The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-05905-4

Love Evolution Algorithm: a stimulus–value–role theory-inspired evolutionary algorithm for global optimization

Yuansheng Gao[1] · Jiahui Zhang[2] · Yulin Wang[3] · Jinpeng Wang[1] · Lang Qin[4]

Accepted: 4 January 2024

Abstract
This paper proposes the Love Evolution Algorithm (LEA), a novel evolutionary algorithm inspired by the stimulus–value–role theory. The optimization process of the LEA includes three phases: stimulus, value, and role. Both partners evolve through these phases and benefit from them regardless of the outcome of the relationship. This inspiration is abstracted into mathematical models for global optimization. The efficiency of the LEA is validated through numerical experiments with CEC2017 benchmark functions, outperforming seven metaheuristic algorithms as evidenced by the Wilcoxon signed-rank test and the Friedman test. Further tests using the CEC2022 benchmark functions confirm the competitiveness of the LEA compared to seven state-of-the-art metaheuristics. Lastly, the study extends to real-world problems, demonstrating the performance of the LEA across eight diverse engineering problems. Source codes of the LEA are publicly available at https://ww2.mathworks.cn/matlabcentral/fileexchange/159101-love-evolution-algorithm.

Keywords Optimization · Metaheuristic · Evolutionary algorithm · Love Evolution Algorithm

1 Introduction

Optimization techniques are widely employed to optimize the design of real-world problems to raise the efficiency of systems, human resources, etc. However, the complexity of most practical problems, characterized by numerous design

✉ Yuansheng Gao
gaoyuansheng2021@163.com

[1] College of Science, Liaoning Technical University, Fuxin 123000, China

[2] College of Mining Engineering, Liaoning Technical University, Fuxin 123000, China

[3] College of Electrical and Control Engineering, Liaoning Technical University, Huludao 125105, China

[4] School of Mechanics and Engineering, Liaoning Technical University, Fuxin 123000, China

Published online: 12 February 2024

variables and constraints, often exceeds the capability of classical optimization algorithms [1, 2]. Numerous metaheuristic algorithms are designed to comprehensively explore the search space, utilizing practical information to circumvent local optimization pitfalls. Additionally, metaheuristic algorithms offer significant advantages, including gradient-free operations, reduced computational complexity, and enhanced flexibility. With these advantages, metaheuristic algorithms have been widely used in many fields of optimization problems and have been paid more and more attention by many scholars [3].

Metaheuristic algorithms have two most important behaviors: exploration and exploitation [4]. In the early iterative phase, algorithms prefer to search the entire solution space extensively, which is related to avoiding local optima. If over-explored, the quality of the solution will be poor. Conversely, if over-exploited, it can make the algorithms vulnerable to local optima. Therefore, achieving a satisfactory solution necessitates a good balance between exploration and exploitation.

Recently, the application of metaheuristic algorithms to highly complex problems has seen a substantial increase. Unlike traditional optimization algorithms, metaheuristic algorithms are gradient-free optimization techniques for solving near-optimal solutions, which can solve black-box optimization problems similar to machine learning algorithms with hyperparametric optimization [5]. Metaheuristic algorithms are considered to be of greater research value due to the consideration of its irreplaceable advantages in certain problems, like black-box optimization problems. As posited by the no free lunch theorem [6], it is acknowledged that no individual metaheuristic algorithm is universally effective for all optimization problems. In other words, the different characteristics exhibited by different problems make the metaheuristic algorithms behave differently during the optimization process, which results in varied performance of metaheuristic algorithms; some may excel in certain problem classes while faltering in others. Although numerous metaheuristic algorithms are proposed nowadays, we still need to develop new metaheuristic algorithms that provide new search processes to adapt to possible problems to be solved.

The proposed Love Evolution Algorithm (LEA) is an evolutionary algorithm inspired by the stimulus–value–role (SVR) theory [7]. Drawing from the SVR theory, the search process of the LEA encompasses three distinct phases: the stimulus phase, the value phase, and the role phase. The greatest novelty of LEA lies in its unique search operations, including inter-variable convolution, multiplication, and division for crossover and mutation. In addition, the proposed LEA is verified to be highly competitive using the CEC2017 and CEC2022 benchmark functions. Eight real-world optimization problems were used to validate the capability of the LEA to solve practical problems.

The main contributions of this paper are as follows:

(1) Proposed an evolutionary algorithm: the Love Evolution Algorithm.
(2) Tested the proposed algorithm on 41 benchmark functions.

(3) Verified the competitiveness of the proposed algorithm in comparison with strong metaheuristic algorithms.
(4) Proposed algorithm was used to solve eight real-world optimization problems.

The structure of the remainder of this paper is as follows: Related studies on metaheuristic algorithms are presented in Sect. 2. Section 3 provides a detailed introduction to the proposed LEA. Section 4 tests and analyzes the LEA on CEC2017 and CEC2022 benchmark functions. In Sect. 5, the LEA is applied to eight real-world optimization problems. Section 6 is the conclusion of this paper and the future research direction.

2 Related studies

2.1 Classification of metaheuristic algorithms

There are hundreds of metaheuristic algorithms, most inspired by nature. Among the many algorithms, the swarm intelligence class has the largest number of algorithms [8]. Generally, these algorithms are bifurcated into two primary categories: single-solution-based and population-solution-based metaheuristics. The classification of the algorithms is given in Fig. 1.

Current single-solution-based metaheuristic algorithms include the simulated annealing (SA) [9], the tabu search (TS) [10], the guided local search (GLS) [11], the iterated local search (ILS) [12], the random search (RS) [13], the variable neighborhood search (VNS) [14], and the large neighborhood search (LNS) [15].

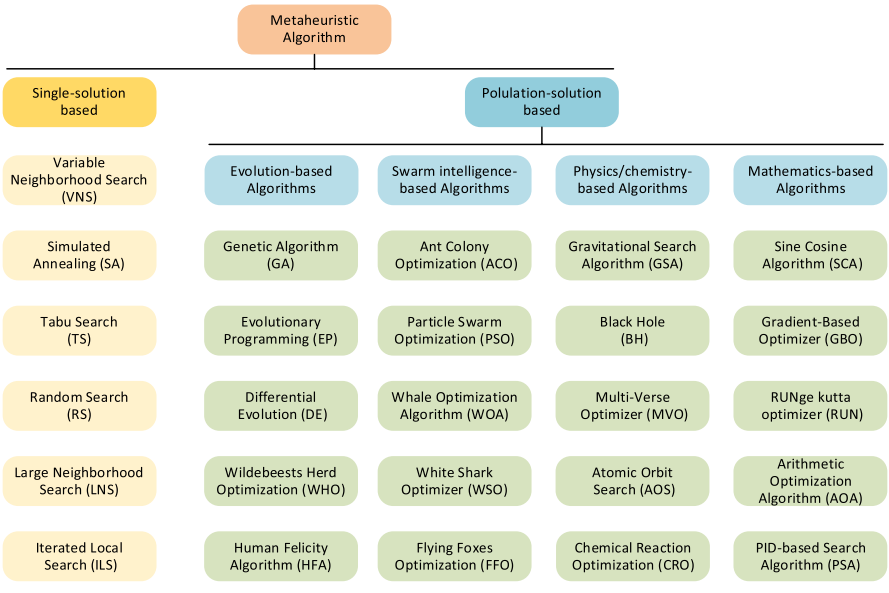

Fig. 1 Classification of metaheuristic algorithms

Population-solution-based metaheuristics are typically categorized into four distinct groups because of the different search methods, i.e., evolution-based algorithms, swarm-based algorithms, physics/chemistry-based algorithms, and mathematics-based algorithms.

Evolution-based algorithms draw inspiration from the genetic evolution of organisms. Compared with traditional optimization algorithms such as calculus-based methods and exhaustive methods, these algorithms are a mature global optimization method with high robustness and wide applicability. It has the characteristics of self-organization, self-adaptation, and self-learning, which can effectively deal with complex problems that are difficult to be solved by traditional optimization algorithms regardless of the nature of the problem. Genetic algorithm (GA) [16] is an optimization model that simulates Darwin's theory of biological evolution, which originates from the simulation of behaviors such as chromosomal crossover variation, acting in the genetic space where information is encoded. Such algorithms also include the evolutionary programming (EP) [17], the evolutionary strategy (ES) [18], the genetic programming (GP) [19], the differential evolution (DE) [20], the gene expression programming (GEP) [21], the biogeography-based optimization (BBO) [22], the differential search (DS) [23], the Wildebeests herd optimization (WHO) [24], and the human felicity algorithm (HFA) [25].

Swarm intelligence-based algorithms are a common algorithm in computing intelligence. These algorithms, for instance, simulate the natural behavior of fish, birds, wolves, and bacteria in nature. They utilize information exchange and cooperation among groups, optimizing through simple yet limited interactions among individuals. In 1992, M. Dorigo et al. proposed the ant colony optimization (ACO) [26] by simulating an ant colony to choose the shortest path from an anthill to a food source for obstacle avoidance. In 1995, J. Kennedy et al. proposed the particle swarm optimization (PSO) [27] inspired by the predatory behavior of bird flocks. Others have since proposed the bacterial foraging (BF) algorithm [28], the moth flame optimization (MFO) [29], the whale optimization algorithm (WOA) [30], the spotted hyena optimizer (SHO) [31], the butterfly optimization algorithm (BOA) [32], the Harris hawk optimization (HHO) [33], the tunicate swarm algorithm (TSA) [34], the African vultures optimization algorithm (AVOA) [35], the snake optimizer (SO) [36], the white shark optimizer(WSO) [37], the dwarf mongoose optimization (DMO) [38], the flying foxes optimization (FFO) [39], the escape bird search (EBS) [40], the FOX optimizer (FOX) [41], the walrus optimizer (WO) [42], and the snow geese algorithm (SGA) [43].

Physics/chemistry-based algorithms are inspired by the major physics and chemistry rules found in the universe. These rules usually constrain the interaction of searching individuals in such methods. Moreover, most of these laws are related to gravity, electromagnetic force, chemical reaction, etc. Examples of these algorithms include the Big Bang–Big Crunch (BBBC) [44], the gravitational search algorithm (GSA) [45], the chemical reaction optimization (CRO) [46], the artificial chemical

reaction optimization algorithm (ACROA) [47], the black hole (BH) algorithm [48], the multi-verse optimizer (MVO) [49], the thermal exchange optimization (TEO) [50], the Archimedes optimization algorithm (AOA) [51], the equilibrium optimizer (EO) [52], the string theory algorithm (STA) [53], and the atomic orbit search (AOS) [54].

Mathematics-based algorithms are a new type of metaheuristic algorithms that have been gradually proposed in recent years for classification. It is not inspired by the complex group life of species or some difficult physical phenomena like other metaheuristic algorithms but is mainly inspired by the arithmetic laws or some basic mathematical formulas in the field of mathematics. For example, sine–cosine algorithm (SCA) [55], proposed by Seyedali Mirjalili in 2016, enables each individual to adjust the direction of motion and thus search the whole space according to the fluctuation changes of sine and cosine functions, which enables effective global exploration. In recent years, scholars have also proposed the gradient-based optimizer (GBO) [56], the Runge–Kutta optimizer (RUN) [57], the arithmetic optimization algorithm (AOA) [58], the weighted mean of vectors (INFO) algorithm [59], the Lévy flight distribution (LFD) [60], the PID-based search algorithm (PSA) [61], and the cubature Kalman optimizer (CKO) [62].

2.2 Basic elements of metaheuristic algorithms

The solution process of metaheuristic algorithms is usually divided into initialization and iterative optimization. The initialization phase usually consists of initializing the parameters, creating a population and a vector containing the values of the objective function. Moreover, the iterative optimization phase consists mainly of selecting guide individual, searching (exploring and exploiting), and updating the population. Note that most of these descriptions are for population-solution-based metaheuristic algorithms.

There are many ways to initialize the population, the most common method is to initialize it randomly in the search space using random numbers obeying a uniform distribution. In addition, initializing the population by chaotic map [63] is an effective improvement, which usually leads to a more even distribution of the population. A more comprehensive description of population initialization methods can be found in the literature [64].

Once the population has been initialized, selection of guide individuals generally begins. Selection methods used by metaheuristic algorithms can be generally categorized into three groups: random selection methods, probabilistic selection methods, and greedy selection methods. In addition, fitness–distance balance (FDB) [65] has been proposed as a greedy selection method in recent years and has successfully improved many algorithms. In the face of constrained optimization problems, these methods mentioned above will not consider whether the best individual violates the

constraints when choosing the guided individual. To solve this problem well, Burcin et al. proposed the fitness–distance–constraint-based guide selection method [66].

The search operation of different algorithms varies because of the design of the formulas. In general, many strategies are employed to balance the exploration and exploitation of the algorithm. A popular strategy is to use Lévy flight [67] to increase the possibility of the algorithm jumping out of the local optimal solutions and drive the algorithm to explore. In addition, random walk [68], adaptive weighting [69], double learning [70], etc. are also used to improve search quality. There is also linear population size reduction [71] mechanism in population size.

In metaheuristic algorithms, the updating mechanism of the population considers only the fitness or directly retains the new individuals to the next generation. For example, the GBO [56] retains individuals with good fitness value. Besides, the PSA [61] directly retains all newly produced individuals into the next generation. These methods, while simple and commonly applied, may suffer from the problem of premature convergence of the algorithm due to improper selection of individuals. Surprisingly, the natural survivor method (NSM) [72] proposed by Hamdi Tolga Kahraman et al. avoids premature convergence of algorithms to some extent. NSM calculates scores that represent an adaptation of an individual to nature to identify survivors, discarding the greedy survival process based on fitness values.

3 Love Evolution Algorithm

This section focuses on the inspiration, mathematical model, and pseudocode of the proposed algorithm. A theoretical analysis of time complexity and space complexity is also given.

3.1 Inspiration

3.1.1 Stimulus-value-role theory

Falling in love is an intimate relationship that a person is able to establish autonomously, voluntarily and freely in the course of his or her life, and it is an expression of adoration between people of the opposite or the same sex. An influential psychological theory of relationships was proposed by the American psychologist Murstein in 1970: the stimulus–value–role (SVR) theory [7]. SVR theory divides falling in love (marriage choice) into three phases, which are the stimulus phase, the value phase, and the role phase. The stimulus phase includes value satisfaction through visual, auditory, and non-interactive means. The value phase consists of values appreciated through verbal interaction. The role phase involves the couple's ability to function in mutually assigned roles.

In every relationship, whether it fails or succeeds, both partners change to a greater or lesser extent. This change may be positive or negative. However, it is undeniable that on the road to finding love, we have more or less progressed in terms of our minds, perceptions and so on. The SVR theory provides an important perspective and

analytical path for us to explain meeting–loving–getting along. It is also enlightened that the process of falling in love bears some similarity to the optimization process of the metaheuristic algorithm.

3.1.2 Abstractions and metaphors

Variable Different people have different temperaments, personalities, hobbies and so on. In this paper, these are uniformly called the characteristics of a person. Then, a certain characteristic is abstracted as a variable of a certain dimension.

Candidate solution Obviously, the happiness of both partners in the relationship process is closely related to these characteristics. Therefore, the entire set of characteristics that a person possesses is abstracted into a candidate solution.

Objective function value The combination of these characteristics affects all aspects of a person and determines a person's happiness. Thus, the happiness degree

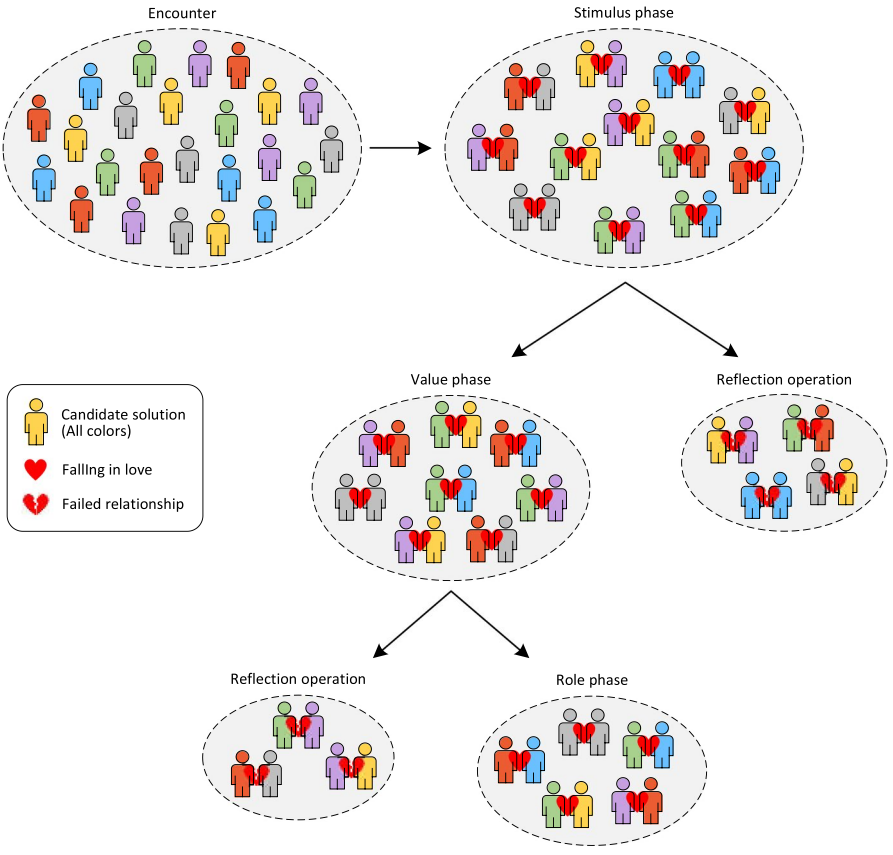

Fig. 2 Structure of the optimization process

of a particular person is abstracted as the objective function value of a particular candidate solution.

The best solution In the course of falling in love, people's characteristics progress. Thus, the characteristics of the best people in history are metaphorically used as the best solution. Then, a particular best characteristic can be likened to a particular variable of the best solution.

Optimization process Inspired by SVR theory, the optimization process of the proposed algorithm is divided into three phases: stimulus phase, value phase, and role phase. Inevitably, breakups will occur during these three phases. After the breakup, both partners will reflect and improve. This situation is abstracted as reflection operation. The structure of the optimization process is shown in Fig. 2.

3.2 Mathematical model and algorithm

3.2.1 Initialization

The optimization problem usually consists of a set of decision variables, constraints and an objective function. It may be assumed that the number of decision variables is d and the upper and lower bounds of the variables are $\mathbf{u}$ and $\mathbf{l}$, respectively (1 row and d columns). The maximum number of function evaluations (MaxFEs) is denoted as T. Since the number of people in love is two, the number of people in the population should be an even number. This does not prevent optimization, so it is reasonable to specify an even number of people n for this algorithm. The population at the initial moment $\mathbf{X}$ is

$$\mathbf{X} = (\mathbf{u} - \mathbf{l}) \cdot \mathbf{R} + \mathbf{l}, \tag{1}$$

where $\mathbf{R}$ is a matrix of n rows and d columns composed of random numbers between 0 and 1. Then $\mathbf{X}$ is a matrix of n rows and d columns. Its matrix form is

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d-1} & x_{1,d} \\ x_{2,1} & x_{1,2} & \cdots & x_{2,d-1} & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n-1,1} & x_{n-1,2} & \cdots & x_{n-1,d-1} & x_{n-1,d} \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d-1} & x_{n,d} \end{pmatrix}_{n \times d} \tag{2}$$

Taking the minimization problem as an example, the happiness degree of each person is $\mathbf{H}$. The smaller the $\mathbf{H}$, the happier people are. Then, the characteristics of the best people in history is defined as

$$\mathbf{G} = \mathbf{X}_{\text{find}(\mathbf{H}=\min(\mathbf{H}))} \tag{3}$$

where min (·) is the function that takes the minimum value and find(·) is the function that gets the index of the equal values of **H** and min (**H**).

3.2.2 Encounter

Consider the random nature of people's acquaintance and the fact that this randomness may be able to increase the diversity of the proposed algorithm. A randomized strategy is used in the encounter, i.e., the romantic partners are generated randomly. The generation method is as follows

$$\begin{cases} \mathbf{r} = \text{randperm}(d) \\ \mathbf{a} = \left[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{n/2}\right] \\ \mathbf{b} = \left[\mathbf{r}_{n/2+1}, \mathbf{r}_{n/2+2}, \ldots, \mathbf{r}_n\right] \end{cases} \tag{4}$$

where randperm(d) denotes the random integer arrangement that generates 1 to d; and **a** and **b** are the indexes of the love partners **A** and **B**, respectively. Then **A** and **B** can be expressed as

$$\mathbf{A} = \mathbf{X}_{\mathbf{a}}; \mathbf{B} = \mathbf{X}_{\mathbf{b}} \tag{5}$$

Apparently, $\mathbf{A}_i$ and $\mathbf{B}_i$ are a couple, $i = 1, 2, \ldots n/2$. In this way, the population is divided into two parts, i.e., $\mathbf{A} \cup \mathbf{B} = \mathbf{X}$.

3.2.3 Stimulus phase

During the stimulus phase, both partners are stimulated by their respective appearances, behaviors, and personalities. Understanding at this phase is generally superficial and hardly makes it possible for the respective characteristics to be influenced. Therefore, this phase does not involve updating of characteristics. In other words, no candidate solutions are updated.

However, at this phase, an acceptance degree was proposed to measure the fit between the partners during the stimulus phase. Since the two partners at this phase know each other superficially and do not have the knowledge to explore the "characteristics" in depth, the acceptance degree only takes into account the degree of proximity to the value of the objective function. The acceptance degree **c** is defined as

$$\begin{cases} \mathbf{c} = \mathbf{r}_{\mathbf{c}} \cdot \left(\mathbf{H}_{\mathbf{A}} - \mathbf{H}_{\mathbf{B}}\right) \cdot \left(\mathbf{H}_{\mathbf{A}} - \mathbf{H}_{\mathbf{B}}\right) \\ \mathbf{c} = \mathbf{c} / \left(\max\left(\mathbf{c}\right) + \min\left(\mathbf{c}\right) + \varepsilon\right) \end{cases} \tag{6}$$

where $\mathbf{r}_{\mathbf{c}}$ is a vector of $n/2$ rows and 1 column consisting of 0.5 to 1.5 random numbers; $\mathbf{H}_{\mathbf{A}}$ and $\mathbf{H}_{\mathbf{B}}$ are the happiness degrees of **A** and **B** ($n/2$ rows and 1 column),

respectively; ε is a very small number greater than 0; and max $(\cdot)$ is the function that takes the maximum value. The $\mathbf{r_c}$ ensures that partners with smaller differences in well-being are also likely to perform reflective operation and partners with larger differences in well-being are likely to enter the next phase. This operation favors diversity.

