO, LEO, and ACEO on 23 100-dimensional functions.

Function	Results	EO	AEO	LEO	ACEO
f_1	Mean	4.42×10^{-29}	3.17×10^{-291}	2.49×10^{-178}	0
	Std	9.42×10^{-29}	0	0	0
	f-rank	4	2	3	1
f_2	Mean	1.21×10^{-29}	1.08×10^{-291}	4.52×10^{-179}	0
	Std	1.78×10^{-29}	0	0	0
	f-rank	4	2	3	1
f_3	Mean	20.3786	1.69×10^{-251}	2.28×10^{-147}	0
	Std	74.1112	0	4.89×10^{-147}	0
	f-rank	4	2	3	1
f_4	Mean	2.5457	2.68×10^{-138}	2.34×10^{-82}	4.10×10^{-215}
	Std	13.9351	7.33×10^{-138}	2.29×10^{-82}	0
	f-rank	4	2	3	1
f_5	Mean	0	0	0	0
	Std	0	0	0	0
	f-rank	1	1	1	1
f_6	Mean	8.62×10^{-51}	0	0	0
	Std	1.55×10^{-50}	0	0	0
	f-rank	4	1	1	1
f_7	Mean	0.0024	1.72×10^{-04}	1.00×10^{-04}	1.22×10^{-05}
	Std	0.0012	1.65×10^{-04}	6.99×10^{-05}	9.33×10^{-05}
	f-rank	4	3	2	1
f_8	Mean	5.03×10^{-127}	0	0	0
	Std	2.65×10^{-126}	0	0	0
	f-rank	4	1	1	1
f_9	Mean	1.46×10^{-25}	8.51×10^{-288}	3.62×10^{-174}	0
	Std	2.80×10^{-25}	0	0	0
	f-rank	4	2	3	1
f_{10}	Mean	1.90×10^{-48}	0	0	0
	Std	5.96×10^{-48}	0	0	0
	f-rank	4	1	1	1
f_{11}	Mean	7.44×10^{-66}	0	0	0
	Std	2.58×10^{-65}	0	0	0
	f-rank	4	1	1	1
f_{12}	Mean	0	0	0	0
	Std	0	0	0	0

Continued on next page

Function	Results	EO	AEO	LEO	ACEO
	f-rank	1	1	1	1
f_{13}	Mean	3.58×10^{-14}	8.88×10^{-16}	4.44×10^{-15}	8.88×10^{-16}
	Std	7.87×10^{-15}	0	0	0
	f-rank	4	1	3	1
f_{14}	Mean	0	0	0	0
	Std	0	0	0	0
	f-rank	1	1	1	1
f_{15}	Mean	5.03×10^{-18}	3.50×10^{-150}	8.81×10^{-93}	1.14×10^{-223}
	Std	5.82×10^{-18}	5.17×10^{-150}	1.26×10^{-92}	0
	f-rank	4	2	3	1
f_{16}	Mean	3.90×10^{-25}	1.18×10^{-289}	4.53×10^{-177}	0
	Std	1.89×10^{-24}	0	0	0
	f-rank	4	2	3	1
f_{17}	Mean	0	0	0	0
	Std	0	0	0	0
	f-rank	1	1	1	1
f_{18}	Mean	1.27×10^{-27}	1.14×10^{-290}	2.51×10^{-178}	0
	Std	4.57×10^{-27}	0	0	0
	f-rank	4	2	3	1
f_{19}	Mean	8.83×10^{-32}	2.86×10^{-294}	7.64×10^{-181}	0
	Std	1.42×10^{-31}	0	0	0
	f-rank	4	2	3	1
f_{20}	Mean	1.46×10^{-09}	1.32×10^{-75}	5.40×10^{-47}	0
	Std	6.53×10^{-10}	1.03×10^{-75}	2.22×10^{-47}	0
	f-rank	4	2	3	1
f_{21}	Mean	0	0	0	0
	Std	0	0	0	0
	f-rank	1	1	1	1
f_{22}	Mean	0.2910	8.84×10^{-63}	0.0025	0
	Std	0.0630	1.90×10^{-62}	0.0035	0
	f-rank	4	2	3	1
f_{23}	Mean	9.62×10^{-65}	0	0	0
	Std	4.21×10^{-64}	0	0	0
	f-rank	4	1	1	1
	Average f-rank	3.3478	1.5652	2.0870	1
	Overall f-rank	4	2	3	1

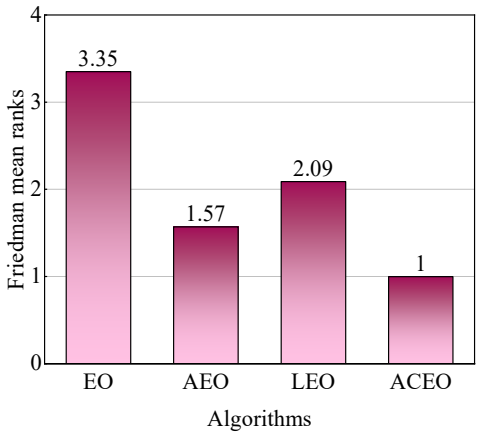

Figure 5. Chart of the Friedman's rank test results obtained by two mechanisms.

