Applied Intelligence
https://doi.org/10.1007/s10489-022-03999-y

A knowledge-driven monarch butterfly optimization algorithm with self-learning mechanism

Tianpeng Xu [1] · **Fuqing Zhao** [1] · **Jianxin Tang** [1] · **Songlin Du** [1] · **Jonrinaldi** [2]

Accepted: 12 July 2022

Abstract
The Monarch Butterfly Optimization (MBO) algorithm has been proved to be an efficient meta-heuristic to directly address continuous optimization problems. In the MBO algorithm, the migration operator cooperates with the butterfly adjusting operator to generate the entire offspring population. Since the individual iterations of the MBO algorithm are not self-learning, the cooperative intelligence mechanism is a random process. In this study, an improved MBO algorithm with a knowledge-driven learning mechanism (KDLMBO) is presented to enable the algorithm to evolve effectively with a self-learning capacity. The neighborhood information extracted from the candidate solutions is treated as the prior knowledge of the KDLMBO algorithm. The learning mechanism consists of the learning migration operator and the learning butterfly adjusting operator. Then, the self-learning collective intelligence is realized by the two cooperative operators in the iterative process of the algorithm. The experimental results demonstrate and validate the efficiency and significance of the proposed KDLMBO algorithm.

Keywords Continuous optimization problem · Monarch butterfly optimization · Knowledge-driven · Self-learning mechanism

1 Introduction

Various real-world applications are transformed into continuous optimization problems. These optimization problems become complex with the aggrandizing of the problem scales and are challenging to be addressed by the traditional optimization algorithms [1, 2]. The current research focuses on feasible and efficient optimization algorithms for addressing complicated continuous optimization problems [3, 4] Generally, a continuous optimization problem is defined as $min\ f(x)$, $x = [x_1, x_2, \ldots, x_D]$, where the objective is to determine the maximal or minimum value of $f(x)$ [5]. Various optimization algorithms and the disparate mechanisms of meta-heuristics have been presented and utilized for addressing complicated continuous optimization problems [6, 7]. The Swarm Intelligence (SI) optimization algorithms are feasible and efficient methods to address complicated continuous optimization problems. Over the past decades, the study of SI algorithms has become an arrestive research and practice area [8, 9]. Due to the collective and intelligent mechanisms, the SI provides satisfactory results for the high-dimensional complicated and variable continuous optimization problems [10–13].

In the past decades, the continuous development of SI algorithms has led to the emergence of a series of optimization algorithms. For instance, the Genetic Algorithm (GA) [14], Evolutionary Strategy (ES) [15], Evolutionary Programming (EP), and Genetic Programming (GP). Other SI algorithms imitate the social behavior of swarm animals in nature, such as Cuckoo Search (CS) Algorithm [16], Bat Algorithm (BA) [17], Ant Colony Optimization (ACO) Algorithm [18], Whale Optimization Algorithm (WOA) [19], Artificial Bee Colony (ABC) Optimization Algorithm [20], Particle Swarm Optimization (PSO) Algorithm [21–23]. In most SI algorithms, the learning mechanism is that the current individuals randomly switch the information of the neighborhood with the

✉ Fuqing Zhao
Fzhao2000@hotmail.com

Tianpeng Xu
xutp@lut.edu.cn

Jianxin Tang
tangjx@lut.edu.cn

Songlin Du
Dsonglin@outlook.com

Jonrinaldi
jonrinaldi@eng.unand.ac.id

1 School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China

2 Department of Industrial Engineering, Universitas Andalas, Padang 25163, Indonesia

Published online: 20 September 2022

other individuals. The strategy easily leads to the deterioration of the population in the iterative process. Therefore, the learning mechanism of the individuals is significant during the iteration of the algorithm [24–26].

The Monarch Butterfly Optimization (MBO) algorithm is one of the promising swarm-based meta-heuristic algorithms inspired by the migration behaviors of monarch butterfly individuals in the northern USA and southern Canada [27, 28]. Unlike other existing SI algorithms with only one population, the MBO algorithm has two fundamental operators belonging to two different subpopulations. The migration operator (MO) and the butterfly adjusting operator (BAO) perform separately in subpopulations 1 and 2, respectively, but cooperate synergistically during the evolutionary process. The offspring individuals located in different subpopulations exchange the neighborhood information with other individuals to achieve individual migration, leading to a high-level cooperative learning mechanism in the MBO algorithm. The MO and BAO in the MBO algorithm cooperatively control the local search and global search operations, which leads to the equilibration of exploration and exploitation. However, the unique crossover of MO and BAO restricts the MBO algorithm to high-dimensional complicated and variable continuous optimization problems. Therefore, various improvements have been proposed in the last few years to improve the effectiveness of the MBO algorithm by enhancing the accuracy and increasing the convergence speed [13, 28, 29].

The information of the individual fitness was ignored in the original MBO algorithm. Therefore, a novel improvement of the MBO algorithm with a greedy mechanism and a self-adapting crossover operator (GCMBO) was proposed in [30]. In the GCMBO algorithm, the self-adapting crossover operator improved the diversity of the population during the late exploitation. The greedy mechanism was presented in the BAO to accelerate the convergence of the GCMBO algorithm. However, the performance of the GCMBO algorithm for the high-dimensional complicated continuous optimization problems was not satisfactory [31, 32]. Opposition-based learning (OBL) and random local perturbation (RLP) were introduced in the OPMBO algorithm [32]. The OBL combined the optimization algorithms with the learning mechanism at initialization. However, the performance of the learning mechanism in the OPMBO algorithm was not outstanding for high-dimensional complicated optimization problems.

The combination of the MBO algorithm with other algorithms has attracted considerable attention. Yazdani proposed the LMBO-DE that combined Differential Evolution (DE) and the MBO [33]. In the LMBO-DE, the mutation operator of DE was integrated into the MBO to enhance the exploration, and the offspring individuals obtained the neighborhood information from the differential item. However, the linearized migration operator was controlled by a random migration parameter *Mn*. The inseparability of the variables was easily destroyed making the LMBO-DE algorithm fail to outperform in non-separable problems. An algorithm combined with DE and a local search mechanism based on the MBO (DE-LSMBO) was proposed by Cui, Chen, and Yin [34]. The local search strategy improved the searching capability in the late iterations. The mechanism enhanced the iterative learning of offspring individuals. Then, the DE was incorporated to equilibrate the exploration and exploitation. Ghanem and Jantan combined the well-known ABC optimization with the MBO algorithm and proposed the Hybrid ABC/MBO [35]. The algorithm was proposed to equilibrate the exploration and exploitation by increasing the diversity of the original ABC algorithm utilizing the modified operator of the MBO algorithm. Besides, the original and the enhanced MBO algorithms have also been widely utilized for various real applications [36–38].

Numerous studies have shown that the learning mechanism is a promising way to enhance collective intelligence during the iteration of the algorithm [39, 40]. To the best of the authors' knowledge, there are few reports on improving the learning mechanism of the MBO algorithm by utilizing self-learning and self-adapting procedures. In this study, an algorithm based on the MBO algorithm with a knowledge-driven learning mechanism (KDLMBO) is provided to strengthen the self-learning and self-adapting capabilities of the algorithm. In the proposed KDLMBO, the neighborhood information obtained by the offspring individuals is the prior knowledge. The iterative learning process of the offspring individuals is instructed by the prior knowledge during the search. The proposed learning mechanism of the KDLMBO algorithm is aimed at increasing the self-learning, self-adapting, and self-organizing of the MBO algorithm. The characteristics of the KDLMBO algorithm are summarized as follows:

1. A knowledge-driven learning mechanism is introduced in the KDLMBO algorithm to strengthen the self-learning and self-adapting capabilities of the MBO algorithm during the iterations.
2. A knowledge-driven dynamic decision-making mechanism is introduced into the iterative process of the KDLMBO algorithm.
3. Three basic operations The proposed KDLMBO algorithm provides a promising framework for improving the learning mechanism of the algorithms.of WWO are redesigned to solve the considered problem.

The remainder of this paper is organized as follows. The original MBO algorithm is described in Section 2. The proposed KDLMBO algorithm is presented in Section 3. The experiments and performance analysis are provided in Section 4. The conclusion and future research are presented in Section 5.

2 Monarch butterfly optimization algorithm

2.1 Migration behavior of monarch butterfly

The MBO is an efficient meta-heuristic algorithm inspired by the migration behavior of monarch butterflies in southern Canada and the northern USA, from where the special butterfly population migrates to Mexico and California in the late summer/autumn every year by flying thousands of miles [27]. In the MBO algorithm, an individual monarch butterfly represents a candidate solution during the iteration of the algorithm. At the beginning of the evolutionary process, the candidate solutions are sorted by the fitness values. Then the population is divided into two parts: Subpopulation 1 in the northern USA and Subpopulation 2 in Mexico. The MO and BAO generate the offspring individuals in Subpopulations 1 and 2. The numbers of monarch butterflies in Subpopulations 1 and 2 are NP_1 and NP_2, respectively, as given by Eqs. (1) and (2).

$$NP_1 = \lceil p*N \rceil \tag{1}$$

$$NP_2 = N - NP_1 \tag{2}$$

where N is the scale of the population and $p = 5/12$ is the migration ratio of Subpopulation 1 [27].

2.2 Migration operato

In Subpopulation 1, the candidate solutions are generated according to Eqs. (3) and (4) as:

$$x_{i,k}^{t+1} = \begin{cases} x_{r1,k}^{t}, \text{if } r \le p \\ x_{r2,k}^{t}, \text{elsewise} \end{cases}, k \in \{1, 2, \ldots, D\} \tag{3}$$

$$r = rand*peri \tag{4}$$

where t is the current generation of the algorithm, $x_{i,k}^{t+1}$ indicates the kth element of the ith solution at iteration $t + 1$, $x_{r1,k}^{t}$ is the kth element of the r1th solution at t, $x_{r2,k}^{t}$ is the kth element of the r2th solution at t, r is a random number based on Eq. (4), and peri is set to 1 [27]. r1 and r2 are random numbers representing random individuals in Subpopulations 1 and 2, respectively.

2.3 Butterfly adjusting operator

In Subpopulation 2, the candidate solutions are generated as:

$$x_{j,k}^{t+1} = \begin{cases} x_{best,k}^{t}, \text{if } rand \le p \\ x_{r3,k}^{t}, \text{ elsewise} \end{cases}, k \in \{1, 2, \ldots, D\} \tag{5}$$

where $x_{j,k}^{t+1}$ is the kth element of the jth solution at $t + 1$, $x_{best,k}^{t}$ is the kth element of the best solution of the entire population at iteration t in Subpopulation 1 and Subpopulation 2, $x_{r3,k}^{t}$ is the kth element of the $r3$th solution at t and $r3$ is a random individual drawn from Subpopulation 2. If $rand > p$ & $rand > 5/12$, the candidate solutions are further improved according to Eqs. (6, 7 and 8) as:

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha \times (d_{x_k} - 0.5), k \in \{1, 2, \ldots, D\} \tag{6}$$

$$\alpha = S_{max}/t^2 \tag{7}$$

$$dx = Levy\left(x_j^t\right) \tag{8}$$

where the kth element of the jth solution at iteration $t + 1$ is updated based on Eq. (6), α is the weighting factor based on Eq. (7)and $S_{max}$is the maximum step size, which is a key parameter for equilibrating exploration and exploitation. At the beginning of the iteration, S_{max} is set larger than that in the late iteration. In Eq. (8), dx is the step size from the Levy flight [41].

2.4 The mechanism of the MBO algorithm

In the MBO algorithm, firstly, the candidate solutions are randomly produced at the beginning of the iterations. Secondly, the fitness values of the candidate solutions are calculated, and the solutions are sorted based on their fitness values. Thirdly, the population is divided into two subpopulations (Subpopulation 1 and Subpopulation 2). The offspring candidate solutions located in different subpopulations exchange the neighborhood information with other individuals to achieve individual migration. After each new subpopulation is generated, the two newly-generated subpopulations are combined into the offspring population. The two cooperative operators perform their duties in different subpopulations to achieve the collective evolutionary process. Finally, the best solution is outputted when the maximum number of iterations is reached. The framework of the original MBO is shown in Algorithm 1.

Algorithm 1 The framework of MBO

```
Begin
  Initialize the value of t, D, N, MaxGen
  Calculate the fitness of the candidate solutions.
  t = 1
  while (t<MaxGen)
    Sort the population
    Divide the population into Subpopulations 1 and 2
     Generate candidate solutions in Subpopulation 1 based
     on the Migration operator
     Generate candidate solutions in Subpopulation 2 based
     on the Butterfly adjusting operator
    Combine the two newly-generated subpopulations
    Evaluate the new candidate solutions
  t = t + 1
  end while
  Output the best solution
End
```

3 Knowledge-driven learning monarch butterfly optimization

The crossover and mutation operators used in this study drawn on the idea from the variants of DE, $x_i^t + F_i*\left(x_{pbest}^t - x_i^t\right) + F_i*\left(x_{r1}^t - x_{r2}^t\right)$ is from "DE/current-to-best/1" and $x_i^t + F_i*\left(x_{r3}^t - x_i^t\right) + F_i*\left(x_{r4}^t - x_{r5}^t\right)$ is from "DE/rand/2". In particular, "DE/rand/2" is excellent at maintaining population diversity, while "DE/current-to-best/1" is conducted to speed up population convergence. Two mutation operators are rationally combined with the learning mechanism in the whole evolution process. It can be concluded from the literature that the other operators would generate poor diversity solutions in the searching space [42]. Compared with other operators, the combination of these two operators can effectively balance the exploration and exploitation, this can be verified from the experiment.

3.1 Initialization

In the proposed KDLMBO algorithm, the initial solutions of the problem are randomly scattered in a D dimensional problem space, and each candidate represents an initial solution to the problem. Then the initial population is sorted by the fitness of each candidate. Next, the population is divided into two parts Subpopulation 1 and Subpopulation 2 as the original MBO.

