Swarm and Evolutionary Computation 83 (2023) 101387

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Deep reinforcement learning assisted co-evolutionary differential evolution for constrained optimization

Zhenzhen Hu [a,b], Wenyin Gong [a,*], Witold Pedrycz [b,c,d], Yanchi Li [a]

[a] School of Computer Science, China University of Geosciences, Wuhan 430074, China
[b] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada
[c] Systems Research Institute, Polish Academy of Sciences, 00-901 Warsaw, Poland
[d] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Istinye University, Sariyer/Istanbul, Turkiye

ARTICLE INFO

Keywords:
Constraint handling technique
Deep reinforcement learning
Differential evolution
Co-evolution
Evolutionary operator

ABSTRACT

Solving constrained optimization problems (COPs) with evolutionary algorithms (EAs) is a popular research direction due to its potential and diverse applications. One of the key issues in solving COPs is the choice of constraint handling techniques (CHTs), as different CHTs can lead to different evolutionary directions. Combining EAs with deep reinforcement learning (DRL) is a promising and emerging approach for solving COPs. Although DRL can help solve the problem of pre-setting operators in EAs, neural networks need to obtain diverse training data within a limited number of evaluations in EAs. Based on the above considerations, this work proposes a DRL assisted co-evolutionary differential evolution, named CEDE-DRL, which can effectively use DRL to help EAs solve COPs. (1) This method incorporates co-evolution into the extraction of training data for the first time, ensuring the diversity of samples and improving the accuracy of the neural network model through information exchange between multiple populations. (2) Multiple CHTs are used for offspring selection to ensure the algorithm's generality and flexibility. (3) DRL is used to evaluate the population state, taking into account feasibility, convergence, and diversity in the state setting and using the overall degree of improvement as a reward. The neural network selects suitable parent populations and corresponding archives for mutation. Finally, (4) to avoid premature convergence and local optima, an adaptive operator selection and individual archive elimination mechanism is added. Comparisons with state-of-the-art algorithms on benchmark functions CEC2010 and CEC2017 show that the proposed method performs competitively and produced robust solutions. The results of the application test set CEC2020 show that the proposed algorithm is also effective in real-world problems.

1. Introduction

Constrained optimization problems (COPs) are inevitable for technological development [1]. Solving COPs by intelligent algorithms has become a long-standing problem explored by researchers [2]. COPs mainly consists of one objective function to be optimized and more than one constraint (q inequality constraints and m equation constraints) that control the range of solutions. A standard COP can be described as [3]:

$$\begin{aligned} &\text{minimize } f(\mathbf{x}), \mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{S}, \\ &\text{subject to: } g_i(\mathbf{x}) \le 0, \forall i = 1, \ldots, q, \\ &\qquad\qquad h_j(\mathbf{x}) = 0, \forall j = q+1, \ldots, m, \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function and $\mathbf{x} = (x_1, \ldots, x_D)$ are the D-dimensional decision variable and the decision space $\mathbb{S} = \{\mathbf{x} \in R \mid L_i \le \mathbf{x}_i \le U_i\}$ $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are the ith inequality constraint and the jth equality constraint, respectively.

According to the No Free Lunch (NFL) theorem [4], there is no single algorithm that is superior to all algorithms to solve all problems. One of the cornerstones of solving COPs is the selection of constraint handling techniques (CHTs). Compared to unconstrained problems the constraints and objective functions of COPs are usually more complex, the population is subject to constraints and the need to find the optimal objective function value in a finite, dispersed and disconnected or dynamic feasible regions [5]. A single-minded targeting of feasible solutions may miss those individuals with certain bootstraps on the edge of feasible solutions or infeasible regions. Sometimes individuals with low constraint violation values may bring important information to the population [6,7], First, solutions with low constraint violation degree values may lie near locally optimal solutions in the space,

* Corresponding author.
E-mail addresses: zhenzhenhu@cug.edu.cn (Z. Hu), wygong@cug.edu.cn (W. Gong), wpedrycz@ualberta.ca (W. Pedrycz), int_lyc@cug.edu.cn (Y. Li).

https://doi.org/10.1016/j.swevo.2023.101387
Received 29 March 2023; Received in revised form 16 June 2023; Accepted 18 August 2023
Available online 25 August 2023

which are not globally optimal but provide strong clues. Second, in COPs, many times relaxing the constraints can lead to a larger search space. When faced with extremely complex COPs, the solution with low constraint violation value may be the best solution available, providing important information about the problem. That is why we also need to adequately consider the trade-off between constraints and objective functions. A well-balanced algorithm has to balance between diversity and convergence [8]. The treatment of constraints is particularly important in COPs, and the exploratory study of COPs reveals that the setting of CHTs relies on the different requirements of the state and problem during the search. The focus of the search varies from problem to problem and from time to time. Therefore, rational determination of CHTs is usually a very tough task.

With the development of evolutionary algorithms (EAs), the use of EAs to solve COPs has become a popular trend. EAs are search algorithms developed based on the principle of biological evolution, which is executed iteratively to find the desired solution through iterative mutation and selection of populations [9]. It mainly includes differential evolution (DE) [10], particle swarm optimization(PSO) [11], ant colony optimization (ACO) [12], genetic algorithm (GA) [13], etc. EAs have wide applicability, where DE is widely used due to its simple operating principle, robustness, and promising results in solving various variants of COPs. When dealing with complex COPs, unitary EAs with single CHT reveal their limitations, such as operators need to be artificially pre-designed, premature algorithms, and fall into local optimality [14]. Consequently, EAs need to add a population-autonomous learning component to strengthen the merit-seeking ability.