When the acceptance degree is greater than λ_c (called acceptance rate, equal to 0.5), it is considered that the two partners break up and proceed to the reflection operation; otherwise, it is considered that the two partners continue to be in love and enter the value phase.

3.2.4 Reflection operation

After the breakup, the ith couple's reflection on the jth characteristic should be at the j th characteristic itself. This behavior is defined in Eq. (7).

$$s_{ij}^{A} = \alpha_A \frac{\mathbf{A}_{ij}}{\mathbf{B}_{ij} + \varepsilon}; s_{ij}^{B} = \alpha_B \frac{\mathbf{B}_{ij}}{\mathbf{A}_{ij} + \varepsilon} \tag{7}$$

where s_{ij}^{A} (s_{ij}^{B}) denotes the self-reflection operator of the ith $\mathbf{A}$ ($\mathbf{B}$) for the jth characteristic; and α_A and α_B are both a random number from -1.5 to 1.5.

The enhancement of the jth characteristic may be related to other characteristics in addition to the jth characteristic. An inspiring example is that if a person is a quiet character (one characteristic), then his hobby paintings (another characteristic) may be landscape paintings; if the person is a fanatic character (one characteristic), then his hobby paintings (another characteristic) may be crazy and abstract. The formula for this behavior is given in Eq. (8).

$$\delta = \frac{1}{2}\left(\frac{\mathbf{A}_{iz}}{\mathbf{u}_z - \mathbf{l}_z} + \frac{\mathbf{B}_{ik}}{\mathbf{u}_k - \mathbf{l}_k}\right) \tag{8}$$

where δ is called learning operator; and z and k are random integers from 1 to d.

In order to really enhance the characteristics, make both characteristics mutate on the basis of the best characteristics. The definition of this behavior can be obtained by combining Eqs. (7) to (8):

$$\mathbf{A}_{ij} = \mathbf{G}_j + \mu s_{ij}^{A} \delta; \mathbf{B}_{ij} = \mathbf{G}_j + \mu s_{ij}^{B} \delta \tag{9}$$

where μ is called the characteristic distance and is defined in Eq. (10).

$$\mu = \frac{1}{nd} \sum_{i} \sum_{j} \left\| \mathbf{X}_{ij} - \mathbf{G}_j \right\|_2 + \varepsilon \tag{10}$$

As the number of iterations increases, the candidate solutions get closer to the best solution, which in turn causes μ to decrease gradually. This change can help the proposed algorithm shift from exploration to exploitation, and it may be able to exhibit different exploration and exploitation behaviors for different problems. This phase is shown schematically in Fig. 3.

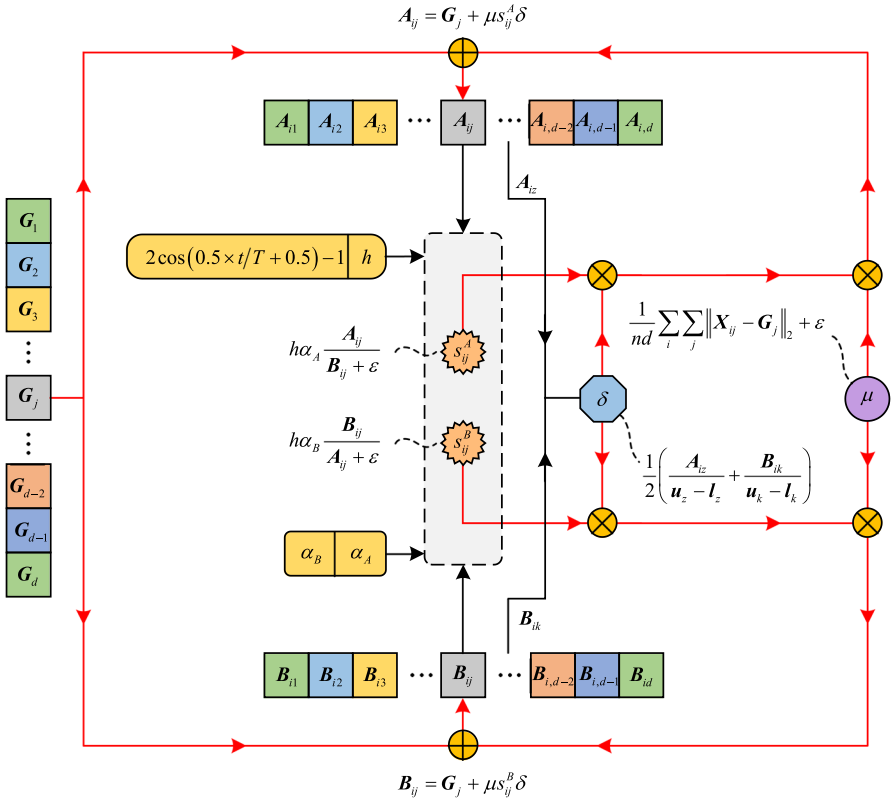

Fig. 3 The process of the reflection operation

3.2.5 Value phase

When the acceptance degree (defined in Eq. (6)) is less than 0.5, the partners enter the value phase. The value phase will take into account deeper thoughts and behaviors, which involves a change in characteristics. Equation (11) uses the convolution to define the convolution operator.

$$\left[\phi_1, \phi_2, \phi_3\right] = \left[\mathbf{A}_{ij}, \mathbf{G}_j\right] * \left[\mathbf{G}_j, \mathbf{B}_{ij}\right] \tag{11}$$

The convolution operator ϕ_1, ϕ_2, and ϕ_3 obtained from the convolution of $\left[\mathbf{A}_{ij}, \mathbf{G}_j\right]$ and $\left[\mathbf{G}_j, \mathbf{B}_{ij}\right]$ can be specifically expressed as

$$\phi_1 = \mathbf{G}_j\mathbf{A}_{ij};\ \phi_2 = \mathbf{G}_j^2 + \mathbf{A}_{ij}\mathbf{B}_{ij};\ \phi_3 = \mathbf{G}_j\mathbf{B}_{ij} \tag{12}$$

After that, Eq. (13) defines the depth operator to inscribe the value phase to understand each other deeply and change the characteristics.

$$\rho_A = \left\|\phi_2 - \phi_1\right\|_2;\ \rho_B = \left\|\phi_2 - \phi_3\right\|_2 \tag{13}$$

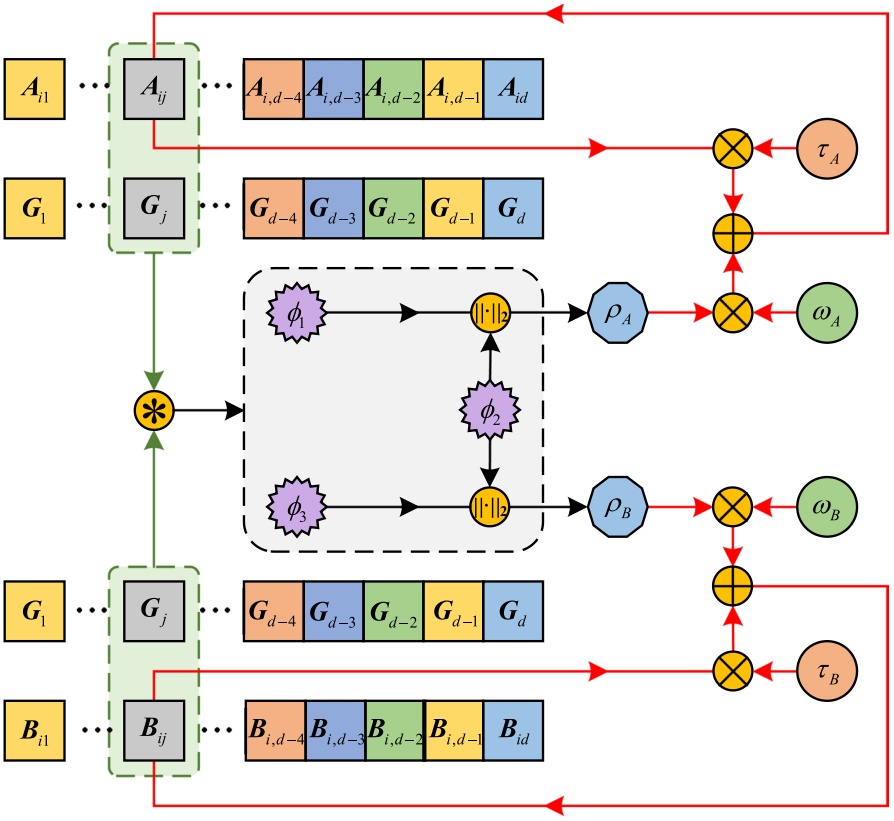

Fig. 4 The process of the value phase

At this phase, the change in the characteristics of both partners is defined in Eq. (14).

$$\mathbf{A}_{ij} = \tau_A \mathbf{A}_{ij} + \omega_A \rho_A; \mathbf{B}_{ij} = \tau_B \mathbf{B}_{ij} + \omega_B \rho_B \tag{14}$$

where τ_A and τ_B are random numbers between 0 and 1; and ω_A and ω_B are random numbers that obey the standard normal distribution. The value phase is shown schematically in Fig. 4.

3.2.6 Adaptation degree

After the value phase, adaptation degree **p** is defined in Eq. (15) to determine which couples are able to enter the role phase.

$$\mathbf{p}_i = \frac{r_p \mathbf{c}_i}{d\mu} \sum_j \left\| \mathbf{A}_{ij} - \mathbf{B}_{ij} \right\|_2 \tag{15}$$

where r_p is a vector of $n/2$ rows and 1 column consisting of 0.5 to 1.5 random numbers, and it has the same role as $\mathbf{r_c}$ defined in Eq. (6). When $\mathbf{p}_i$ is less than λ_p (called adaptation rate, equal to 0.5), it indicates that $\mathbf{A}_i$ and $\mathbf{B}_i$ are very close, at which point the role phase is entered. Otherwise, both partners are considered broken up and the reflection operation is executed.

3.2.7 Role phase

The inspired behaviors of the role phase are assigning roles and complementing each other. Both partners more or less want the other to be a certain role as they envision it. Considering this behavior, the role operator operating on characteristics is defined in Eq. (16).

$$\begin{cases} \boldsymbol{\xi}_i = \mathbf{A}_i \cdot \mathbf{B}_i \\ \boldsymbol{\xi}_{ij} = \dfrac{\boldsymbol{\xi}_{ij} - \min\left(\boldsymbol{\xi}_i\right)}{\max\left(\boldsymbol{\xi}_i\right) - \min\left(\boldsymbol{\xi}_i\right) + \varepsilon} + h \end{cases} \tag{16}$$

where h is called the convergence factor, which is defined in Eq. (17).

$$h = (1 - t/T)\left(h_{\max} - h_{\min}\right) + h_{\min} \tag{17}$$

where $h_{\max}$ and $h_{\min}$ are convergence constants and t is the number of function evaluations (FEs). From the analysis, h is a linearly decreasing function with t.

The update of the characteristics of the role stage is defined in Eq. (18).

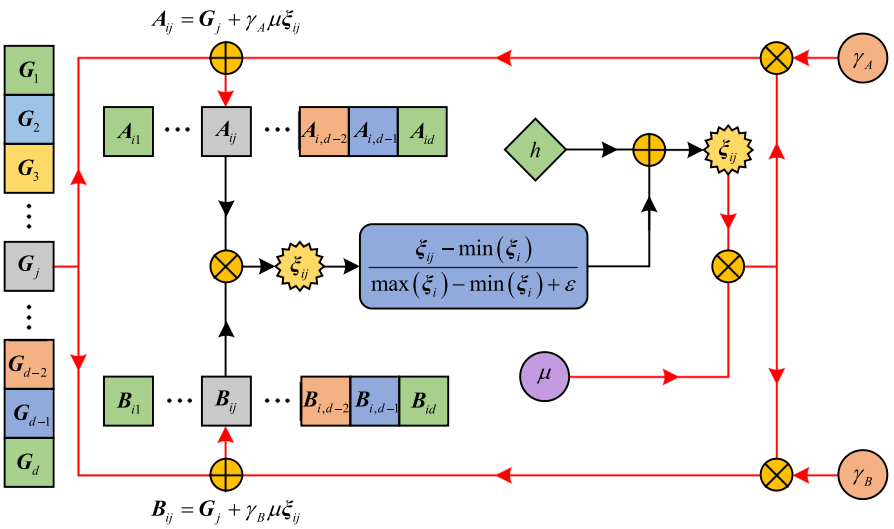

Fig. 5 Process of the role phase

$$\mathbf{A}_{ij} = \mathbf{G}_j + \gamma_A \mu \boldsymbol{\xi}_{ij};\ \mathbf{B}_{ij} = \mathbf{G}_j + \gamma_B \mu \boldsymbol{\xi}_{ij} \tag{18}$$

where γ_A and γ_B are random numbers obeying the standard normal distribution.

The schematic of the role phase is shown in Fig. 5. The role of h in Eq. (16) is to balance exploration and exploitation. If the effect of ε is ignored, $\boldsymbol{\xi}_{ij}$ takes values between h and $1 + h$. In the early stage of the iteration, the value of h is larger and more values greater than 1 in $\boldsymbol{\xi}_i$ are computed. This means that the main tendency is to explore. As the number of iterations increases, the value of h decreases, which means that progressively more and more values in $\boldsymbol{\xi}_i$ are less than 1. This implies a gradual shift from exploration to exploitation.

3.2.8 Update of the population

In this paper, two formulas for the cross-boundary processing of $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ are given, defined in Eqs. (19) and (20).

$$\begin{aligned}
\mathbf{A}_{ij} &= \begin{cases} \mathbf{l}_j + \left(\mathbf{u}_j - \mathbf{l}_j\right)\dfrac{\mathrm{mod}\left(\mathbf{A}_{ij}, \mathbf{u}_j + \varepsilon\right)}{\mathbf{u}_j + \varepsilon}, & \mathbf{A}_{ij} > \mathbf{u}_j \\ \mathbf{A}_{ij}, & \mathbf{l}_j < \mathbf{A}_{ij} < \mathbf{u}_j \\ \mathbf{l}_j + \left(\mathbf{u}_j - \mathbf{l}_j\right)\dfrac{\mathrm{mod}\left(\mathbf{A}_{ij}, \mathbf{l}_j + \varepsilon\right)}{\mathbf{l}_j + \varepsilon}, & \mathbf{A}_{ij} < \mathbf{l}_j \end{cases}; \\
\mathbf{B}_{ij} &= \begin{cases} \mathbf{l}_j + \left(\mathbf{u}_j - \mathbf{l}_j\right)\dfrac{\mathrm{mod}\left(\mathbf{B}_{ij}, \mathbf{u}_j + \varepsilon\right)}{\mathbf{u}_j + \varepsilon}, & \mathbf{B}_{ij} > \mathbf{u}_j \\ \mathbf{B}_{ij}, & \mathbf{l}_j < \mathbf{B}_{ij} < \mathbf{u}_j \\ \mathbf{l}_j + \left(\mathbf{u}_j - \mathbf{l}_j\right)\dfrac{\mathrm{mod}\left(\mathbf{B}_{ij}, \mathbf{l}_j + \varepsilon\right)}{\mathbf{l}_j + \varepsilon}, & \mathbf{B}_{ij} < \mathbf{l}_j \end{cases}
\end{aligned} \tag{19}$$

where mod denotes the modulo operation, and $\mathrm{mod}(x, y)$ returns the remainder after x is divided by y; and ε is used to prevent the upper or lower bounds from being equal to 0. Equation (19) retains a portion of the search information while performing transgression processing.

$$\mathbf{A}_{ij} = \begin{cases} \mathbf{u}_j, & \mathbf{A}_{ij} > \mathbf{u}_j \\ \mathbf{A}_{ij}, & \mathbf{l}_j < \mathbf{A}_{ij} < \mathbf{u}_j \\ \mathbf{l}_j, & \mathbf{A}_{ij} < \mathbf{l}_j \end{cases}; \mathbf{B}_{ij} = \begin{cases} \mathbf{u}_j, & \mathbf{B}_{ij} > \mathbf{u}_j \\ \mathbf{B}_{ij}, & \mathbf{l}_j < \mathbf{B}_{ij} < \mathbf{u}_j \\ \mathbf{l}_j, & \mathbf{B}_{ij} < \mathbf{l}_j \end{cases} \tag{20}$$

After a round of iterations, a new population is created (i.e., $\mathbf{X}$=[$\mathbf{A}$;$\mathbf{B}$]). The new population will completely replace the old population into the next cycle. The flowchart of the proposed algorithm is shown in Fig. 6. Moreover, the pseudocode of the algorithm is provided in Algorithm 1.

Algorithm 1 Love Evolution Algorithm

Initialize the population size n, the MaxFEs T and the FEs $t = 0$
Initialize the number of characteristics d, the upper boundary $\boldsymbol{u}$, the lower boundary $\boldsymbol{l}$, and the function F
Create the population $\boldsymbol{X}$ using Eq. (1) *%% Create the population*
$\boldsymbol{H}$ = F($\boldsymbol{X}$) *%% Create the vector of happiness degree*
Select the characteristics of the best people in history $\boldsymbol{G}$ from the population
while $t < T$ *%% Main loop*
    Calculate the convergence factor h using Eq. (17)
    Population $\boldsymbol{X}$ is randomly and equally divided into $\boldsymbol{A}$ and $\boldsymbol{B}$ using Eqs. (4) to (5) *% Encounter*
    Calculate the acceptance degree $\boldsymbol{c}$ using Eq. (6) *% Stimulus phase*
    for i = 1:n/2
        if $\boldsymbol{c}_i < \lambda_c$
            for j = 1:d *% Value phase*
                Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eqs. (12) to (14)
                Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eq. (19)
            end for
            $t = t + 1$; **if** $t > T$; **break**; **end if**
            Select the characteristics of the best people in history $\boldsymbol{G}$ from the population
            Calculate the adaptation degree $\boldsymbol{p}_i$ using Eq. (15)
            if $\boldsymbol{p}_i > \lambda_p$
                for j = 1:d *% Reflection operation*
                    Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eqs. (7) to (10)
                end for
            else *% Role phase*
                Calculate the role operator $\boldsymbol{\xi}_i$ using Eq. (16)
                for j = 1:d
                    Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eq. (18)
                    Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eq. (20)
                end for
            end if
            $t = t + 1$; **if** $t > T$; **break**; **end if**
            Select the characteristics of the best people in history $\boldsymbol{G}$ from the population
        else
            for j = 1:d *% Reflection operation*
                Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eqs. (7) to (10)
                Update the $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ using Eq. (20)
            end for
            $t = t + 1$; **if** $t > T$; **break**; **end if**
            Select the characteristics of the best people in history $\boldsymbol{G}$ from the population
        end if
    end for
    $\boldsymbol{X}$ = [$\boldsymbol{A}$; $\boldsymbol{B}$] *%% Update the population*
end while
return $\boldsymbol{G}$

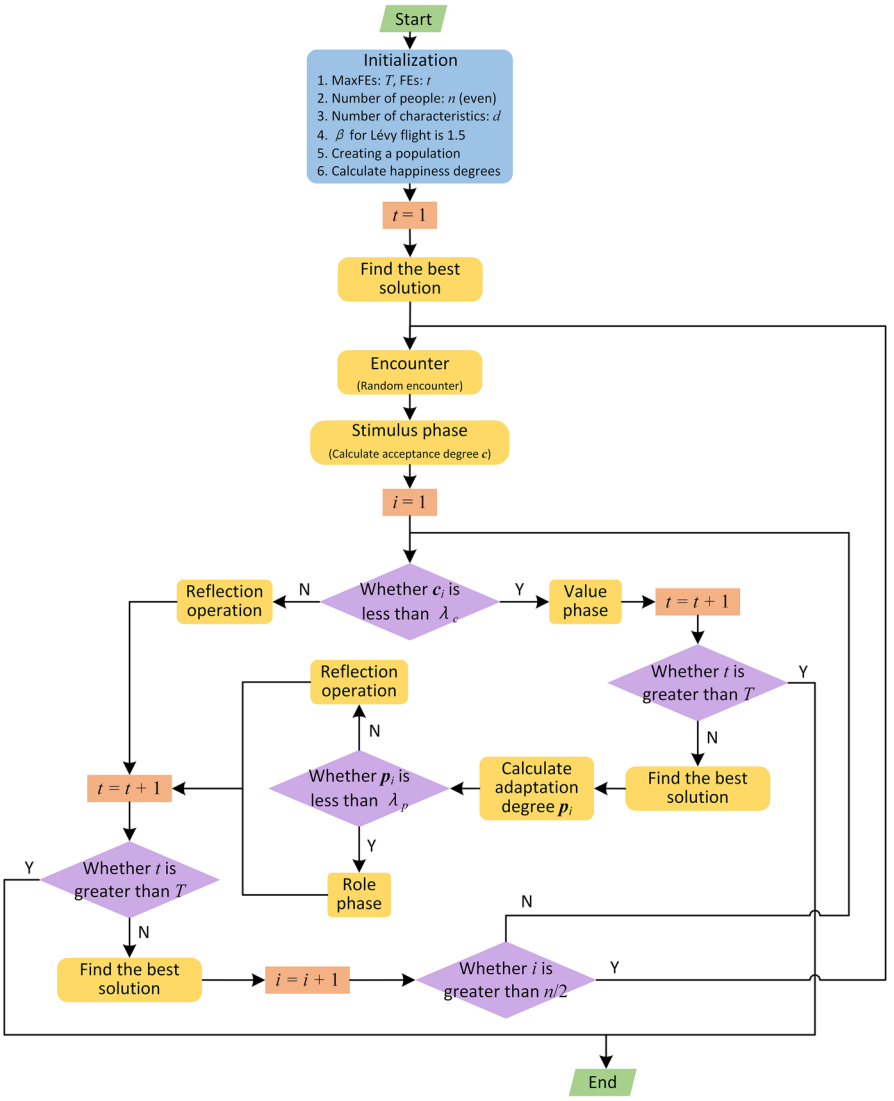

Fig. 6 Flowchart of the proposed Love Evolution Algorithm

3.3 Theoretical analysis of time and space complexity

Time complexity and space complexity are two important metrics to evaluate the performance of an algorithm. The time complexity of the LEA is mainly influenced by three factors: the population size (n), the maximum number of iterations (T) and the number of variables (d) for solving the problem. In the proposed algorithm, the initialization problem requires O(nd). During the iterative process, the computation requires O(Tnd). Therefore, the time complexity of the LEA is O(Tnd). The spatial complexity of the LEA is mainly affected by the population size (n) and the number of variables (d) for solving the problem. In the proposed algorithm, the features of the stored population occupy the main memory space. Thus, the space complexity of the LEA is O(nd).