3.2 Mutation

3.2.1 Historical archive

In this study, the historical archive is provided in the learning migration operator to maintain the diversity of the population. The maximum size of the historical archive population A is N*2. In the first generation, the historical archive population A is the initial population. Then, the historical archive population A is generated as:

$$A^{t+1} = A^t \cup P_{worse}^t \tag{9}$$

where A^{t+1} is the combination of the A^t and the P_{worse}^t. The P_{worse}^t is the set of individuals that are worse than the parents in the t generation. Whenever the historical archive scale exceeds $|A|$, elements are randomly selected and deleted to store the newly inserted individuals.

3.2.2 Learning migration operator

In the KDLMBO algorithm, the candidate solutions in Subpopulation 1 are generated by the learning migration operator defined as:

$$x_i^{t+1} = x_i^t + \sigma_i^t,\ i \in \{1, 2, \ldots, NP_1\} \tag{10}$$

$$\sigma_i^1 = \begin{cases} F_i*\left(x_{pbest}^1 - x_i^1 + x_{r1}^1 - x_{r2}^1\right), & if\ r \le 0.5 \\ F_i*\left(x_{r3}^1 - x_i^1 + x_{r4}^1 - x_{r5}^1\right) & , elsewise \end{cases} \tag{11}$$

$$\sigma_i^t = \begin{cases} F_i*\left(x_{pbest}^t - x_i^t + x_{r1}^t - x_{r2}^t\right), & if\ r \le LR_{MO} \\ F_i*\left(x_{r3}^t - x_i^t + x_{r4}^t - x_{r5}^t\right) & elsewise \end{cases} \tag{12}$$

where x_i^{t+1} and x_i^t are the ith solutions in generations $t + 1$ and t, respectively. The individual x_{pbest}^t is randomly selected from the top $N \times rand$ members in generation t. N is the number of individuals in the population. F_i is the F parameter used by individual x_i. The parameter $F \in [0, 1]$ is the mutation operator that controls the magnitude of the mutation. The individual x_{r1}^t is randomly selected from the population. The individual x_{r2}^t is randomly selected from Subpopulation 2 and x_{r3}^t is randomly selected from Subpopulation 1. The individuals x_{r4}^t and x_{r5}^t are randomly selected from Subpopulation 2. LR_{MO} is the learning rate. Eq. (11) is utilized in the first generation, while Eq. (12) is used in the following generations. The main process of the migration operator is shown in Algorithm 2.

Algorithm 2 Main process of LMO

```
Begin
  for  i = 1 to NP_1
    if  t = 1
        Generate the candidate solution based on Eq. (11)
    Else
        Generate the candidate solution based on Eq. (12)
    end if
  end for
End
```

3.2.3 Learning butterfly adjusting operator

In the KDLMBO, the candidate solutions in Subpopulation 2 are generated by the learning butterfly adjusting operator defined as:

$$x_j^{t+1} = x_j^t + \sigma_j^t,\ j \in \{NP_1 + 1, NP_1 + 2, \ldots, NP_1 + NP_2\} \tag{13}$$

$$\sigma_j^1 = \begin{cases} F_j*\left(x_{best}^1 - x_j^1 + x_{r6}^1 - x_{worst}^1\right), & if\ r \le 0.5 \\ F_j*\left(x_{best}^1 - x_j^1 + x_{r7}^1 - x_{r8}^1\right) & , elsewise \end{cases} \tag{14}$$

$$\sigma_j^t = \begin{cases} F_j*\left(x_{best}^t - x_j^t + x_{r6}^t - x_{worst}^t\right), & if\ r \le LR_{BAO} \\ F_j*\left(x_{best}^t - x_j^t + x_{r7}^t - x_{r8}^t\right) & elsewise \end{cases} \tag{15}$$

where x_j^{t+1} and x_j^t are the jth solutions in generations $t + 1$ and t, respectively. The individual x_{best}^t is the best individual of the population in generation t. F_j is the F parameter used by individual x_j. The parameter $F \in [0, 1]$ is the mutation operator that controls the magnitude of mutation. The individual x_{r6}^t is randomly selected from Subpopulation 1. x_{worst}^t is the worst individual of the population in generation t. The individuals x_{r7}^{t} and x_{r8}^{t} are randomly selected from the population and LR_{BAO} is the learning rate. The primary process of the butterfly adjusting operator is shown in Algorithm 3.

Algorithm 3 Primary process of LBAO

```
Begin
  for j = 1 to NP_2
    if t = 1
      Generate the candidate solution based on Eq. (14)
    Else
      Generate the candidate solution based on Eq. (15)
    end if
  end for
End
```

3.3 Crossover and parameter adaptation

In the KDLMBO, the operators LMO and LBAO generate the candidate solutions at the mutation process. Then, the two Subpopulations are combined into an offspring candidate population. The widely used crossover operator in DE [42–44] is considered in the next iteration of the KDLMBO, which is defined by Eq. (16) as:

$$x_{i,k}^{t+1} = \begin{cases} x_{i,k}^{t+1}, if\ rand \le CR\ or\ k = k_{rand} \\ x_{i,k}^{t}\ , \qquad elsewise \end{cases} \tag{16}$$

$i \in \{1, 2, \ldots, N\}$

where $x_{i,k}^{t+1}$ and $x_{i,k}^{t}$ are the kth elements of the ith solutions in generations $t+1$ and t, respectively, k_{rand} is an index number randomly selected from $[1, \quad D]$ and CR is the crossover rate.

In this study, the mutation operator F and the crossover rate CR are generated according to Eqs. (17) and (18), respectively, as:

$$CR_i = randn_i(\mu_{CR}, 0.1) \tag{17}$$

$$F_i = randc_i(\mu_F, 0.1) \tag{18}$$

where $randn_i(\mu, \delta^2)$ is randomly selected from normal distributions with mean μ and variance δ^2, $randc_i(\mu, \delta^2)$ is randomly selected from Cauchy distributions with mean μ and variance δ^2, $CR_i \in [0, \quad 1]$ and $F_i \in [0, \quad 1]$. At the first iteration, both μ_{CR} and μ_F are set to 0.5 and generated according to Eqs. (19) and (20), respectively, as:

$$\mu_{CR} = 0.9 * \mu_{CR} + 0.1 * mean_{WA}(S_{CR}) \tag{19}$$

$$\mu_F = 0.9 * \mu_F + 0.1 * mean_L(S_F) \tag{20}$$

where S_{CR} is the combination of the successful CR_i in which the offspring individual is preeminent andS_F is the combination of the successful F_i in which the offspring individual is superior while $mean_L(*)$ and $mean_A(*)$ are Lehmer mean and weighted mean, respectively, defined as in Eqs. (21, 22 and 23).

$$mean_{WA}(S_{CR}) = \sum_{i=1}^{|S_{CR}|} w_i * S_{CR,i} \tag{21}$$

$$w_i = \frac{\left| f\left(x_i^{t+1}\right) - f\left(x_i^{t}\right) \right|}{\sum_{i=1}^{|S_{CR}|} \left| f\left(x_i^{t+1}\right) - f\left(x_i^{t}\right) \right|} \tag{22}$$

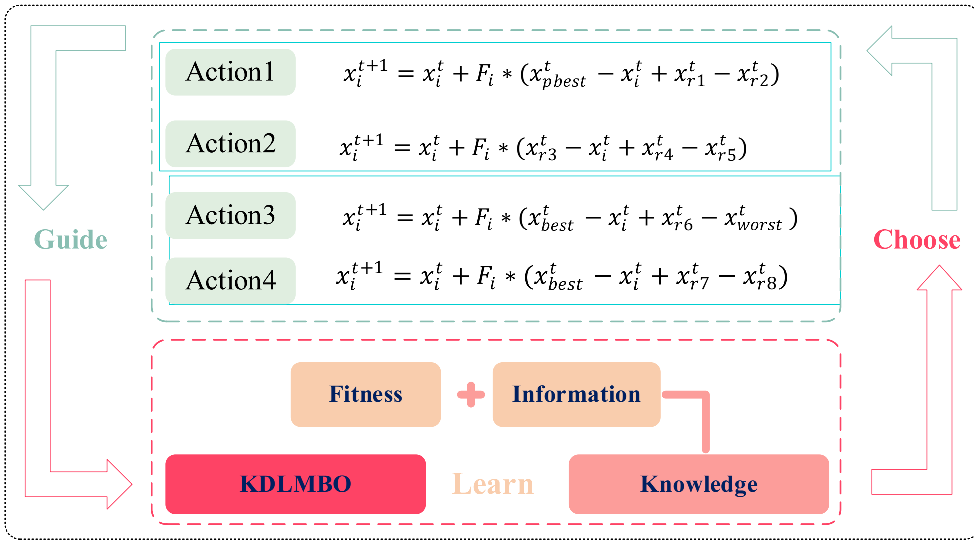
$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{23}$$

3.4 Learning mechanism

To strengthen the learning capacity of the algorithm on continuous optimization problems, the learning mechanism is provided in the KDLMBO algorithm. At iteration t, the prior knowledge of the algorithm involves the fitness of the functional problems and the information obtained from the neighborhood of the candidate solutions. The offspring individuals based on the fitness learn the neighborhood information in different phases of the iteration by choosing different actions. The neighborhood information is represented as four kinds of actions. Action 1 is efficient in the LMO compared with Action 2 in the early iteration of the algorithm to improve the convergence rate. Furthermore, in the late iterations, Action 2 makes Subpopulation 1 avoid the local optimal. Action 3 is useful in the LBAO compared with Action 4 in the early iteration to approach the best solution and escape the worst solution. In the late iterations, Action 4 also makes it difficult for Subpopulation 2 to trap into the local optimal. The four actions worked in different stages of the iteration to message the offspring. Collective intelligence is realized by the aforementioned mechanisms. Thus, there is no clear distinction between exploration and exploitation in the proposed algorithm. Therefore, exploration and exploitation of the algorithm are equilibrated by the cooperation of the LMO and LBAO. In general, the knowledge-driven learning mechanisms effectively increase the self-learning and self-adapting abilities of the algorithm. The strategy of collective intelligence is represented by the self-organizing learning mechanism. At the beginning of the generation, the prior knowledge of the algorithm is null. Therefore, the action is randomly selected according to Eqs. (11) and (14). The process of the learning mechanism is shown in Fig. 1.

As shown in Fig. 1, Actions 1 and 2 are the new mutation operators proposed in the LMO. The original actions in MO restrict the performance of the MBO algorithm. Action 1 is represented to accelerate the convergence of the improved algorithm. Action 2 is aimed at increasing the diversity of Subpopulation 1. Actions 3 and 4 are the novel mutation actions given in the LBAO. The original BAO destroys the inseparability of variables, making the MBO unsuited for inseparable problems. Therefore, Actions 3 and 4 are provided

Fig. 1 The general view of the learning mechanism



to avoid the inseparability of variables being destroyed. The cooperative learning mechanism equilibrates the exploration and exploitation of the proposed KDLMBO.

According to the roles of the four actions played in the proposed KDLMBO, a novel parameter named *Score*, defined as a real number, is introduced to quantitate the prior knowledge of each action. Specifically, if the fitness value of the current iteration is better than the previous iteration, the score of the current action is calculated according to Eq. (24) to indicate that the current action achieves one more acceptable evolution, otherwise, the score value remains the same. At the beginning, the initial score of each action is set to 0. Based on the score values obtained at the current iteration, the learning rates LR_{MO} and LR_{BAO} are redefined separately. When the prior knowledge of tth iteration is successfully learned, the subsequent action of $t+1$ iteration is guided by the knowledge at t iteration, and the learning rates LR_{MO} and LR_{BAO} are defined and updated according to Eqs. (25) and (26), respectively. The primary procedure of this condition is shown in Fig. 2.

$$\text{if } fitness^t < fitness^{t-1}$$

$$Score^t_{action\ n} = Score^{t-1}_{action\ n} + 1 \tag{24}$$

$$\text{if } Score^t_{action\ 1} \text{ and } Score^t_{action\ 2} \neq 0$$

$$LR_{MO}{}^{t+1} = \frac{Score^t_{action\ 1}}{Score^t_{action\ 1} + Score^t_{action\ 2}} \tag{25}$$

$$\text{if } Score^t_{action\ 3} \text{ and } Score^t_{action\ 4} \neq 0$$

$$LR_{BAO}{}^{t+1} = \frac{Score^t_{action\ 3}}{Score^t_{action\ 3} + Score^t_{action\ 4}} \tag{26}$$

When the fitness value at the tth iteration fails to be updated, meaning that all the offsprings are worse than the parents, then the action score is set to 0. Therefore, the score values from 1st iteration to $t-1$th iteration are considered, meaning that the prior knowledge of tth iteration comes from historical knowledge, and the action in tth iteration is guided by historical knowledge. Then the learning rates LR_{MO} and LR_{BAO} are generated by Eqs. (27) and (28), respectively. The learning mechanism in this second condition is shown in Fig. 3.

$$\text{if } Score^t_{action\ 1} \text{ or } Score^t_{action\ 2} = 0$$

$$LR_{MO}{}^{t+1} = \frac{\sum_{i=1}^t Score^i_{action\ 1}}{\sum_{i=1}^t Score^i_{action\ 1} + \sum_{i=1}^t Score^i_{action\ 2}} \tag{27}$$

$$\text{if } Score^t_{action\ 3} \text{ or } Score^t_{action\ 4} = 0$$

$$LR_{BAO}{}^{t+1} = \frac{\sum_{i=1}^t Score^i_{action\ 3}}{\sum_{i=1}^t Score^i_{action\ 3} + \sum_{i=1}^t Score^i_{action\ 4}} \tag{28}$$

3.5 Learning mechanism

The framework of the proposed KDLMBO is shown in Algorithm 4. Firstly, similar to the original MBO, the candidate solutions in the KDLMBO are randomly produced at the beginning of the iterations. Then, the fitness values of the candidate solutions are calculated, and the solutions are sorted based on their fitness values. After that, the entire population is divided into Subpopulations 1 and 2. Subpopulations 1 and 2 are evolved by Algorithms 2 and 3, respectively. After each new subpopulation is generated, the two newly-generated subpopulations are combined into the offspring population. The two cooperative operators perform their duties in different subpopulations to