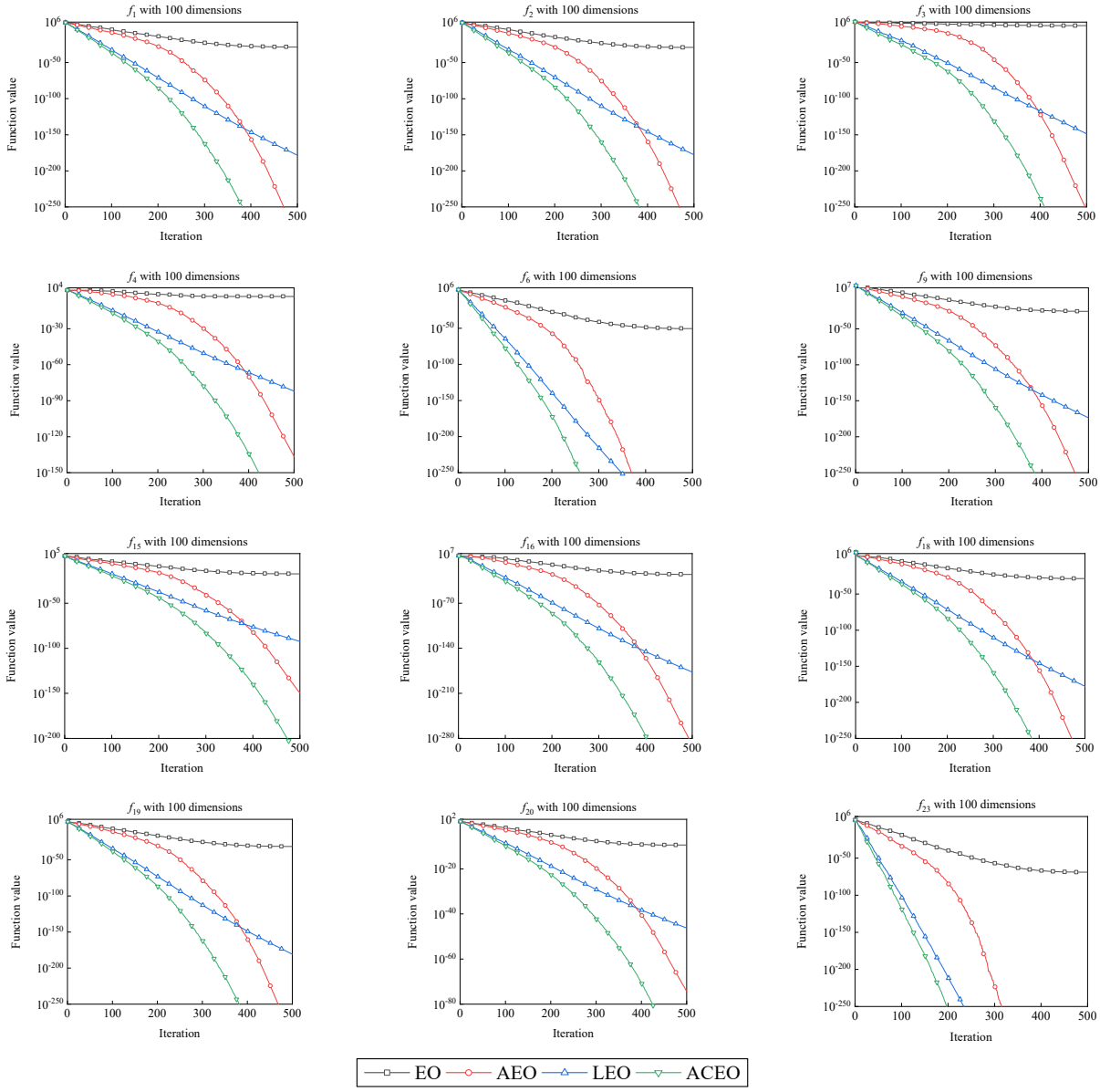

Figure 6. Convergence curves of EO, AEO, LEO and ACEO on some selected functions.

5. Application in path planning

In recent years, with the application and popularity of mobile robots, this technology has been developed rapidly. Path planning, which is the key technology of mobile robots, has also become a critical research direction for mobile robots. MRPP is the process of finding an optimal or near-optimal path around obstacles from a starting position to a target position in an established map environment.

In order to improve the efficiency and path smoothness of MRPP, a number of path planning algorithms have been introduced successively. The current traditional path-planning methods such as the A* algorithm, artificial potential field method, visibility graph method, and RRT algorithm all have corresponding disadvantages [45]. The A* algorithm can improve the search path efficiency, but it will consume a lot of energy if the target point is unreachable. The artificial potential field method is more likely to be trapped in the local optimum problem. The visibility graph method has difficulty obtaining the optimal path in the environment with more obstacle information. RRT is a stochastic sampling planning algorithm, its complexity is not influenced by the dispersion of the map and it still has high search accuracy in high-dimensional space. However, the paths that the RRT algorithm finds tend to have more path inflection points and longer path lengths. To deal with these disadvantages, and because of the increasing space complexity of the robot's environment, numerous metaheuristic algorithms have emerged to solve these problems. Li et al. [46] propose an MRPP technique based on a modified GA and dynamic window method. Wang et al. [47] employ an orthogonal opposition-based learning-driven dynamic SSA to deal with the MRPP problem. Wang et al. [48] propose a modified SSA method, and use the new SSA variant to resolve the MRPP problem. Parhi and Kashyap [49] employ a hybrid-based enhanced gravitational search algorithm (IGSA) to solve the best path planning of humanoid robots in rugged terrain. Wu et al. [50] propose a novel ACO variant (MAACO) to resolve the MRPP problem.