In the concept of reinforcement learning (RL) [15,16], reward maximization and fitness function optimization in DE solve the same problem. The shortcomings of using EAs to address COPs can be addressed by RL due to its autonomous learning capability. Q-Learning or strategy gradients can be used to determine the evolutionary direction, mutation strategies [17], etc. In recent years, combining RL and EAs to solve COPs has been studied more and more extensively [18]. RL is performed for discrete states, and a limited number of state settings often does not bring promising results when the constrained problems are continuous and with complex feasible regions. The combination of deep learning and RL can be an excellent solution to this problem, and the integration of RL and deep learning (named deep reinforcement learning, DRL) has led to impressive achievements in a wide range of challenging tasks [19]. In DRL, an important issue is the extraction of the training set. Neural networks require a large amount of data for training, and the training set should ensure sufficient data and favorable diversity of samples. In EAs, the information obtained by a single population using a limited number of evaluations is not enough for the training of neural networks, but consuming a large number of evaluations reduces the competitiveness of EAs. Therefore, the aid of co-evolution needs to be added to ensure the data diversity of the training set by co-evolving multiple populations to ensure that enough information is obtained with a limited evaluation system. Also the introduction of group collaboration and information exchange in the face of different CHTs can also enhance the search capability of the algorithm, thus in order to better deal with COPs.

Based on the above state-of-the-research and problematic areas, we propose a co-evolution DE based on DQN, denoted as CEDE-DRL. The main contributions of this study are summarized as follows:

- By incorporating co-evolution, multiple populations can ensure diversity of the training data set, thereby improving the accuracy of neural network models. Additionally, using different CHTs in different populations can enhance the versatility of the algorithm, making it applicable to a wider range of problems.
- DRL is used to evaluate the potential future rewards of each population based on the current evolutionary state and select the most suitable population and corresponding archive as the parents to generate offspring, thus obtaining the most useful set of new solutions.
- The feasibility, convergence, and diversity of sub-populations are taken into account in the state setting of DRL. The exploration and exploitation of the algorithm are ensured. Rewards are evaluated based on the overall improvement of the population, which improved the reliability of rewards.
- Compared with state-of-the-art algorithms in recent years, the proposed method can achieve satisfactory results on two COP benchmark suites and 33 real-world COPs.

In short, the originality of this paper lies in combining co-evolution with DRL to solve COPs. The rest of the paper is organized as follows. In Section 2, the preliminary knowledge and related work are described, including DQN, co-evolution and CHTs. Section 3 elaborates on the proposed CEDE-DRL algorithm. In Section 4, the experimental results are presented and analyzed. Finally, conclusions are given in Section 5.

2. Preliminary and related work

2.1. Deep Q-Network (DQN)

DQN is the combination of Q-learning and neural networks. Deep learning has a strong capability of prediction while RL has a decision-making capability. Combining the two therefore allows for complementary strengths. The DQN was developed by DeepMind in 2015 [20], which is a method that combines neural networks and Q-learning. DQN is the replacement of a value function estimator with a deep neural network to handle a high-dimensional, complex state space. The core idea of DQN is to estimate the Q-value for each action in each state via a deep neural network. DQN also uses a technique called experience replay, which stores experiences in a replay buffer and randomly samples from it to train the neural network. This helps to improve the stability and efficiency of learning [21]. DQN is about finding optimal actions in finite or even infinite dynamic problems, while evolutionary computation is about finding optimal solutions in statics. The key to adding DQN to evolutionary computation then is how to make the intelligence find feasible optimal solutions in dynamics [22]. As shown in Fig. 1, the first step is to initialize the network and output the $state_t$, output all Q values in that $state_t$ through the neural network, select an action using a strategy such as ϵ-greedy, and invoke this action in the environment to obtain a new $state_{t+1}$ and $reward$. The loss function (Eq. (2)) is then calculated and the network is updated. The loop repeats until the end of the iteration.

The training of a neural network is an optimization problem where the loss function represents the difference between the network output and the labeled value, and the objective is to minimize the loss function.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')}[(r + \gamma max\boldsymbol{Q}(s', a', \theta) - \boldsymbol{Q}(s, a, \theta))^2], \tag{2}$$

where γ is the discount factor between 0 and 1, θ and θ' represent the weight of the predicted value and target value respectively. $r + \gamma max\boldsymbol{Q}(s', a', \theta)$ is the TD (temporal difference) target, which represents the $reward$ obtained by this step plus the maximum Q value that can be obtained in the next $state$, and they subtract the Q value of this step, which is the discount sum of future rewards.

In the current research on combining DRL with EAs, most of the algorithms address operator selection in the optimization process of EAs [23]. Sharma et al. [24] proposed a double DQN-based Adaptive Operator Selection (AOS). Neural networks were trained offline to control the mutation strategies in DE by collecting data and the reward of applying each mutation strategy over multiple runs. Tan [25] proposed a DE based on DQN (DEDQN) that uses DQN to select mutation strategies from a mixed pool of evolutionary strategies. DQN trains the test function by collecting data related to the fitness landscape and the risk-return corresponding to the various mutation strategies. After that the mutation strategy for each generation is predicted based on the fitness value. To address the dilemma of the relationship between

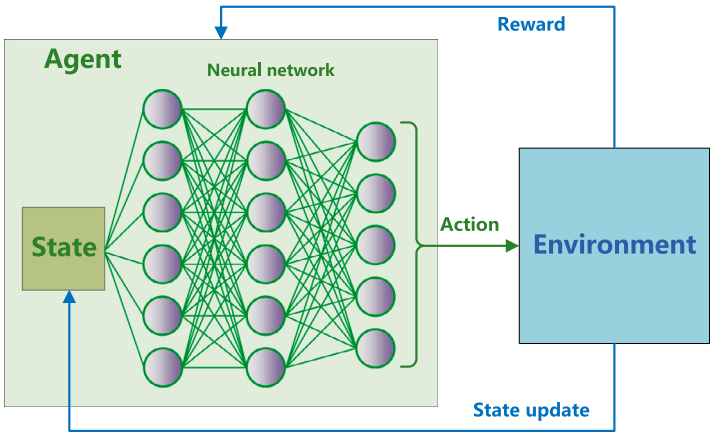


Fig. 1. The framework of DQN.

exploration and exploitation in operator selection, a new method for operator selection based on RL is proposed [22]. The method treats decision variables as states and candidate operators as actions. By using a DQN to learn a strategy for estimating the Q-value of each action in a given state, the proposed method can determine the best operator for each parent population and maximize its cumulative improvement.