4 Experimental results and discussion

The section focuses on comparing and analyzing the test results of the LEA and some state-of-the-art metaheuristic algorithms on the CEC2017 and CEC2022 benchmark functions.

4.1 Experimental setup

This study implements all algorithms using MATLAB R2023b on a computer with 64-bit Windows 11. The CEC2017 benchmark functions [73] are quite challenging test sets. It greatly simulates the optimization problems in the real world. Among them, F1 and F3 are unimodal functions, F4–F10 are multimodal functions, F11–F20 are hybrid functions, and F21–F30 are composition functions. The dimension is set to 50. Besides, the profiles of the CEC2017 benchmark functions are shown in Table 1. Moreover, there are a total of 12 single-objective test functions with boundary constraints in the CEC2022 benchmark functions [74]. These functions are unimodal function (F1), basic functions (F2–F5), hybrid functions (F6–F8) and composition functions (F9–F12). The dimensions of two experiments on the CEC2022 benchmark functions are set to 10 and 20, respectively. In addition, the CEC2022 benchmark functions are described in Table 2.

To verify the superiority of the proposed algorithm, the nutcracker optimization algorithm (NOA) [75], the golden jackal optimization (GJO) [76], the atomic orbital search (AOS) [54], the tunicate swarm algorithm (TSA) [34], the seagull optimization algorithm (SOA) [77], the Harris hawk optimization (HHO) [33], and the random drift particle swarm optimization (RDPSO) [78] are compared with the proposed LEA on the CEC2017 benchmark functions. The population size of these algorithms is set to 50. Furthermore, the maximum number of function evaluations (MaxFEs) is 500,000 (10,000* dimensions). The settings of other parameters are shown in Table 3.

Further, some strong algorithms are used to compare with the LEA on the CEC2022 benchmark functions. These algorithms are the success history-based

Table 1 A brief description of the CEC2017 benchmark functions

No.	Type	Function	Minimum
F1	Unimodal function	Shifted and rotated bent cigar function	100
F3	Unimodal function	Shifted and rotated Zakharov function	300
F4	Multimodal function	Shifted and rotated Rosenbrock's function	400
F5	Multimodal function	Shifted and rotated Rastrigin's function	500
F6	Multimodal function	Shifted and rotated expanded Scaffer's F6 function	600
F7	Multimodal function	Shifted and rotated Lunacek Bi_Rastrigin function	700
F8	Multimodal function	Shifted and rotated non-continuous Rastrigin's function	800
F9	Multimodal function	Shifted and rotated levy function	900
F10	Multimodal function	Shifted and rotated Schwefel's function	1000
F11	Hybrid function	Hybrid function 1 ($N=3$)	1100
F12	Hybrid function	Hybrid function 2 ($N=3$)	1200
F13	Hybrid function	Hybrid function 3 ($N=3$)	1300
F14	Hybrid function	Hybrid function 4 ($N=4$)	1400
F15	Hybrid function	Hybrid function 5 ($N=4$)	1500
F16	Hybrid function	Hybrid function 6 ($N=4$)	1600
F17	Hybrid function	Hybrid function 6 ($N=5$)	1700
F18	Hybrid function	Hybrid function 6 ($N=5$)	1800
F19	Hybrid function	Hybrid function 6 ($N=5$)	1900
F20	Hybrid function	Hybrid function 6 ($N=6$)	2000
F21	Composition function	Composition function 1 ($N=3$)	2100
F22	Composition function	Composition function 2 ($N=3$)	2200
F23	Composition function	Composition function 3 ($N=4$)	2300
F24	Composition function	Composition function 4 ($N=4$)	2400
F25	Composition function	Composition function 5 ($N=5$)	2500
F26	Composition function	Composition function 6 ($N=5$)	2600
F27	Composition function	Composition function 7 ($N=6$)	2700
F28	Composition function	Composition function 8 ($N=6$)	2800
F29	Composition function	Composition function 9 ($N=3$)	2900
F30	Composition function	Composition function 10 ($N=3$)	3000

Search range: $[-100, 100]^D$

adaptive differential evolution with linear population size reduction (L-SHADE) [71], the adaptive L-SHADE (AL-SHADE) [79], the L-SHADE with semi-parameter adaptation hybrid with CMA-ES (L-SHADE-spacma) [80], the adaptive fitness–distance balance-based artificial rabbits optimization (AFDB-ARO) [81], the fitness–distance balance-based adaptive guided differential evolution (FDB-AGDE) [82], the fitness–distance balance-based adaptive gaining–sharing knowledge (FDB-AGSK) [83], and the fitness–distance balance-based phasor particle swarm optimization (FDB-PPSO) [84]. L-SHADE, AL-SHADE, and L-SHADE-spacma are

Table 2 A brief description of the CEC2022 benchmark functions

No.	Type	Function	Minimum
F1	Unimodal function	Shifted and full rotated Zakharov function	300
F2	Basic function	Shifted and full rotated Rosenbrock's function	400
F3	Basic function	Shifted and full rotated expanded Schaffer's F6 function	600
F4	Basic function	Shifted and full rotated non-continuous Rastrigin's function	800
F5	Basic function	Shifted and full rotated levy function	900
F6	Hybrid function	Hybrid function 1 ($N=3$)	1800
F7	Hybrid function	Hybrid function 2 ($N=6$)	2000
F8	Hybrid function	Hybrid function 3 ($N=5$)	2200
F9	Composition function	Composition function 1 ($N=5$)	2300
F10	Composition function	Composition function 2 ($N=4$)	2400
F11	Composition function	Composition function 3 ($N=5$)	2600
F12	Composition function	Composition function 4 ($N=6$)	2700

Search range: $[-100, 100]^D$

Table 3 Parameter settings for algorithms experimented on the CEC2017 benchmark functions

Algorithm	Parameter	Value	Algorithm	Parameter	Value
LEA	Convergence constant $h_{\max}$	0.7	AOS	Maximum number of nucleus layers	10
	Convergence constant $h_{\min}$	0		Photon rate PR	0.1
	Acceptance rate λ_c	0.5	TSA	Initial speed $P_{\min}$	1
	Adaptation rate λ_p	0.5		Subordinate speed $P_{\max}$	4
NOA	Probability δ	0.05	HHO	Constant of levy flight β	1.5
	Probability P_{a2}	0.2	RDPSO	Acceleration coefficient c_1	2
	Probability P_{rp}	0.2		Acceleration coefficient c_2	2
GJO	Constant c_1	1.5		Thermal coefficient α (max)	0.9
	Constant of levy flight β	1.5		Thermal coefficient α (min)	0.3
SOA	Frequency control parameter f_c	2		Drift coefficient β	1.5

recognized as strongly competitive algorithms. Besides, AFDB-ARO, FDB-AGDE, FDB-AGSK, and FDB-PPSO are strong algorithms that have been improved using fitness–distance balance-based guide mechanism. Comparison with these algorithms better demonstrates the competitive nature of the LEA. In both experiments, the population size is set to 50. Moreover, the MaxFEs is 1,000,000 (20 dimensions), respectively. The settings of other parameters are given in Table 4.

Table 4 Parameter settings for algorithms experimented on the CEC2022 benchmark functions

Algorithm	Parameter	Value	Algorithm	Parameter	Value
LEA	Convergence constant $h_{\max}$	0.73	AL-SHADE	Maximum population size	50
	Convergence constant $h_{\min}$	0		Minimum population size	4
	Acceptance rate λ_c	0.5		Historical memory M_{CR}	0.5
	Adaptation rate λ_p	0.5		Historical memory M_F	0.5
L-SHADE	Maximum population size	50		Historical memory size	6
	Minimum population size	4		p best rate	0.11
	Historical memory size	5		Archive rate	2.6
	p best rate	0.11	L-SHADE-spacma	L rate	0.8
	Archive rate	1.4		Maximum population size	50
AFDB-ARO	Parameter-less			Minimum population size	4
FDB-AGDE	Parameter-less			Historical memory size	5
FDB-AGSK	Maximum population size	50		p best rate	0.11
	Minimum population size	12		Archive rate	1.4
	Knowledge factor pool k_f	[0.1,1,0.5,1]		First class percentage	0.5
	Knowledge ratio pool k_r	[0.2,0.1,0.9,0.9]	FDB-PPSO	Parameter-less	

4.2 Performance comparison

This subsection evaluates the performance of the LEA on the CEC2017 benchmark functions using the average and variance metrics (Ave and Var, respectively). Also, the average, variance, minimum, and maximum values (Ave, Var, Min, and Max, respectively) are used to evaluate the competitiveness of LEAs and powerful algorithms on the CEC2022 benchmark functions.

Table 5 Optimization results for CEC2017 unimodal and multimodal functions

F	Metrics	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
F1	Ave	**9.39E+03**	1.40E+11	3.12E+10	4.94E+06	6.52E+10	2.42E+10	3.77E+07	1.36E+11
	Var	**1.01E+08**	1.17E+20	3.34E+19	1.67E+12	1.23E+20	3.45E+19	3.54E+13	7.18E+19
F3	Ave	**3.00E+02**	2.45E+05	9.51E+04	2.92E+04	1.51E+05	8.11E+04	1.70E+04	2.31E+05
	Var	**3.94E−09**	6.59E+08	1.68E+08	3.94E+07	4.72E+08	2.47E+08	3.39E+07	3.51E+08
F4	Ave	**5.18E+02**	3.56E+04	4.22E+03	6.28E+02	1.41E+04	2.02E+03	6.27E+02	3.50E+04
	Var	**2.75E+03**	1.08E+07	2.00E+06	3.63E+03	2.37E+07	1.80E+05	2.28E+03	1.36E+07
F5	Ave	**7.55E+02**	1.35E+03	8.92E+02	8.26E+02	1.01E+03	8.43E+02	8.83E+02	1.33E+03
	Var	1.88E+03	**7.14E+02**	3.86E+03	1.09E+03	7.87E+02	1.91E+03	1.01E+03	9.26E+02
F6	Ave	**6.22E+02**	7.08E+02	6.46E+02	6.62E+02	6.87E+02	6.51E+02	6.70E+02	7.05E+02
	Var	1.03E+02	2.22E+01	7.47E+01	3.02E+01	4.22E+01	5.55E+01	1.18E+01	**1.11E+01**
F7	Ave	**1.14E+03**	4.14E+03	1.37E+03	1.59E+03	1.86E+03	1.45E+03	1.77E+03	4.10E+03
	Var	7.87E+03	2.75E+04	9.51E+03	1.45E+04	**5.50E+03**	7.15E+03	7.02E+03	2.91E+04
F8	Ave	**1.05E+03**	1.65E+03	1.18E+03	1.15E+03	1.34E+03	1.18E+03	1.16E+03	1.63E+03
	Var	2.29E+03	1.71E+03	2.47E+03	2.34E+03	1.45E+03	2.20E+03	2.79E+03	**9.72E+02**
F9	Ave	**1.01E+04**	5.08E+04	1.54E+04	1.37E+04	2.84E+04	1.37E+04	1.96E+04	4.78E+04
	Var	1.07E+07	1.66E+07	2.91E+07	**4.25E+06**	1.08E+07	1.31E+07	6.66E+06	1.91E+07
F10	Ave	**6.84E+03**	1.48E+04	9.70E+03	8.42E+03	1.23E+04	9.23E+03	8.32E+03	1.44E+04
	Var	7.66E+05	1.28E+05	3.02E+06	1.30E+06	9.22E+05	1.43E+06	6.43E+05	**9.96E+04**

Bolded+ values represent the smallest values

4.2.1 Exploitation and exploration

Unimodal functions have only one strictly optimal solution in the selected interval and are often used to test the exploitation capability of an algorithm. Moreover, multimodal functions have a large number of locally optimal solutions in the considered interval. The number of solutions increases with the dimension of the problem. Therefore, the multimodal functions are commonly used to evaluate the exploration capability of an algorithm. The results of the LEA and the comparison algorithms on the CEC2017 unimodal and multimodal functions are given in Table 5.

The Ave and Var of the LEA are minimized on F1 and F3 and are much smaller in order of magnitude than the comparison algorithms. This indicates that the LEA has a strong exploitation capability. Considering the exploration capability of the LEA, the Ave of the LEA are ranked first on F4–F10. In terms of the Var, the LEA ranks 1st on F4, 3rd on F9, 4th on F7 and F10, 5th on F8, 6th on F5, and 8th on F6. It can be found that the Ave of the LEA has a large advantage among the competitors, while the Var does not have an advantage. The reason for this phenomenon may be that the LEA has a greater chance of exploring ideal areas compared to its competitors, which makes the LEA not superior to other algorithms in terms of variance under the condition of better mean value.

Table 6 Optimization results for CEC2017 hybrid functions

F	Metrics	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
F11	Ave	1.44E+03	2.48E+04	7.33E+03	1.46E+03	1.75E+04	4.21E+03	**1.43E+03**	2.15E+04
	Var	5.18E+03	1.24E+07	4.82E+06	**3.77E+03**	2.58E+07	2.04E+06	8.98E+03	8.90E+06
F12	Ave	**1.18E+07**	4.63E+10	7.66E+09	9.26E+07	3.66E+10	2.82E+09	5.30E+07	4.08E+10
	Var	**4.59E+13**	4.89E+19	1.72E+19	2.48E+15	1.45E+20	2.88E+18	5.25E+14	2.87E+19
F13	Ave	**1.62E+05**	1.63E+10	1.05E+09	2.57E+05	1.50E+10	4.69E+08	1.35E+06	1.52E+10
	Var	**9.57E+09**	9.18E+18	1.72E+18	1.22E+10	7.42E+19	6.50E+17	9.41E+11	5.61E+18
F14	Ave	**7.86E+04**	1.19E+07	9.61E+05	3.22E+05	1.64E+07	6.65E+05	3.51E+05	8.25E+06
	Var	**3.43E+09**	1.50E+13	5.37E+11	3.17E+10	2.62E+14	5.70E+11	7.44E+10	8.70E+12
F15	Ave	**5.94E+04**	4.42E+09	1.26E+08	7.15E+04	2.37E+09	2.21E+07	1.82E+05	3.63E+09
	Var	**9.45E+08**	1.33E+18	5.71E+16	4.06E+09	5.02E+18	8.41E+14	6.83E+09	8.32E+17
F16	Ave	**3.49E+03**	7.56E+03	3.56E+03	3.96E+03	5.13E+03	3.54E+03	4.20E+03	7.30E+03
	Var	1.77E+05	1.28E+05	**6.46E+04**	3.99E+05	5.37E+05	2.16E+05	2.06E+05	1.11E+05
F17	Ave	**3.08E+03**	7.57E+03	3.41E+03	3.48E+03	5.72E+03	3.22E+03	3.55E+03	7.12E+03
	Var	1.51E+05	1.40E+06	2.52E+05	1.06E+05	7.77E+06	**7.58E+04**	1.76E+05	3.75E+05
F18	Ave	**3.79E+05**	6.35E+07	8.19E+06	1.48E+06	3.56E+07	3.06E+06	3.54E+06	4.82E+07
	Var	**2.93E+10**	4.04E+14	3.48E+14	8.21E+11	2.01E+15	1.41E+12	9.00E+12	2.31E+14
F19	Ave	**2.18E+04**	1.75E+09	3.78E+07	2.28E+06	1.28E+09	2.00E+06	5.50E+05	1.46E+09
	Var	**1.15E+08**	2.53E+17	4.89E+15	1.68E+12	1.60E+18	7.25E+12	1.43E+11	9.52E+16
F20	Ave	**2.92E+03**	4.06E+03	3.10E+03	3.36E+03	3.64E+03	3.22E+03	3.37E+03	4.04E+03
	Var	4.60E+04	3.19E+04	6.32E+04	5.38E+04	1.16E+05	1.32E+05	5.59E+04	**1.59E+04**

Bolded values represent the smallest values

4.2.2 Capability of avoiding locally optimal solutions

Hybrid and composition functions are often considered the most challenging optimization problems. An algorithm is better able to avoid local optimal solutions when it achieves some balance between exploitation and exploration. Hence, these functions are often used to evaluate the capability of an algorithm to avoid local optimal solutions. The optimization results of different algorithms on the CEC2017 hybrid and composition functions are given in Tables 6 and 7, respectively.

For the hybrid functions, the LEA ranks 1st on F12–F15, F18, and F19 in terms of the Ave and the Var. Besides, the LEA ranks 2nd in terms of the Ave and the Var on F11. In addition, on F16 and F17, the LEA is ranked 1st in terms of the Ave. On F20, the LEA is ranked 1st in terms of the Ave and 3rd in terms of the Var.

For the composition functions, the Ave of the LEA ranks 1st on F21 ~ F30 except F24. Furthermore, the Var of the LEA ranks 1st on F28–F30, 2nd on F23 and F27. In worse result is that the LEA is ranked 5th and 8th in F21 and F22 in terms of the Var, respectively. The LEA, on the other hand, has more chances of jumping out of the local extremes, which makes the mean of LEA relatively better but the variance worse. Collectively, the results of LEA on the composition functions are much better than its competitors.

Overall, on Ave, the optimization results of the LEA are smaller than those of the competitors. Moreover, the Var of the LEA is similarly smaller than those of

Table 7 Optimization results for CEC2017 composition functions

F	Metrics	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
F21	Ave	**2.56E+03**	3.16E+03	2.65E+03	2.73E+03	2.94E+03	2.65E+03	2.82E+03	3.13E+03
	Var	2.62E+03	**1.19E+03**	1.49E+03	4.48E+03	2.98E+03	2.43E+03	5.00E+03	1.15E+03
F22	Ave	**8.11E+03**	1.64E+04	1.12E+04	1.02E+04	1.46E+04	1.05E+04	1.08E+04	1.62E+04
	Var	5.00E+06	**7.12E+04**	3.14E+06	1.36E+06	3.89E+05	1.17E+06	8.63E+05	1.07E+05
F23	Ave	**3.02E+03**	4.05E+03	3.19E+03	3.62E+03	3.89E+03	3.07E+03	3.66E+03	4.02E+03
	Var	4.41E+03	5.69E+03	6.56E+03	3.39E+04	2.94E+04	**2.23E+03**	2.24E+04	5.77E+03
F24	Ave	3.22E+03	4.34E+03	3.41E+03	3.91E+03	4.13E+03	**3.16E+03**	4.21E+03	4.30E+03
	Var	1.22E+04	1.15E+04	1.09E+04	4.18E+04	3.42E+04	**2.60E+03**	4.46E+04	8.66E+03
F25	Ave	**3.02E+03**	2.39E+04	5.38E+03	3.09E+03	8.54E+03	4.65E+03	3.13E+03	2.35E+04
	Var	1.61E+03	6.43E+06	6.62E+05	**6.96E+02**	1.99E+06	2.56E+05	1.60E+03	3.37E+06
F26	Ave	**6.32E+03**	1.82E+04	8.94E+03	1.16E+04	1.48E+04	7.29E+03	7.75E+03	1.78E+04
	Var	5.34E+05	1.11E+06	9.49E+05	9.71E+05	1.99E+06	**3.45E+05**	1.14E+07	3.81E+05
F27	Ave	**3.46E+03**	5.59E+03	3.89E+03	4.15E+03	4.99E+03	3.59E+03	4.07E+03	5.47E+03
	Var	7.27E+03	4.37E+04	2.68E+04	8.74E+04	1.77E+05	**6.97E+03**	1.33E+05	2.95E+04
F28	Ave	**3.29E+03**	1.35E+04	5.89E+03	3.36E+03	8.19E+03	8.35E+03	3.37E+03	1.30E+04
	Var	**7.52E+02**	5.15E+05	4.17E+05	8.33E+02	1.18E+06	1.15E+06	8.86E+02	4.12E+05
F29	Ave	**4.19E+03**	1.30E+04	5.39E+03	6.42E+03	1.08E+04	5.49E+03	5.28E+03	1.24E+04
	Var	**8.34E+04**	3.44E+06	2.61E+05	5.27E+05	2.67E+07	3.64E+05	1.92E+05	2.82E+06
F30	Ave	**1.57E+06**	3.06E+09	2.26E+08	6.93E+07	2.53E+09	1.55E+08	1.64E+07	2.84E+09
	Var	**2.13E+11**	2.77E+17	2.00E+16	7.75E+14	4.66E+18	3.50E+15	1.13E+13	2.72E+17

Bolded values represent the smallest values

the comparison algorithms on most composition functions. Although the LEA is not ranked high in terms of the Var on the F21 and F22 functions, it is in the same or neighboring order of magnitude with the algorithms ranked 1st. The above results show that the LEA has a certain capability of avoiding falling into local optimum.