To address the problems such as low search efficiency, and many path redundancies in global MRPP, this study employs the developed ACEO to tackle these problems. The five maps provided in [51] are used to inspect the performance of the ACEO-based path-planning task. The environment maps are described in detail in Table 5. Furthermore, the ACEO algorithm is compared with some classical metaheuristics, such as SSA [12], GWO [10], artificial bee colony algorithm (ABC) [52], PSO [9], and firefly algorithm (FA) [53]. The general parameter settings are the same as in the original literature, and the parameter settings of ACEO are the same as in Section 4.2. The shortest path lengths produced by the six approaches on the five maps are listed in Table 6. The generated shortest path is marked in bold. The trajectory routes of the six algorithms are displayed in Figures 7–11.

From the data in Table 6, it can be noted that the developed ACEO can plan the shortest paths that avoid collision on all five terrains compared with other techniques. The trajectory routes generated by the six algorithms are shown in detail as follows.

Figure 7(a)–(f) plots the comparison of the trajectory routes of the six approaches on the first map. From Figure 7, this environment is a simple terrain, and all six algorithms can plan a collision-free route, and ACEO plans a shortest path. FA and GWO plot an identical path, which has more redundancy and inflection points, resulting in a longer path length. ABC, SSA and PSO design an identical trajectory path. The routes of PSO and ABC have no extra inflection points, but are not an optimal path. The route of SSA has more inflection points and the longest path length.

Figure 8(a)–(f) depicts the trajectory routes of the second environment map. It can be observed from the figure that ACEO plans the shortest route. The routes planned by FA and GWO have more inflection points than ACEO, producing longer routes. The routes of ABC and PSO have high search efficiency, but the routes are not optimal. The route of SSA gets stuck in a local optimum in the initial search phase, which leads to more path redundancy.

Table 5. Type of environment.

Terrain	No. of obstacles	Initial coordinates	Final coordinates	X axis	Y axis	Obstacle radius
Map 1	3	0, 0	4, 6	[1 1.8 4.5]	[1 5.0 0.9]	[0.8 1.5 1]
Map 2	6	0, 0	10, 10	[1.5 8.5 3.2 6.0 1.2 7.0]	[4.5 6.5 2.5 3.5 1.5 8.0]	[1.5 0.9 0.4 0.6 0.8 0.6]
Map 3	13	3, 3	14, 14	[1.5 4.0 1.2 5.2 9.5 6.5 10.8 5.9 3.4 8.6 11.6 3.3 11.8]	[4.5 3.0 1.5 3.7 10.3 7.3 6.3 9.9 5.6 8.2 8.6 11.5 11.5]	[0.5 0.4 0.4 0.8 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7]
Map 4	30	3, 3	14, 14	[10.1 10.6 11.1 11.6 12.1 11.2 11.7 12.2 12.7 13.2 11.4 11.9 12.4 12.9 13.4 8 8.5 9 9.5 10 9.3 9.8 10.3 10.8 11.3 5.9 6.4 6.9 7.4 7.9]	[8.8 8.8 8.8 8.8 8.8 11.7 11.7 11.7 11.7 11.7 9.3 9.3 9.3 9.3 9.3 5.3 5.3 5.3 5.3 5.3 6.7 6.7 6.7 6.7 6.7 8.4 8.4 8.4 8.4 8.4]	[0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4]
Map 5	45	0, 0	15, 15	[2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 6 6 6 8 8 8 8 8 8 8 8 8 10 10 10 10 10 10 10 10 10 12 12 12 12 12 14 14 14 14]	[8 8.5 9 9.5 10 10.5 3 3.5 4 4.5 5 5.5 6 6.5 7 11 11.5 12 1 1.5 2 2.5 3 3.4 4 4.5 5 6 6.5 7 7.5 8 8.5 9 9.5 10 10 10.5 11 11.5 12 10 10.5 11 11.5]	[0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4]

Figure 9(a)–(f) visually shows the routes of the six algorithms on the third map with 13 obstacle areas of different sizes. It can be seen that ACEO has the shortest route and the highest efficiency. The routes of FA and GWO have more inflection points than ACEO. PSO and ABC plan different styles of routes that have no redundancy and inflection points, but are not optimal paths. The route of SSA is trapped in a local optimum at the initial search phase generating redundancy and inflection points, which causes an increase in length.

The trajectories of the six algorithms on an environmental map with 30 threat areas are shown in Figure 10(a)–(f). It can be shown from the figure that ACEO has the highest efficiency. GWO and SSA plan a trajectory route that bypasses all threat areas. Although this route avoids collisions with obstacle areas, it increases the path length and the fuel consumption of the robot. The routes of FA and ABC have more path inflection points than ACEO, and the path of FA is not smooth. The route of PSO is another style of route, and the path is not smooth and has inflection points.

Figure 11(a)–(f) gives the simulation results of the fifth environmental map, which consists of 45 dangerous areas and is the most complex terrain among the five maps. The results show that the route of ACEO has the shortest and fastest efficiency. Compared with ACEO, the routes of PSO, ABC and FA are not optimal routes even though they do not have redundancy or redundant inflection points. The routes of GWO and SSA have more inflection points, and the path of GWO is not smooth. Based on the above analysis, the ACEO method can quickly find the optimal collision-free path that has better path quality, and improve operational efficiency, which is a feasible and effective path-planning algorithm.