Since Q-learning is an off-policy offline learning method that learns both current and past historical experiences, the past archives can be randomly selected for learning each time the DQN is updated. Therefore, in order to better train the neural network, it requires a high diversity of data to be extracted. So we have used multiple population co-evolution in our algorithm to avoid homogeneity of populations and to ensure diversity in the training set. In our proposed method, we choose discrete actions by setting up continuous states, so that the performance of the DQN can be invoked well. Today there are also many evolutionary algorithms aided by Q-learning [26], but the use of DRL is mostly for algorithmic parameter tuning, which is different from what we are doing. In Section 3, the proposed method as well as CEDE-DRL are elaborated on.

2.2. Co-evolution

Co-evolution was proposed by American ecologists Ehrlich and Raven in 1964 and refers to the self-evolution of traits of one species in response to those of another species [27]. It is a method of improving overall competitiveness through the interaction of individuals in multiple populations. Co-evolution is the achievement of higher levels of evolution through the interaction and evolution of different populations. It is a strategy for improving overall evolutionary efficiency through interactions and collaboration between populations. In co-evolution, different populations evolve collaboratively by interacting and adapting to produce better solutions [28].

Co-evolution is often applied to solve complex optimization problems, for example, when designing complex systems, co-evolution can help different subsystems to coordinate with each other, thus improving the performance and stability of the whole system. Co-evolution can also be used in the field of artificial intelligence, for example in the optimization of neural networks, where co-learning between different neural networks can be used to improve the performance of the whole system [29]. Co-evolution is used to assign fitness through the interaction of solutions with other solutions, and was originally used as an alternative solution to the problem of difficult to construct fitness functions that can be used in optimization-based EAs, where the search direction is guided by competition in the population, and these interactions provide the basis for improvement in fitness values, which will also influence the selection process between populations as fitness values are ranked and subsequent mutation [30].

Co-evolutionary CHT was used to deal with a constrained single-objective optimization problem [31]. It described a method for adjusting the penalty factor in the fitness function in a GA through co-evolution. Co-evolution can also be used in DE, where multiple populations evolve in concert to balance the objective function and constraints by assigning each population an independent penalty factor [32].

2.3. CHTs

The superiority of feasible solutions (SF) is a simple and effective criterion proposed by Deb [31] in 2000, mainly uses the following three criteria to compare two individuals. (1) Feasible individuals and infeasible individuals, the feasible solution dominates; (2) If both are feasible individuals, the individual with better objective value is dominant; (3) If both are infeasible individuals, the individual with the smallest constraint violation value is dominant. Due to its simple principle and convenient operation, the SF can quickly make the population converge to the feasible region, and is widely used in CHTs. However, due to paying too much attention to the solution in the feasible region, it is easy to ignore the exploration of the boundary area of the feasible region. In the work of Zielinski [33], a CHT based on a modified selection procedure is applied in DE without additional parameters and information about the number of constraint violations is used to guide individuals to feasible regions. Venkatraman et al. [34] used a two-stage framework to use GA for solving COPs. In the first stage the search is performed only for the constraint to find a feasible solution. The individual with the least constraint violation is considered as the elite individual. In the second stage the optimization of the objective function and the satisfaction of the constraint is considered as a bio-objective optimization problem and the non-dominated ranking among individuals is found to be beneficial for optimizing the performance of the algorithm. The procedure of SF is simple and the convergence speed is fast, but at the same time, the search direction is relatively single and it is easy to ignore the information of infeasible regions, causing the population to fall into local optimum.

ϵ-constraint is a CHT proposed by Takahama et al. [35]. The core idea is to take the individual whose constraint violation degree is less than ϵ as a feasible solution, through a pre-set ϵ value, so as to make full use of the information of better infeasible solutions in the infeasible region and improve the convergence of the algorithm. The value of ϵ changes with the deepening of the evolution process, and the following function is usually used:

$$\epsilon(g) = \begin{cases} \epsilon(o)(1 - G/T_c)^{cp}, & 0 < g < T_c \\ 0, & g \geq T_c \end{cases} \tag{3}$$

where cp is a parameter that controls the rate at which ϵ decrease, g is the evolutionary number of generations and Tc controls when ϵ is changed to 0.

Fan et al. [36] proposes an improved ϵ-constraint method to embed into LSHADE44 to solve COPs, named LSHADE44-IEpsilon, which can adjust the value of ϵ adaptively according to the percentage of feasible solutions in the population, and therefore can solve the search imbalance between feasible and infeasible regions in the evolutionary process. Domínguez et al. [37] proposed a DE algorithm for adaptive local search coordination using e as a CHT. A library of local search operators is added directly to the conventional DE. ϵ-constraint can introduce information about the better individual of the objective function value in the infeasible solution by the adjustment of ϵ. However, there are also limitations because the value of ϵ needs to be set and adjusted artificially.