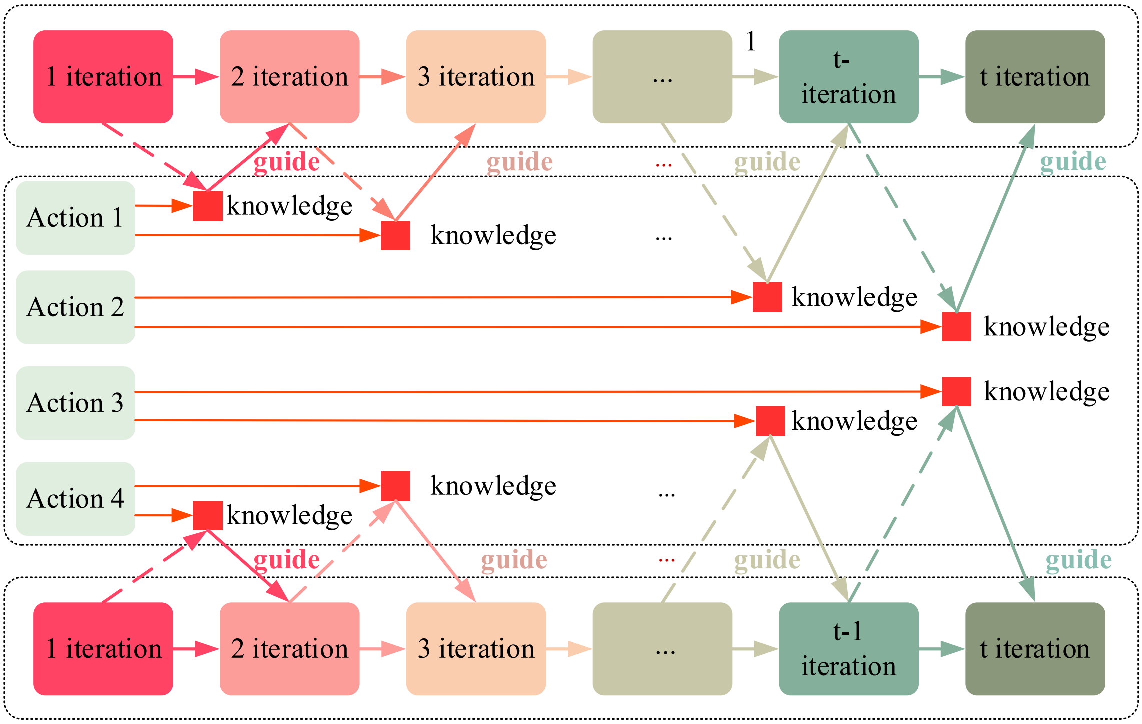


Fig. 2 The process of the learning mechanism in the first condition

achieve the collective evolutionary process. These solutions are operated by crossover after the combination. Finally, the best solution is outputted when the number of maximum iterations is reached. Furthermore, the flowchart of KDLMBO is shown in Fig. 4, we can clearly learn that how the LMO and LBAO work cooperatively.

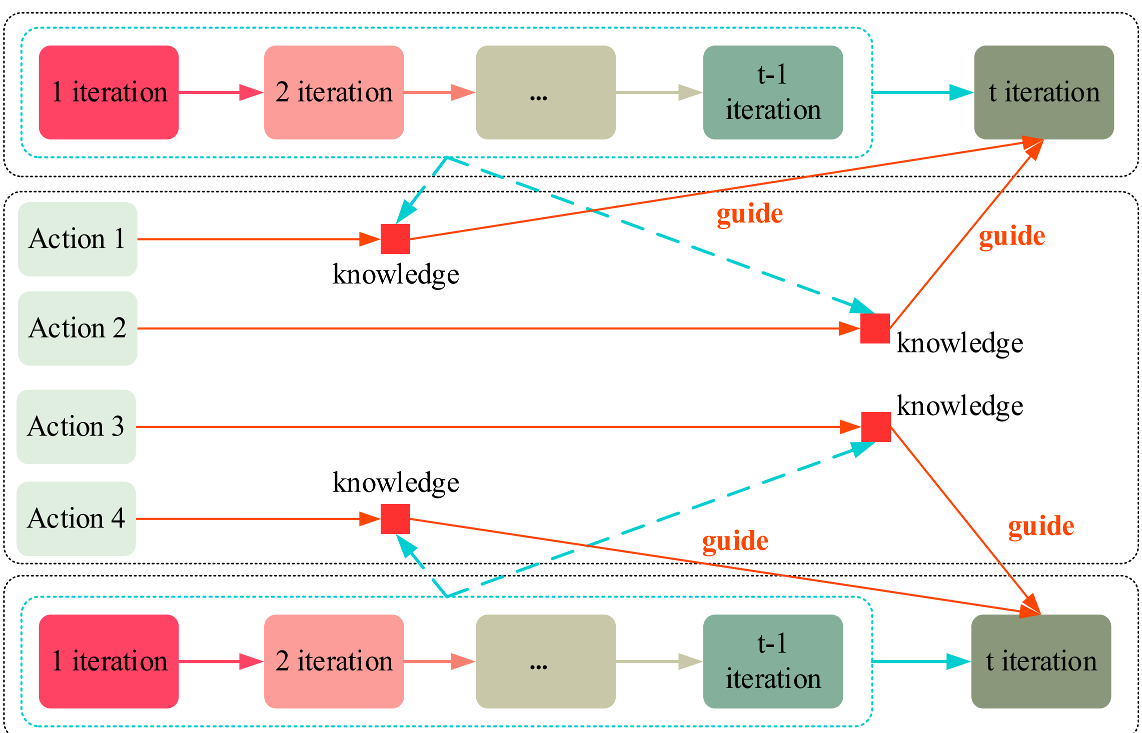


Fig. 3 The framework of the learning mechanism in the second condition

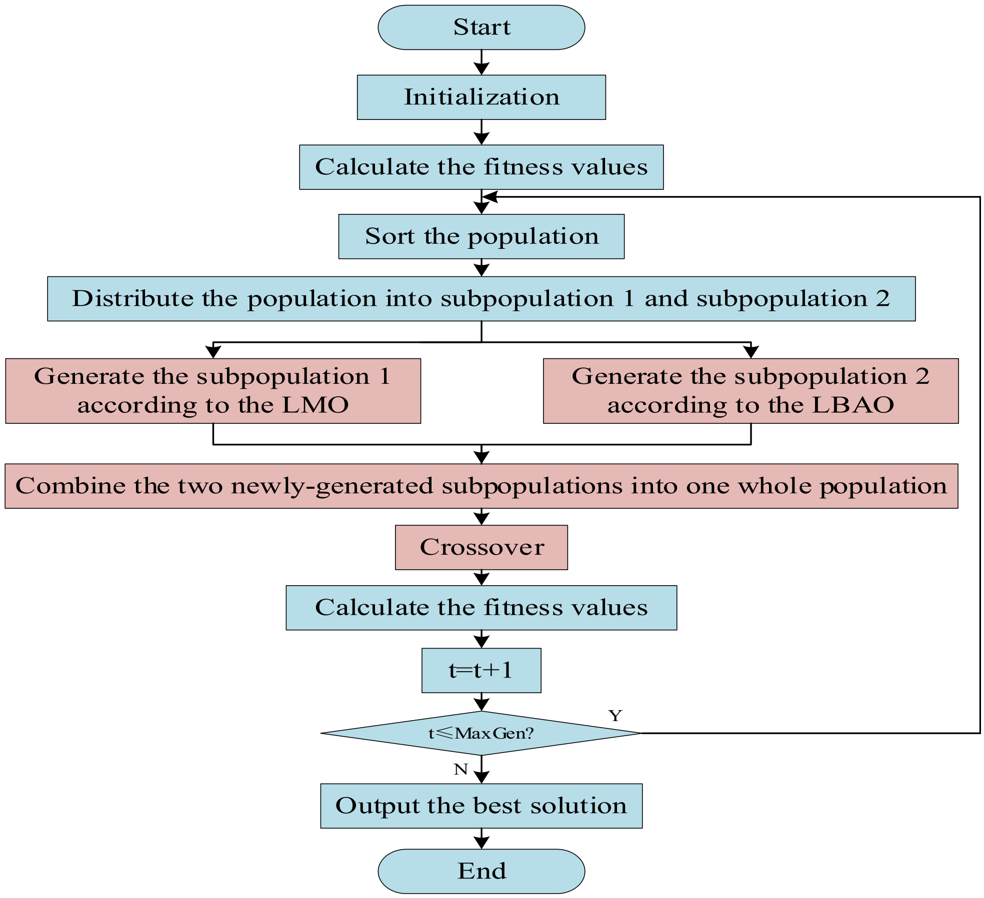


Fig. 4 Flowchart of the proposed KDLMBO algorithm

Algorithm 4 The framework of the proposed KDLMBO

Begin
  Initialize the values of $t,\ D,\ N,\ MaxGen, F,\ CR$
  Calculate the fitness of each candidate solution.
  $t = 1$
  while ($t < MaxGen$)
    Sort the population
    Divide the population into Subpopulation 1 and Subpopulation 2
    Generate candidate solutions in Subpopulation 1 based on **Algorithm 2**
    Generate candidate solutions in Subpopulation 1 based on **Algorithm 3**
    Combine the two newly-generated subpopulations
    Crossover
    Evaluate the new candidate solutions
    $t = t + 1$
  end while
  Output the best solution
End

3.6 The computational complexity of KDLMBO

This section analyzes the computational complexity of the proposed KDLMBO algorithm. N is the number of individuals in the population, while $NP1$ and $NP2$ represent the numbers of individuals in Subpopulations 1 and 2, respectively. In the KDLMBO, the complexity of fitness evaluation is $O(N)$ and the complexity of sorting the population is $O(Nlog(N))$.With respect to the generation of candidate solutions, the complexities in. Subpopulations 1 and 2 are $O(NP1)$ and $O(NP2)$, respectively. Lastly, the complexity of crossover is roughly $O(N)$. As a result, under the condition of $MaxGen$ iterations, the overall complexity of the KDLMBO algorithm is $O(MaxGen * Nlog(N))$.

4 Experimental results and analysis

4.1 Initialization

To validate the performance of the proposed KDLMBO algorithm, it was compared with the standard MBO [26], the variant of MBO GCMBO [28] and some state-of-the-art algorithms including BBO [45], IWO [46], Jaya [47], CMA-ES [48], LMBO-DE [33], and the reinforcement learning brain storm optimization algorithm (RLBSO) [49] on the 29 test functions of the CEC-2017 benchmark.

The 29 test functions of the CEC-2017 benchmark involve unimodal functions $f_1 - f_3$, multimodal functions $f_4 - f_{10}$, hybrid functions $f_{11} - f_{20}$ and composition functions $f_{21} - f_{30}$. The maximum fitness of the function evaluations was set to D×10,000. The baseline algorithms were carefully reimplemented in MATLAB and run on a PC with a 2.60 GHz Intel(R)-Core i7-9750H CPU, 16 GB of RAM and 64-bit OS. All algorithms were run independently 51 times to ensure the reliability of the experimental results.

4.2 Parameters analysis

In this study, the parameters of the mutation are self-learning. In the crossover, the parameters are self-adapting at the iteration except the beginning of the iteration. Therefore, the number of individuals in the population N, the mutation operator F, and the crossover operator CR were selected to analyze the effect of the parameters in the KDLMBO. There were three levels of the different combinations of the parameters including large, medium and small scales as shown in Table 1. Thus, the orthogonal design of the parameters was necessary. The experimental results depended on the three factors, and each factor had three levels. Hence, there were 9 test combinations. The orthogonal array of the parameters is listed in Table 2. The experimental results at 10D and 30D of the 9 test combinations are shown in Table 2. All the results of the combinations were followed by the Monte Carlo rule. The AVE and TAVE show the performance of each combination.

Table 3 lists the importance of each parameter calculated according to the TAVE in Table 2. The main effect diagram of parameters is shown in Fig. 5. F was the most significant parameter among the three parameters. The second was CR. The two parameters controlled the range of variation of the elements for the candidate solutions. The diversity of the population was ensured by the selection mechanism of the number of candidates. According to Table 3 and Fig. 5, the selected parameters of the KDLMBO algorithm were set as followed. N=16×D, F=0.3, CR=0.5.

4.3 Effect analysis of LMO and LBAO

The two main operators of the proposed KDLMBO are first analyzed for convergence performance on four types of functions from CEC-2017. These functions include the unimodal function f_1, the multimodal function f_4, the hybrid function f_{14} and the composition function f_{26}. It can be seen from Figs. 6, 7, 8 and 9 that the LMO improves the convergence speed of the algorithm and the LBAO enhances the precision of the solution. In addition, the LBAO can make Subpopulation 2 jump out of the local regions in the later iterations, which can ensure that the algorithm could find the best in the later iterations. Meanwhile, the experimental results from the proposed KDLMBO with LMO and LBAO algorithms in D = 10, 30, 50, and 100 dimensions are analyzed. The experimental results demonstrate that the two proposed operators cooperated and played complementary roles in improving the performance of the algorithm.