Table 6. The minimum route length comparison of ACEO-based MRPP method and comparison methods under five environmental setups.

Terrain	ACEO Path length	SSA Path length	GWO Path length	ABC Path length	PSO Path length	FA Path length
Map1	**7.4222**	8.9979	7.6470	7.7817	7.7355	7.7055
Map2	**14.3051**	16.5335	14.4742	14.3513	14.3102	14.3321
Map3	**15.7640**	17.1855	16.0913	17.5347	15.8486	15.8782
Map4	**15.7321**	16.8938	16.3021	15.8795	16.2471	15.8196
Map5	**21.5114**	23.1577	21.9080	21.7147	21.9139	21.8897

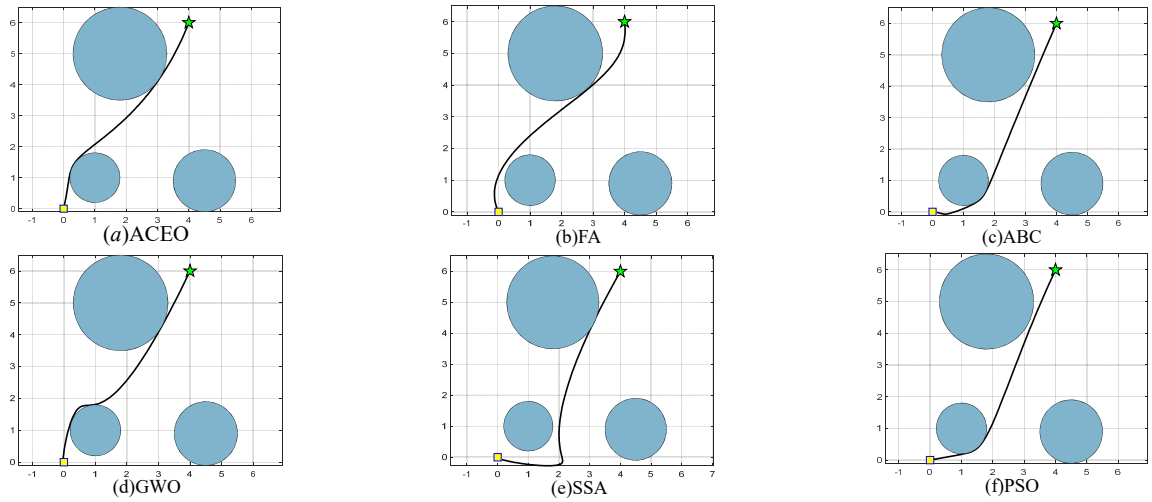

Figure 7. Map 1 (a) ACEO, (b) FA, (c) ABC, (d) GWO, (e) SSA and (f) PSO.

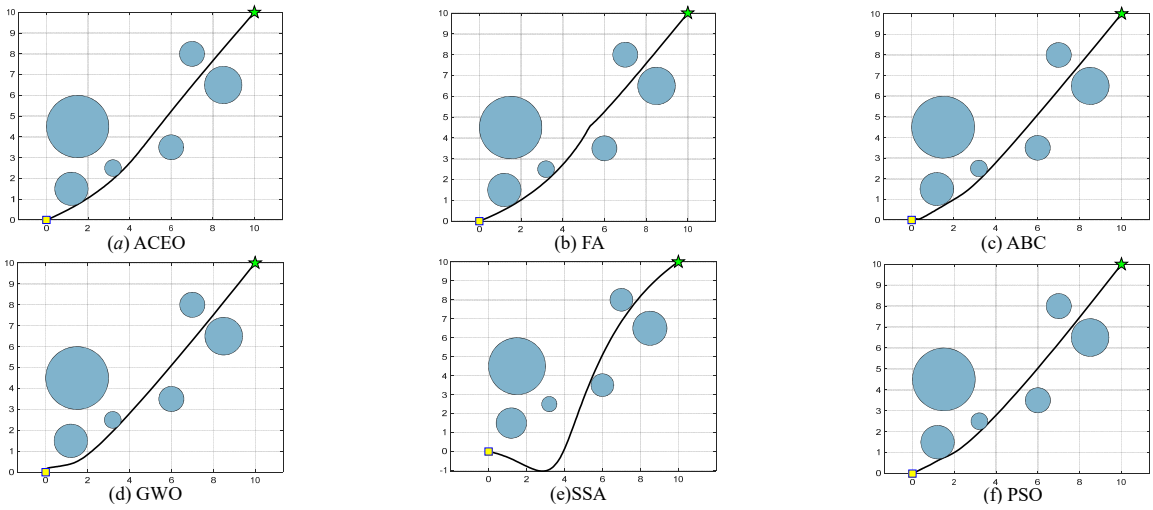

Figure 8. Map 2 (a) ACEO, (b) FA, (c) ABC, (d) GWO, (e) SSA and (f) PSO.