4.2.3 Capability of finding optimal solutions

It is generally considered that better optimization results are available when there is some balance between exploitation and exploration. By comparing the optimization results of LEA with other algorithms on the CEC2017 benchmark functions, it can be observed that the LEA is usually ranked in the first or top position on the unimodal, multimodal, hybrid, and composition functions. In order to visually compare the performance of the different algorithms, boxplot is chosen to show the quality of the solutions produced by the different algorithms. Figures 7, 8, and 9 give some boxplots of the different algorithms in the CEC2017 benchmark functions. On the F3, F4, F7, F12–F15, F18, F19, and F27–F30, the LEA obtains solutions of higher quality than the competitors with less fluctuation. Statistically, it is obtained that the LEA is ranked 1st on the Ave of 27 functions and

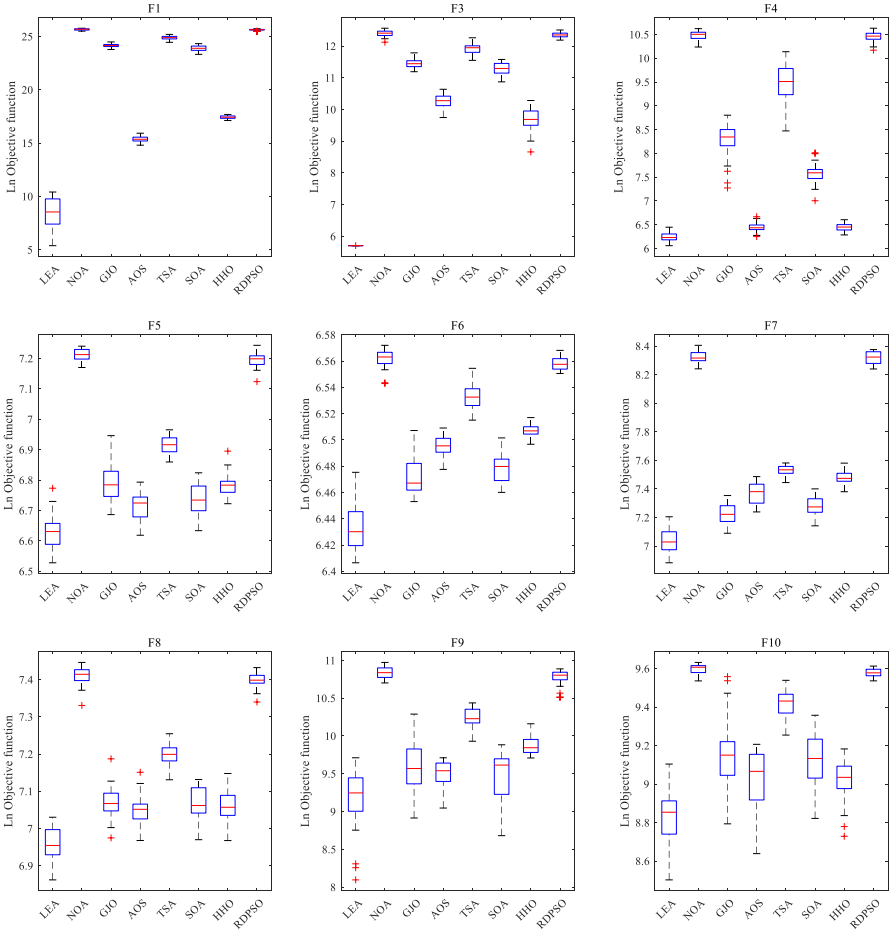

Fig. 7 Boxplots of different algorithms on some CEC2017 unimodal and multimodal functions

2nd on the Ave of 2 functions, which verifies the capability of the LEA to find the optimal solutions.

4.2.4 Time complexity analysis

The time complexity analysis methodology defined by the CEC2017 benchmark test [73] was used to evaluate the time complexity of the LEA. The steps of time complexity analysis are: (1) Run the program defined by Eq. (21) 1,000,000 times when $x=0.55$ to get the time T_0; (2) time T_1 is obtained by evaluating the Function 18 using 200,000 evaluations in 50 dimensions; (3) the algorithm to be evaluated is

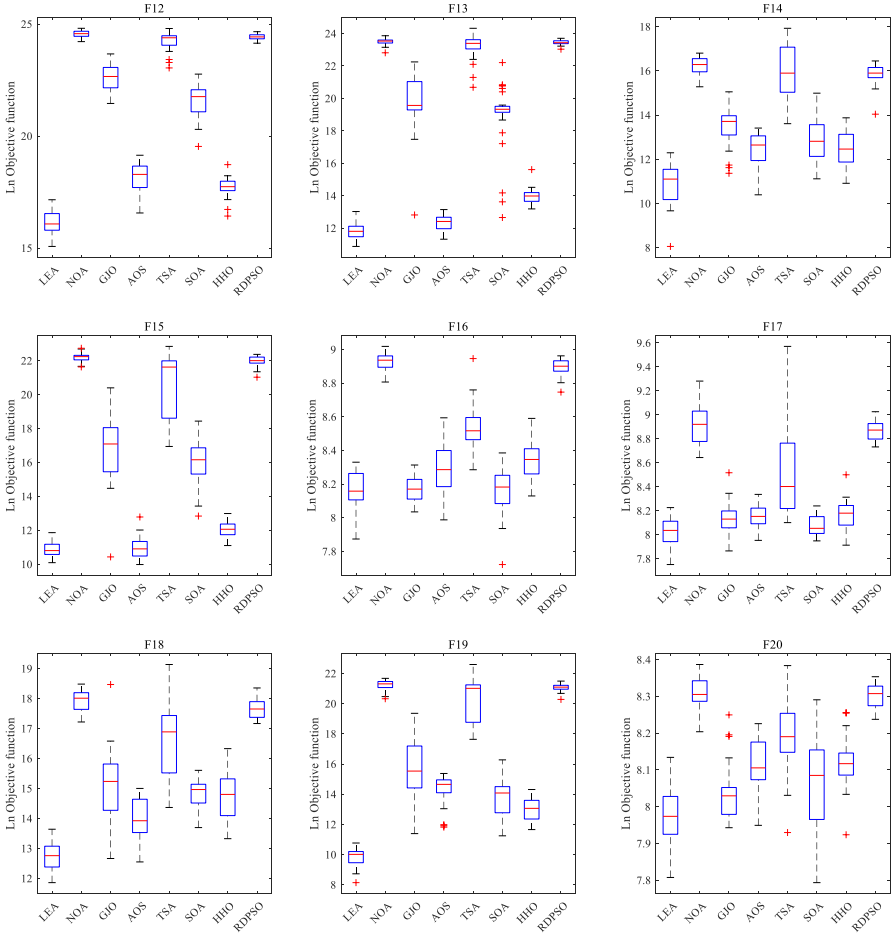

Fig. 8 Boxplots of different algorithms on some CEC2017 hybrid functions

evaluated on the Function 18 (50 dimensions) for 200,000 to get time T_2; (4) execute step (3) five times and take the average of the five times T_2 to get T_{mean}; and (5) calculate $(T_{\text{mean}} - T_1)/T_0$ and analyze.

$$x = x + x; x = x/2; x = x \times x; x = \sqrt{x}; x = \ln x; x = e^x; x = x/(x+2) \quad (21)$$

Table 8 gives information on the computational cost of all algorithms tested on the CEC2017 benchmark functions. Although the computational cost of the SOA, the HHO and the RDPSO is much less than that of the LEA, the performance of the LEA in the CEC2017 benchmark set is much better than that of these three competitors. Therefore, it can be concluded that although the time complexity of LEA is not dominant, it is acceptable with guaranteed optimization quality.

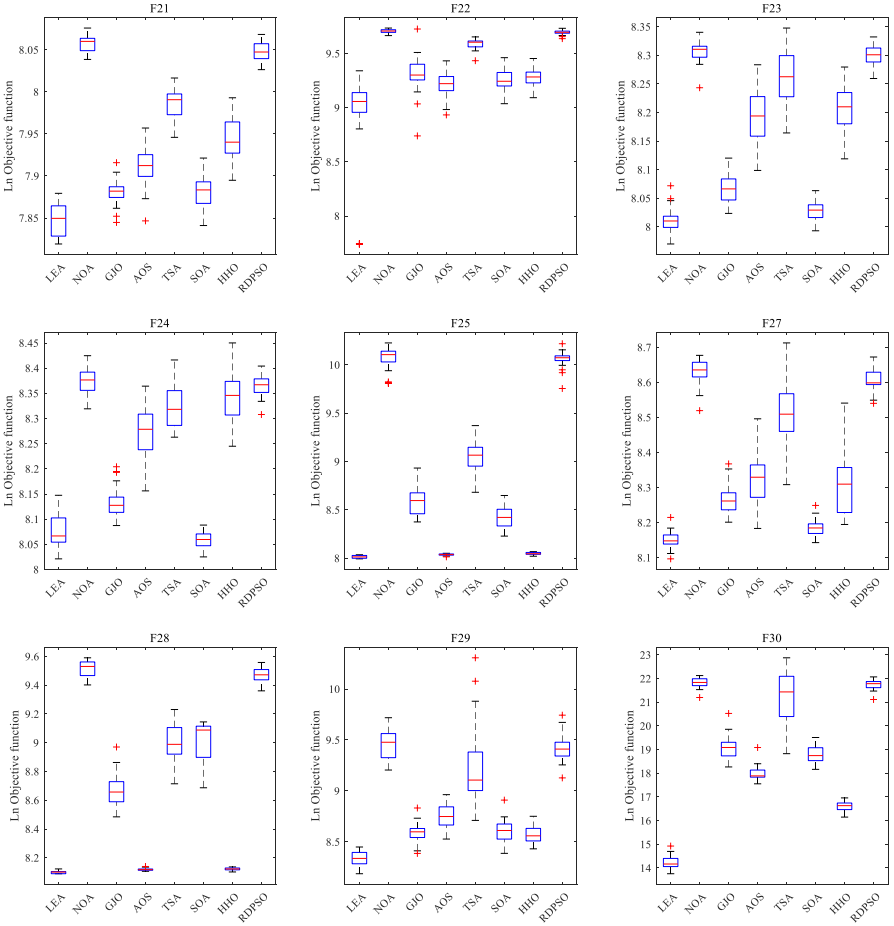

Fig. 9 Boxplots of different algorithms on some CEC2017 composition functions

4.2.5 Comparison with strong algorithms

The CEC2022 benchmark test set encompasses a series of test functions of multiple dimensions and complexity for testing and evaluating the capabilities of optimization algorithms in solving real-world problems. These functions are characterized as nonlinear, multi-peaked, non-convex, non-derivable and highly complex, which can fully examine the search capability and robustness of the optimization algorithms.

Tables 9 and 10 show the optimization results of different algorithms on the CEC2022 benchmark functions (20 dimensions). On the unimodal function F1, the LEA shows a strong competitive capability. Not only does the LEA have the smallest Ave, but it also has the smallest Max and Min. On multimodal functions F2–F5, the results exhibited by the LEA and the strong comparison algorithms are strongly competitive. The Ave index of the LEA on F2 is even smaller than that of

Table 8 Computational cost of the algorithms

Algorithm	Properties	Result	Algorithm	Properties	Result
LEA	T_0	0.0707	TSA	T_0	0.0618
	T_1	1.0392		T_1	1.2537
	T_{mean}	3.1958		T_{mean}	3.4902
	$(T_{\text{mean}} - T_1)/T_0$	30.5070		$(T_{\text{mean}} - T_1)/T_0$	36.2176
NOA	T_0	0.0711	SOA	T_0	0.0786
	T_1	1.0068		T_1	1.3266
	T_{mean}	2.7441		T_{mean}	3.3869
	$(T_{\text{mean}} - T_1)/T_0$	24.4396		$(T_{\text{mean}} - T_1)/T_0$	26.2041
GJO	T_0	0.0627	HHO	T_0	0.0747
	T_1	1.3163		T_1	1.4822
	T_{mean}	3.2417		T_{mean}	2.8766
	$(T_{\text{mean}} - T_1)/T_0$	30.6993		$(T_{\text{mean}} - T_1)/T_0$	18.6679
AOS	T_0	0.0709	RDPSO	T_0	0.0712
	T_1	1.1482		T_1	1.6347
	T_{mean}	3.2758		T_{mean}	2.6697
	$(T_{\text{mean}} - T_1)/T_0$	30.0118		$(T_{\text{mean}} - T_1)/T_0$	14.5468

the L-SHADE, the AL-SHADE, the L-SHADE-spacma, the FDB-AGDE, and the FDB-PPSO. On F3, the LEA has the smallest Ave, which ranks 1st in the Min. Furthermore, the Min of the LEA is ranked 1st on F2. On the hybrid functions F6 and F7, the result is comparable to that of the other competitors, although none of the Ave, the Var, the Max, and the Min of the LEA are ranked 1st. Compared to other competitors, LEA ranks 1st in the Ave on F8 and F9, and also reaches the smallest in the Max and the Min on F9 and F11. On F12, the LEA, although not 1st on the all metrics, achieved acceptably good results compared to the competitors.

From the test results with the strongly competitive algorithms on the CEC2022 benchmark test set, it can be concluded that although the LEA as a whole is not yet up to the optimization capabilities of these strong algorithms, it can compete with these algorithms on most of the functions. Figure 10 shows the change in the percentage of exploration and exploitation of the LEA during the iteration process using the method defined in the literature [85]. It can be seen that the LEA exhibits different search behaviors on different functions. Combined with the numerical results, it can be concluded that the LEA is adaptable on different problems. Therefore, the proposed LEA is highly competitive and has potential for further development.

4.3 Convergence analysis

The ultimate goal of all optimization algorithms is to find the global optimal solution accurately and quickly. Generally speaking, in the early stages of optimization,

Table 9 Ave and the Var of optimization results for the CEC2022 benchmark functions

No.	Metrics	LEA	L-SHADE	AL-SHADE	L-SHADE−spacma	AFDB-ARO	FDB-AGDE	FDB-AGSK	FDB-PPSO
F1	Ave	3.00E+02	1.08E+03	3.00E+02	2.74E+03	3.00E+02	3.00E+02	3.00E+02	2.52E+04
	Var	2.20E−15	1.02E+07	1.00E−27	4.74E+07	1.07E−26	1.06E−26	0.00E+00	1.06E+08
F2	Ave	4.34E+02	4.47E+02	4.48E+02	4.47E+02	4.33E+02	4.41E+02	4.12E+02	6.02E+02
	Var	4.82E+02	8.05E+01	3.25E+00	8.04E+01	5.36E+02	3.43E+02	3.55E+02	1.26E+04
F3	Ave	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.16E+02
	Var	1.91E−01	3.87E−04	1.18E−03	1.24E−03	1.33E−06	3.96E−04	2.93E−21	6.21E+01
F4	Ave	8.62E+02	8.13E+02	8.09E+02	8.12E+02	8.69E+02	8.24E+02	8.25E+02	8.49E+02
	Var	3.37E+02	1.26E+01	4.40E+00	9.04E+00	5.72E+02	5.06E+01	2.26E+01	1.68E+02
F5	Ave	1.14E+03	9.01E+02	9.00E+02	9.01E+02	1.25E+03	9.02E+02	9.00E+02	2.55E+03
	Var	6.70E+04	7.34E−01	4.12E−02	4.24E−01	7.75E+04	1.08E+01	5.16E−04	5.60E+05
F6	Ave	8.73E+03	1.85E+03	1.84E+03	1.87E+03	1.83E+03	1.85E+03	1.80E+03	1.83E+08
	Var	5.21E+07	1.06E+03	9.54E+02	1.84E+03	1.86E+03	3.01E+03	2.49E+00	3.23E+17
F7	Ave	2.04E+03	2.02E+03	2.02E+03	2.02E+03	2.05E+03	2.02E+03	2.01E+03	2.12E+03
	Var	3.12E+02	4.27E+01	6.57E+01	4.07E+01	2.50E+02	7.35E+01	3.59E+01	1.66E+03
F8	Ave	2.22E+03	2.22E+03	2.22E+03	2.22E+03	2.22E+03	2.22E+03	2.22E+03	2.28E+03
	Var	9.01E+00	2.42E−01	2.87E−01	2.61E+00	1.02E+00	2.44E+01	3.21E+00	5.96E+03
F9	Ave	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.49E+03	2.48E+03	2.65E+03
	Var	1.01E−06	7.13E−27	0.00E+00	1.28E−25	3.57E−25	3.30E+02	0.00E+00	1.20E+04
F10	Ave	2.52E+03	2.50E+03	2.51E+03	2.51E+03	2.48E+03	2.51E+03	2.50E+03	3.29E+03
	Var	2.59E+03	2.37E+03	2.65E+03	1.81E+03	6.05E+03	6.44E+02	9.76E−04	1.08E+06
F11	Ave	2.90E+03	2.91E+03	2.91E+03	2.93E+03	2.86E+03	2.92E+03	2.89E+03	4.44E+03
	Var	1.27E+04	1.20E+03	9.31E+02	7.68E+03	1.08E+04	1.66E+03	1.50E+04	1.57E+06
F12	Ave	2.95E+03	2.94E+03	2.94E+03	2.95E+03	2.98E+03	2.98E+03	2.93E+03	3.04E+03
	Var	1.33E+02	3.23E+01	9.30E+01	1.59E+02	3.73E+02	1.39E+04	1.74E+00	3.33E+03

Table 10 Min and the Max of optimization results for the CEC2022 benchmark functions

No.	Metrics	LEA	L-SHADE	AL-SHADE	L-SHADE-spacma	AFDB-ARO	FDB-AGDE	FDB-AGSK	FDB-PPSO
F1	Min	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	3.00E+02	9.18E+03
	Max	3.00E+02	1.64E+04	3.00E+02	2.76E+04	3.00E+02	3.00E+02	3.00E+02	4.78E+04
F2	Min	4.00E+02	4.00E+02	4.45E+02	4.00E+02	4.00E+02	4.00E+02	4.00E+02	4.83E+02
	Max	4.49E+02	4.49E+02	4.49E+02	4.49E+02	4.49E+02	4.49E+02	4.45E+02	1.02E+03
F3	Min	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.07E+02
	Max	6.02E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.00E+02	6.44E+02
F4	Min	8.30E+02	8.07E+02	8.04E+02	8.08E+02	8.30E+02	8.12E+02	8.13E+02	8.27E+02
	Max	9.21E+02	8.22E+02	8.12E+02	8.19E+02	9.29E+02	8.42E+02	8.32E+02	8.81E+02
F5	Min	9.09E+02	9.00E+02	9.00E+02	9.00E+02	9.56E+02	9.00E+02	9.00E+02	1.53E+03
	Max	2.14E+03	9.04E+02	9.01E+02	9.02E+02	1.82E+03	9.18E+02	9.00E+02	4.16E+03
F6	Min	1.93E+03	1.80E+03	1.80E+03	1.81E+03	1.80E+03	1.80E+03	1.80E+03	4.69E+05
	Max	2.06E+04	1.94E+03	1.96E+03	2.03E+03	2.04E+03	2.03E+03	1.81E+03	2.22E+09
F7	Min	2.02E+03	2.00E+03	2.00E+03	2.00E+03	2.02E+03	2.00E+03	2.00E+03	2.05E+03
	Max	2.09E+03	2.03E+03	2.02E+03	2.03E+03	2.08E+03	2.04E+03	2.02E+03	2.21E+03
F8	Min	2.22E+03	2.22E+03	2.22E+03	2.21E+03	2.22E+03	2.20E+03	2.21E+03	2.23E+03
	Max	2.24E+03	2.22E+03	2.22E+03	2.22E+03	2.23E+03	2.22E+03	2.22E+03	2.49E+03
F9	Min	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.53E+03
	Max	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.48E+03	2.56E+03	2.48E+03	2.99E+03
F10	Min	2.50E+03	2.40E+03	2.40E+03	2.40E+03	2.40E+03	2.50E+03	2.50E+03	2.50E+03
	Max	2.67E+03	2.62E+03	2.63E+03	2.62E+03	2.68E+03	2.64E+03	2.50E+03	5.70E+03
F11	Min	2.60E+03	2.90E+03	2.90E+03	2.90E+03	2.60E+03	2.90E+03	2.60E+03	3.21E+03
	Max	3.00E+03	3.00E+03	3.00E+03	3.36E+03	2.90E+03	3.00E+03	3.00E+03	9.03E+03
F12	Min	2.94E+03	2.93E+03	2.93E+03	2.93E+03	2.94E+03	2.94E+03	2.93E+03	2.95E+03
	Max	2.99E+03	2.96E+03	2.98E+03	2.99E+03	3.01E+03	3.59E+03	2.94E+03	3.18E+03

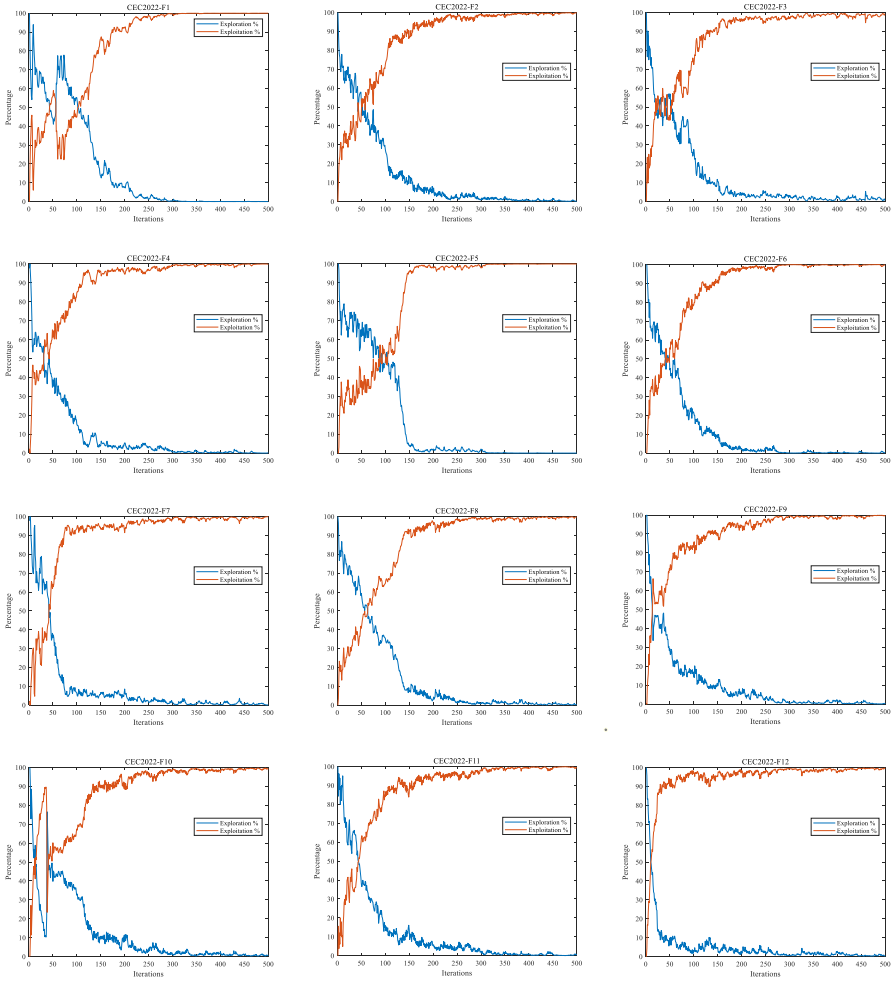

Fig. 10 Changes in the percentage of exploration and exploitation by the LEA during the iteration process

there are sudden changes in the solution, which are more favorable for the algorithm to explore unknown areas. As optimization progresses, the fluctuation of the solution decreases appropriately so that an algorithm can focus on exploitation. The F3, F4, F7, F9, F22, and F27 functions (2 dimensions) of the CEC2017 benchmark test set are selected to evaluate the convergence behavior of the LEA.