Table 1 The combinations of parameters

Levels	N	F	CR
1	8×D	0.3	0.3
2	12×D	0.5	0.5
3	16×D	0.2	0.2

Table 2 The orthogonal array of parameters

NO.	Parameters Levls			AVE		TAVE
	N	F	CR	10 D	30 D	
1	1	1	1	1.03E+02	4.77E+02	2.90E+02
2	1	2	2	1.16E+02	5.65E+02	3.40E+02
3	1	3	3	1.16E+02	1.56E+03	8.36E+02
4	2	1	2	1.06E+02	6.62E+02	3.84E+02
5	2	2	3	1.08E+02	1.21E+03	6.61E+02
6	2	3	1	1.11E+02	8.93E+02	5.02E+02
7	3	1	3	1.12E+02	5.41E+02	3.26E+02
8	3	2	1	1.10E+02	5.95E+02	3.53E+02
9	3	3	2	1.14E+02	6.62E+02	3.88E+02

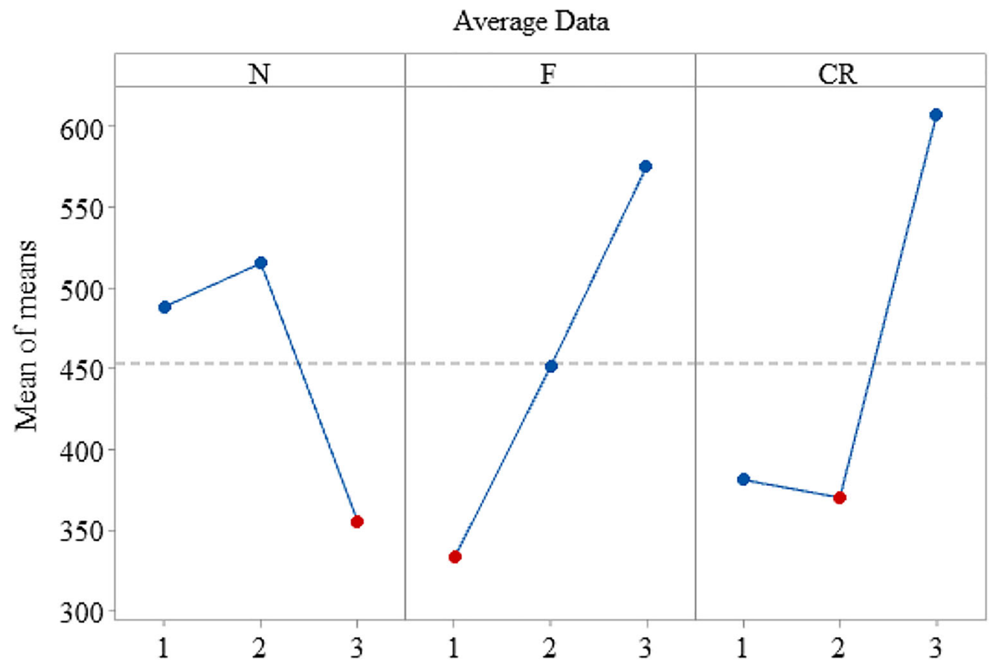
4.4 Results and analysis

In this experiment, the KDLMBO and the eight baselines algorithms were independently run 51 times on the selected benchmarks. The mean values with dimensions D = 10, 30, 50, and 100 are shown in Tables 4, 5, 6 and 7, respectively, where the mean values smaller than 1e-8 are taken as zero and the text of the minimum value in the experimental results is set to bold. It can be seen from the tables that the proposed KDLMBO outperforms the other eight baselines on the majority of the test benchmarks except the f21 and f30 when the dimension D = 10. With the increase of the search dimension, some of the baseline algorithms achieve better results than the proposed KDLMBO. However, the KDLMBO always returns the best solutions on most of the multimodal and hybrid functions when the dimensions are D = 30, 50, and 100.

Table 3 The importance of parameters

Levels	N	F	CR
1	4.89E+02	3.33E+02	3.82E+02
2	5.16E+02	4.51E+02	3.71E+02
3	3.56E+02	5.75E+02	6.08E+02
Delta	1.60E+02	2.42E+02	2.37E+02
Rank	3	1	2

Fig. 5 The main effect diagram of three parameters



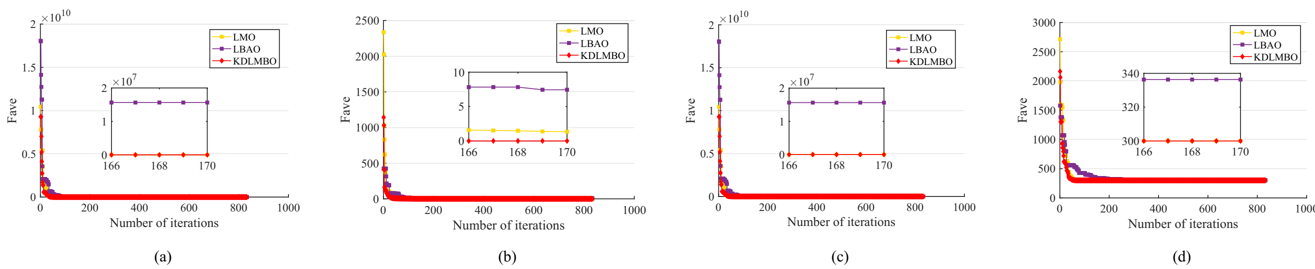


Fig. 6 The convergence curves of four functions from CEC-2017 with D = 10. **a** The convergence of f_1 with $D = 10$, **b** The convergence of f_4 with $D = 10$, **c** The convergence of f_{14} with $D = 10$, **d** The convergence of f_{26} with$D = 10$

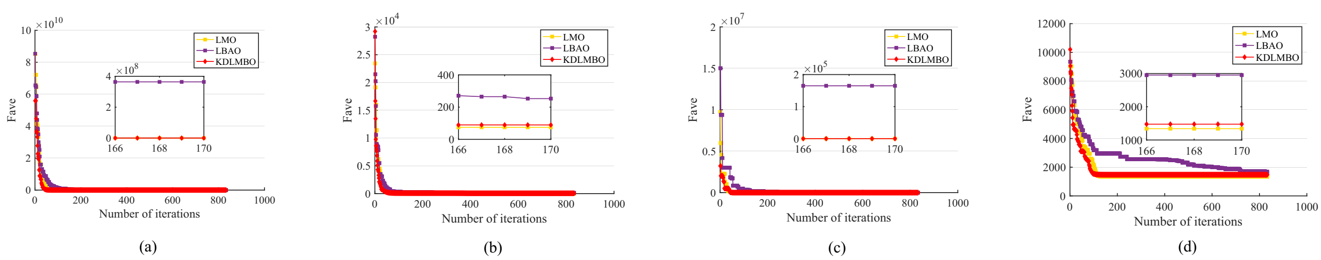


Fig. 7 The convergence curves of four functions from CEC-2017 with $D = 30$. **a** The convergence of $f_1$with D = 30, **b** The convergence of f_4 with D = 30, **c** The convergence of f_{14} with D = 30, **d** The convergence of f_{26} with$D = 30$

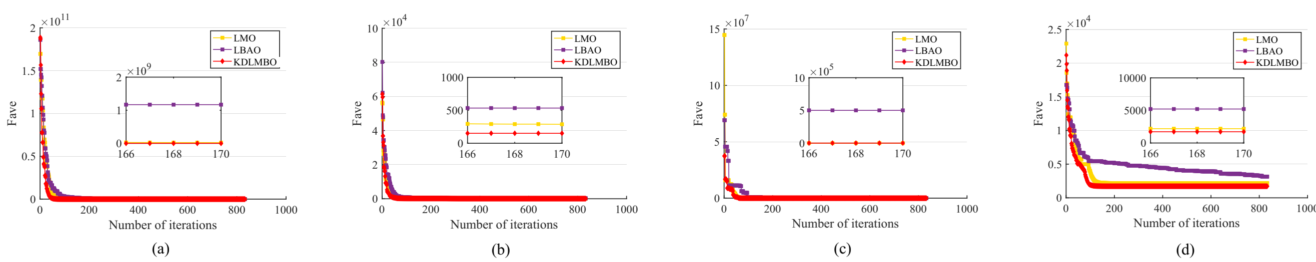


Fig. 8 The convergence curves of four functions from CEC-2017 with $D = 50$. **a** The convergence of $f_1$with D = 5, **b** The convergence of f_4 with D = 50, **c** The convergence of f_{14} with$D = 50$, **d** The convergence of f_{26} with D = 50

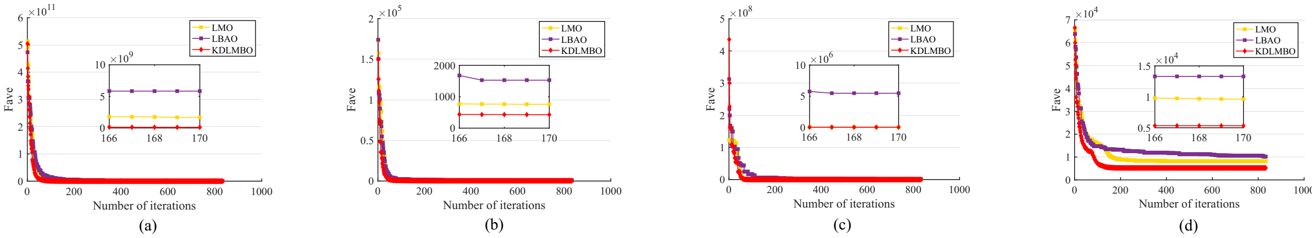


Fig. 9 The convergence curves of four functions from CEC-2017 with $D = 100$. **a** The convergence of f_1 with D = 100, **b** The convergence of f_4 with $D = 100$, **c** The convergence of $f_{14}$with D = 100, **d** The convergence of f_{26} with D = 100

4.4.1 Visual analysis

For a clear comparison, the visualization of the experimental results is shown in Fig. 10. The horizontal axis labels the number of the test functions. The vertical axis represents the normalized mean values. To clearly show the gaps in the performance, the mean values in Tables 4, 5, 6 and 7 were normalized by Eq. (29):

$$Normalized\ Value = log_{10}(Mean) \tag{29}$$

where Mean is the data in Tables 4, 5, 6 and 7. It can be seen from Fig. 10 that most of the curves of the proposed

Table 4 The results on the benchmarks when D = 10

Func	MBO	GCMBO	BBO	IWO	Jaya	CMA-ES	LMBODE	RLBSO	KDLMBO
1	5.93E+05	3.29E+03	3.61E+04	3.25E+03	1.36E+08	**0.00E+00**	5.71E+03	6.55E-14	**0.00E+00**
3	1.43E+04	4.87E+01	4.50E+03	3.28E-06	1.35E+03	5.55E+04	**0.00E+00**	3.34E-15	**0.00E+00**
4	2.47E+01	5.40E+00	8.94E+00	5.01E-02	8.30E+00	2.26E+03	5.47E-01	3.13E-01	**0.00E+00**
5	2.14E+01	1.19E+01	8.08E+00	9.38E+01	3.69E+01	2.62E+02	9.75E+00	9.31E+00	**6.62E+00**
6	6.73E-01	3.93E-04	1.25E-01	2.93E+01	5.19E+00	8.34E+01	1.44E-06	1.66E-03	**1.42E-06**
7	3.66E+01	2.10E+01	2.26E+01	1.86E+01	4.94E+01	4.10E+02	1.88E+01	1.93E+01	**1.61E+01**
8	2.18E+01	1.18E+01	1.04E+01	6.86E+01	3.65E+01	1.69E+02	8.39E+00	8.80E+00	**6.75E+00**
9	7.33E+01	7.56E-07	1.27E+00	8.26E+02	9.32E+00	3.17E+03	1.76E-03	**0.00E+00**	**0.00E+00**
10	6.92E+02	5.52E+02	3.20E+02	1.06E+03	9.48E+02	1.76E+03	4.06E+02	5.63E+02	**1.37E+02**
11	1.58E+02	9.58E+00	9.60E+00	7.00E+01	4.69E+01	6.54E+03	8.74E+00	2.73E+00	**2.07E+00**
12	6.55E+05	2.99E+05	1.43E+06	1.37E+04	2.87E+06	2.36E+08	1.72E+04	1.63E+02	**1.19E+02**
13	1.22E+04	8.04E+03	1.02E+04	1.24E+04	8.25E+03	3.79E+08	1.55E+02	7.41E+00	**5.21E+00**
14	7.47E+03	2.32E+03	7.26E+03	1.97E+02	8.42E+01	6.47E+04	8.66E+01	1.73E+01	**2.77E+00**
15	7.35E+03	3.94E+03	6.97E+03	1.34E+03	3.82E+02	6.87E+04	4.61E+01	1.91E+00	**9.86E-01**
16	1.76E+02	7.50E+01	1.80E+02	4.43E+02	6.70E+01	5.98E+02	5.53E+01	7.99E+01	**6.25E-01**
17	6.93E+01	3.78E+01	4.49E+01	2.97E+02	7.22E+01	8.91E+02	3.69E+01	1.53E+01	**1.28E+00**
18	1.45E+04	2.17E+04	9.15E+03	1.21E+04	3.85E+04	1.16E+09	4.82E+03	1.69E+01	**1.02E+00**
19	9.83E+03	7.68E+03	8.85E+03	6.90E+02	9.56E+02	8.63E+07	4.84E+01	1.09E+00	**1.07E-01**
20	5.61E+01	1.77E+01	6.36E+00	2.50E+02	5.44E+01	1.16E+03	2.70E+01	2.97E+01	**5.63E-02**
21	2.13E+02	1.58E+02	2.02E+02	2.54E+02	2.35E+02	4.78E+02	1.71E+02	**1.09E+02**	1.72E+02
22	1.04E+02	9.30E+01	1.03E+02	6.12E+02	1.10E+02	2.55E+03	9.75E+01	1.78E+02	**9.17E+01**
23	3.29E+02	3.16E+02	3.16E+02	4.58E+02	3.42E+02	1.81E+03	3.14E+02	3.13E+02	**3.07E+02**
24	3.60E+02	3.33E+02	3.36E+02	4.00E+02	3.63E+02	6.03E+02	3.20E+02	3.45E+02	**3.01E+02**
25	4.40E+02	4.30E+02	4.31E+02	**4.11E+02**	4.48E+02	7.97E+02	4.19E+02	4.20E+02	4.16E+02
26	4.67E+02	3.24E+02	4.24E+02	1.36E+03	5.35E+02	2.40E+03	3.23E+02	3.50E+02	**3.00E+02**
27	4.12E+02	4.46E+02	3.99E+02	4.75E+02	3.97E+02	1.70E+03	4.33E+02	3.80E+02	**3.95E+02**
28	4.84E+02	4.76E+02	5.30E+02	4.81E+02	5.82E+02	5.00E+02	4.92E+02	4.64E+02	**3.17E+02**
29	3.24E+02	2.95E+02	2.80E+02	5.10E+02	2.85E+02	3.02E+04	2.85E+02	2.55E+02	**2.44E+02**
30	1.06E+04	7.20E+02	5.81E+05	3.81E+05	2.24E+05	2.84E+08	2.88E+02	**2.08E+02**	4.51E+02