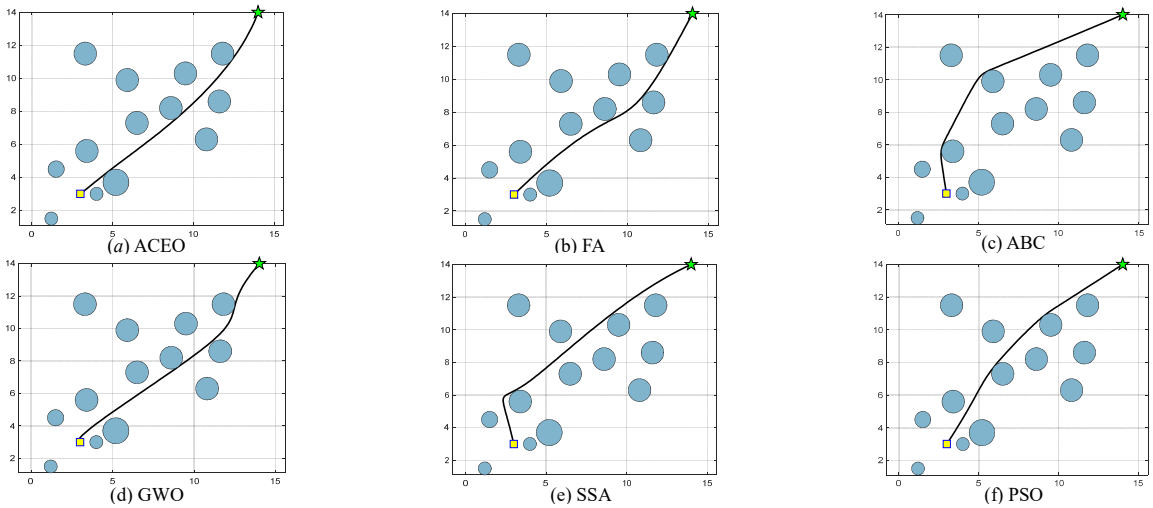

Figure 9. Map 3 (a) ACEO, (b) FA, (c) ABC, (d) GWO, (e) SSA and (f) PSO.

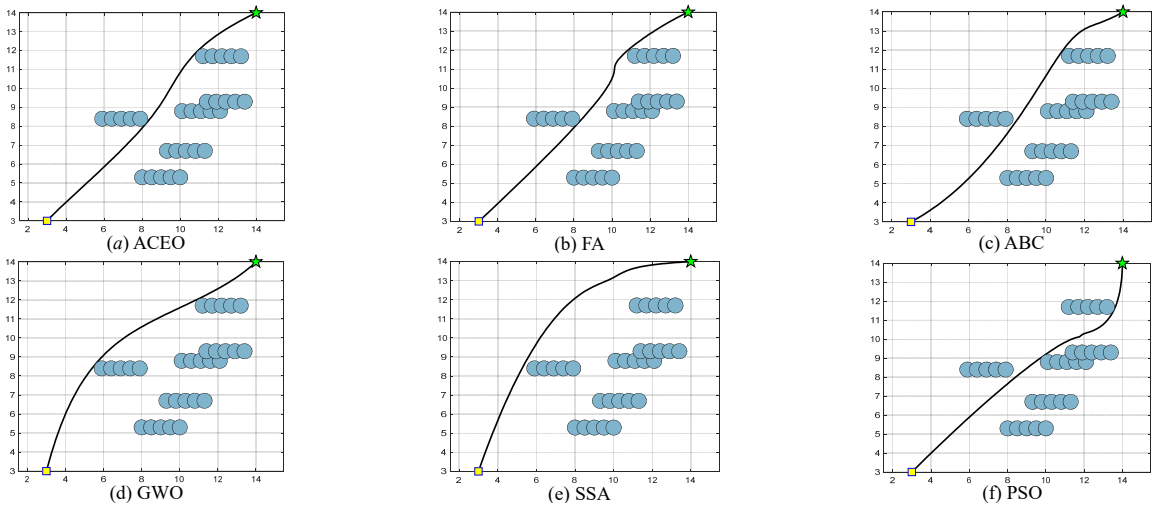

Figure 10. Map 4 (a) ACEO, (b) FA, (c) ABC, (d) GWO, (e) SSA and (f) PSO.