Another popular CHT is the transformation of COPs into multi-objective optimization problems (MO-CHT), which is to transform the constraints into other objective functions: $(f(\vec{x}), G(\vec{x}))$ or $(f(\vec{x}), G_1(\vec{x}), \ldots, G_n(\vec{x}))$. Wang et al. [38] proposed a novel EA based on a multi-objective technique that replaces individuals in the parent if they are dominated by non-dominant individuals in the offspring. Zhou et al. [39] proposed a tri-goal evolutionary framework that takes into account convergence and diversity, respectively, and translates constraints as a feasibility consideration into a third indicator, using a multi-objective concept for COPs. Peng et al. [40] proposed a CHT using biased dynamic weights, where biased weights are used to select different individuals with low objective function values and low constraint violations, and the algorithm is tilted toward individuals with low objective function values and low constraint violations by adjusting the weights to ensure convergence and diversity. DeCODE [41] is the transformation of a COP into a bio-objective optimization problem, which is then decomposed into several scalar optimization sub-problems and selected using a weighted sum method in DE. The multi-objective optimization method can avoid the equilibrium problem of constraint and objective function, but its solution difficulty is high, and its performance advantage is insufficient in the face of multi-constraint problems transformed into super-multi-objective algorithms. Wang et al. [42] proposed an algorithm for single-objective constrained problems called CORCO. In CORCO, a method is proposed to explore the correlation between constraints and objective functions and to generate correlations into exponential mergers into single objective problems. The relationship between the objective and the constraint is explored through a previous learning phase, while archiving and replacement mechanisms are added. The previously learned correlations are used in a subsequent phase for offspring selection and to guide evolution. In summary, we combine the four CHTs (SF, ϵ-constraint, MO-CHT and CORCO-CHT) and present the proposed algorithm in detail in the next sections.

3. Proposed method

In this section, we present the proposed algorithm (CEDE-DRL) in detail.

3.1. CEDE-DRL evolutionary process

There are four sub-populations in total, corresponding to four archives (for the preservation of eliminated individuals). As shown in Fig. 2, each population $(population_1^t, population_2^t, population_3^t, population_4^t)$ matching one CHT $(CHT_1, CHT_2, CHT_3, CHT_4)$, and when mutation is carried out, at the tth iteration, the mutation strategy is judged among the three strategies based on the success rate of each previously chosen strategy. By the action of the DQN output, one population $(population_i^t)$ is selected by the neural network, combined with the corresponding archival population $(Archive_i^t)$ to become a union population $(Union_i^t)$, and then mutated to produce the offspring population $(offspring_i^t)$ with size Np. In the selection section, according to the corresponding CHT, by comparing the original population $(population_i^t)$ with the offspring $(offspring_i)$ to update each sub-populations $(population_1^{t+1}, population_2^{t+1}, population_3^{t+1}, population_4^{t+1})$ at iteration $t+1$. Individuals

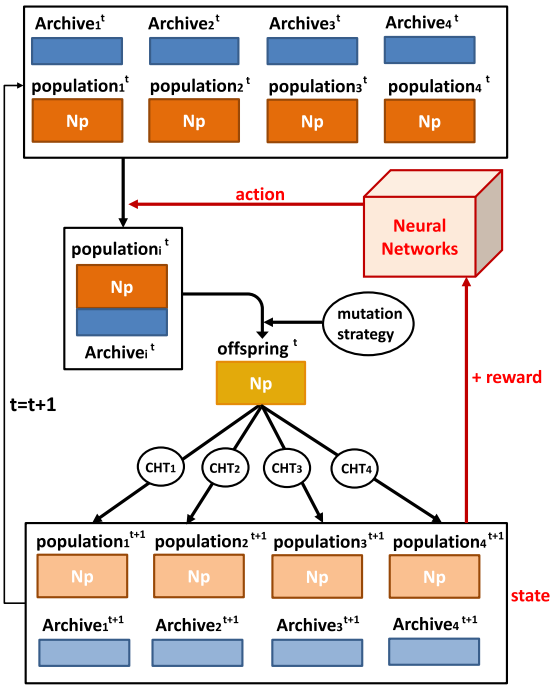


Fig. 2. The framework of population evolution.

eliminated in the competition are saved in four archives to increase overall diversity and avoid rapid convergence of populations to local optima. Algorithm 1 demonstrates the pseudocode of CEDE-DRL. In the initialization process, population $P_o = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ randomly generated from the search space, while the corresponding archive populations A_o is empty. In order to increase the sample diversity and ensure the training data set capacity, instead of using DQN for action selection in the first 2000 iterations at the beginning, one of the four actions is randomly selected. After the random selection, action selection is performed based on the neural network. According to the four CHTs (SF, ϵ-constraint, MO-CHT, CORCOR-CHT) mentioned in Section 2.3, updates are made according to different parameter requirements based on the population information. The selected population is merged with its archive into a parent population (*union*), and then the mutation strategy selected according to the success rate of each strategy is mutated to generate offspring. The rewards of the agent are calculated according to Eq. (8) and $(ave\_f, ave\_cv, ave\_dis), (ave\_f1, ave\_cv1, ave\_dis1)$ denote the respective sets of states of the population and the offspring, respectively. The training set data of the neural network is updated. Then comes the co-evolution part, which is in Algorithm 2, where the four populations select offspring according to different CHTs. It is worth mentioning that to save computational time cost and avoid frequent training, the network is updated every 500 generations after the neural network is constructed. It is repeated until the iteration is completed.