Table 5 The results on the benchmarks when D = 30

Func	MBO	GCMBO	BBO	IWO	Jaya	CMA-ES	LMBODE	RLBSO	KDLMBO
1	2.28E+08	3.23E+03	1.37E+05	4.55E+03	6.40E+09	**0.00E+00**	4.16E+03	1.63E-05	**0.00E+00**
3	2.88E+07	3.38E+04	7.43E+04	7.76E-05	5.24E+04	4.10E+05	1.38E-06	3.35E+03	**0.00E+00**
4	9.85E+01	7.91E+01	9.85E+01	6.92E+01	2.83E+02	1.69E+04	2.16E+01	2.44E+01	**2.09E+01**
5	1.37E+02	5.44E+01	5.71E+01	3.11E+02	2.31E+02	5.71E+02	4.72E+01	5.63E+01	**3.41E+01**
6	8.24E+00	2.74E-02	1.37E-01	5.50E+01	2.26E+01	9.05E+01	**8.36E-04**	1.16E-01	1.32E-01
7	2.09E+02	8.83E+01	1.09E+02	7.69E+01	3.45E+02	3.39E+03	7.70E+01	8.30E+01	**5.91E+01**
8	1.36E+02	5.52E+01	6.05E+01	2.83E+02	2.41E+02	6.06E+02	3.98E+01	5.33E+01	**3.36E+01**
9	3.56E+03	2.02E+02	3.36E+02	8.04E+03	3.02E+03	1.95E+04	6.90E+01	1.32E+01	**4.65E+00**
10	3.32E+03	2.74E+03	**2.22E+03**	3.12E+03	6.86E+03	7.79E+03	2.62E+03	2.97E+03	3.35E+03
11	3.29E+03	9.52E+01	2.16E+03	1.73E+02	8.03E+02	4.05E+03	9.26E+01	**2.18E+01**	7.20E+01
12	7.35E+06	2.91E+06	3.09E+06	8.78E+05	7.34E+07	1.25E+09	7.59E+04	3.05E+04	**6.35E+03**
13	1.66E+06	7.82E+04	5.14E+04	1.18E+05	8.52E+06	7.29E+08	1.75E+04	2.20E+02	**1.10E+02**
14	4.12E+05	9.22E+04	1.84E+06	2.80E+03	8.57E+04	4.93E+06	7.54E+02	4.50E+01	**5.56E+01**
15	2.40E+05	1.70E+04	2.82E+04	8.04E+04	4.38E+06	5.22E+05	2.20E+04	7.30E+01	**7.51E+01**
16	1.14E+03	8.17E+02	1.16E+03	1.08E+03	1.62E+03	4.19E+03	5.37E+02	9.17E+02	**5.18E+02**
17	6.80E+02	3.60E+02	4.85E+02	8.35E+02	5.76E+02	6.51E+02	2.68E+02	4.33E+02	**1.15E+02**
18	2.73E+06	1.12E+06	2.44E+06	8.73E+04	1.81E+06	2.74E+08	3.42E+04	9.50E+03	**5.91E+01**
19	1.12E+05	1.23E+04	2.16E+04	7.80E+04	4.40E+05	1.09E+09	9.65E+03	1.59E+01	**5.06E+01**
20	6.13E+02	3.95E+02	4.99E+02	8.77E+02	6.12E+02	2.58E+03	2.01E+02	4.43E+02	**1.69E+02**
21	3.19E+02	2.60E+02	2.69E+02	4.86E+02	4.21E+02	1.06E+03	2.48E+02	2.59E+02	**2.34E+02**
22	3.63E+03	1.78E+03	1.98E+03	3.66E+03	6.23E+03	9.99E+03	1.05E+03	3.38E+03	**1.00E+02**
23	5.19E+02	4.22E+02	4.22E+02	7.90E+02	6.02E+02	1.62E+03	4.19E+02	4.12E+02	**3.86E+02**
24	6.05E+02	4.99E+02	4.96E+02	7.63E+02	6.65E+02	2.63E+03	5.05E+02	4.83E+02	**4.52E+02**
25	4.08E+02	3.87E+02	3.93E+02	3.88E+02	4.72E+02	1.87E+03	**3.79E+02**	**3.79E+02**	3.87E+02
26	2.62E+03	1.39E+03	1.93E+03	3.26E+03	3.80E+03	1.43E+04	1.51E+03	1.42E+03	**1.31E+03**
27	5.00E+02	5.00E+02	5.35E+02	5.99E+02	5.40E+02	5.00E+02	5.00E+02	**5.00E+02**	5.27E+02
28	4.93E+02	4.91E+02	4.23E+02	3.46E+02	8.66E+02	5.00E+02	4.93E+02	4.99E+02	**3.45E+02**
29	9.46E+02	6.13E+02	8.37E+02	1.31E+03	1.43E+03	4.78E+03	**5.43E+02**	5.58E+02	5.87E+02
30	2.14E+04	5.68E+03	1.92E+04	3.54E+05	5.98E+06	1.65E+06	3.90E+03	2.20E+02	**2.58E+03**

KDLMBO are at the bottom of the graph, meaning that the results of the KDLMBO for most benchmark functions are better than the other algorithms. For the hybrid functions f_11-f_20, the KDLMBO is an excellent algorithm compared with the other baseline algorithms. In some simple unimodal, multimodal and composition functions, the KDLMBO is better than the other baseline algorithms. The inseparability of variables was maintained by the mutation and crossover of the KDLMBO. The learning mechanism played an essential role to maintain the equilibration of the convergence and diversity of the KDLMBO. The visualization of experimental results shows that the proposed mechanisms in the KDLMBO algorithm are feasible, and the performance of the KDLMBO is expectable.

To analyze the performance of the KDLMBO on different benchmark functions, the convergence curves of the KDLMBO and the eight baseline algorithms are shown in Figs. 11, 12, 13 and 14. The figures show that the KDLMBO has the best accuracy among all algorithms on f_5 and f_{21} with D=10, 30, 50, and 100. The KDLMBO is better than the MBO on the simple multimodal and the hybrid functions. The reason is that the learning mechanism balances the convergence speed and the accuracy of the KDLMBO algorithm. The LMO and LBAO of the KDLMBO avoid the population falling into local optima. Therefore, the proposed KDLMBO algorithm is feasible and effective for addressing continuous optimization problems in terms of convergence speed and accuracy.

In order to analyze the stability of the proposed algorithm, the box plots of the KDLMBO and the eight baseline algorithms for f_1, f_5, f_{14} and f_{21} are shown in Figs. 15, 16, 17 and 18, respectively. In this study, the actions of the KDLMBO

Table 6 The results on the benchmarks when D = 50

Func	MBO	GCMBO	BBO	IWO	Jaya	CMA-ES	LMBODE	RLBSO	KDLMBO
1	8.89E+10	5.39E+09	3.24E+05	1.53E+04	2.52E+10	**0.00E+00**	3.08E+03	6.77E-01	1.33E-02
3	5.60E+05	1.61E+05	9.50E+04	2.97E-02	1.18E+05	3.20E+05	4.68E+01	1.37E+05	**1.21E-04**
4	1.68E+04	9.73E+02	1.52E+02	1.14E+02	1.86E+03	2.33E+03	7.67E+01	3.92E+01	**7.21E+01**
5	6.53E+02	2.15E+02	1.19E+02	7.87E+01	5.02E+02	1.35E+03	1.06E+02	1.20E+02	**5.87E+01**
6	7.67E+01	2.09E+01	**1.43E-01**	6.25E-01	4.10E+01	9.37E+01	2.61E-01	1.66E+00	1.11E+00
7	2.49E+03	2.74E+02	2.15E+02	1.31E+02	7.94E+02	7.51E+03	1.67E+02	1.67E+02	**1.03E+02**
8	6.51E+02	2.14E+02	1.21E+02	8.31E+01	5.42E+02	1.44E+03	1.05E+02	1.13E+02	**6.15E+01**
9	3.25E+04	3.50E+03	1.15E+03	**1.31E+00**	1.33E+04	3.75E+04	9.01E+02	3.83E+02	4.51E+01
10	1.25E+04	8.26E+03	4.22E+03	**3.39E+03**	1.34E+04	1.43E+04	4.93E+03	5.86E+03	6.47E+03
11	3.14E+04	4.98E+03	6.03E+03	2.26E+02	2.23E+03	9.40E+04	3.05E+02	5.44E+01	**1.38E+02**
12	3.02E+10	1.38E+09	8.26E+06	7.37E+06	3.86E+09	2.80E+09	1.64E+06	2.96E+05	**4.61E+04**
13	1.46E+10	3.78E+08	8.64E+04	2.03E+05	5.66E+08	1.45E+06	1.21E+04	2.33E+03	**1.79E+03**
14	3.74E+07	3.53E+06	4.80E+06	2.12E+04	8.98E+05	3.90E+06	1.39E+04	1.34E+04	**1.62E+02**
15	5.01E+09	4.36E+07	4.33E+04	8.54E+04	1.16E+08	6.91E+08	1.35E+04	5.97E+02	**2.56E+02**
16	5.28E+03	2.03E+03	1.80E+03	**7.41E+02**	3.36E+03	6.68E+03	1.17E+03	1.71E+03	9.30E+02
17	8.82E+03	1.40E+03	1.40E+03	8.00E+02	2.38E+03	1.50E+05	8.01E+02	1.21E+03	**7.98E+02**
18	1.14E+08	1.65E+07	7.17E+06	1.81E+05	9.58E+06	2.24E+06	1.57E+05	1.07E+05	**3.17E+02**
19	2.16E+09	8.96E+06	2.90E+04	2.17E+05	2.67E+07	8.76E+05	1.62E+04	7.20E+02	**7.97E+01**
20	2.42E+03	1.16E+03	1.09E+03	**5.22E+02**	1.85E+03	2.47E+03	6.10E+02	9.07E+02	5.98E+02
21	8.46E+02	4.31E+02	3.25E+02	2.87E+02	6.85E+02	1.62E+03	3.03E+02	3.23E+02	**2.58E+02**
22	1.29E+04	8.66E+03	5.19E+03	**3.72E+03**	1.33E+04	1.96E+04	5.39E+03	6.87E+03	4.98E+03
23	1.40E+03	7.81E+02	5.89E+02	5.04E+02	1.02E+03	4.07E+03	5.22E+02	5.39E+02	**5.00E+02**
24	1.44E+03	8.81E+02	6.58E+02	5.68E+02	1.02E+03	3.41E+03	6.86E+02	6.29E+02	**5.57E+02**
25	1.22E+04	1.27E+03	5.53E+02	4.91E+02	1.25E+03	2.45E+03	**4.64E+02**	4.42E+02	5.24E+02
26	1.15E+04	4.62E+03	2.84E+03	1.90E+03	7.63E+03	3.63E+04	1.81E+03	2.44E+03	**1.76E+03**
27	1.96E+03	1.23E+03	7.18E+02	5.43E+02	8.68E+02	5.00E+02	5.00E+02	**5.00E+02**	6.84E+02
28	6.80E+03	2.17E+03	5.10E+02	**4.72E+02**	3.53E+03	5.00E+02	4.96E+02	5.00E+02	4.93E+02
29	6.29E+03	2.01E+03	1.19E+03	9.09E+02	2.80E+03	1.11E+04	8.49E+02	1.16E+03	**7.88E+02**
30	2.95E+09	9.10E+07	1.13E+06	1.30E+07	7.48E+07	5.81E+08	7.37E+03	**3.07E+02**	1.34E+06

are guided by the learning mechanism. The generation of the KDLMBO is not just a random search. The process of the KDLMBO is intelligent. Figures 15, 16, 17 and 18 demonstrate that the KDLMBO is superior to the baseline algorithms.

4.4.2 Friedman test

To make a statistical comparison between KDLMBO and other algorithms, the Friedman test was conducted. The Friedman test is a non-parametric statistical test used to determine significant differences in multiple (related) samples. The results of the Friedman test are shown in Tables 8, 9, 10 and 11. As shown in Figs. 19, 20, 21 and 22, it can be observed that the Mean Rank of the KDLMBO is the minimum. Furthermore, the performance of KDLMBO is excellent compared with the other algorithms at 90% and 95% confidence intervals. There are significant differences between the algorithms, and the Mean Rank of the KDLMBO algorithm is the smallest among all algorithms with D=10,30,50,100.