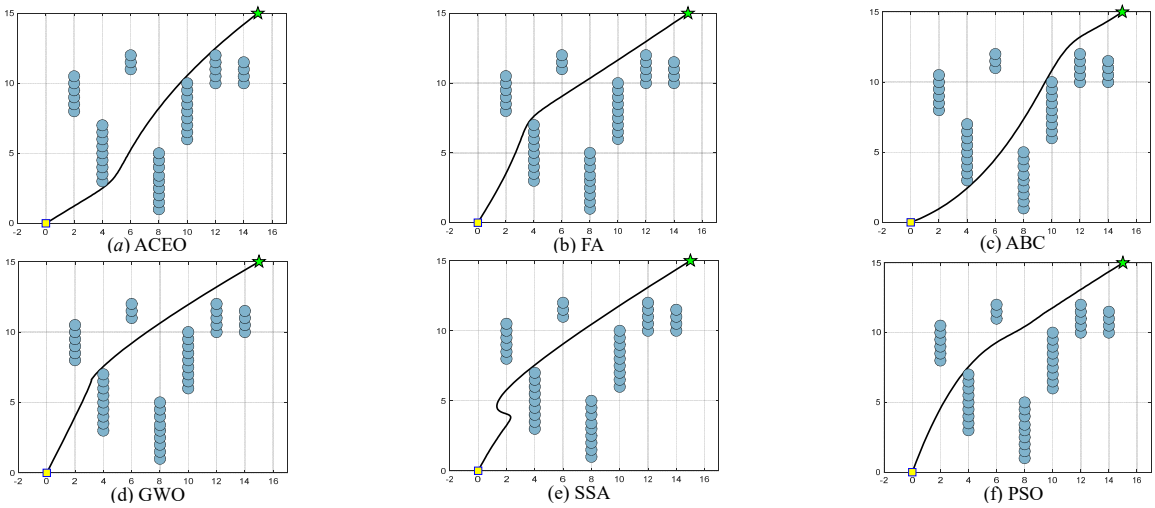

Figure 11. Map 5 (a) ACEO, (b) FA, (c) ABC, (d) GWO, (e) SSA and (f) PSO.

6. Conclusions

In this paper, a new EO version, known as ACEO, is developed to focus on solving the imbalance between exploration and exploitation and its tendency to fall into local optima of the canonical EO. The canonical EO has a simple structure, strong search capability and easy implementation. However, the position of the population is always determined by the equilibrium state equation, which causes the algorithm to fall into the local optimum and stagnation phenomenon, thus reducing the algorithm's ability to search for the optimal. Meanwhile, the candidate solutions are selected within the equilibrium pool by ranking according to the size of the fitness value, which increases the similarity risk of the obtained solutions and reduces the population diversity. Hence, the developed ACEO introduces two mechanisms to improve the EO performance. First, the inertia weight embedded by the adaptive gbest-guided search mechanism is adopted to enrich the population diversity of the equilibrium candidate. Next, a chaos mechanism is utilized to increase the possibility of avoiding dropping into a local optimum, thus improving the search capability. The combination of the two mechanisms achieves a proper balance between exploration and exploitation. The performance of the proposed ACEO is proven on 23 classical benchmark test functions and compared with canonical EO, EO variants, and other metaheuristic methods. The analysis reveals that the mechanism added based on the canonical EO is effective, and ACEO has the best performance and is a highly competitive algorithm. In addition, this paper employs the ACEO method to resolve the problems of slow convergence speed and high ratio of the number of path inflection points of traditional path-planning algorithms, and is compared with some classical metaheuristic techniques. The experimental results illustrate that the ACEO method is a prospective tool for solving the MRPP task. From the above research, ACEO presents enormous potential for addressing various optimization problems. Consequently, in our future work, we will primarily work on the application of the ACEO algorithm to tackle various practical engineering optimization problems, such as vehicle scheduling problems and feature selection problems.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (grant no. 61461053, 61461054, and 61072079), Yunnan Provincial Education Department Scientific Research Fund Project (grant no. 2022Y008), the Yunnan University's Research Innovation Fund for Graduate Students, China (grant no. Y-200211) and Ding Hongwei expert grassroots research station of Youbei Technology Co, Yunnan Province.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. W. Y. Wang, Z. H. Xu, Y. H. Fan, D. D. Pan, P. Lin, X. T Wang, Disturbance inspired equilibrium optimizer with application to constrained engineering design problems, *Appl. Math. Modell.*, **116** (2023), 254–276. https://doi.org/10.1016/j.apm.2022.11.016
2. F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, S. Mirjalili, Henry gas solubility optimization: A novel physics-based algorithm, *Future Gener. Comput. Syst.*, **101** (2019), 646–667. https://doi.org/10.1016/j.future.2019.07.015
3. Z. Wang, H. Ding, J. Yang, P. Hou, G. Dhiman, J. Wang, et al., Orthogonal pinhole-imaging-based learning salp swarm algorithm with self-adaptive structure for global optimization, *Front. Bioeng. Biotechnol.*, **10** (2022), 1018895. https://doi.org/10.3389/fbioe.2022.1018895
4. F. Zhao, L. Zhang, J. Cao, J. Tang, A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem, *Comput. Ind. Eng.*, **153** (2021), 107082. https://doi.org/10.1016/j.cie.2020.107082
5. Y. Zhang, Y. Zhou, G. Zhou, Q. Luo, An effective multi-objective bald eagle search algorithm for solving engineering design problems, *Appl. Soft Comput.*, **145** (2023) 110585. https://doi.org/10.1016/j.asoc.2023.110585
6. T. Zhang, Y. Zhou, G. Zhou, W. Deng, Q. Luo, Discrete Mayfly Algorithm for spherical asymmetric traveling salesman problem, *Expert Syst. Appl.*, **221** (2023), 119765. https://doi.org/10.1016/j.eswa.2023.119765
7. A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, *Knowl.-Based Syst.*, **191** (2020), 105190. https://doi.org/10.1016/j.knosys.2019.105190
8. K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, S. Dam, A genetic algorithm (GA) based load balancing strategy for cloud computing, *Proc. Technol.*, **10** (2013), 340–347. https://doi.org/10.1016/j.protcy.2013.12.369
9. R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization: An overview, *Swarm Intell.*, **1** (2007), 33–57. https://doi.org/10.1007/s11721-007-0002-0
10. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
11. E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, *Inf. Sci.*, **179** (2009), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004
12. S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Software*, **114** (2017), 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
13. N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evol. Comput.*, **11** (2003), 1–18. https://doi.org/10.1162/106365603321828970
14. C. Zhong, G. Li, Z. Meng, W. He, Opposition-based learning equilibrium optimizer with Levy flight and evolutionary population dynamics for high-dimensional global optimization problems, *Expert Syst. Appl.*, **215** (2023), 119303. https://doi.org/10.1016/j.eswa.2022.119303