3.2. Co-evolution of multiple populations

In this work, four populations undergo co-evolution and information can be passed between individuals in each population, thus providing a better direction for subsequent evolution. As shown in Algorithm 2, DRL produces an offspring of Np size (OP^t) after selecting a union population as the parent and mutating it among the four populations $(P_{1,2,3,4}^t)$. In the selection stage, this offspring (OP^t) is selected with each of the four populations, according to their different CHTs, and the new four populations $(P_{1,2,3,4}^{t+1})$ are selected according to the corresponding rules. This is used as the parent population for the new iteration. Through co-evolution among the four populations, evolution

Algorithm 1: The framework of CEDE-DRL

Input: Np, $CHTnum$, parameters of Deep Q-network
Output: The optimal solution: P_{best}

Initialize the population P_i^0 (i= 1,2,3,4) and Deep Q-network randomly;
Initializes the parameters of CHTs;
Initializes the $archive_i$ (i= 1,2,3,4);
while *Termination condition not met* **do**
  if $Gen \leq 2000$ **then**
    Action = randi($CHTnum$)
  else
    Build the model;
    if $rand > greedy$ **then**
      Action = randi($CHTnum$)
    else
      Action ⟵ Deep Q-network;
  Update parameters of four CHTs;
  Mutation strategy ⟵ success rate;
  $union = [population, archive]$;
  Generate offspring $OP = \{\boldsymbol{u}_1^t, \ldots, \boldsymbol{u}_{Np}^t\}$ according to the chosen mutation strategy;
  Calculate Reward based on Eq. (8);
  Update the state = $[ave\_f, ave\_cv, ave\_dis, current\_at, R, ave\_f1, ave\_cv1, ave\_dis1]$;
  Data = [Data;$state$];
  Update P_i (i= 1,2,3,4) and $archive_i$ (i= 1,2,3,4) according to Algorithm 2;
  if *the model is built* **then**
    Update Q-net model every 500 generations;
  return P_{best} ;

can be performed under the consideration of four CHTs, which gets rid of the limitation brought by a single CHT and makes the algorithm generalizable. Meanwhile the offspring generated by mutation can pass the information in the four parent populations to each other through selection to influence the individuals in other populations, thus improving the overall evolutionary direction of the algorithm in the search region. Elimination selection based on different criteria also ensures the exploration ability of the algorithm, with those selected down into the next generation and those eliminated stored in the corresponding archives.

Algorithm 2: Co-evolution of multiple populations

Input: Np, $CHTnum$, P_{ch}^t (ch= 1,2,3,4), $archive_{ch}^t$ (ch= 1,2,3,4), OP^t
Output: P_{ch}^{t+1} (ch= 1,2,3,4), $archive_{ch}^{t+1}$ (ch= 1,2,3,4)

for $ch = 1 : CHTnum$ **do**
  for $i = 1 : Np$ **do**
    $\boldsymbol{x}_{i,ch} \in P_{ch}^t$ and $\boldsymbol{u}_i \in OP^t$;
    if $\boldsymbol{u}_i$ *is better than* $\boldsymbol{x}_{i,ch}$ *according to CHTi* **then**
      replace $\boldsymbol{x}_{i,ch}$ by $\boldsymbol{u}_i$
      $P_{ch}^{t+1} = P_{ch}^t$
      Store the eliminated individuals in $archive_{ch}^{t+1}$
return P_{ch}^{t+1} (ch= 1,2,3,4), $archive_{ch}^{t+1}$ (ch= 1,2,3,4) ;

3.3. Model design for DRL

This subsection focuses on the design of the various components of the DQN in our proposed algorithm. Since the DQN is applicable to the continuous space problem, we choose to set the state as shown in the following:

$$state = \{ave\_obj,\ ave\_conV,\ ave\_dis\}, \tag{4}$$

$$ave\_obj = \frac{\sum_{i=1}^{Np} f(x_i)}{Np}, \tag{5}$$

$$ave\_conV = \frac{\sum_{i=1}^{Np} conV(x_i)}{Np}, \tag{6}$$

$$ave\_dis = \frac{\sum_{i=1}^{Np} \sqrt{\sum_{i=1}^{D} (x_k^i - \sum_{j=1}^{Np} x_j^i)}}{Np}, \tag{7}$$

ave_obj, ave_conV, ave_dis denote the average objective function value of all individuals in the population, the average constraint violation value, and the average Euclidean distance between all individuals, respectively. As shown in Eqs. (5)–(7). Where Np and D are the number and dimension of the population, respectively.

In order to balance the convergence and exploratory of the population, we have to consider both the overall population toward the feasible region and also those individuals whose constraint values are small and near the feasible domain, so the state contains both the objective function and the constraint violation. In general the improvement of function fitness values and the maintenance of population diversity are in conflict with each other, so the effective balance between the two has important implications for ensuring the global convergence of the algorithm. Finally, in order to avoid the whole population converging too quickly and falling into a local optimum, the diversity of the population should also be ensured, so we add the average Euclidean distance between individuals to the state as a tabulation to examine the diversity of the population.

For the action settings, each population with a different CHT is treated as an action, so that each of the four actions corresponds to four populations, which are constrained using SF (The superiority of feasible solutions [31]), ϵ-constraint [35], MO-CHT (DeCODE [41]) and CORCO-CHT (CORCO [42]) respectively. In order to take into account the change of the overall population, the reward is set as:

$$reward = \frac{num\_better}{Np}, \tag{8}$$

This means the sum of the number of individuals that became better after each mutation divided by the total population.

The inputs to the DQN are the states and actions of the population (state, action), and the outputs are the Q values of all actions, leading to action selection based on a greedy strategy, which is the parent population selection.

The network contains two hidden layers and dropout. Since DE is solved with a limited number of evaluations, the use of dropout can reduce overfitting in the case of less data [43], and using dropout can reduce the time used to train the network and reduce the algorithm time cost. DQN requires a set that holds the historical data of the trained neural network to break down the correlation between the data, called experience replay (EP), recorded as follows:

$$Data = \left\{f, cv, d, a, r, f', cv', d'\right\}, \tag{9}$$

where f, cv, d represents the $state$ of the agent, a is the action performed by the agent, r is the reward calculated by Eq. (8), and the last three components represent the next $state$ of agent.