The 2D images, search history, trajectory, and average fitness (the population size and the maximum number of iterations are set to 30 and 500, respectively) are given in Fig. 11. Search history and trajectory provides a visual representation of exploration and exploitation. In the search history, the less dense areas are exploration, and the denser areas are exploitation. It is obvious from F24 that the LEA jumps out of

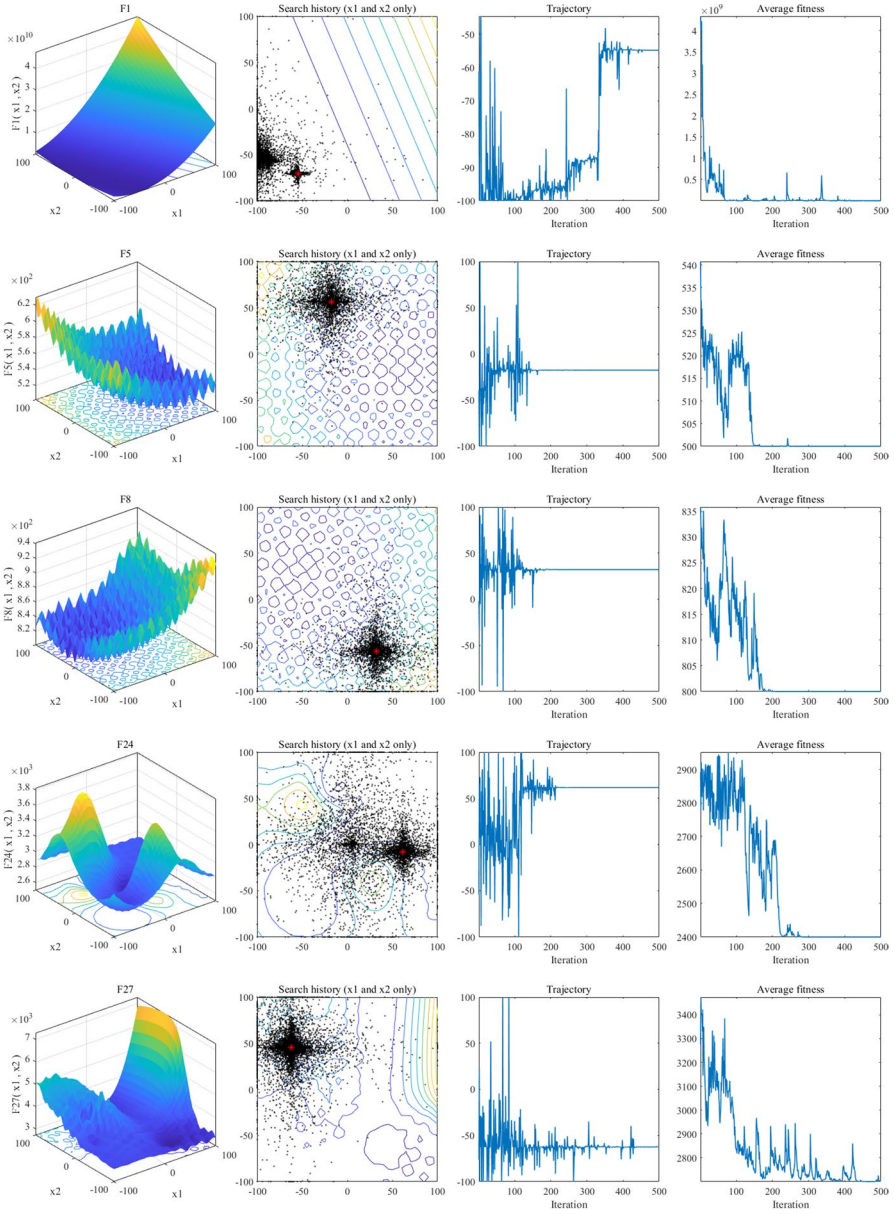

Fig. 11 Function images, search history, trajectory, and average fitness on some CEC2017 benchmark functions

the local optimal solution and then quickly converges to the vicinity of the global optimal solution. It is noteworthy that after 200 iterations, the trajectory graph of F27 still shows relatively large changes. This indicates that the LEA has the motivation to explore better areas even in the late iterations. It can be noticed that there

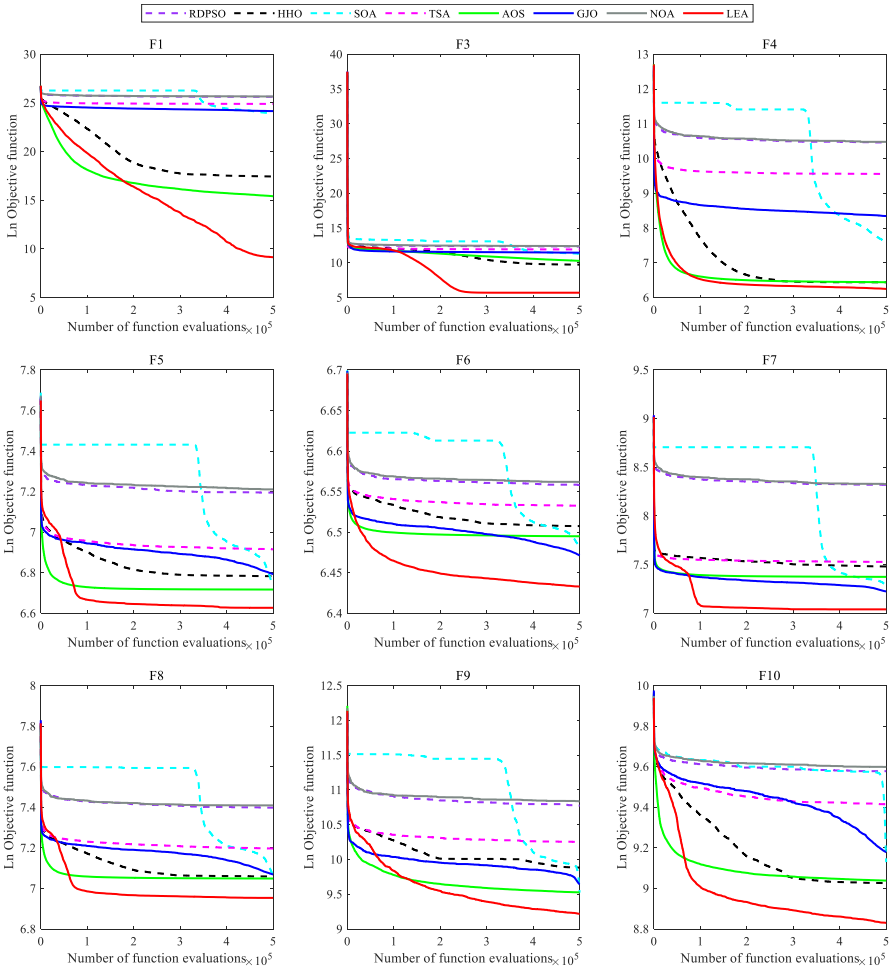

Fig. 12 Convergence curves on the unimodal and multimodal functions of the CEC2017 benchmark test set

are two regions of high density in the search history of F1 and that the trajectory of F1 fluctuates a lot in the early stage and shows large fluctuations between iterations of 300 and 400. This indicates that the richer population diversity of the LEA can lead to a greater drive for exploitation. On F5 and F8, the solution fluctuates more in the early stage, but the location is still clustered in an ideal area. This indicates that for some complex optimization problems, LEA can quickly explore to have an ideal area. The average fitness values of all the functions show a decreasing trend with the number of iterations, which verifies the convergence of the LEA.

Figures 12, 13, and 14 give the convergence curves of the LEA and the competitors for some CEC2017 benchmark test functions. The convergence speed of the LEA is obviously faster than the other compared algorithms on F1, F3,

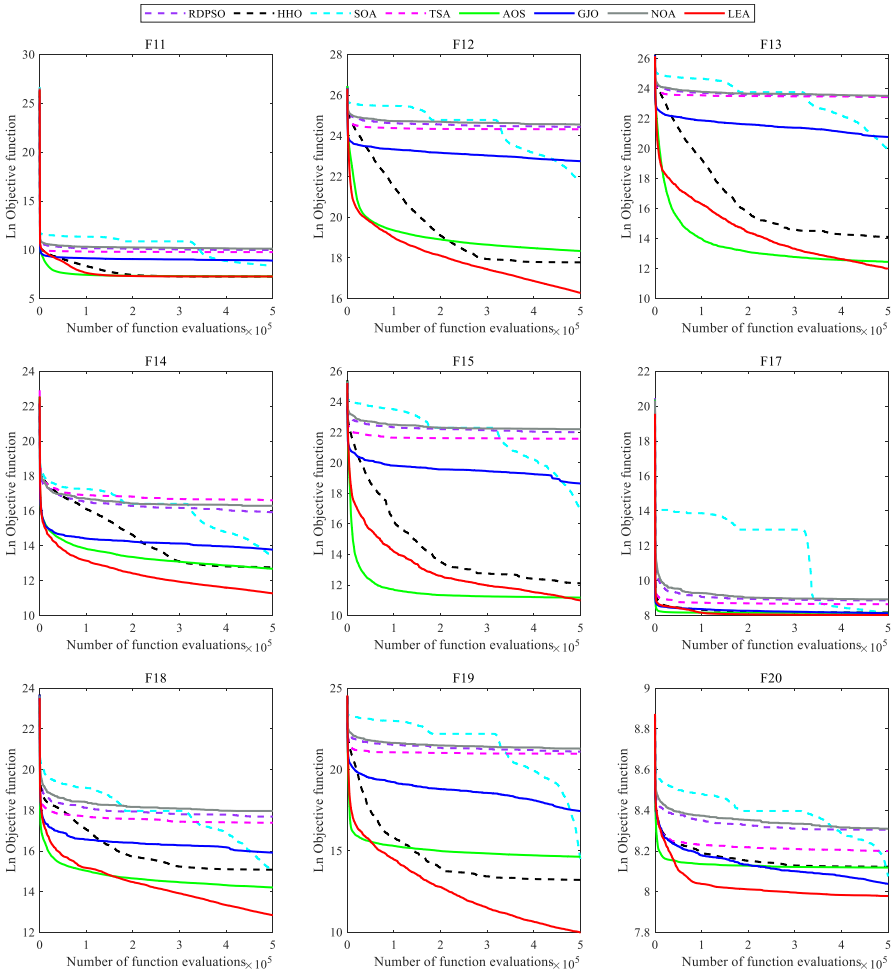

Fig. 13 Convergence curves on some hybrid functions of the CEC2017 benchmark test set

F5–F10, F12, F18, F19, F22, and F30. On F3, F9, and F15, although the LEA does not have outstanding convergence speed in the early iterations, it converges faster in the middle and late iterations, and eventually achieves the best optimization results as well. In addition, AOS ranks 2nd on F1, F5, F8, F9, F13, F15, F18, and F22 obviously; HHO ranks 2nd on F3, F10, F12, F19, and F30; and GJO obviously ranks 2nd no F6 and F7. Specifically, SOA converges slowly in the early stage, but significantly faster in the middle of the iteration, and also achieves better optimization results. It can be seen that the NOA, the TSA and the RDPSO perform the worst and do not achieve good convergence results on most of the functions. The variation of the convergence curves indicates that the

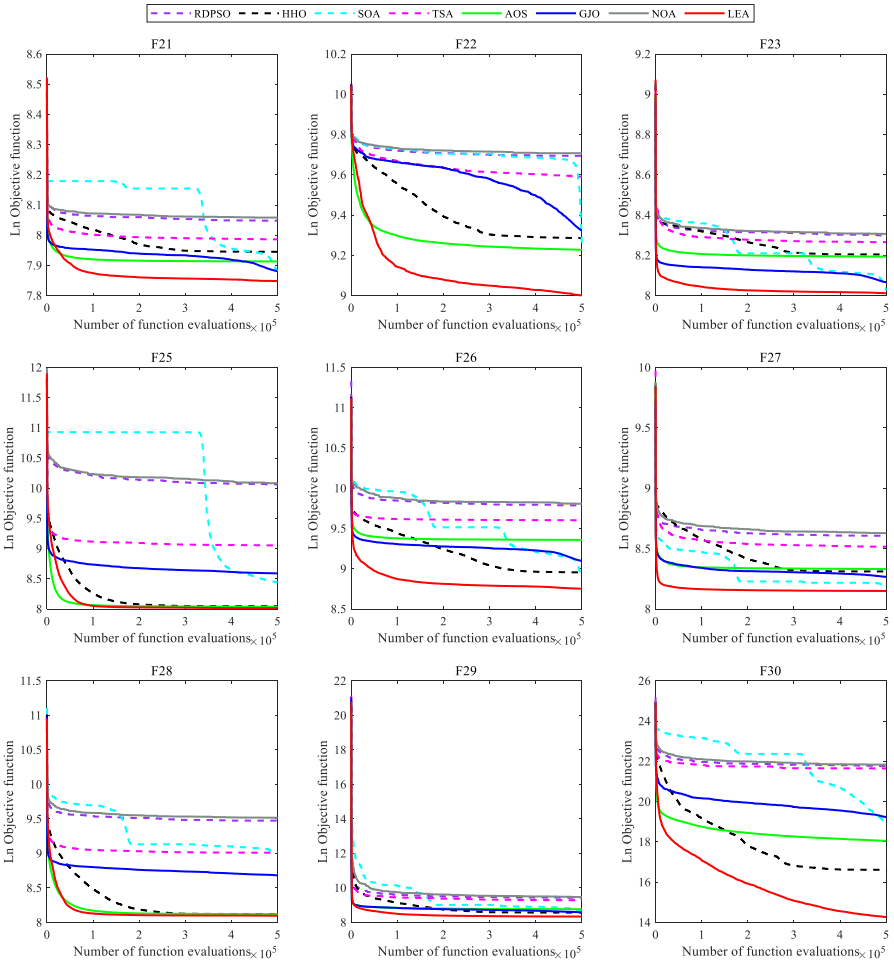

Fig. 14 Convergence curves on some composition functions of the CEC2017 benchmark test set

LEA converges faster on most of the functions compared to the competitors and satisfactory optimization results can be obtained.

4.4 Statistical tests

The optimization results of the metaheuristic algorithm are random. A simple comparison between algorithms only does not indicate algorithmic merit. For this reason, this section performs statistical tests on the optimization results on the CEC2017 benchmark functions. The Wilcoxon signed-rank test [86] is used for statistical testing at a significance level of 0.05. The original hypothesis H_0: The median of the LEA's 30

Table 11 *p*-values obtained by Wilcoxon signed-rank test at a significance level of 0.05

No.	LEA vs. NOA	LEA vs. GJO	LEA vs. AOS	LEA vs. TSA	LEA vs. SOA	LEA vs. HHO	LEA vs. RDPSO
F1	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11
F3	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11
F4	1.51E−11	1.51E−11	1.42E−08	1.51E−11	1.51E−11	2.98E−09	1.51E−11
F5	1.51E−11	1.08E−10	3.54E−08	1.51E−11	7.15E−09	5.47E−11	1.51E−11
F6	1.51E−11	2.10E−10	1.51E−11	1.51E−11	6.03E−11	1.51E−11	1.51E−11
F7	1.51E−11	6.44E−10	1.51E−11	1.51E−11	2.75E−11	1.51E−11	1.51E−11
F8	1.51E−11	8.88E−11	2.09E−09	1.51E−11	7.32E−11	9.28E−10	1.51E−11
F9	1.51E−11	3.83E−05	8.18E−06	1.51E−11	1.59E−04	1.84E−11	1.51E−11
F10	1.51E−11	4.05E−10	9.30E−07	1.51E−11	7.73E−10	5.78E−08	1.51E−11
F11	1.51E−11	1.51E−11	1.66E−01	1.51E−11	1.51E−11	8.02E−01	1.51E−11
F12	1.51E−11	1.51E−11	5.47E−11	1.51E−11	1.51E−11	7.32E−11	1.51E−11
F13	1.51E−11	1.67E−11	2.36E−04	1.51E−11	2.25E−11	1.51E−11	1.51E−11
F14	1.51E−11	1.30E−10	1.82E−08	1.51E−11	2.50E−09	2.77E−08	1.51E−11
F15	1.51E−11	1.74E−10	3.92E−01	1.51E−11	1.51E−11	9.28E−10	1.51E−11
F16	1.51E−11	3.37E−01	1.03E−03	3.35E−11	2.95E−01	4.77E−07	1.51E−11
F17	1.51E−11	3.49E−03	1.06E−04	2.31E−10	9.79E−02	3.38E−05	1.51E−11
F18	1.51E−11	8.88E−11	9.28E−10	1.51E−11	1.51E−11	2.49E−11	1.51E−11
F19	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11
F20	1.51E−11	3.81E−03	1.54E−08	4.45E−10	4.76E−04	5.05E−09	1.51E−11
F21	1.51E−11	1.98E−08	1.19E−10	1.51E−11	1.76E−07	1.51E−11	1.51E−11
F22	1.51E−11	3.01E−08	3.61E−06	1.51E−11	1.01E−07	1.30E−08	1.51E−11
F23	1.51E−11	1.22E−09	1.51E−11	1.51E−11	2.11E−04	1.51E−11	1.51E−11
F24	1.51E−11	1.29E−07	1.51E−11	1.51E−11	9.76E−01	1.51E−11	1.51E−11
F25	1.51E−11	1.51E−11	5.51E−09	1.51E−11	1.51E−11	3.06E−10	1.51E−11
F26	1.51E−11	1.51E−11	1.51E−11	1.51E−11	3.54E−08	3.39E−02	1.51E−11
F27	1.51E−11	2.04E−11	1.84E−11	1.51E−11	5.97E−07	2.49E−11	1.51E−11
F28	1.51E−11	1.51E−11	1.58E−10	1.51E−11	1.51E−11	2.31E−10	1.51E−11
F29	1.51E−11	6.64E−11	1.51E−11	1.51E−11	8.88E−11	2.04E−11	1.51E−11
F30	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11	1.51E−11

test results is greater than the median of 30 test results of a specific comparison algorithm. Alternative hypothesis H_1: A comparison algorithm's median of 30 test results is greater than the median of 30 test results of the LEA.

Table 11 gives the *p*-values obtained by Wilcoxon signed-rank test for the LEA and each comparison algorithms at a significance level of 0.05. From the results, it can be concluded that the vast majority of *p*-values are less than 0.05, i.e., the vast majority of results are accepted for the alternative hypothesis H_1, which verifies the superiority of the LEA.

Table 12 Friedman test for ranking results

No.	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
F1	1	8	5	2	6	4	3	7
F3	1	8	5	3	6	4	2	7
F4	1	8	5	3	6	4	2	7
F5	1	8	5	2	6	3	4	7
F6	1	8	2	4	6	3	5	7
F7	1	8	2	4	6	3	5	7
F8	1	8	5	2	6	4	3	7
F9	1	8	4	3	6	2	5	7
F10	1	8	5	3	6	4	2	7
F11	2	8	5	3	6	4	1	7
F12	1	8	5	3	6	4	2	7
F13	1	8	5	2	6	4	3	7
F14	1	7	5	2	8	4	3	6
F15	1	8	5	2	6	4	3	7
F16	1	8	3	4	6	2	5	7
F17	1	8	3	4	6	2	5	7
F18	1	8	5	2	6	3	4	7
F19	1	8	5	4	6	3	2	7
F20	1	8	2	4	6	3	5	7
F21	1	8	3	4	6	2	5	7
F22	1	8	5	2	6	3	4	7
F23	1	8	3	4	6	2	5	7
F24	2	8	3	4	5	1	6	7
F25	1	8	5	2	6	4	3	7
F26	1	8	4	5	6	2	3	7
F27	1	8	3	5	6	2	4	7
F28	1	8	4	2	5	6	3	7
F29	1	8	3	5	6	4	2	7
F30	1	8	5	3	6	4	2	7
FAR	1.07	7.97	4.10	3.17	6.00	3.24	3.48	6.97
Rank	1	8	5	2	6	3	4	7

FAR is Friedman average rank

To further demonstrate the capabilities of the LEA to solve the benchmark test problems, Friedman test [87] is performed on all algorithms. The ranking results of each algorithm are shown in Table 12. From the ranking results, LEA ranks 1st, AOS ranks 2nd, HHO ranks 3rd, GJO ranks 4th, SOA ranks 5th, LFD ranks 6th, EBS ranks 7th, FHO ranks 8th, SPO ranks 9th, and TSA ranks 10th.