15. Y. Sun, J. S. Pan, P. Hu, S. C. Chu, Enhanced equilibrium optimizer algorithm applied in job shop scheduling problem, *J. Intell. Manuf.*, **34** (2023), 1639–1665. https://doi.org/10.1007/s10845-021-01899-5
16. E. H. Houssein, E. Çelik, M. A. Mahdy, R. M. Ghoniem, Self-adaptive equilibrium optimizer for solving global, combinatorial, engineering, and multi-objective problems, *Expert Syst. Appl.*, **195** (2022), 116552. https://doi.org/10.1016/j.eswa.2022.116552
17. M. Premkumar, P. Jangir, R. Sowmya, H. H. Alhelou, S. Mirjalili, B. S. Kumar, Multi-objective equilibrium optimizer: Framework and development for solving multi-objective optimization problems, *J. Comput. Des. Eng.*, **9** (2022), 24–50. https://doi.org/10.1093/jcde/qwab065
18. E. H. Houssein, M. H. Hassan, M. A. Mahdy, S. Kamel, Development and application of equilibrium optimizer for optimal power flow calculation of power system, *Appl. Intell.*, **53** (2023), 7232–7253. https://doi.org/10.1007/s10489-022-03796-7
19. R. M. Rizk-Allah, A. E. Hassanien, A hybrid equilibrium algorithm and pattern search technique for wind farm layout optimization problem, *ISA Trans.*, **132** (2023), 402–418. https://doi.org/10.1016/j.isatra.2022.06.014
20. Q. Luo, S. Yin, G. Zhou, W. Meng, Y. Zhao, Y. Zhou, Multi-objective equilibrium optimizer slime mould algorithm and its application in solving engineering problems, *Struct. Multidisc. Optim.*, **66** (2023), 114. https://doi.org/10.1007/s00158-023-03568-y
21. B. Zhu, Q. Luo, Y. Zhou, Quantum-inspired equilibrium optimizer for linear antenna array, *Comput. Model. Eng. Sci.*, **137** (2023), 385–413. https://doi.org/10.32604/cmes.2023.026097
22. S. Yin, Q. Luo, Y. Zhou, EOSMA: an equilibrium optimizer slime mould algorithm for engineering design problems, *Arab. J. Sci. Eng.*, **47** (2022), 10115–10146. https://doi.org/10.1007/s13369-021-06513-7
23. J. J. Wang, L. Wang, A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling, *Comput. Ind. Eng.*, **168** (2022), 108126. https://doi.org/10.1016/j.cie.2022.108126
24. F. Zhao, H. Zhang, L. Wang, A pareto-based discrete jaya algorithm for multiobjective carbon-efficient distributed blocking flow shop scheduling problem, *IEEE Trans. Ind. Inform.*, **19** (2022), 8588–8599. https://doi.org/10.1109/TII.2022.3220860
25. Z. Wang, H. Ding, J. Wang, P. Hou, A. Li, Z. Yang, et al., Adaptive guided salp swarm algorithm with velocity clamping mechanism for solving optimization problems, *J. Comput. Des. Eng.*, **9** (2022), 2196–2234. https://doi.org/10.1093/jcde/qwac094
26. V. K. Pathak, A. K. Srivastava, A novel upgraded bat algorithm based on cuckoo search and Sugeno inertia weight for large scale and constrained engineering design optimization problems, *Eng. Comput.*, **38** (2022), 1731–1758. https://doi.org/10.1007/s00366-020-01127-3
27. Y. Chen, J. Xi, H. Wang, X. Liu, Grey wolf optimization algorithm based on dynamically adjusting inertial weight and levy flight strategy, *Evol. Intell.*, **16** (2023) 917–927. https://doi.org/10.1007/s12065-022-00705-2
28. H. Ding, X. Cao, Z. Wang, G. Dhiman, P. Hou, J. Wang, et al., Velocity clamping-assisted adaptive salp swarm algorithm: balance analysis and case studies, *Math. Biosci. Eng.*, **19** (2022), 7756–7804. https://doi.org/10.3934/mbe.2022364
29. C. Yin, S. Mao, Fractional multivariate grey Bernoulli model combined with improved grey wolf algorithm: Application in short-term power load forecasting, *Energy*, **269** (2023), 126844. https://doi.org/10.1016/j.energy.2023.126844