3.4. Mutation strategy selection

In order to balance the convergence and exploratory nature of the algorithm, this paper uses three mutation operators with different properties. The next round of strategy selection is based on the success rate of each strategy. $DE/current-to-pbest/1$ makes the population exploratory and the inclusion of an archive prevents the population from falling into local optimum. $DE/rand/1$ ensures that the population is diverse and enhances the ability to find the optimal individual. $DE/rand{-}to{-}pbest/1$ ensures that the population has strong exploitation capability. Archiving has been added to both DE/current-to-rand/1 and DE/rand-to-pbest/1, which can preserve some promising solutions during the iterations to avoid their loss in subsequent evolution, and can also add some diversity to the overall population evolution, which can be used as a proper reference. Three DE strategies of agent (population) are expressed as:

- $DE/current-to-pbest/1$

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + F \cdot \left(\boldsymbol{x}_{r_1}^t - \boldsymbol{x}_{pbest}^t\right) + F \cdot \left(\boldsymbol{x}_{r_2}^t - \hat{\boldsymbol{x}}_{r_3}^t\right), \tag{10}$$

Table 1
Experimental results of C2oDE, LSHADE44, FROFI, DeCODE, CORCO and CEDE-DRL on IEEE CEC2010 test functions (30D) with 30 independent runs.

	C2oDE	LSHADE44	FROFI	DeCODE	CORCO	CEDE-DRL
CEC10-F1	−8.2042e−01 (2.25e−03) +	−8.0089e−01 (1.53e−02) =	−8.1042e−01 (5.35e−03) =	**−8.2060e−01 (1.97e−03) +**	−3.9578e−01 (2.68e−02) −	−8.0923e−01 (1.17e−02)
CEC10-F2	4.5849e−01 (4.92e−01) −	2.9329e+00 (6.22e−01) −	−2.1508e+00 (5.30e−02) −	−2.1962e+00 (3.13e−02) −	−2.1422e+00 (5.79e−02) −	**−2.2771e+00 (2.60e−03)**
CEC10-F3	2.8680e+01 (3.02e−03) −	1.4554e+13 (1.09e+13) −	2.6104e+01 (8.66e+00) −	2.2939e+01 (1.18e+01) −	**3.7729e−23 (5.90e−25) +**	1.1843e−22 (2.70e−22)
CEC10-F4	1.0955e−02 (1.57e−02) −	3.2834e−01 (3.77e−01) −	NaN (NaN) −	**−3.3333e−06 (5.95e−11) +**	1.7435e−05 (3.78e−05) =	2.2430e−03 (5.68e−03)
CEC10-F5	1.2967e+02 (2.52e+02) −	5.4384e+02 (4.11e+01) −	−4.8341e+02 (1.06e−01) −	−4.8346e+02 (1.11e−01) −	−4.4180e+02 (4.80e+01) −	**−4.8361e+02 (4.93e−04)**
CEC10-F6	−3.2494e+01 (1.48e+02) −	5.8837e+02 (2.05e+01) −	**−5.3055e+02 (9.86e−02) +**	−5.1696e+02 (3.81e+00) −	−5.1696e+02 (3.81e+00) −	−5.3013e+02 (3.46e−01)
CEC10-F7	8.2120e−12 (2.03e−11) −	3.9866e−01 (1.23e+00) −	1.5467e+04 (1.24e+04) −	1.9933e−01 (8.91e−01) −	**8.5012e−25 (9.71e−30) +**	1.5086e−24 (8.38e−25)
CEC10-F8	1.1119e−05 (1.97e−05) +	2.4292e+01 (3.46e+01) +	3.3585e+01 (1.87e+01) −	**1.2093e−24 (2.09e−25) =**	4.5901e+00 (2.05e+01) +	2.7827e+01 (7.61e+01)
CEC10-F9	4.5571e+12 (4.74e+12) −	4.3821e+13 (1.94e+13) −	3.7474e+00 (1.58e+01) −	2.4743e+01 (3.13e+01) −	**4.8403e−24 (2.22e−25) +**	3.5779e+00 (1.02e+01)
CEC10-F10	2.5874e+12 (1.57e+12) −	3.4054e+13 (1.29e+13) −	3.1349e+01 (9.07e−03) −	3.1309e+01 (1.22e−05) −	**6.6631e−01 (2.98e+00) +**	6.6967e+00 (1.52e+00)
CEC10-F11	NaN (NaN) −	−7.9766e−05 (9.36e−05) −	NaN (NaN) −	**−3.9234e−04 (1.66e−10) +**	−3.9234e−04 (2.56e−11) +	−3.9231e−04 (1.35e−07)
CEC10-F12	−1.9345e−01 (4.19e−03) −	3.2221e−02 (2.11e−01) −	NaN (NaN) −	**−1.9926e−01 (7.98e−09) =**	**−1.9926e−01 (9.42e−06) =**	**−1.9926e−01 (2.96e−08)**
CEC10-F13	−3.4048e+01 (2.25e+00) −	−3.1432e+01 (2.54e+00) −	−6.3411e+01 (2.15e+00) −	−6.7831e+01 (1.39e+00) −	−6.7416e+01 (0.00e+00) −	**−6.8217e+01 (6.51e−01)**
CEC10-F14	3.4651e+09 (1.24e+10) −	8.1666e+13 (6.99e+13) −	4.5387e+02 (8.53e+02) −	9.8154e−23 (1.14e−22) −	**1.0498e−23 (1.93e−25) +**	**1.0863e−23 (1.75e−23)**
CEC10-F15	3.6846e+13 (2.09e+13) −	2.2808e+14 (7.06e+13) −	2.8043e+01 (1.02e+00) −	2.1603e+01 (7.50e−07) −	**1.0716e−23 (3.37e−25) +**	9.9558e−21 (2.10e−20)
CEC10-F16	8.0789e−01 (1.71e−01) −	1.1022e+00 (1.49e−02) −	**0.0000e+00 (0.00e+00) =**	**0.0000e+00 (0.00e+00) =**	**0.0000e+00 (0.00e+00) =**	7.9727e−03 (3.57e−02)
CEC10-F17	3.4505e+02 (3.40e+02) −	1.1687e+03 (1.77e+02) −	9.2685e−02 (1.57e−01) +	**4.0686e−02 (1.19e−01) +**	2.7047e−01 (3.50e−01) −	5.5754e−01 (1.27e+00)
CEC10-F18	7.0987e+03 (3.29e+03) −	2.4277e+04 (2.94e+03) −	1.7269e−06 (5.25e−06) −	**1.8917e−25 (5.67e−25) +**	2.6929e+03 (1.01e+04) −	2.9975e−10 (5.31e−10)
+/−/=	2/16/0	1/16/1	2/14/2	5/10/3	8/7/3	Base