Table 13 Optimization results for CEC2017 benchmark functions (30, 50, and 100 dimensions)

No.	Metrics	30	50	100	No.	Metrics	30	50	100
F1	Ave	7.84E+03	9.39E+03	2.60E+04	F1	Var	5.39E+07	1.01E+08	5.23E+08
F3	Ave	3.00E+02	3.00E+02	3.00E+02	F3	Var	1.13E−10	3.94E−09	4.42E−06
F4	Ave	4.81E+02	5.18E+02	6.47E+02	F4	Var	4.42E+02	2.75E+03	1.28E+03
F5	Ave	6.23E+02	7.55E+02	1.21E+03	F5	Var	7.03E+02	1.88E+03	6.12E+03
F6	Ave	6.06E+02	6.22E+02	6.52E+02	F6	Var	3.07E+01	1.03E+02	2.50E+01
F7	Ave	9.00E+02	1.14E+03	2.13E+03	F7	Var	1.75E+03	7.87E+03	6.18E+04
F8	Ave	9.13E+02	1.05E+03	1.52E+03	F8	Var	6.76E+02	2.29E+03	1.15E+04
F9	Ave	2.98E+03	1.01E+04	2.69E+04	F9	Var	1.56E+06	1.07E+07	2.43E+07
F10	Ave	3.89E+03	6.84E+03	1.44E+04	F10	Var	3.30E+05	7.66E+05	2.29E+06
F11	Ave	1.30E+03	1.44E+03	2.59E+03	F11	Var	3.32E+03	5.18E+03	8.45E+04
F12	Ave	1.98E+06	1.18E+07	4.00E+07	F12	Var	1.34E+12	4.59E+13	2.53E+14
F13	Ave	7.44E+04	1.62E+05	1.28E+05	F13	Var	1.36E+09	9.57E+09	1.80E+09
F14	Ave	8.28E+03	7.86E+04	4.01E+05	F14	Var	1.99E+07	3.43E+09	4.26E+10
F15	Ave	2.10E+04	5.94E+04	1.12E+05	F15	Var	1.50E+08	9.45E+08	2.79E+09
F16	Ave	2.50E+03	3.49E+03	5.65E+03	F16	Var	1.21E+05	1.77E+05	1.93E+05
F17	Ave	2.12E+03	3.08E+03	5.40E+03	F17	Var	3.37E+04	1.51E+05	1.94E+05
F18	Ave	1.97E+05	3.79E+05	7.13E+05	F18	Var	2.12E+10	2.93E+10	7.03E+10
F19	Ave	1.32E+04	2.18E+04	8.26E+04	F19	Var	1.57E+08	1.15E+08	9.57E+08
F20	Ave	2.35E+03	2.92E+03	4.62E+03	F20	Var	2.14E+04	4.60E+04	1.70E+05
F21	Ave	2.41E+03	2.56E+03	3.11E+03	F21	Var	1.03E+03	2.62E+03	8.84E+03
F22	Ave	2.55E+03	8.11E+03	1.73E+04	F22	Var	8.69E+05	5.00E+06	2.61E+06
F23	Ave	2.77E+03	3.02E+03	3.51E+03	F23	Var	7.88E+02	4.41E+03	7.45E+03
F24	Ave	2.96E+03	3.22E+03	4.07E+03	F24	Var	3.21E+03	1.22E+04	1.43E+04
F25	Ave	2.89E+03	3.02E+03	3.27E+03	F25	Var	7.59E+01	1.61E+03	2.97E+03
F26	Ave	5.07E+03	6.32E+03	1.35E+04	F26	Var	3.10E+05	5.34E+05	1.65E+06
F27	Ave	3.23E+03	3.46E+03	3.54E+03	F27	Var	4.37E+02	7.27E+03	6.18E+03
F28	Ave	3.18E+03	3.29E+03	3.38E+03	F28	Var	4.45E+03	7.52E+02	1.99E+03
F29	Ave	3.73E+03	4.19E+03	6.74E+03	F29	Var	2.23E+04	8.34E+04	3.11E+05
F30	Ave	3.44E+04	1.57E+06	8.12E+05	F30	Var	1.79E+08	2.13E+11	7.91E+10

4.5 Scalability analysis

Scalability can be used to evaluate the performance of metaheuristic algorithms in the face of growing problem size, complexity, or resource constraints. Algorithms are usually considered competitive and better scalable when they can effectively handle high-dimensional spaces or large-scale problems without sacrificing performance and efficiency. In this section, the scalability of the LEA is evaluated using the CEC2017 benchmark functions (30, 50, and 100 dimensions).

The optimization results are recorded in Table 13. From the analysis, it can be obtained that (1) on the unimodal functions, the results do not become much larger as the dimensionality increases (F3 is particularly reflective of this); (2)

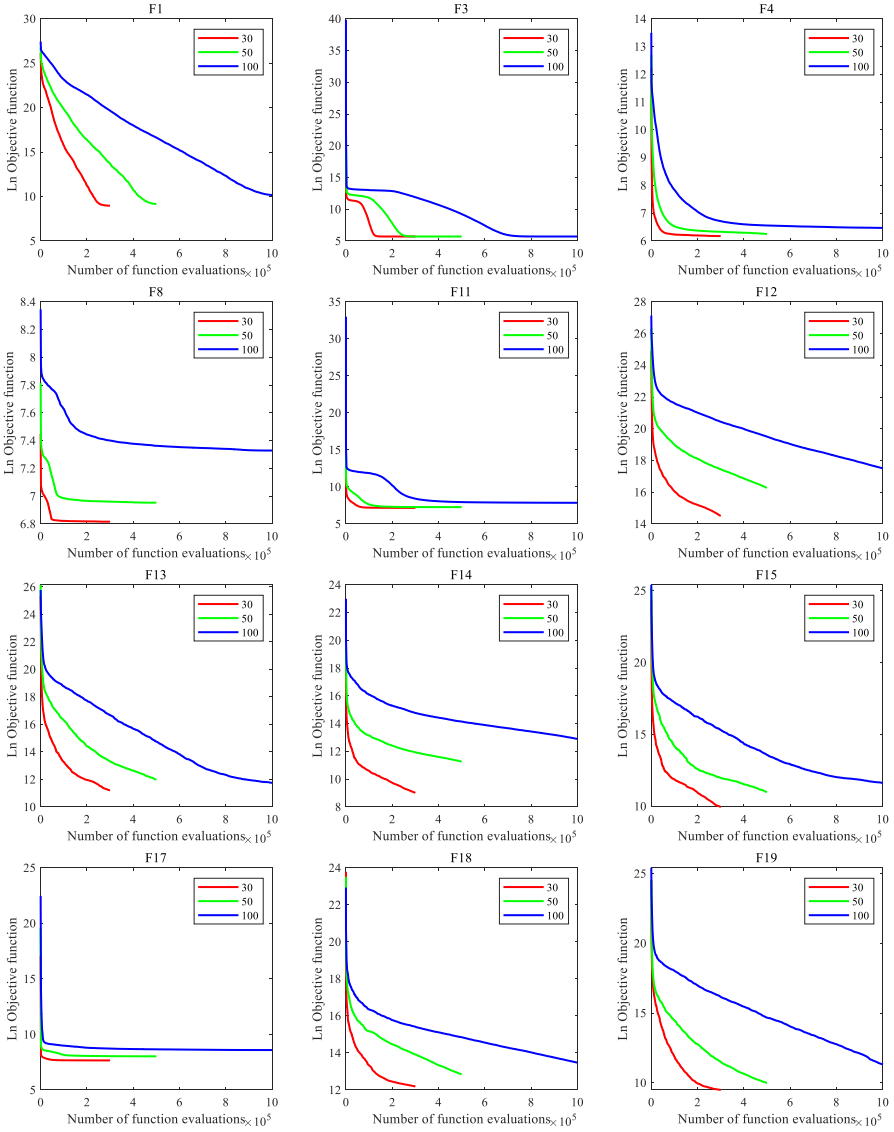

Fig. 15 Convergence curves of the LEA on some CEC2017 unimodal, multimodal and hybrid functions (30, 50, and 100 dimensions)

on the multimodal functions F5 to F10, the growth of the optimization results of the LEA on all dimensions is not significant, except for F5, F9, and F10; (3) it is worth noting that the optimization results do not show an order of magnitude increase with increasing dimensions on the hybrid functions F11, F13, and F16–F20; (4) on the composition functions F21, F23, F24, F25, and F27–F29, the changes in the optimization results are relatively small; and (5) on F30, the result

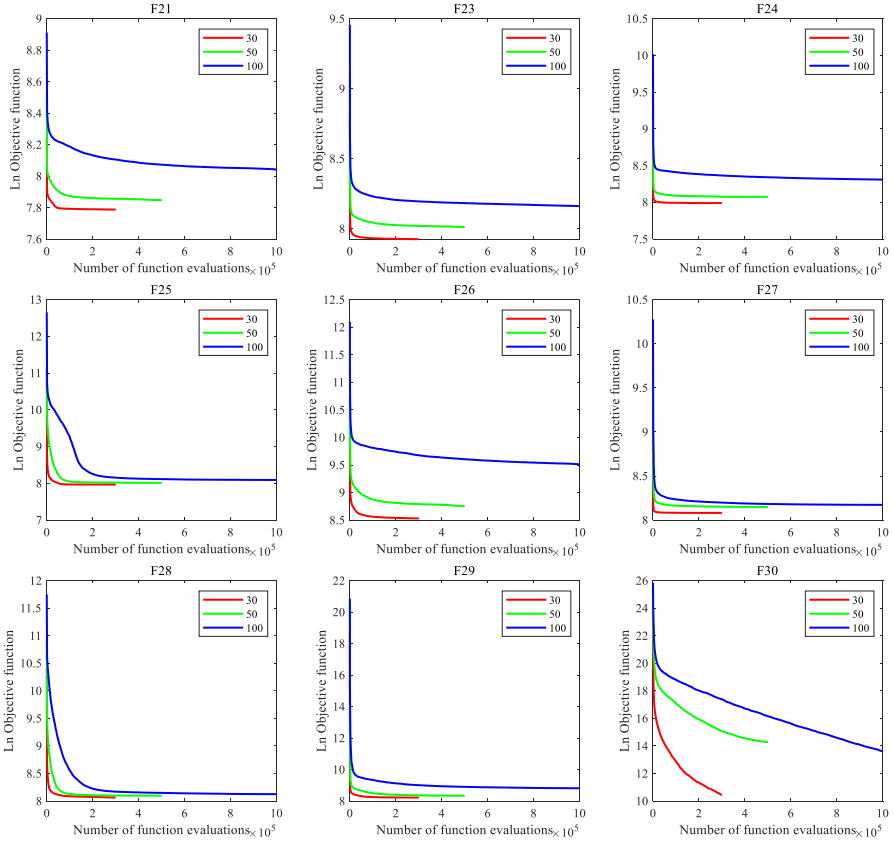

Fig. 16 Convergence curves of the LEA on some CEC2017 composition functions (30, 50, and 100 dimensions)

reaches its maximum at 50 dimensions and gets decreased again on at 100 dimensions, which may be due to the increase of MaxFEs with the stronger search capability of the LEA.

Figures 15 and 16 give the convergence curves obtained by the LEA solving part of the CEC2017 benchmark functions (30, 50, and 100 dimensions). It can be seen that the convergence curves of F1, F12–F15, F18, F19, and F30 still do not converge at the later stages. It can be assumed that sufficient MaxFEs have a higher probability of giving similar results for the LEA in 30, 50, and 100 dimensions. Moreover, the convergence curves for F3, F4, F11, F17, F25, and F27–F29 level off in the later stages, but the optimization results do not get much larger with increasing dimensionality. Some bad performance is reflected in F8, F21, F23, F24, and F26. On these functions, the LEA tends to converge at a later stage and the optimization results vary more with increasing dimensions. However, as given in Table 13, these increases are still within acceptable limits.

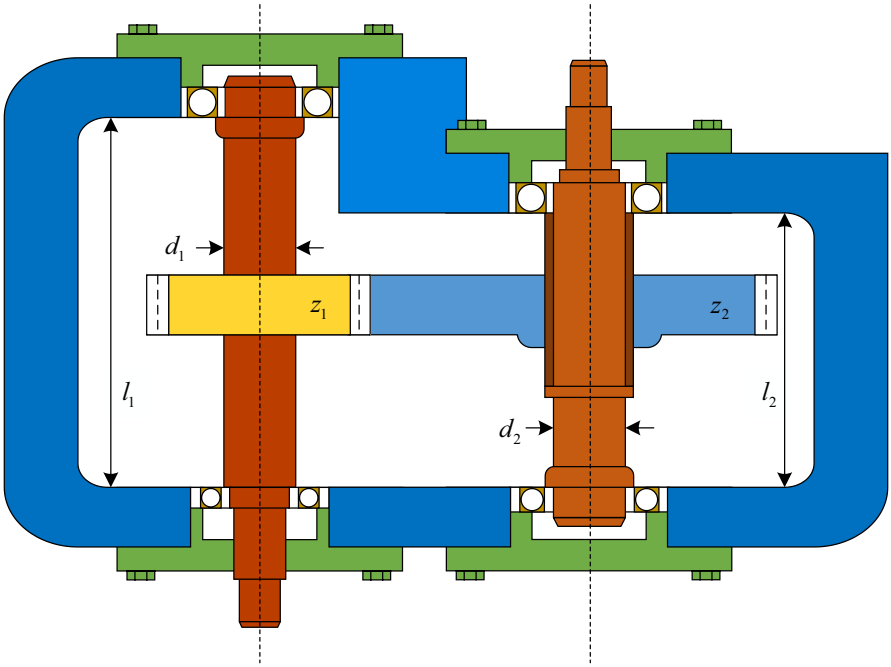

Fig. 17 Schematic diagram of the speed reducer design problem

5 Applications to real-world optimization problems

Most practical engineering problems are difficult to solve with more constraints. In order to test the capability of the LEA to solve real-world optimization problems, eight engineering problems are selected, which are speed reducer design problem, pressure vessel design problem, cantilever beam design problem, I-beam design problem, tubular column design problem, piston lever design problem, rolling element bearing design problem, and welded beam design problem. The LEA and its competitors in Table 3 are run independently 30 times. Furthermore, the optimization performance of different algorithms is analyzed through the best of these 30 results.

5.1 Speed reducer design problem

In a mechanical system, the speed reducer is one of the essential components of the gearbox. The design of the speed reducer (Fig. 17) is a challenging problem. In this problem, the weight of the reducer is minimized subject to 11 constraints [88]. The problem has seven variables, which are the face width of teeth (z_1), module of teeth (z_2), the number of teeth in the pinion (p), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2), the diameter of first shafts (d_1)

and the diameter of second shafts (d_2). The mathematical form of the problem is given in Eq. (22).

Consider:

$$\mathbf{x} = \left[x_1, x_2, x_3, x_4, x_5, x_6, x_7\right] = \left[z_1, z_2, p, l_1, l_2, d_1, d_2\right].$$

Minimize:

$$\begin{aligned} f(\mathbf{x}) = {} & 0.7854 x_1 x_2^2 \left(3.3333 x_3^2 + 14.9334 x_3 - 43.0934\right) - 1.508 x_1 \left(x_6^2 + x_7^2\right) \\ & + 7.4777 \left(x_6^3 + x_7^3\right) + 0.7854 \left(x_4 x_6^2 + x_5 x_7^2\right). \end{aligned} \tag{22}$$

Subject to:

$$g_1(\mathbf{x}) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0, g_2(\mathbf{x}) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0,$$

$$g_3(\mathbf{x}) = \frac{1.93 x_4^3}{x_2 x_6^4 x_3} - 1 \le 0, g_4(\mathbf{x}) = \frac{1.93 x_5^3}{x_2 x_7^4 x_3} - 1 \le 0,$$

$$g_5(\mathbf{x}) = \frac{\sqrt{\left(745 x_4 \big/ x_2 x_3\right)^2 + 16.9 \times 10^6}}{110 x_6^3} - 1 \le 0,$$

$$g_6(\mathbf{x}) = \frac{\sqrt{\left(745 x_5 \big/ x_2 x_3\right)^2 + 157.5 \times 10^6}}{85 x_7^3} - 1 \le 0,$$

$$g_7(\mathbf{x}) = \frac{x_2 x_3}{40} - 1 \le 0, g_8(\mathbf{x}) = \frac{5 x_2}{x_1} - 1 \le 0,$$

$$g_9(\mathbf{x}) = \frac{x_1}{12 x_2} - 1 \le 0, g_{10}(\mathbf{x}) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0,$$

$$g_{11}(\mathbf{x}) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0.$$

Variable range:

$2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ x_3 \in \{17, 18, 19, \ldots, 27, 28\},\ 7.3 \le x_4, x_5 \le 8.3,$ $2.9 \le x_6 \le 3.9, 5 \le x_7 \le 5.5.$

The best results obtained by the LEA and the competitors on the speed reducer design problem are shown in Table 14. It can be concluded that the LEA achieves the top ranked result provided that the constraints are satisfied. It is noted that the LEA far outperformed the NOA, TSA, and RDPSO on this problem. Meanwhile, the AOS achieved similar results to the LEA on this problem.

Table 14 Best results of different algorithms for the speed reducer design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	3.5016	3.5249	3.5019	3.5024	3.5466	3.5102	3.5000	3.5481
x_2	0.7	0.7016	0.7003	0.7	0.7	0.7	0.7	0.7004
x_3	17	17.1184	17	17	17.0331	17	17	17.0001
x_4	7.3278	7.3239	7.3335	7.3	7.4235	7.3651	7.3	7.3748
x_5	7.7169	7.8391	7.7653	7.7212	8.3	7.7630	8.0306	8.0136
x_6	3.3507	3.5386	3.3531	3.3506	3.3847	3.3671	3.3507	3.4598
x_7	5.2867	5.2895	5.2870	5.2885	5.2987	5.2889	5.2911	5.3364
g_1	− 2.168020	− 2.700088	− 2.197155	− 2.174922	− 2.601041	− 2.239779	− 2.155018	− 2.586354
g_2	− 98.356341	− 110.918885	− 98.851627	− 98.473681	− 106.698607	− 99.576249	− 98.135306	− 105.469849
g_3	− 1.882125	− 2.863187	− 1.885959	− 1.925333	− 1.895085	− 1.898767	− 1.925726	− 2.323273
g_4	− 18.297506	− 17.585963	− 17.935336	− 18.292360	− 14.507754	− 17.972795	− 16.079348	− 16.832144
g_5	− 0.104332	− 166.287588	− 2.465911	− 0.045105	− 32.757406	− 16.063081	− 0.129230	− 100.817644
g_6	− 0.001352	− 1.349149	− 0.138438	− 0.897463	− 5.698164	− 1.072400	− 2.101201	− 23.504459
g_7	− 28.1	− 27.990188	− 28.094669	− 28.1	− 28.076804	− 28.1	− 28.1	− 28.093798
g_8	− 0.002233	− 0.024316	− 0.000507	− 0.003417	− 0.066619	− 0.014539	− 0.000003	− 0.066088
g_9	− 6.997767	− 6.975684	− 6.999493	− 6.996583	− 6.933381	− 6.985461	− 6.999997	− 6.933912
g_{10}	− 0.401755	− 0.116055	− 0.403800	− 0.374120	− 0.446461	− 0.414362	− 0.373992	− 0.285088
g_{11}	− 0.001581	− 0.120662	− 0.049611	− 0.003811	− 0.571407	− 0.045239	− 0.310346	− 0.243572
f	2995.36071	3087.48576	2998.83561	2996.68852	3048.99746	3005.72116	3004.23387	3082.91734

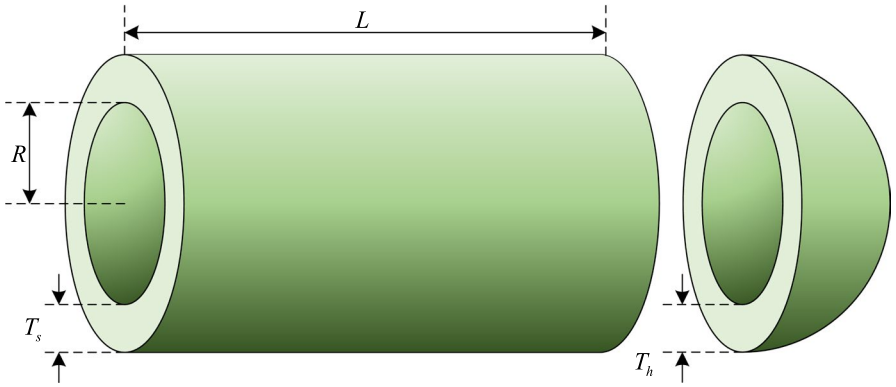

Fig. 18 Schematic diagram of the pressure vessel design problem

5.2 Pressure vessel design problem

In this problem, the ends of the cylindrical container are covered by hemispherical caps (as shown in Fig. 18). There are four variables and four constraints with the objective of minimizing the total cost in this problem [89]. The four variables are thickness of the shell (T_s), thickness of the head (T_h), the inner radius (R), and the length of the cylindrical section of the vessel without the head (L). The mathematical description of the problem is given in Eq. (23).

Consider:

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

Minimize:

$$f(\mathbf{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3. \tag{23}$$

Subject to:

$$\begin{aligned} g_1(\mathbf{x}) &= -x_1 + 0.0193x_3 \le 0, \\ g_2(\mathbf{x}) &= -x_2 + 0.00954x_3 \le 0, \\ g_3(\mathbf{x}) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0, \\ g_4(\mathbf{x}) &= x_4 - 240 \le 0. \end{aligned}$$

Variable range:

$x_1, x_2 \in \{1 \times 0.0625, 2 \times 0.0625, \ldots, 99 \times 0.0625\}, 10 \le x_3, x_4 \le 200$.

The best results obtained by the different algorithms on the pressure vessel design problem are shown in Table 15. The LEA, the GJO, the SOA, and the HHO have obtained similar optimization results. But the LEA is still number one on this problem.

Table 15 Best results of different algorithms for the pressure vessel design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	12.7560	15.3250	12.7662	12.5582	16.3766	13.4356	12.6660	14.2849
x_2	6.8187	7.2589	7.4821	6.5484	8.0850	6.5396	6.7190	7.3643
x_3	42.0981	41.8698	42.0896	42.0888	51.2783	42.0938	41.8928	43.8871
x_4	176.6409	179.6195	176.7559	176.7645	90.6529	176.7263	179.2025	168.7176
g_1	−0.00001	−0.12941	−0.00017	−0.00019	−0.01033	−0.00009	−0.00397	−0.02798
g_2	−0.03588	−0.03806	−0.03596	−0.03597	−0.01080	−0.03592	−0.03784	−0.01882
g_3	−0.26402	−709.23679	−56.85303	−45.99499	−17,652.76052	−180.71967	−3.70714	−78,983.14077
g_4	−63.35906	−60.38052	−63.24409	−63.23554	−149.34706	−63.27372	−60.79750	−71.28243
f	6059.75763	6981.97412	6061.02531	6061.08291	6535.35339	6061.03561	6084.93480	6606.46794

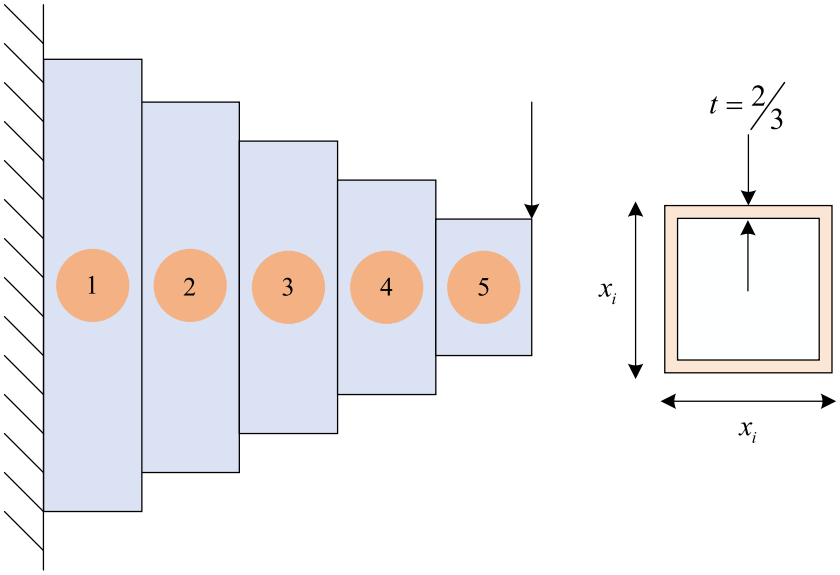

Fig. 19 Schematic diagram of the cantilever beam design problem

Table 16 Best results of different algorithms for the cantilever beam design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	6.0112	7.0416	6.0201	6.1425	5.9591	5.9921	6.0464	4.7158
x_2	5.2974	4.7512	5.2971	5.1556	5.2226	5.3039	5.2237	16.0349
x_3	4.5057	5.9815	4.4796	4.6156	4.5505	4.5222	4.5718	5.4816
x_4	3.5112	12.9275	3.5220	3.3931	3.6176	3.5062	3.5074	3.6315
x_5	2.1485	2.4502	2.1553	2.1982	2.1474	2.1501	2.1306	2.3615
g	−0.000013	−0.320301	−0.000004	−0.000230	−0.001517	−0.000039	−0.0000002	−0.071908
f	1.33997	2.06868	1.33998	1.34191	1.34143	1.34001	1.34034	2.01086

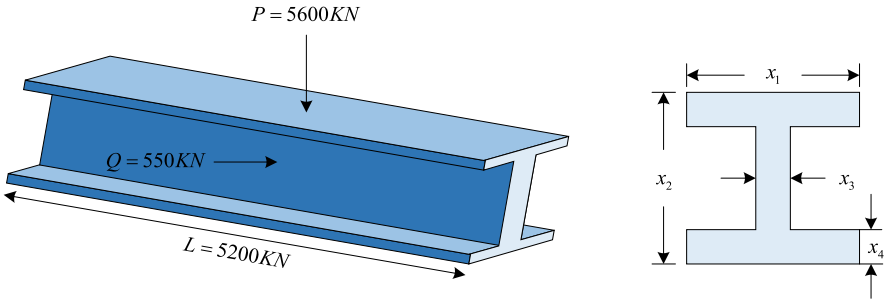

Fig. 20 Schematic diagram of the I-beam design problem

5.3 Cantilever beam design problem

The problem is an example of structural engineering design for weight optimization of a cantilever beam with a square section [90]. As depicted in Fig. 19, one end of the beam is rigidly supported, and the cantilever-free node is affected by force in the vertical direction. The beam consists of five hollow square blocks. The thickness of the blocks is kept constant 2/3 in this problem. Its height (or width) is considered a decision variable (x_1, x_2, x_3, x_4, x_5), and the problem can be expressed as Eq. (24).