30. H. Gezici, H. Livatyalı, Chaotic Harris hawks optimization algorithm, *J. Comput. Des. Eng.*, **9** (2022), 216–245. https://doi.org/10.1093/jcde/qwab082
31. S. Liang, Y. Pan, H. Zhang, J. Zhang, F. Wang, Z. Chen, Marine predators algorithm based on adaptive weight and chaos factor and its application, *Sci. Program.*, (2022), 4623980. https://doi.org/10.1155/2022/4623980
32. J. Feng, H. Kuang, L. Zhang, EBBA: An enhanced binary bat algorithm integrated with chaos theory and lévy flight for feature selection, *Future Internet*, **14** (2022), 178. https://doi.org/10.3390/fi14060178
33. F. S. Gharehchopogh, M. H. Nadimi-Shahraki, S. Barshandeh, B. Abdollahzadeh, H. Zamani, CQFFA: A chaotic quasi-oppositional farmland fertility algorithm for solving engineering optimization problems, *J. Bionic Eng.*, **20** (2023), 158–183. https://doi.org/10.1007/s42235-022-00255-4
34. S. K. Joshi, Chaos embedded opposition based learning for gravitational search algorithm, *Appl. Intell.*, **53** (2023), 5567–5586. https://doi.org/10.1007/s10489-022-03786-9
35. W. Long, J. Jiao, X. Liang, T. Wu, M. Xu, S. Cai, Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection, *Appl. Soft Comput.*, **103** (2021), 107146. https://doi.org/10.1016/j.asoc.2021.107146
36. S. Gupta, K. Deep, S. Mirjalili, An efficient equilibrium optimizer with mutation strategy for numerical optimization, *Appl. Soft Comput.*, **96** (2020), 106542. https://doi.org/10.1016/j.asoc.2020.106542
37. J. Liu, W. Li, Y. Li, LWMEO: An efficient equilibrium optimizer for complex functions and engineering design problems, *Expert Syst. Appl.*, **198** (2022), 116828. https://doi.org/10.1016/j.eswa.2022.116828
38. X. Zhang, Q. Lin, Information-utilization strengthened equilibrium optimizer, *Artif. Intell. Rev.*, **55** (2022), 4241–4274. https://doi.org/10.1007/s10462-021-10105-0
39. L. Yang, Z. Xu, Y. Liu, G. Tian, An improved equilibrium optimizer with a decreasing equilibrium pool, *Symmetry*, **14** (2022), 1227. https://doi.org/10.3390/sym14061227
40. H. Ren, J. Li, H. Chen, C. Li, Stability of salp swarm algorithm with random replacement and double adaptive weighting, *Appl. Math. Modell.*, **95** (2021), 503–523. https://doi.org/10.1016/j.apm.2021.02.002
41. M. M. Saafan, E. M. El-Gendy, IWOSSA: An improved whale optimization salp swarm algorithm for solving optimization problems, *Expert Syst. Appl.*, **176** (2021), 114901. https://doi.org/10.1016/j.eswa.2021.114901
42. S. Dhargupta, M. Ghosh, S. Mirjalili, R. Sarkar, Selective opposition based grey wolf optimization, *Expert Syst. Appl.*, **151** (2020), 113389. https://doi.org/10.1016/j.eswa.2020.113389
43. X. Yu, W. Xu, C. Li, Opposition-based learning grey wolf optimizer for global optimization, *Knowl.-Based Syst.*, **226** (2021), 107139. https://doi.org/10.1016/j.knosys.2021.107139
44. L. Ma, C. Wang, N. Xie, M. Shi, Y. Ye, L. Wang, Moth-flame optimization algorithm based on diversity and mutation strategy, *Appl. Intell.*, **51** (2021), 5836–5872. https://doi.org/10.1007/s10489-020-02081-9
45. Z. Wang, H. Ding, B. Li, L. Bao, Z. Yang, Q. Liu, Energy efficient cluster based routing protocol for WSN using firefly algorithm and ant colony optimization, *Wireless Pers. Commun.*, **125** (2022), 2167–2200. https://doi.org/10.1007/s11277-022-09651-9

46. Y. Li, J. Zhao, Z. Chen, G. Xiong, S. Liu, A robot path planning method based on improved genetic algorithm and improved dynamic window approach, *Sustainability*, **15** (2023), 4656. https://doi.org/10.3390/su15054656
47. Z. Wang, H. Ding, J. Yang, J. Wang, B. Li, Z. Yang, et al., Advanced orthogonal opposition-based learning-driven dynamic salp swarm algorithm: framework and case studies, *IET Control Theory Appl.*, **16** (2022), 945–971. https://doi.org/10.1049/cth2.12277
48. Z. Wang, H. Ding, Z. Yang, B. Li, Z. Guan, L. Bao, Rank-driven salp swarm algorithm with orthogonal opposition-based learning for global optimization, *Appl. Intell.*, **52** (2022), 7922–7964. https://doi.org/10.1007/s10489-021-02776-7
49. D. R. Parhi, A. K. Kashyap, Humanoid robot path planning using memory-based gravity search algorithm and enhanced differential evolution approach in a complex environment, *Expert Syst. Appl.*, **215** (2023), 119423. https://doi.org/10.1016/j.eswa.2022.119423
50. L. Wu, X. Huang, J. Cui, C. Liu, W. Xiao, Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot, *Expert Syst. Appl.*, **215** (2023), 119410. https://doi.org/10.1016/j.eswa.2022.119410
51. D. Agarwal, P. S. Bharti, Implementing modified swarm intelligence algorithm based on Slime moulds for path planning and obstacle avoidance problem in mobile robots, *Appl. Soft Comput.*, **107** (2021), 107372. https://doi.org/10.1016/j.asoc.2021.107372
52. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J Glob. Optim.*, **39** (2007), 459–471. https://doi.org/10.1007/s10898-007-9149-x
53. I. Fister, I. Fister Jr, X. Yang, J. Brest, A comprehensive review of firefly algorithms, *Swarm Evol. Comput.*, **13** (2013), 34–46. https://doi.org/10.1016/j.swevo.2013.06.001

AIMS Press