4.4.3 Wilcoxon test

In addition, the Wilcoxon test was also adopted to make a pairwise comparison between the KDLMBO and the other algorithms. The method was developed based on the sign test of paired observation data and was more effective than the traditional plus or minus sign alone. The results of the Wilcoxon sign test are shown in Table 12. The R+ represents outstanding results compared with the other contrast algorithm. R- is the opposite. In the Wilcoxon test, if the p-value is less than α, it means that there are significant differences in

Table 7 The results on the benchmarks when D = 100

Func	MBO	GCMBO	BBO	IWO	Jaya	CMA-ES	LMBODE	RLBSO	KDLMBO
1	2.25E+11	1.81E+10	1.13E+06	9.98E+03	1.11E+11	**0.00E+00**	5.69E+03	6.93E+02	3.36E+00
3	9.82E+05	3.68E+05	1.82E+05	**2.70E-03**	4.12E+05	5.87E+05	1.03E+00	1.20E+05	5.47E-01
4	4.24E+04	2.84E+03	2.99E+02	2.26E+02	1.31E+04	2.01E+03	1.95E+02	**1.51E+02**	2.05E+02
5	1.45E+03	5.72E+02	3.18E+02	9.81E+02	1.27E+03	1.65E+03	3.70E+02	3.67E+02	**1.73E+02**
6	8.78E+01	2.92E+01	**1.58E-01**	7.45E+01	6.84E+01	8.32E+01	4.71E+00	1.54E+01	7.51E+00
7	5.96E+03	7.17E+02	6.12E+02	3.93E+02	2.71E+03	1.16E+04	6.32E+02	5.31E+02	**3.15E+02**
8	1.50E+03	5.69E+02	3.30E+02	9.64E+02	1.34E+03	2.36E+03	3.89E+02	3.76E+02	**1.78E+02**
9	7.64E+04	1.32E+04	5.21E+03	4.34E+04	5.57E+04	5.27E+04	9.24E+03	7.06E+03	**9.46E+02**
10	2.74E+04	1.91E+04	**1.09E+04**	1.09E+04	3.02E+04	1.59E+04	1.29E+04	1.46E+04	1.27E+04
11	2.77E+05	8.53E+04	5.79E+04	1.33E+03	4.41E+04	2.93E+05	2.51E+03	5.89E+02	**7.23E+02**
12	8.31E+10	5.38E+09	2.55E+07	1.44E+07	1.98E+10	3.39E+08	2.11E+06	1.38E+06	**1.52E+05**
13	1.82E+10	4.68E+08	3.47E+04	1.01E+05	2.66E+09	8.50E+04	1.38E+04	4.37E+03	**5.33E+03**
14	8.20E+07	1.67E+07	6.92E+06	6.46E+04	1.61E+07	2.22E+05	2.01E+04	7.25E+04	**2.07E+03**
15	8.19E+09	9.72E+07	1.95E+04	7.83E+04	8.01E+08	5.13E+05	8.29E+03	3.53E+03	**8.23E+02**
16	1.18E+04	4.77E+03	4.06E+03	3.38E+03	9.45E+03	2.10E+04	2.84E+03	3.93E+03	**2.76E+03**
17	9.93E+04	3.41E+03	3.07E+03	2.93E+03	8.54E+03	5.80E+03	2.97E+03	2.97E+03	**2.29E+03**
18	1.05E+08	9.26E+06	5.66E+06	2.07E+05	3.06E+07	1.20E+07	1.03E+05	4.48E+05	**1.49E+04**
19	7.57E+09	1.03E+08	2.16E+04	6.38E+05	9.76E+08	5.63E+04	5.53E+03	3.83E+03	**6.38E+02**
20	5.96E+03	3.25E+03	3.01E+03	2.89E+03	5.23E+03	5.85E+03	2.41E+03	2.73E+03	**2.37E+03**
21	1.74E+03	9.42E+02	5.86E+02	1.20E+03	1.52E+03	4.49E+03	6.19E+02	6.64E+02	**4.07E+02**
22	2.85E+04	2.08E+04	**1.19E+04**	1.27E+04	3.10E+04	3.02E+04	1.37E+04	1.60E+04	1.51E+04
23	2.43E+03	1.39E+03	**7.62E+02**	1.82E+03	1.98E+03	5.87E+03	9.89E+02	9.83E+02	8.31E+02
24	3.85E+03	2.03E+03	1.29E+03	1.76E+03	2.66E+03	1.39E+04	1.65E+03	1.36E+03	**1.20E+03**
25	2.69E+04	2.79E+03	8.15E+02	**7.22E+02**	7.98E+03	4.06E+03	7.42E+02	7.50E+02	7.40E+02
26	2.93E+04	1.28E+04	7.57E+03	1.11E+04	2.33E+04	2.87E+04	1.11E+04	8.69E+03	**6.23E+03**
27	3.38E+03	1.65E+03	8.39E+02	7.79E+02	1.84E+03	5.00E+02	5.00E+02	**5.00E+02**	9.83E+02
28	2.23E+04	5.99E+03	6.28E+02	5.55E+02	1.76E+04	5.00E+02	5.00E+02	**5.00E+02**	5.88E+02
29	7.05E+04	5.46E+03	3.51E+03	3.94E+03	9.52E+03	9.79E+03	**2.78E+03**	3.08E+03	3.06E+03
30	1.17E+10	4.86E+08	6.07E+04	3.21E+06	1.75E+09	1.10E+08	7.10E+03	**4.46E+02**	9.09E+03

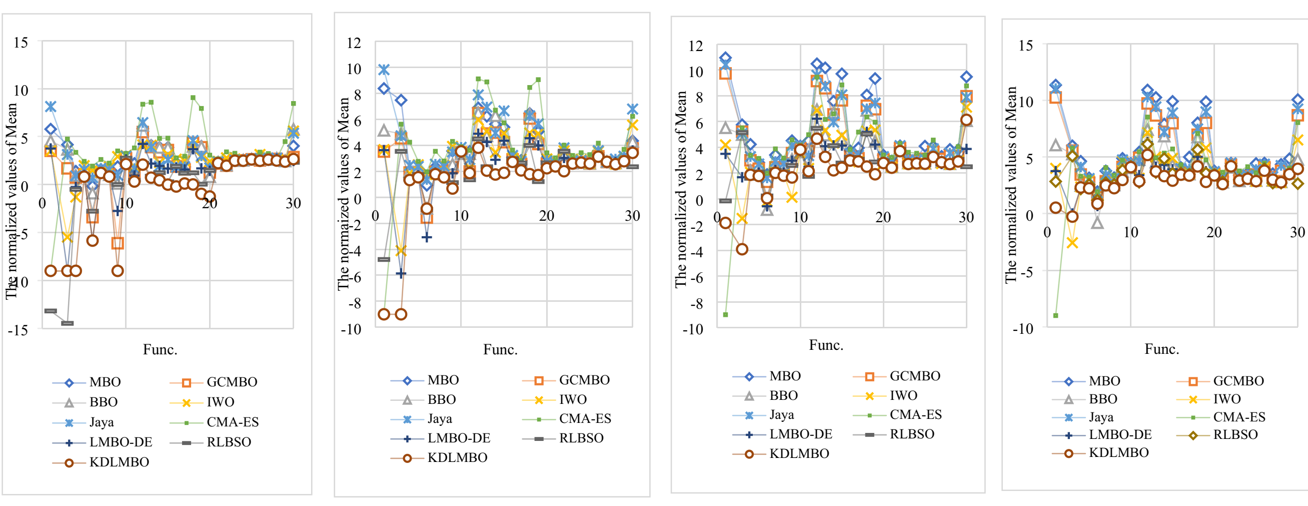


Fig. 10 Experimental results of nine algorithms on the benchmarks with D = 10, 30, 50 and 100. **a** Experimental results of nine algorithms with D = 10, **b** Experimental results of nine algorithms with D = 30, F **c** Experimental results of nine algorithms with D = 50, **d** Experimental results of nine algorithms with D = 100

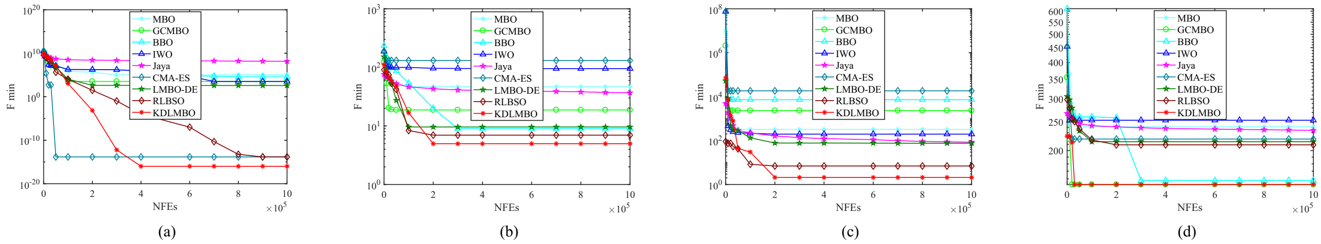


Fig. 11 The convergence curves of different algorithms for four benchmark functions with $D = 10$. **a** The convergence of f_1 with D = 10, **b** The convergence of f_5 with D = 10, **c** The convergence of f_{14} with D = 10, **d** The convergence of f_{21} with D = 10

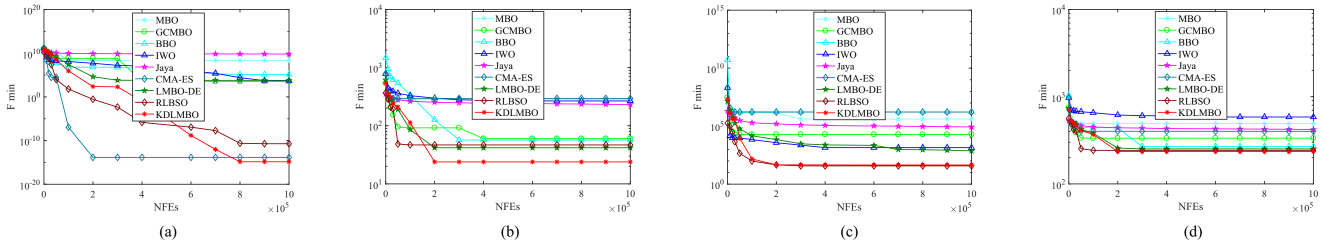


Fig. 12 The convergence curves of different algorithms for four benchmark functions with $D = 30$. **a** The convergence of f_1 with D = 30, **b** The convergence of f_5 with D = 30, **c** The convergence of f_{14} with D = 30, **d** The convergence of f_{21} with D = 30

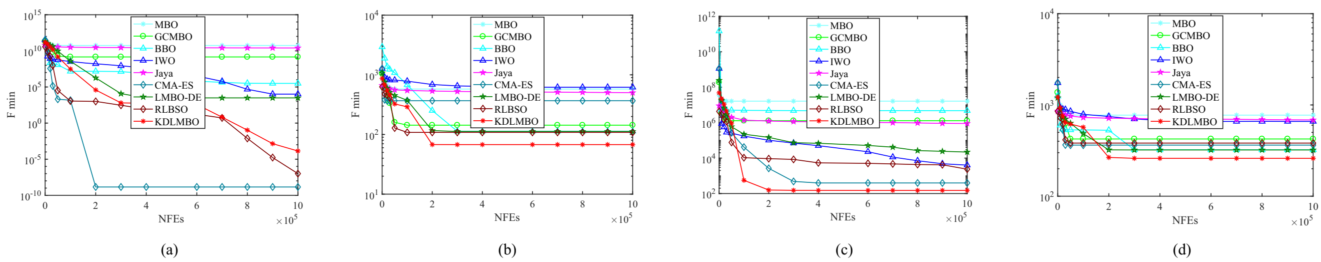


Fig. 13 The convergence curves of different algorithms for four benchmark functions with $D = 50$. **a** The convergence of f_1 with D = 50, **b** The convergence of f_5 with D = 50, **c** The convergence of f_{14} with D = 50, **d** The convergence of f_{21} with D = 50

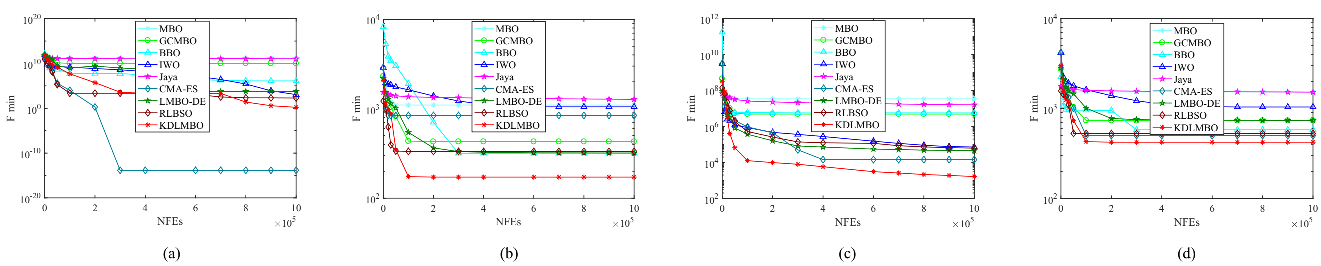


Fig. 14 The convergence curves of different algorithms for four benchmark functions with $D = 100$. **a** The convergence of f_1 with D = 100, **b** The convergence of f_5 with D = 100, **c** The convergence of f_{14} with D = 100, **d** The convergence of f_{21} withD = 100

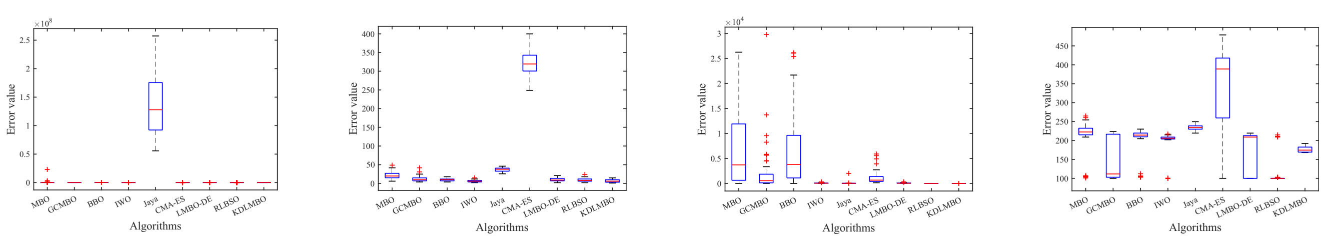


Fig. 15 Boxplots of four typical benchmark functions with different algorithms (10D). **a** The mean plot of f_1, **b** The mean plot of f_5, **c** The mean plot of f_{14}, **d** The mean plot of f_{21}

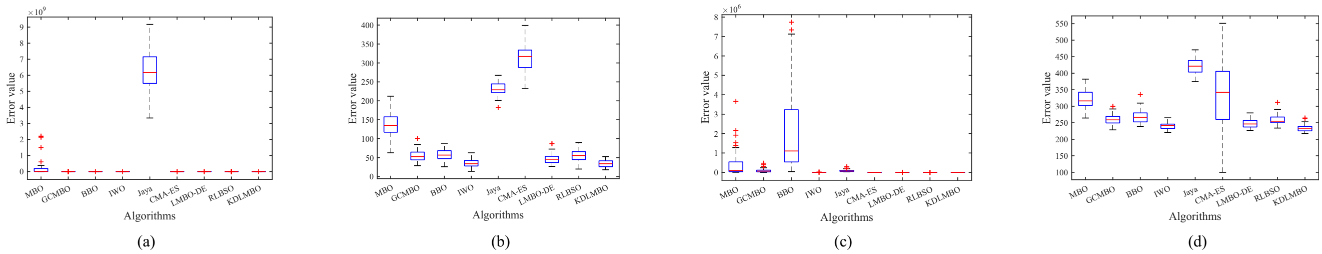


Fig. 16 Boxplots of four benchmark functions with different algorithms (30D). **a** The mean plot of f_1, **b** The mean plot of f_5, **c** The mean plot of f_{14}, **d** The mean plot of f_{21}