Minimize:

$$f(\mathbf{x}) = 0.0624\left(x_1 + x_2 + x_3 + x_4 + x_5\right). \tag{24}$$

Subject to:

$$g(\mathbf{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0.$$

Variable range:

$$0.01 \le x_i \le 100, \quad i = 1, \ldots, 5.$$

The best solutions to this problem obtained by the LEA and the comparison algorithms are shown in Table 16. In terms of results, the result obtained by LEA are better than those of other algorithms. Among them, the optimization results of the LEA and the GJO are very close, but the LEA still has certain advantages.

5.4 I-beam design problem

The optimal design of the I-beam vertical deflection (Fig. 20) is a typical engineering optimization problem. The target of the optimization is to minimize the vertical deflection of the beam subject to given condition of the cross-sectional area as well as the stress constraint [91]. This optimization problem covers four decision

Table 17 Best results of different algorithms for the I-beam design problem

	x_1	x_2	x_3	x_4	g_1	g_2	f
LEA	80	50.0000	0.9	2.3217931	0	− 1.5702280	0.01307411916
NOA	79.6370	47.4508	0.9396	2.3686131	− 4.841887	− 1.3523652	0.01356782111
GJO	80	50	0.9	2.3217916	− 0.000065	− 1.5702273	0.01307412207
AOS	80	50	0.9	2.3217922	− 0.000007	− 1.5702284	0.01307411924
TSA	80	50	0.9	2.3211758	− 0.060534	− 1.5691801	0.01307705751
SOA	80	50	0.9	2.3217895	− 0.000271	− 1.5702238	0.01307413207
HHO	80	50	0.9	2.3217922	− 0.000010	− 1.5702283	0.01307411938
RDPSO	79.7060	37.0861	0.9243	3.1187869	− 0.768549	− 1.0201020	0.01351537425

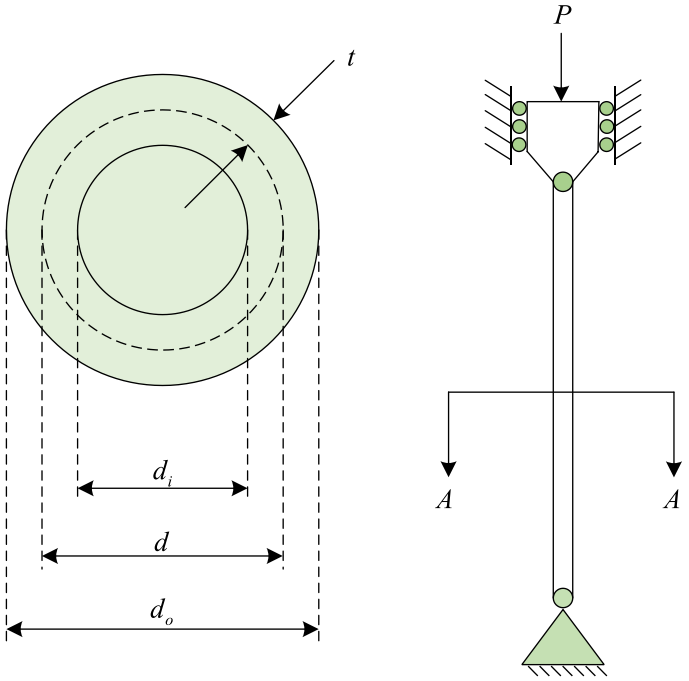

Fig. 21 Schematic diagram of the tubular column design problem

variables, including the width of flange (b), the height of section (h), the thickness of the web (t_w), and the thickness of the flange (t_f). The maximum vertical deflection of the beam is $PL^3/48EI$, where the beam length (L) is 5200 cm and the modulus of elasticity (E) is 523.104 kN/cm^2. The objective function of the problem and the constraints are as in Eq. (25).

Consider:

$$\mathbf{x} = [x_1, x_2, x_3, x_4] = [b, h, t_w, t_f].$$

Minimize:

$$f(\mathbf{x}) = \frac{5000}{x_3(x_2 - 2x_4)^3 \Big/ 12 + (x_1 x_4^3/6) + 2bx_4(x_2 - x_4/2)^2}. \tag{25}$$

Subject to:

$$g_1(\mathbf{x}) = 2x_1x_3 + x_3(x_2 - 2x_4) \le 300,$$

$$g_2(\mathbf{x}) = \frac{18x_2 \times 10^4}{x_3(x_2 - 2x_4)^3 + 2x_1x_3[4x_4^2 + 3x_2(x_2 - 2x_4)]} + \frac{15x_1 \times 10^3}{(x_2 - 2x_4)x_3^2 + 2x_3x_1^3} \le 56.$$

Variable range:

$10 \leq x_1 \leq 50, 10 \leq x_2 \leq 80, 0.9 \leq x_3 \leq 5, 0.9 \leq x_4 \leq 5.$

From Table 17, the best results of the design problem of I-beam can be seen under the optimization of different algorithms. It is found that the performance of LEA optimization search is slightly better than the AOS, the HHO, the GJO, the SOA, the TSA, and even significantly better than the RDPSO and the NOA, which ranks 1st overall.

5.5 Tubular column design problem

The optimal design of the tubular column (Fig. 21) is a typical engineering optimization problem and is an example of how engineers can minimize costs by designing a uniform column with a tubular cross section to withstand compressive loads [92]. The two main variables of this problem are the average diameter of the column (d) and the thickness of the column (t). Moreover, the yield stress of this engineering material is 500 $\mathrm{kgf}/\mathrm{cm}^2$ and the modulus of elasticity is 8.5×10^5 $\mathrm{kgf}/\mathrm{cm}^2$. The objective function of the problem and the constraints are as in Eq. (26).

Consider:

$$\mathbf{x} = \left[x_1, x_2\right] = [b, t].$$

Minimize:

$$f(\boldsymbol{x}) = 9.8x_1x_2 + 2x_1. \tag{26}$$

Subject to:

Table 18 Best results of different algorithms for the tubular column design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	5.4522	5.4456	5.4532	5.4521	5.4543	5.4510	5.4509	5.4885
x_2	0.2916	0.2937	0.2916	0.2916	0.2922	0.2918	0.2918	0.2897
g_1	−2.89E−14	−9.44E−03	−4.05E−05	−5.94E−05	−3.61E−03	−7.82E−04	−7.47E−04	−2.64E−04
g_2	−1.78E−13	−1.70E−01	−1.96E−02	−5.22E−05	−1.44E−01	−3.43E−03	−8.19E−06	−6.38E−01
g_3	−0.633174	−0.632734	−0.633246	−0.633168	−0.633314	−0.633096	−0.633087	−0.635601
g_4	−0.610559	−0.611025	−0.610482	−0.610566	−0.610410	−0.610641	−0.610651	−0.607965
g_5	−0.314191	−0.319057	−0.314075	−0.314229	−0.315482	−0.314674	−0.314676	−0.309739
g_6	−0.318477	−0.319294	−0.318344	−0.318490	−0.318217	−0.318622	−0.318639	−0.313940
f	26.48636	26.56584	26.48889	26.48674	26.52587	26.49171	26.49109	26.56155

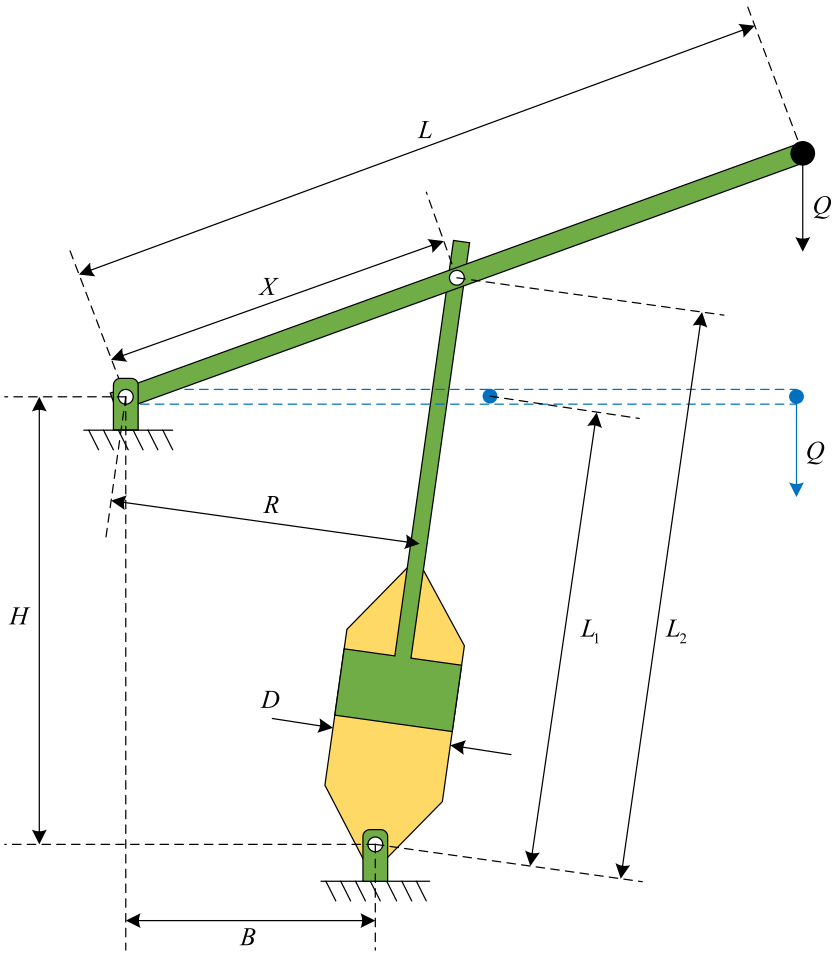

Fig. 22 Schematic diagram of the piston lever design problem

$$g_1(x) = \frac{P}{\pi x_1 x_2 \sigma_y} - 1 \le 0,$$

$$g_2(x) = \frac{8PL^2}{\pi^3 E x_1 x_2 \left(x_1^2 + x_2^2\right)} - 1 \le 0,$$

$$g_3(x) = \frac{2}{x_1} - 1 \le 0,$$

$$g_4(x) = \frac{x_1}{14} - 1 \le 0,$$

$$g_5(x) = \frac{0.2}{x_2} - 1 \le 0,$$

$$g_6(x) = \frac{x_1}{8} - 1 \le 0$$

Variable range:
$2 \le x_1 \le 14, 0.2 \le x_2 \le 0.8$.

As shown in Table 18, the best results of the optimization design problem for the tubular column in different algorithms. Experiments show that the optimization performance of the LEA outperforms the comparison algorithms. In particular, the proposed algorithm finds the best solution better than the NOA and the RDPSO, and performs similarly to and better than the GJO, the AOS, and the TSA.

5.6 Piston lever design problem

The main objective of the design of piston lever problem is to determine the parameters of the piston assembly during the lifting of the piston rod from 0° to 45°, so that the oil volume is minimized [93]. The position of the piston assembly (H, B, D, and X) is presented in Fig. 22, and its mathematical model can be expressed as Eq. (27).

Consider:

$$\mathbf{x} = \left[x_1, x_2, x_3, x_4\right] = [H, B, D, X].$$

Minimize:

$$f(\mathbf{x}) = \frac{1}{4}\pi x_3^2\left(L_2 - L_1\right). \tag{27}$$

Subject to:

$$\begin{aligned} g_1(\mathbf{x}) &= QL\cos\theta - R \times F \le 0, \\ g_2(\mathbf{x}) &= Q\left(L - x_4\right) - M_{\max} \le 0, \\ g_3(\mathbf{x}) &= 1.2\left(L_2 - L_1\right) - L_1 \le 0, \\ g_4(\mathbf{x}) &= \frac{x_3}{2} - x_2 \le 0. \end{aligned}$$

Table 19 Best results of different algorithms for the piston lever design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	0.05	2.25339	0.05	0.05009	0.05	0.05026	0.05	14.11759
x_2	2.04162	20.91339	2.04254	2.04208	2.05720	2.04180	2.07417	18.52856
x_3	4.08314	4.47145	4.08378	4.08390	4.08998	4.08302	4.10912	3.93935
x_4	120	118.752	120	119.94	120	120	118.604	119.352
g_1	−98.92	−801,591.77	−644.56	−27.35	−6010.01	−10.12	−2606.14	−581,343.39
g_2	−600,000	−587,518.07	−600,000	−599,489.9	−600,000	−600,000	−586,043.0	−593,522.6
g_3	−117.1873	−87.1167	−117.1861	−117.1356	−117.1661	−117.1869	−115.7472	−81.7720
g_4	−0.00005	−18.67767	−0.00065	−0.00013	−0.01221	−0.00028	−0.01962	−16.55888
f	8.41361	140.64335	8.41988	8.41940	8.50355	8.41628	8.65278	203.49531

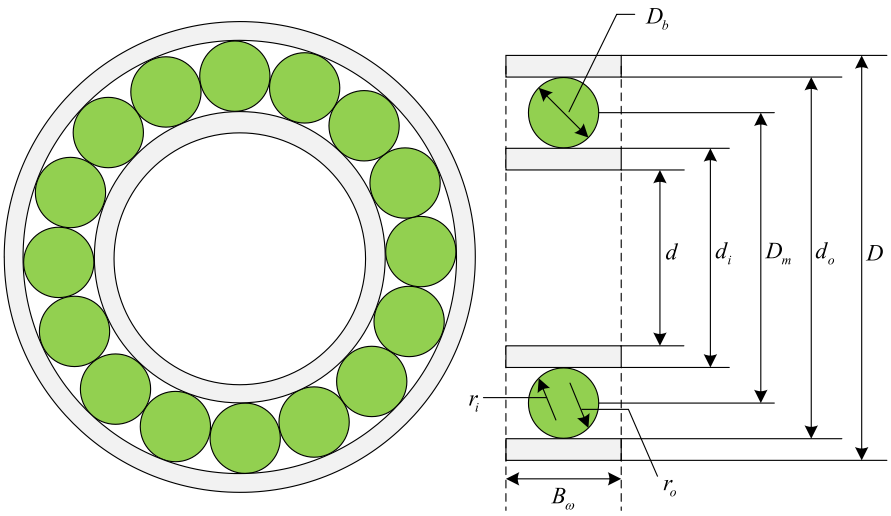

Fig. 23 Schematic diagram of the rolling element bearing design problem

where

$$R = \frac{\left|-x_4\left(x_4 \sin\theta + x_1\right) + x_1\left(x_2 - x_4\cos\theta\right)\right|}{\sqrt{\left(x_4 - x_2\right)^2 + x_1^2}},$$

$$F = \frac{\pi P x_3^2}{4},$$

$$L_1 = \sqrt{\left(x_4 - x_2\right)^2 + x_1^2},$$

$$L_2 = \sqrt{\left(x_4 \sin\theta + x_1\right)^2 + \left(x_2 - x_4\cos\theta\right)^2},$$

$$\theta = 45°,$$

$$Q = 10000 \text{ lbs},$$

$$L = 240 \text{ in},$$

$$M_{\max} = 1.8 \times 10^6 \text{ lbs in},$$

$$P = 1500 \text{ psi}.$$

Variable range:

$0.05 \le x_1, x_2, x_4 \le 500, 0.05 \le x_3 \le 120$.

The best results of the piston lever design problem optimized with different algorithms are given in Table 19. From the experiments, it can be concluded that the results of the LEA are ranked 1st while meeting the constraints. On this problem, the LEA performs far better than the NOA, the TSA, the HHO, and the RDPSO. Meanwhile, the GJO, the AOS, and the SOA showed similar results to the LEA.

Table 20 Best results of different algorithms for the rolling element bearing design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	125.7190	125.1349	125.7066	125.7235	125.6564	125.7121	125.7184	126.2746
x_2	21.4255	19.9850	21.4189	21.4218	21.3585	21.4174	21.4254	20.2788
x_3	11.2800	10.9132	10.9847	10.7159	10.9768	11.1204	11.4314	11.2958
x_4	0.515	0.5162	0.515	0.5150	0.515	0.515	0.515	0.5156
x_5	0.5150	0.5398	0.5516	0.5154	0.5567	0.5832	0.5153	0.5270
x_6	0.4290	0.4619	0.4958	0.4105	0.5	0.4690	0.4998	0.4175
x_7	0.6990	0.6447	0.6312	0.6121	0.7	0.6222	0.6996	0.6214
x_8	0.3	0.3701	0.3	0.3000	0.3	0.3004	0.3	0.3287
x_9	0.0310	0.0714	0.0220	0.0685	0.02	0.0294	0.0224	0.0375
x_{10}	0.6013	0.6179	0.6594	0.6000	0.6	0.6092	0.6017	0.6047
g_1	−0.00002	−0.53586	−0.00152	−0.00178	−0.02038	−0.00254	−0.00001	−0.50567
g_2	−12.81984	−7.63910	−8.13008	−14.11003	−7.71704	−10.00569	−7.86446	−11.33093
g_3	−6.07972	−5.15756	−1.34273	−0.00677	−6.28296	−0.71813	−6.12297	−2.94360
g_4	−8.47341	−17.97521	−6.21485	−17.85616	−5.65638	−8.05652	−6.33088	−10.64779
g_5	−7.03550	−17.70541	−4.80168	−16.40908	−4.34362	−6.63233	−4.89400	−8.09853
g_6	−0.71896	−0.13490	−0.70659	−0.72354	−0.65638	−0.71210	−0.71844	−1.27463
g_7	−0.00012	−0.04433	−0.01157	−0.00055	−0.08500	−0.00247	−0.00043	−0.05724
g_8	−3.38619	−1.44904	−1.63735	−3.42163	−3.35852	−3.14154	−3.37546	−2.13890
g_9	0	−1.17E−03	0	−4.07E−07	0	0	0	−5.83E−04
g_{10}	−0.00005	−0.02483	−0.03659	−0.00042	−0.04168	−0.06819	−0.00029	−0.01198
f	85,548.6168	73,345.1098	85,494.0330	85,521.8274	85,066.2631	85,475.6918	85,548.1078	76,418.3804

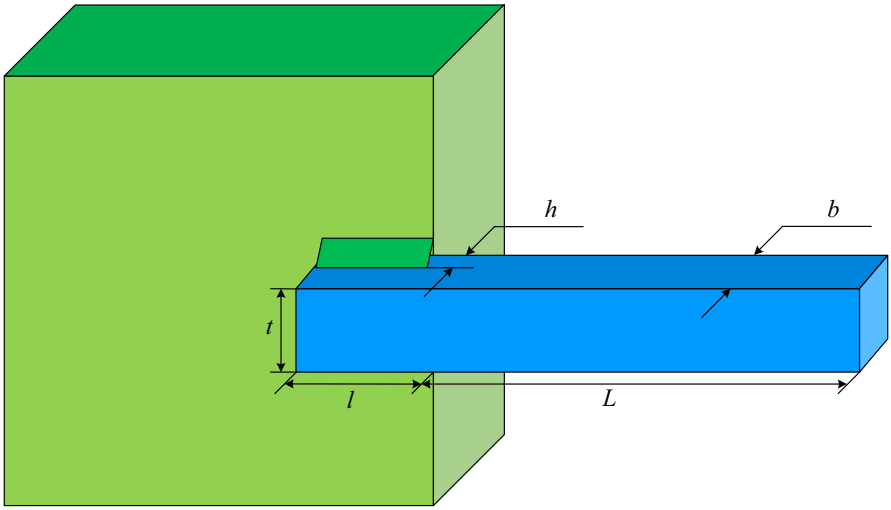

Fig. 24 Schematic diagram of the welded beam design problem

5.7 Rolling element bearing design problem

The goal of the rolling element bearing design problem is to maximize fatigue life [94]. Moreover, fatigue life is closely related to its dynamic load carrying capacity. The problem involves 10 decision variables and 9 constraints. The schematic diagram of the rolling element bearing design is shown in Fig. 23, and its mathematical expression is defined in Eq. (28).