Table 2
Friedman's test ranking of CEDE-DRL and other methods on 18 benchmark functions in IEEE CEC2010 (30D).

Algorithm	Ranking	p-value
CEDE-DRL	**2.1389**	
DeCODE	2.5833	0.476033
CORCO	2.8611	0.24681
FROFI	3.7778	0.008587
C2oDE	4.2778	0.000604
LSHADE44	5.3611	0

Table 3
Wilcoxon signed-rank test results for CEDE-DRL and the other five compared methods on 18 benchmark functions in IEEE CEC2010 (30D).

CEDE-DRL VS	R^+	R^-	p-value	$\alpha = 0.1$	$\alpha = 0.05$
C2oDE	160.0	11.0	1.09E−03	YES	YES
LSHADE44	164.0	7.0	5.81E−04	YES	YES
FROFI	154.0	16.5	2.34E−03	YES	YES
DeCODE	119.0	51.5	0.132971	NO	NO
CORCRO	100.5	70.5	0.499656	NO	NO

- $DE/rand/1$

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + F \cdot \left(\boldsymbol{x}_{r_2}^t - \boldsymbol{x}_{r_3}^t \right), \tag{11}$$

- $DE/rand - to - pbest/1$

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + \text{rand} \cdot \left(\boldsymbol{x}_{best}^t - \boldsymbol{x}_{r_1}^t \right) + F \cdot \left(\boldsymbol{x}_{r_2}^t - \hat{\boldsymbol{x}}_{r_3}^t \right), \tag{12}$$

$$\boldsymbol{u}_{i,j} = \begin{cases} \boldsymbol{v}_{i,j}, & \text{if (rand} \le \text{CR) or } \left(j = j_{\text{rand}} \right) \\ \boldsymbol{x}_{i,j}, & \text{otherwise} \end{cases} \tag{13}$$

where $r1$ $r2$ and $r3$ are mutually different integers randomly selected from the range $[1, Np]$, t is the current generation number, x_{best}^t is the best individual in the population at generation t. F is the scaling factor used to control the difference vector and $rand$ is a uniformly distributed random number between 0 and 1, $j = 1, \ldots,$ D, CR is the crossover constant. $\hat{\boldsymbol{x}}_{r_3}^t$ is chosen randomly from the union of population and archive.

Different operators have different advantages, so how to select different mutation operators according to different features in the evolutionary process. In this work, we use the historical empirical probability for selection, calculate the selection probability by the accumulation of each success, and then decide whether to select the current operator based on the roulette approach. Thus the success rate of the mth mutation strategy at each iteration is shown below.

$$q_m = \frac{n_m + n_0}{\sum_{i=1}^{M} (n_i + n_0)}, \tag{14}$$

where n_m is the total number of successful individuals for the mth DE strategy. n_0 is set to 3 to ensure that each strategy has a certain selection probability. M is the total number of mutation strategies.

3.5. Computational complexity

According to the algorithm, the time complexity of the proposed CEDE-DRL is mainly composed of operator selection, neural network training and co-evolution to update the offspring. The operator selection part has a time complexity of $o(N \cdot log(N))$ because only one offspring is generated in each loop and the population size is N. The number of populations is M, so the time complexity of co-evolution for population update is $o(M * N)$, and since M differs significantly compared to N, M can be considered almost constant, so the time complexity of the co-evolution part is $o(N)$. The training data set size is T, the neural network has L layers, the number of neurons per layer is a, and the number of training iterations is K. The training time complexity of the DQN algorithm is $O(TLK * a^2)$, where a^2 denotes the number of connections between neurons. Since the number of layers is a smaller constant, the complexity is $o(TKa^2)$. So the time complexity of the whole algorithm combining the three parts is $o(N \cdot log(N)) + o(TKa^2)$.

4. Experimental results and analysis

4.1. Experimental settings

The tables show the average minimum objective function value (variance) in the feasible solution. In order to detect significant differences between the two algorithms, we performed the Friedman's test and the Wilcoxon signed-rank test via KEEL software [44] at a significance level of 0.05.