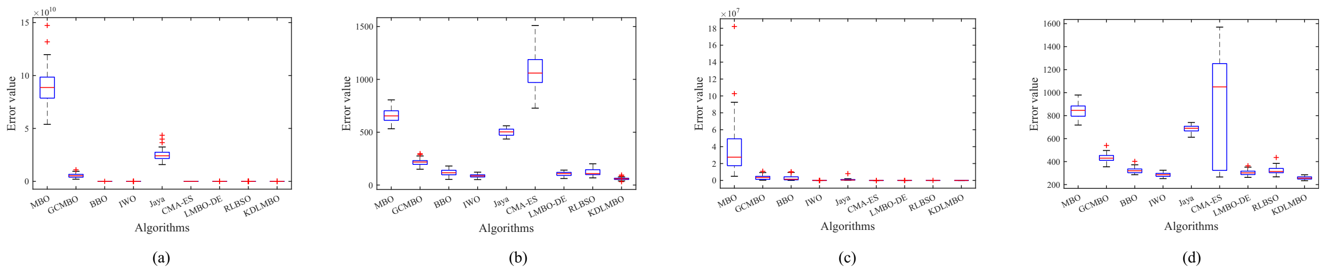


Fig. 17 Boxplots of four benchmark functions with different algorithms (50D). **a** The mean plot of f_1, **b** The mean plot of f_5, **c** The mean plot of f_{14}, **d** The mean plot of f_{21}

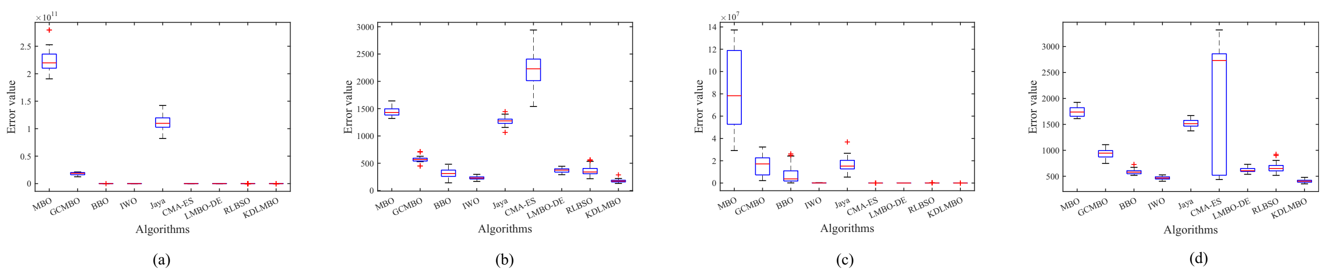


Fig. 18 Boxplots of four typical benchmark functions with different algorithms (100D). **a** The mean plot of f_1, **b** The mean plot of f_5, **c** The mean plot of f_{14}, **d** The mean plot of f_{21}

Table 8 Friedman-test results with D = 10

Algorithm	Mean Rank
MBO	6.55
GCMBO	4.43
BBO	5.29
IWO	6.24
Jaya	6.29
CMA_ES	8.67
LMBO_DE	3.31
RLBSO	2.91
KDLMBO	1.29
Crit. Diff α=0.05	2.69
Crit. Diff α=0.1	2.45

Table 9 Friedman-test results with D = 30

Algorithm	Mean Rank
MBO	6.59
GCMBO	3.86
BBO	5.31
IWO	6.10
Jaya	7.41
CMA_ES	8.33
LMBO_DE	2.76
RLBSO	2.81
KDLMBO	1.83
Crit. Diff α=0.05	2.69
Crit. Diff α=0.1	2.45

Table 10 Friedman-test results with D = 50

Algorithm	Mean Rank
MBO	8.38
GCMBO	6.36
BBO	4.74
IWO	2.83
Jaya	6.97
CMA_ES	7.67
LMBO_DE	3.02
RLBSO	3.17
KDLMBO	1.86
Crit. Diff α=0.05	2.69
Crit. Diff α=0.1	2.45

Table 11 Friedman-test results with D = 100

Algorithm	Mean Rank
MBO	8.62
GCMBO	6.24
BBO	3.78
IWO	4.14
Jaya	7.48
CMA_ES	6.83
LMBO_DE	3.03
RLBSO	2.95
KDLMBO	1.93
Crit. Diff α=0.05	2.69
Crit. Diff α=0.1	2.45

the pairwise algorithms. The KDLMBO was compared with the eight baseline algorithms, and all the p-values are less than α at D=10, 30, 50, 100. Therefore, according to the Wilcoxon test, the KDLMBO is promising compared with other algorithms.

5 Conclusion and future research

In this study, a novel algorithm (KDLMBO) based on the MBO with a knowledge-driven learning mechanism (KDLMBO) is proposed to strengthen the self-learning and self-adapting capabilities of the original algorithm. In the proposed KDLMBO algorithm, the prior knowledge of the algorithm is defined. The information of the neighborhood is represented by four kinds of actions. The appropriate actions of the KDLMBO are guided by the cooperative learning mechanism. The Wilcoxon test results demonstrate that the fitness and information are effectively utilized in the iterations of the algorithm. The experiments conducted on 29 benchmarks show that the performance of the KDLMBO algorithm is increased by 240.40% and 459.03% compared with the LMBODE and the original MBO algorithm, respectively. The learning mechanism efficiently increases the self-organizing ability of the algorithm and maintains the inseparability of variables. The collective intelligence is self-learning and is realized by the mechanism in the iterative process of the algorithm.

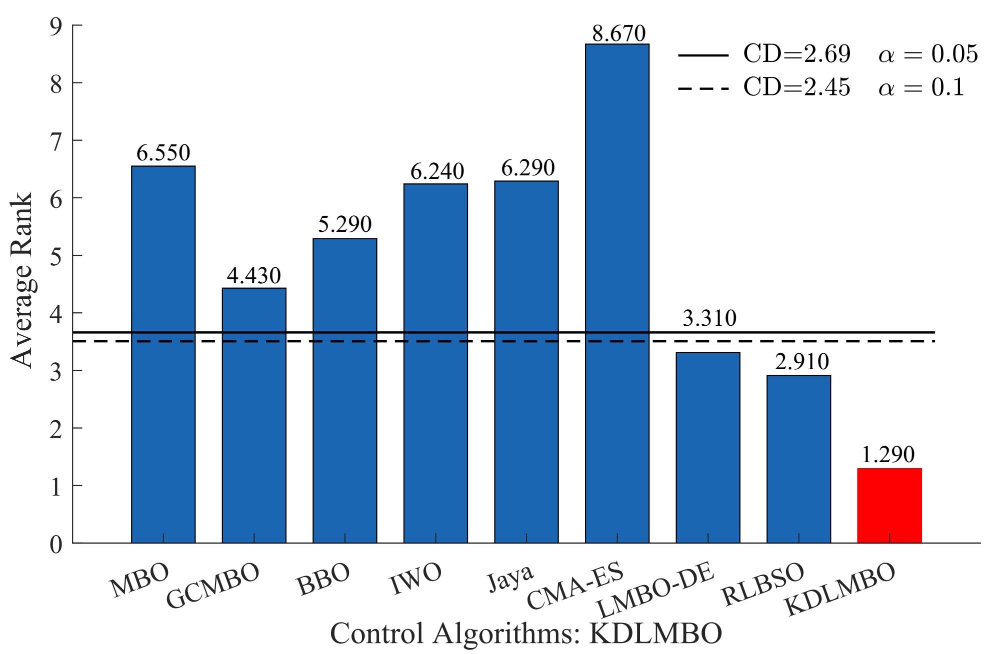


Fig. 19 Rankings with D = 10

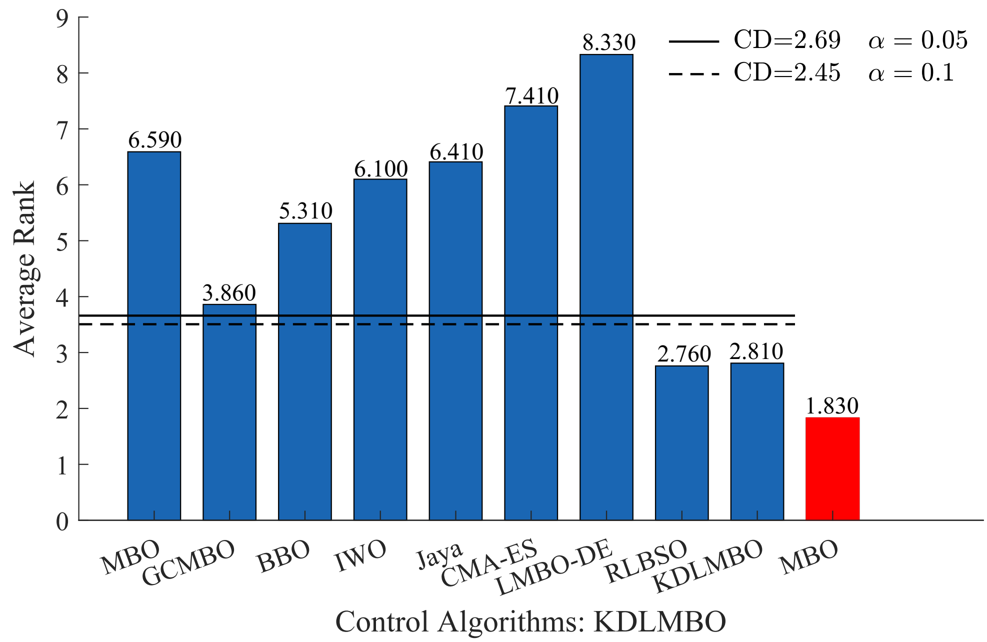


Fig. 20 Rankings with $D = 30$

Therefore, the evolution process of the KDLMBO is not a random search rather an intelligent mechanism. The experimental results demonstrate and validate that the knowledge-driven learning mechanism is feasible, contributing and promising.

In future work, the effectiveness and efficiency of the proposed KDLMBO algorithm for optimizing high-dimensional continuous optimization problems will be further explored. Meanwhile, the learning mechanism in the KDLMBO algorithm can be improved by combining advanced machine learning mechanisms including the typical Apriori heuristic that can enhance learning personalization and autonomy. Moreover, improved MBO versions can be utilized to address practically intractable optimization problems such as the flow shop scheduling problem and the optimal reactive power dispatch, etc.

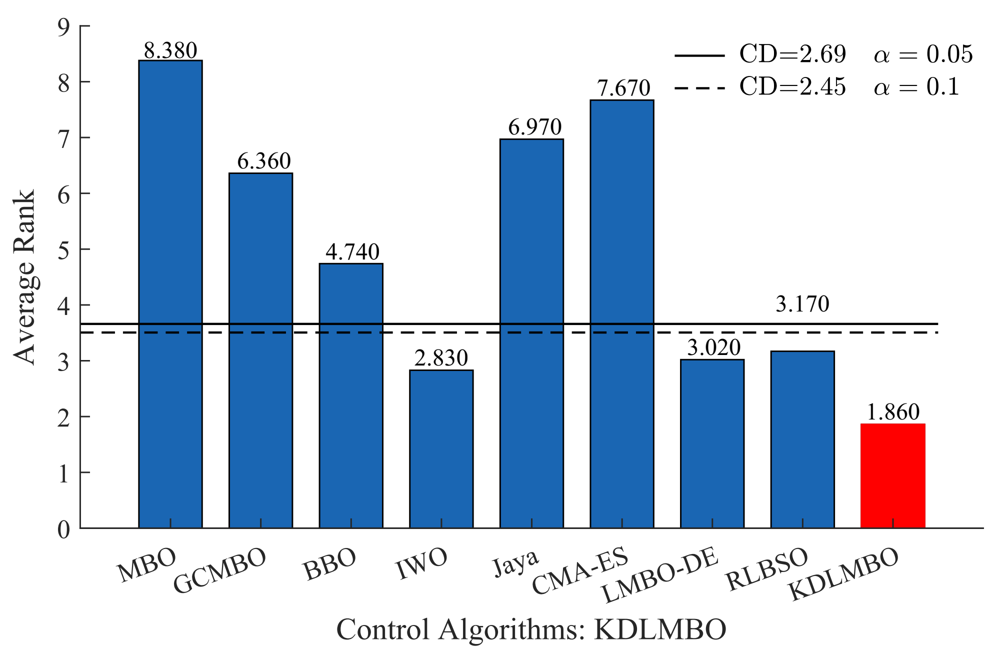


Fig. 21 Rankings with $D = 50$

Fig. 22 Rankings with D = 100

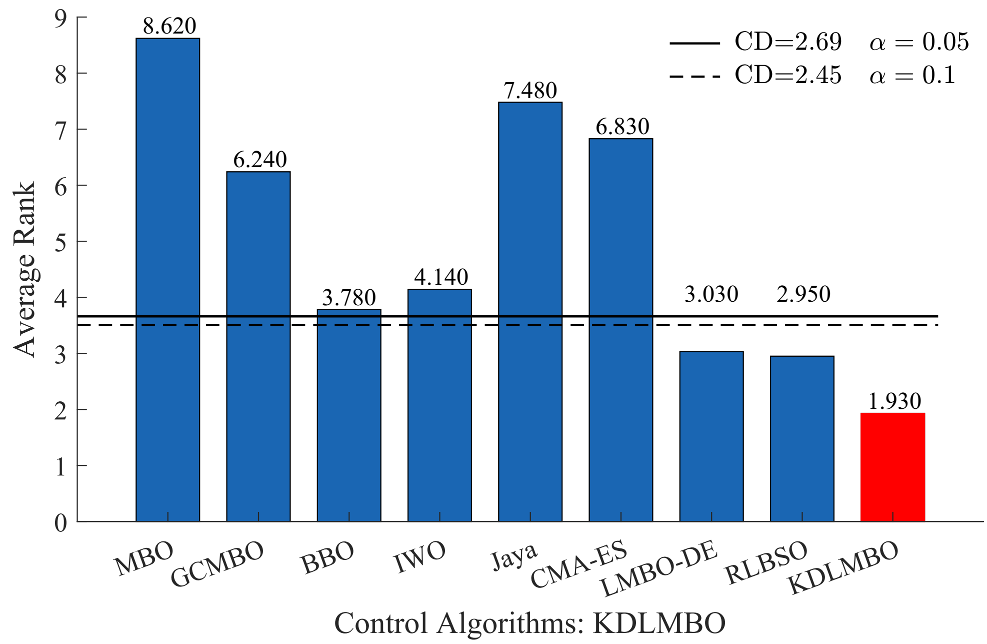


Table 12 Wilcoxon sign test results

D	KDLMBO pk	R+	R-	*p*-value	α=0.1	α=0.05
10	MBO	29	0	2.56E-06	yes	yes
	GCMBO	28	1	7.60E-06	yes	yes
	BBO	29	0	2.56E-06	yes	yes
	IWO	28	1	3.90E-06	yes	yes
	Jaya	29	0	2.56E-06	yes	yes
	CMA-ES	28	0	3.79E-06	yes	yes
	LMBO-DE	25	3	6.13E-05	yes	yes
	RLBSO	26	2	1.95E-03	yes	yes
30	MBO	27	2	5.32E-06	yes	yes
	GCMBO	25	3	5.56E-05	yes	yes
	BBO	28	1	1.60E-05	yes	yes
	IWO	28	1	6.53E-06	yes	yes
	Jaya	29	0	2.56E-06	yes	yes
	CMA-ES	27	1	4.23E-06	yes	yes
	LMBO-DE	24	5	3.75E-04	yes	yes
	RLBSO	19	10	4.55E-02	yes	yes
50	MBO	29	0	2.56E-06	yes	yes
	GCMBO	29	0	2.56E-06	yes	yes
	BBO	26	3	1.91E-04	yes	yes