Consider:

$$\mathbf{x} = \left[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\right] = \left[D_m, D_b, Z, f_i, f_0, K_{Dmin}, K_{Dmax}, \varepsilon, e, \zeta\right].$$

Maximum:

$$f(\mathbf{x}) = \begin{cases} f_c \times Z^{2/3} \times D_b^{1.8} & D_b \leq 25.4 \\ 3.647 \times f_c \times Z^{2/3} \times D_b^{1.4} & D_b > 25.4 \end{cases}. \tag{28}$$

Subject to:

Table 21 Best results of different algorithms for the welded beam design problem

	LEA	NOA	GJO	AOS	TSA	SOA	HHO	RDPSO
x_1	0.20570	0.20710	0.20557	0.20577	0.20037	0.20399	0.19132	0.20305
x_2	3.47153	3.73723	3.48002	3.47590	3.64637	3.51970	3.84924	3.42250
x_3	9.03661	9.77945	9.03991	9.03217	8.96104	9.03675	8.94836	9.46263
x_4	0.20573	0.22010	0.20581	0.20597	0.20982	0.20575	0.20981	0.20384
g_1	−1.39104	−1635.95	−23.1131	−14.1106	−76.6266	−35.1355	−0.67354	−162.537
g_2	−0.14017	−6056.72	−33.9125	−5.88963	−86.5844	−3.81050	−0.29998	−2386.44
g_3	−0.00003	−0.01300	−0.00024	−0.00021	−0.00945	−0.00176	−0.01849	−0.00079
g_4	−3.39057	−3.11585	−3.38868	−3.38908	−3.35941	−3.38687	−3.34734	−3.33771
g_5	−0.08070	−0.08210	−0.08057	−0.08077	−0.07537	−0.07899	−0.06632	−0.07805
g_6	−0.23554	−0.23934	−0.23556	−0.23554	−0.23546	−0.23554	−0.23540	−0.23729
g_7	−0.13959	−1727.13	−8.69591	−19.38058	−329.94	−1.85256	−323.08	−12.4242
f	1.724961	2.013841	1.727101	1.726728	1.757961	1.728961	1.767866	1.772630

$$
\begin{aligned}
g_1(x) &= -\frac{\varphi_0}{2\arcsin\left(D_b/D_m\right)} + Z - 1 \le 0,\\
g_2(x) &= -2D_b + K_{D\min}(D-d) \le 0,\\
g_3(x) &= -K_{D\max}(D-d) + 2D_b \le 0,\\
g_4(x) &= -D_m + (0.5-e)(D+d) \le 0,\\
g_5(x) &= D_m - (0.5+e)(D+d) \le 0,\\
g_6(x) &= -D_m + 0.5(D+d) \le 0,\\
g_7(x) &= -0.5\left(D - D_m - D_b\right) + \varepsilon D_b \le 0,\\
g_8(x) &= \zeta B_\omega - D_b \le 0,\\
g_9(x) &= 0.515 - f_i \le 0,\\
g_{10}(x) &= 0.515 - f_o \le 0.
\end{aligned}
$$

where

$$
\begin{aligned}
f_c &= 37.91\left\{1 + \left[1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right]^{10/3}\right\}^{-0.3}\\
&\quad \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{f_o(1+\gamma)^{\frac{1}{3}}}\right] \times \left[\frac{2f_i}{2f_i-1}\right]^{0.41},\\
\varphi_0 &= 2\pi - 2\arccos\frac{\left[(D-d)/2 - 3(T/4)\right]^2 + \left(D/2 - T/4 - D_b\right)^2 - (d/2 + T/4)^2}{2\left[(D-d)/2 - 3(T/4)\right]\left(D/2 - T/4 - D_b\right)},\\
T &= D - d - 2D_b,\ B_\omega = 30,\ D = 160,\ d = 90, r_i = r_0 = 11.033.
\end{aligned}
$$

Variable range:

$0.5(D+d) \le x_1 \le 0.6(D+d)$, $0.15(D-d) \le x_2 \le 0.45(D-d)$, $4 \le x_3 \le 50$, $0.515 \le x_4, x_5 \le 0.6$, $0.4 \le x_6 \le 0.5$, $0.6 \le x_7 \le 0.7$, $0.3 \le x_8 \le 0.4$, $0.02 \le x_9 \le 0.1, 0.6 \le x_{10} \le 0.85$.

Table 20 shows the best results obtained by different algorithms for optimizing this problem. It can be noted that the NOA and the RDPSO perform the worst on this problem. Moreover, although all algorithms except for the NOA and the RDPSO achieved good optimization results, LEA still came in first place, showing a strong competition.

5.8 Welded beam design problem

The welded beam design problem (Fig. 24) is an engineering problem that was proposed by Coello and solved by many researchers using different methods [95]. The problem is constrained by seven conditions from stress, deflection, welding, and geometry, and the objective is to find the minimum manufacturing cost of the welded beam. The decision variables are weld thickness (h), height (l), length (t), and crossbeam thickness (b). The objective function can be defined in Eq. (29).

Consider:

$$\mathbf{x} = \left[x_1, x_2, x_3, x_4\right] = [h, l, t, b].$$

Minimize:

$$f(\mathbf{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 \left(14 + x_2\right). \tag{29}$$

Subject to:

$$\begin{aligned}
&g_1(x) = \tau(x) - \tau_{\max} \le 0,\\
&g_2(x) = \sigma(x) - \sigma_{\max} \le 0,\\
&g_3(x) = \delta(x) - \delta_{\max} \le 0,\\
&g_4(x) = x_1 - x_4 \le 0,\\
&g_5(x) = P - P_c(x) \le 0,\\
&g_6(x) = 0.125 - x_1 \le 0,\\
&g_7(x) = 1.10471 x_1^2 + 0.04811 x_3 x_4 \left(14.0 + x_2\right) - 5.0 \le 0.
\end{aligned}$$

where

$$
\begin{aligned}
\tau(x) &= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\\
\tau' &= \frac{P}{\sqrt{2}x_1x_2},\\
\tau'' &= MR/J,\\
M &= P\left(L + x_2/2\right),\\
R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},\\
J &= 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},\\
\sigma(x) &= \frac{6PL}{x_4x_3^2},\\
\delta(x) &= \frac{6PL^3}{Ex_3^2x_4},\\
P_c(x) &= \frac{4.013Ex_3x_4^3/6}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
P &= 6000 \text{ lb},\\
L &= 14 \text{ in},\\
\delta_{\max} &= 0.25 \text{ in},\\
E &= 30 \times 10^6 \text{ psi},\\
G &= 12 \times 10^6 \text{ psi},\\
\tau_{\max} &= 13600 \text{ psi},\\
\sigma_{\max} &= 30000 \text{ psi}
\end{aligned}
$$

Variable range:
$0.1 \le x_1, x_4 \le 2, 0.1 \le x_2, x_3 \le 10$.

The best results of the welded beam design problem optimized with different algorithms are given in Table 21. From the experimental results, it is concluded that the LEA ranks 1st when the constraints are satisfied. The LEA performs much better than the NOA on this problem. In addition, the LEA shows a slight advantage over the TSA, the HHO, and the RDPSO. Meanwhile, a host of algorithms such as GJO, AOS, and SOA compared to it showed similar results to the LEA.

6 Conclusion

In this paper, an evolutionary algorithm inspired by the stimulus–value–role theory called Love Evolution Algorithm (LEA) was proposed. The proposed algorithm abstracts human characteristics (e.g., temperament, personality,

hobbies, etc.) into solutions and happiness in relationships into objective function values. In addition, the LEA includes three phases: the stimulus phase, the value phase, and the role phase. The goal of the LEA is to enhance the characteristics of both partners in a relationship through mutual learning and bonding between the two people in the relationship, thus enhancing the happiness of both partners.

This paper verified the optimization performance of the LEA using the CEC2017 and CEC2022 benchmark test sets. Seven recent and excellent metaheuristic algorithms were compared with the LEA on the CEC2017 benchmark functions. These algorithms include the NOA, the GJO, the AOS, the TSA, the SOA, the HHO, and the RDPSO. Then, Wilcoxon signed-rank test and Friedman test were performed for the optimization results of the LEA and the competitors on the CEC2017 benchmark functions. In addition, the LEA and seven state-of-the-art metaheuristic algorithms were used to optimize the CEC2022 benchmark functions to validate the strong competitiveness of the LEA through comparisons. These seven state-of-the-art metaheuristics are L-SHADE, AL-SHADE, L-SHADE-spacma, AFDB-ARO, FDB-AGDE, FDB-AGSK, and FDB-PPSO. Moreover, time complexity analysis, convergence analysis, and scalability analysis were performed. Afterward, the LEA was used to solve eight real-world optimization problems to verify the capability of the LEA to solve engineering problems. These eight real-world optimization problems include the speed reducer design problem, the pressure vessel design problem, the cantilever beam design problem, the I-beam design problem, the tubular column design problem, the piston lever design problem, the rolling element bearing design problem, and the welded beam design problem. The core results obtained from this study are summarized as follows:

(1) The time complexity of the LEA is not dominant compared to the competitors (CEC2017 benchmark set), but it is feasible.
(2) The LEA achieves similar results to the state-of-the-art algorithms on the CEC2022 benchmark functions, validating the stronger competitiveness of the LEA.
(3) The LEA is able to exhibit different optimization behaviors on the CEC2022 benchmark functions.
(4) The LEA converges significantly faster on F1, F3, F5~F10, F12, F18, F19, F22, and F30 (CEC2017 benchmark set) compared to the competitors.
(5) The p-values obtained from the Wilcoxon signed-rank test are mostly less than 0.05, indicating that there is a significant difference between the results of LEA and those of the competitors.
(6) On the Friedman test, the LEA ranks 2nd on F11 and F24, and 1st on the remaining functions; and the final rank of the LEA is 1st.
(7) The LEA shows strong scalability on most functions of the CEC2017 benchmark set.
(8) The LEA demonstrates good optimization capabilities for real-world optimization problems.

The proposed LEA provides some new search operations. From the results of this paper, it can be concluded that the LEA has a large research prospect. In the future, other valuable research based on this study includes:

(1) Provide a multi-objective optimization version of the LEA.
(2) Discuss the applications of the LEA to discrete-valued and binary optimization problems.
(3) Prove theoretically the convergence of the LEA.
(4) Propose improved LEA in terms of population initialization (e.g., chaotic mapping), selection of a guided individual (e.g., the fitness–distance balance), balance between exploration and exploitation (e.g., parameters or new search operations), and update mechanisms (e.g., the natural survivor method).
(5) Propose fusion algorithms with better performance by fusing LEA and other metaheuristics.
(6) Propose improved versions of the LEA for constrained engineering problems (e.g., the fitness–distance–constraint).
(7) Apply the LEA to different real-world optimization problems.

Acknowledgements This work was supported in part by the College Students' Innovative Entrepreneurial Training Plan Program (Project Number: X202310147035).

Author contributions YG was involved in conceptualization, methodology, software, writing—original draft, writing—reviewing and editing, and visualization. JZ was responsible for writing—original draft, data curation, writing—reviewing and editing, and visualization. YW contributed to validation, data curation, and writing—original draft. JW took part in validation, writing—original draft, and writing—reviewing and editing. LQ participated in validation, investigation, and writing—original draft.

Data availability No data were used for the research described in the paper.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Wu G (2016) Across neighborhood search for numerical optimization. Inf Sci 329:597–618
2. Wu G, Pedrycz W, Suganthan PN, Mallipeddi R (2015) A variable reduction strategy for evolutionary algorithms handling equality constraints. Appl Soft Comput 37:774–786
3. Wong WK, Ming CI (2019) A review on metaheuristic algorithms: recent trends, benchmarking and applications. In 2019 7th international conference on smart computing & communications (ICSCC). IEEE, pp 1–5
4. Malik A, Tikhamarine Y, Souag-Gamane D, Kisi O, Pham QB (2020) Support vector regression optimized by meta-heuristic algorithms for daily streamflow prediction. Stoch Env Res Risk Assess 34(11):1755–1773
5. Gao Y, Li C, Huang L (2022) An improved deep extreme learning machine to predict the remaining useful life of lithium-ion battery. Front Energy Res 10:1032660
6. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

7. Murstein BI (1970) Stimulus. Value. Role: a theory of marital choice. J Marriage Fam 465–481
8. Ma Z, Wu G, Suganthan PN, Song A, Luo Q (2023) Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms. Swarm Evol Comput 77:101248
9. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
10. Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549
11. Voudouris C, Tsang EP, Alsheddy A (2010) Guided local search. In: Handbook of metaheuristics. Springer, Boston, pp 321–361
12. Lourenço HR, Martin OC, Stützle T (2019) Iterated local search: framework and applications. Handbook of metaheuristics, pp 129–168
13. Rastrigin LA (1963) The convergence of the random search method in the extremal control of a many parameter system. Autom Remote Control 24:1337–1342
14. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
15. Pisinger D, Ropke S (2019) Large neighborhood search. Handbook of metaheuristics, pp 99–127
16. Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–73
17. Fogel DB (1998) Artificial intelligence through simulated evolution. Wiley-IEEE Press, pp 227–296
18. Ingo R (1973) Evolution strategy: optimization of technical systems by means of biological evolution, vol 104. Fromman-Holzboog, Stuttgart, p 15
19. Koza JR (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4:87–112
20. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
21. Ferreira, C. (2001). Gene expression programming: a new adaptive algorithm for solving problems. Preprint https://arxiv.org/abs/cs/0102027
22. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713
23. Civicioglu P (2012) Transforming geocentric Cartesian coordinates to geodetic coordinates by using differential search algorithm. Comput Geosci 46:229–247
24. Motevali MM, Shanghooshabad AM, Aram RZ, Keshavarz H (2019) WHO: a new evolutionary algorithm bio-inspired by wildebeests with a case study on bank customer segmentation. Int J Pattern Recognit Artif Intell 33(05):1959017
25. Veysari EF (2022) A new optimization algorithm inspired by the quest for the evolution of human society: human felicity algorithm. Expert Syst Appl 193:116468
26. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B (cybern) 26(1):29–41
27. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4. IEEE, pp 1942–1948
28. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52–67
29. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl-Based Syst 89:228–249
30. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
31. Dhiman G, Kumar V (2017) Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 114:48–70
32. Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23:715–734
33. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
34. Kaur S, Awasthi LK, Sangal AL, Dhiman G (2020) Tunicate swarm algorithm: a new bio-inspired based metaheuristic paradigm for global optimization. Eng Appl Artif Intell 90:103541
35. Abdollahzadeh B, Gharehchopogh FS, Mirjalili S (2021) African vultures optimization algorithm: a new nature-inspired metaheuristic algorithm for global optimization problems. Comput Ind Eng 158:107408
36. Hashim FA, Hussien AG (2022) Snake optimizer: a novel meta-heuristic optimization algorithm. Knowl-Based Syst 242:108320

37. Braik M, Hammouri A, Atwan J, Al-Betar MA, Awadallah MA (2022) White shark optimizer: a novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowl-Based Syst 243:108457
38. Agushaka JO, Ezugwu AE, Abualigah L (2022) Dwarf mongoose optimization algorithm. Comput Methods Appl Mech Eng 391:114570
39. Zervoudakis K, Tsafarakis S (2022) A global optimizer inspired from the survival strategies of flying foxes. Eng Comput 1–34
40. Shahrouzi M, Kaveh A (2022) An efficient derivative-free optimization algorithm inspired by avian life-saving manoeuvres. J Comput Sci 57:101483
41. Mohammed H, Rashid T (2023) FOX: a FOX-inspired optimization algorithm. Appl Intell 53(1):1030–1050
42. Han M, Du Z, Yuen K, Zhu H, Li Y, Yuan Q (2023) Walrus optimizer: a novel nature-inspired metaheuristic algorithm. Expert Syst Appl 122413
43. Tian AQ, Liu FF, Lv HX (2023) Snow geese algorithm: a novel migration-inspired meta-heuristic algorithm for constrained engineering optimization problems. Appl Math Model
44. Erol OK, Eksin I (2006) A new optimization method: big bang–big crunch. Adv Eng Softw 37(2):106–111
45. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
46. Lam AY, Li VO (2009) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14(3):381–399
47. Alatas B (2011) ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 38(10):13170–13180
48. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184
49. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
50. Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv Eng Softw 110:69–84
51. Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W (2021) Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 51:1531–1551
52. Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: a novel optimization algorithm. Knowl-Based Syst 191:105190
53. Rodriguez L, Castillo O, Garcia M, Soria J (2021) A new meta-heuristic optimization algorithm based on a paradigm from physics: string theory. J Intell Fuzzy Syst 41(1):1657–1675
54. Azizi M (2021) Atomic orbital search: a novel metaheuristic algorithm. Appl Math Model 93:657–683
55. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
56. Ahmadianfar I, Bozorg-Haddad O, Chu X (2020) Gradient-based optimizer: a new metaheuristic optimization algorithm. Inf Sci 540:131–159
57. Ahmadianfar I, Heidari AA, Gandomi AH, Chu X, Chen H (2021) RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. Expert Syst Appl 181:115079
58. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609
59. Ahmadianfar I, Heidari AA, Noshadian S, Chen H, Gandomi AH (2022) INFO: an efficient optimization algorithm based on weighted mean of vectors. Expert Syst Appl 195:116516
60. Houssein EH, Saad MR, Hashim FA, Shaban H, Hassaballah M (2020) Lévy flight distribution: a new metaheuristic algorithm for solving engineering optimization problems. Eng Appl Artif Intell 94:103731
61. Gao Y (2023) PID-based search algorithm: a novel metaheuristic algorithm based on PID algorithm. Expert Syst Appl 120886
62. Musa Z, Ibrahim Z, Shapiai MI, Tsuboi Y (2023) Cubature Kalman optimizer: a novel metaheuristic algorithm for solving numerical optimization problems. J Adv Res Appl Sci Eng Technol 33(1):333–355
63. Ozer AB (2010) CIDE: chaotically initialized differential evolution. Expert Syst Appl 37(6):4632–4641

64. Kazimipour B, Li X, Qin AK (2014) A review of population initialization techniques for evolutionary algorithms. In: 2014 IEEE congress on evolutionary computation (CEC). IEEE, pp 2585–2592
65. Kahraman HT, Aras S, Gedikli E (2020) Fitness-distance balance (FDB): a new selection method for meta-heuristic search algorithms. Knowl-Based Syst 190:105169
66. Ozkaya B, Kahraman HT, Duman S, Guvenc U (2023) Fitness-distance-constraint (FDC) based guide selection method for constrained optimization problems. Appl Soft Comput 110479
67. Viswanathan GM, Afanasyev V, Buldyrev SV, Murphy EJ, Prince PA, Stanley HE (1996) Lévy flight search patterns of wandering albatrosses. Nature 381(6581):413–415
68. Pearson K (1905) The problem of the random walk. Nature 72(1865):294–294
69. Liu W, Wang Z, Yuan Y, Zeng N, Hone K, Liu X (2019) A novel sigmoid-function-based adaptive weighted particle swarm optimizer. IEEE Trans Cybern 51(2):1085–1093
70. Pan JS, Lv JX, Yan LJ, Weng SW, Chu SC, Xue JK (2022) Golden eagle optimizer with double learning strategies for 3D path planning of UAV in power inspection. Math Comput Simul 193:509–532
71. Tanabe R, Fukunaga AS (2014) Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE congress on evolutionary computation (CEC). IEEE, pp 1658–1665
72. Kahraman HT, Katı M, Aras S, Taşci DA (2023) Development of the natural survivor method (NSM) for designing an updating mechanism in metaheuristic search algorithms. Eng Appl Artif Intell 122:106121
73. Wu G, Mallipeddi R, Suganthan PN (2017) Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report
74. Biedrzycki R, Arabas J, Warchulski E (2022) A version of NL-SHADE-RSP algorithm with midpoint for CEC 2022 single objective bound constrained problems. In: 2022 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
75. Abdel-Basset M, Mohamed R, Jameel M, Abouhawwash M (2023) Nutcracker optimizer: a novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. Knowl-Based Syst 262:110248
76. Chopra N, Ansari MM (2022) Golden jackal optimization: a novel nature-inspired optimizer for engineering applications. Expert Syst Appl 198:116924
77. Dhiman G, Kumar V (2019) Seagull optimization algorithm: theory and its applications for large-scale industrial engineering problems. Knowl-Based Syst 165:169–196
78. Sun J, Palade V, Wu XJ, Fang W, Wang Z (2013) Solving the power economic dispatch problem with generator constraints by random drift particle swarm optimization. IEEE Trans Industr Inf 10(1):222–232
79. Li Y, Han T, Zhou H, Tang S, Zhao H (2022) A novel adaptive L-SHADE algorithm and its application in UAV swarm resource configuration problem. Inf Sci 606:350–367
80. Mohamed AW, Hadi AA, Fattouh AM, Jambi KM (2017) LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In: 2017 IEEE congress on evolutionary computation (CEC). IEEE, pp 145–152
81. Bakır H (2023) Fitness-distance balance-based artificial rabbits optimization algorithm to solve optimal power flow problem. Expert Syst Appl 122460
82. Guvenc U, Duman S, Kahraman HT, Aras S, Katı M (2021) Fitness-distance balance based adaptive guided differential evolution algorithm for security-constrained optimal power flow problem incorporating renewable energy sources. Appl Soft Comput 108:107421
83. Bakır H, Duman S, Guvenc U, Kahraman HT (2023) Improved adaptive gaining-sharing knowledge algorithm with FDB-based guiding mechanism for optimization of optimal reactive power flow problem. Electr Eng 1–40
84. Duman S, Kahraman HT, Korkmaz B, Bakir H, Guvenc U, Yilmaz C (2021) Improved Phasor particle swarm optimization with fitness distance balance for optimal power flow problem of hybrid AC/DC power grids. In: The international conference on artificial intelligence and applied mathematics in engineering. Springer, Cham, pp 307–336
85. Morales-Castañeda B, Zaldivar D, Cuevas E, Fausto F, Rodríguez A (2020) A better balance in metaheuristic algorithms: Does it exist? Swarm Evol Comput 54:100671
86. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics 1:196–202

87. Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701
88. Sattar D, Salim R (2021) A smart metaheuristic algorithm for solving engineering problems. Eng Comput 37(3):2389–2417
89. Sandgren E (1990) NIDP in mechanical design optimization. J Mech Design 112(2):223–229
90. Chickermane HEMIANT, Gea HC (1996) Structural optimization using a new local approximation method. Int J Numer Meth Eng 39(5):829–846
91. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35
92. Bayzidi H, Talatahari S, Saraee M, Lamarche CP (2021) Social network search for solving engineering optimization problems. Comput Intell Neurosci 2021:1–32
93. Rao SS (2019) Engineering optimization: theory and practice. Wiley
94. Gupta S, Tiwari R, Nair SB (2007) Multi-objective design optimisation of rolling bearings using genetic algorithms. Mech Mach Theory 42(10):1418–1443
95. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127