Table 12 (continued)

D	KDLMBO pk	R+	R-	*p*-value	α = 0.1	α = 0.05
	IWO	20	3	4.55E-02	yes	yes
	Jaya	29	0	2.56E-06	yes	yes
	CMA-ES	27	2	4.33E-06	yes	yes
	LMBODE	24	5	2.75E-03	yes	yes
	RLBSO	23	6	5.46E-03	yes	yes
100	MBO	29	0	2.56E-06	yes	yes
	GCMBO	29	0	2.56E-06	yes	yes
	BBO	24	5	2.09E-04	yes	yes
	IWO	23	6	3.18E-04	yes	yes
	Jaya	29	0	2.56E-06	yes	yes
	CMA-ES	26	3	5.90E-06	yes	yes
	LMBO-DE	22	7	3.16E-03	yes	yes
	RLBSO	23	6	1.65E-03	yes	yes

Funding This work was financially supported by the National Natural Science Foundation of China under grant 62063021. It was also supported by the Key talent project of Gansu Province (ZZ2021G50700016), the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (21YF5WA086), Lanzhou Science Bureau project (2018-rc-98), and Project of Gansu Natural Science Foundation (21JR7RA204) respectively.

Declarations

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

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Ali MZ, Awad NH, Reynolds RG, Suganthan PN (2018) A balanced fuzzy cultural algorithm with a modified levy flight search for real parameter optimization. Inf Sci (Ny) 447:12–35. https://doi.org/10.1016/j.ins.2018.03.008
2. Kim H (2018) Parallel genetic algorithm with a knowledge base for a redundancy allocation problem considering the sequence of heterogeneous components. Expert Syst Appl 113:328–338. https://doi.org/10.1016/j.eswa.2018.06.056
3. Zhao F, He X, Wang L (2021) A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem. IEEE Trans Cybern 51: 5291–5303. https://doi.org/10.1109/TCYB.2020.3025662
4. Zhao F, Liu Y, Shao Z et al (2015) A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem. 29:962–981. https://doi.org/10.1080/0951192X.2015.1130240
5. Wang F, Zhang H, Li K, Lin Z, Yang J, Shen XL (2018) A hybrid particle swarm optimization algorithm using adaptive learning strategy. Inf Sci (Ny) 436–437:162–177. https://doi.org/10.1016/j.ins.2018.01.027
6. Lv X, Zhou D, Tang Y (2018) Ma L (2018) an improved test selection optimization model based on fault ambiguity group isolation and chaotic discrete PSO. Complexity 2018:1–10. https://doi.org/10.1155/2018/3942723
7. Cao Y, Zhang H, Li W, Zhou M, Zhang Y, Chaovalitwongse WA (2019) Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions. IEEE Trans Evol Comput 23:718–731. https://doi.org/10.1109/TEVC.2018.2885075
8. Gao G, Mei Y, Jia YH, Browne WN, Xin B (2021) Adaptive coordination ant Colony optimization for multipoint dynamic aggregation. IEEE Trans Cybern PP: https://doi.org/10.1109/TCYB.2020.3042511, PP
9. Mortazavi A, Moloodpoor M (2021) Enhanced butterfly optimization algorithm with a new fuzzy regulator strategy and virtual butterfly concept. Knowledge-Based Syst 228:107291. https://doi.org/10.1016/j.knosys.2021.107291
10. Wang L, Hu H, Ai XY, Liu H (2018) Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm. Energy 153:801–815. https://doi.org/10.1016/J.ENERGY.2018.04.078
11. Zhang H, Heidari AA, Wang M, Zhang L, Chen H, Li C (2020) Orthogonal Nelder-Mead moth flame method for parameters identification of photovoltaic modules. Energy Convers Manag 211: 112764. https://doi.org/10.1016/J.ENCONMAN.2020.112764
12. Zhang X, Xu Y, Yu C, Heidari AA, Li S, Chen H, Li C (2020) Gaussian mutational chaotic fruit fly-built optimization and feature selection. Expert Syst Appl 141:112976. https://doi.org/10.1016/J.ESWA.2019.112976
13. Singh P, Meena NK, Yang J, Vega-fuentes E (2020) Multi-criteria decision making monarch butterfly optimization for optimal distributed energy resources mix in distribution networks. Appl Energy 278:115723. https://doi.org/10.1016/j.apenergy.2020.115723
14. Huang T, Huang J (2008) Zhang J (2008) an orthogonal local search genetic algorithm for the design and optimization of power electronic circuits. IEEE Congr Evol Comput CEC 2008:2452–2459. https://doi.org/10.1109/CEC.2008.4631126
15. Li Z, Zhang Q (2018) A simple yet efficient evolution strategy for large-scale black-box optimization. IEEE Trans Evol Comput 22: 637–646. https://doi.org/10.1109/TEVC.2017.2765682
16. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35. https://doi.org/10.1007/S00366-011-0241-Y/FIGURES/16
17. Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput (Swansea, Wales) 29:464–483. https://doi.org/10.1108/02644401211235834/FULL/PDF
18. Gao SZ, Wang Y, Cheng J et al (2016) Ant colony optimization with clustering for solving the dynamic location routing problem. Appl Math Comput 285:149–173. https://doi.org/10.1016/J.AMC.2016.03.035
19. Tu J, Chen H, Liu J, Heidari AA, Zhang X, Wang M, Ruby R, Pham QV (2021) Knowledge-based systems evolutionary biogeography-based whale optimization methods with communication structure : towards measuring the balance. Knowledge-Based Syst 212:106642. https://doi.org/10.1016/j.knosys.2020.106642
20. Xue Y, Jiang J, Zhao B, Ma T (2018) A self-adaptive artificial bee colony algorithm based on global best for global optimization. Soft Comput 22:2935–2952. https://doi.org/10.1007/S00500-017-2547-1/TABLES/11
21. Liu W, Wang Z, Yuan Y, Zeng N, Hone K, Liu X (2021) A novel sigmoid-function-based adaptive weighted particle swarm optimizer. IEEE Trans Cybern 51:1085–1093. https://doi.org/10.1109/TCYB.2019.2925015
22. Mirghasemi S, Andreae P, Zhang M (2019) Domain-independent severely noisy image segmentation via adaptive wavelet shrinkage using particle swarm optimization and fuzzy C-means. Expert Syst Appl 133:126–150. https://doi.org/10.1016/j.eswa.2019.04.050
23. Djenouri Y, Comuzzi M (2017) Combining Apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. Inf Sci (Ny) 420:1–15. https://doi.org/10.1016/J.INS.2017.08.043
24. Hong Q, Shi Z, Sun J, Du S (2020) Memristive self-learning logic circuit with application to encoder and decoder. Neural Comput Appl 3310 33:4901–4913. https://doi.org/10.1007/S00521-020-05281-Z
25. Sudharsan B, Yadav P, Breslin JG, Intizar Ali M (2021) Train++: an incremental ML model training algorithm to create self-learning IoT devices. 97–106. https://doi.org/10.1109/SWC50871.2021.00023
26. Zhao D, Liu X, Zhao HJ, Wang C, Tang J, Liu J, Shen C (2021) Seamless integration of polarization compass and inertial

navigation data with a self-learning multi-rate residual correction algorithm. Measurement 170:108694. https://doi.org/10.1016/J.MEASUREMENT.2020.108694
27. Wang GG, Deb S, Cui Z (2019) Monarch butterfly optimization. Neural Comput Appl 31:1995–2014. https://doi.org/10.1007/S00521-015-1923-Y/TABLES/7
28. Feng Y, Deb S, Wang G, Alavi AH (2021) Monarch butterfly optimization : a comprehensive review. Expert Syst Appl 168:114418. https://doi.org/10.1016/j.eswa.2020.114418
29. Ibrahim AM, Tawhid MA (2019) A hybridization of differential evolution and monarch butterfly optimization for solving systems of nonlinear equations. J Comput Des Eng 6:354–367. https://doi.org/10.1016/J.JCDE.2018.10.006
30. Wang GG, Deb S, Zhao X, Cui Z (2016) A new monarch butterfly optimization with an improved crossover operator. Oper Res 183 18:731–755. https://doi.org/10.1007/S12351-016-0251-Z
31. Faris H, Aljarah I, Mirjalili S (2017) Improved monarch butterfly optimization for unconstrained global search and neural network training. Appl Intell 482 48:445–464. https://doi.org/10.1007/S10489-017-0967-3
32. Sun L, Chen S, Xu J, et al (2019) Improved Monarch Butterfly Optimization Algorithm Based on Opposition-Based Learning and Random Local Perturbation Complexity 2019:. https://doi.org/10.1155/2019/4182148
33. Yazdani S, Hadavandi E (2019) LMBO-DE: a linearized monarch butterfly optimization algorithm improved with differential evolution. Soft Comput 23:8029–8043. https://doi.org/10.1007/s00500-018-3439-8
34. Cui X, Chen Z, Yin F (2018) Differential evolution and local search based monarch butterfly optimization algorithm with applications. Int J Comput Intell Syst 12:149–163. https://doi.org/10.2991/IJCIS.2018.25905188
35. Ghanem WAHM, Jantan A (2018) Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. Neural Comput Appl 30:163–181. https://doi.org/10.1007/S00521-016-2665-1/FIGURES/5
36. Devikanniga D, Joshua Samuel Raj R (2018) Classification of osteoporosis by artificial neural network based on monarch butterfly optimisation algorithm. Healthc Technol Lett 5:70–75. https://doi.org/10.1049/HTL.2017.0059
37. Feng Y, Yu X, Wang GG (2019) A novel monarch butterfly optimization with global position updating operator for large-scale 0-1 knapsack problems. Mathematics 7:1–32. https://doi.org/10.3390/math7111056
38. Soltani P, Hadavandi E (2019) A monarch butterfly optimization-based neural network simulator for prediction of siro-spun yarn tenacity. Soft Comput 23:10521–10535. https://doi.org/10.1007/S00500-018-3624-9/FIGURES/13
39. Ren Z, Zhang A, Wen C, Feng Z (2014) A scatter learning particle swarm optimization algorithm for multimodal problems. IEEE Trans Cybern 44:1127–1140. https://doi.org/10.1109/TCYB.2013.2279802
40. Gong YJ, Li JJ, Zhou Y, Li Y, Chung HSH, Shi YH, Zhang J (2016) Genetic learning particle swarm optimization. IEEE Trans Cybern 46:2277–2290. https://doi.org/10.1109/TCYB.2015.2475174
41. Ingle KK, Jatoth DRK (2020) An efficient JAYA algorithm with Lévy flight for Non-Linear Channel equalization. Expert Syst Appl 145:112970. https://doi.org/10.1016/J.ESWA.2019.112970
42. Wang W, Yang S, Lin Q, Zhang Q, Wong KC, Coello Coello CA, Chen J (2019) An effective ensemble framework for multi-objective optimization. IEEE Trans Evol Comput 23:645–659. https://doi.org/10.1109/TEVC.2018.2879078
43. Zhan ZH, Wang ZJ, Jin H, Zhang J (2020) Adaptive distributed differential evolution. IEEE Trans Cybern 50:4633–4647. https://doi.org/10.1109/TCYB.2019.2944873
44. Zhao F, Zhao L, Wang L, Song H (2020) A collaborative LSHADE algorithm with comprehensive learning mechanism. Appl Soft Comput 96:106609. https://doi.org/10.1016/J.ASOC.2020.106609
45. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12:702–713. https://doi.org/10.1109/TEVC.2008.919004
46. Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform 1:355–366. https://doi.org/10.1016/J.ECOINF.2006.07.003
47. Rao RV, Saroj A (2019) An elitism-based self-adaptive multi-population Jaya algorithm and its applications. Soft Comput 23:4383–4406. https://doi.org/10.1007/S00500-018-3095-Z/TABLES/24
48. Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the time complexity of the Derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11:1–18. https://doi.org/10.1162/106365603321828970
49. Zhao F, Hu X, Wang L, Zhao J, Tang J, Jonrinaldi (2022) A reinforcement learning brain storm optimization algorithm (BSO) with learning mechanism. Knowledge-Based Syst 235:107645. https://doi.org/10.1016/J.KNOSYS.2021.107645