In Tables 1, 4, 7, 10 and 13, "NaN" indicates that a feasible solution could not be found with 30 consecutive runs. "-", "+" and "≈" indicate that the corresponding competitor performed worse, better, and similarly in the Wilcoxon's rank sum test/t- test than CEDE-DRL did in the test, respectively. The best ranking for each row is highlighted in the table. All experiments were performed in the open-source toolkit MTO-Platform.[1]

Benchmark problems: In order to demonstrate the performance of CEDE-DRL, three sets of benchmark test functions were employed. The first set includes 18 test functions from IEEE CEC2010 [45]. IEEE CEC2017 [46] was selected as the second benchmark test set. Additionally, we selected the first 33 real-world optimization problems in CEC2020 [47] for the algorithm performance test. These 33 problems

[1] https://github.com/intLyc/MTO-Platform

Table 4
Experimental results of C2oDE, LSHADE44, FROFI, DeCODE, CORCO, Cϵ-LDE and CEDE-DRL on IEEE CEC2017 test functions (50D) with 30 independent runs.

	C2oDE	LSHADE44	FROFI	DeCODE	CORCO	Cϵ-LDE	CEDE-DRL
CEC17-F1	5.1890e−12 (5.10e−12) −	**1.4587e−25 (1.39e−25) +**	1.1342e+02 (1.47e+02) −	2.8648e−20 (3.42e−20) −	8.9134e−22 (1.45e−21) =	3.93e−25(9.25e−25)+	1.5604e−21 (2.33e−21)
CEC17-F2	8.4952e−12 (1.31e−11) −	4.2310e+04 (5.51e+03) −	8.4494e+03 (1.15e+03) −	1.2428e−20 (3.49e−20) −	7.4264e−22 (1.99e−21) −	**5.09e−24 (2.27e−23)+**	1.5188e−22 (1.64e−22)
CEC17-F3	9.1773e+04 (5.27e+04) −	1.8261e+06 (1.62e+06) −	1.8185e+04 (3.70e+04) −	1.4599e+03 (6.35e+02) −	**4.1332e−22 (4.14e−22) +**	1.71e+02(1.42e+02)−	2.9929e+00 (1.50e+01)
CEC17-F4	7.1200e+01 (2.21e+01) =	**1.3573e+01 (5.86e−14) +**	8.6781e+01 (3.14e+01) −	1.5319e+01 (1.63e+00) +	1.9226e+01 (1.44e+01) +	1.66e+02(3.17e+01)−	6.4591e+01 (1.27e+01)
CEC17-F5	6.8429e−04 (1.21e−03) −	1.1739e+05 (2.70e+04) −	3.0494e+02 (1.80e+02) −	4.7839e−01 (1.32e+00) −	1.5946e−01 (7.97e−01) =	4.78e−01(1.32e+00)−	**9.2847e−26 (1.63e−25)**
CEC17-F6	2.1852e+03 (0.00e+00) −	8.6523e+03 (1.93e+03) =	NaN (NaN) −	7.0896e+02 (8.69e+01) −	**3.2852e−07 (1.63e−06) +**	2.76e+02(4.60e+01)−	2.0429e+02 (2.77e+02)
CEC17-F7	−5.0209e+02 (1.34e+02) −	−2.7428e+02 (1.54e+02) −	**−2.2564e+03 (2.04e+01) +**	−3.5321e+01 (1.74e+02) −	−2.6580e+01 (2.04e+02) −	−1.38e+03(9.84e+01)−	−1.9823e+03 (6.20e+02)
CEC17-F8	3.8345e−03 (5.08e−04) −	6.1162e−03 (1.19e−03) −	−1.5336e−05 (8.09e−05) +	**−1.3327e−04 (1.32e−06) +**	−1.1725e−04 (1.24e−05) +	−1.30e−04(7.22e−06)+	6.6445e−04 (9.27e−04)
CEC17-F9	2.9411e+00 (9.32e−01) −	1.3258e+01 (1.87e+00) −	**−2.0362e−03 (1.21e−06) +**	−1.9916e−03 (9.97e−05) −	NaN (NaN) =	3.79e−02(2.00e−01)−	−2.0089e−03 (2.57e−05)
CEC17-F10	1.1452e−03 (1.05e−04) −	1.8507e−03 (4.45e−04) −	1.9942e−05 (1.53e−05) −	**−4.8162e−05 (9.91e−08) +**	−4.7355e−05 (7.55e−07) =	−4.80e−05(6.86e−07)+	−2.1988e−05 (1.51e−05)
CEC17-F11	NaN (NaN) =	**−4.7349e−01 (9.43e−01) +**	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN(NaN)=	NaN (NaN)
CEC17-F12	1.8534e+02 (2.01e+01) −	2.4411e+02 (2.35e+01) −	**4.1319e+00 (7.21e−01) +**	1.2787e+01 (7.55e+00) −	8.9404e+00 (6.36e−01) −	6.50e+00 (4.94e+00)−	6.4562e+00 (1.26e+00)
CEC17-F13	2.5450e+03 (5.46e+02) −	3.0678e+03 (6.99e+02) −	4.3371e+01 (2.52e−01) −	7.9579e−11 (1.43e−10) −	5.7127e−24 (3.79e−24) −	6.23e+01(7.80e+01)−	**1.7842e−24 (1.81e−24)**
CEC17-F14	1.5970e+00 (6.55e−02) −	1.5997e+00 (2.52e−02) −	1.1006e+00 (1.27e−03) −	**1.1000e+00 (4.49e−15) +**	1.5900e+00 (1.58e−01) −	1.10e+00(1.05e−02)=	1.1000e+00 (4.82e−16)
CEC17-F15	1.3289e+01 (2.24e+00) −	1.7310e+01 (2.45e+00) −	5.7491e+00 (8.70e−01) −	6.5030e+00 (1.50e+00) −	4.4925e+00 (5.27e+00) −	1.37e+01(3.85e+00)−	**2.4818e+00 (6.28e−01)**
CEC17-F16	1.7021e+02 (2.75e+01) −	2.6176e+02 (2.20e+01) −	6.2831e+00 (4.83e−06) −	6.2831e+00 (8.42e−06) −	**2.0961e−13 (0.00e+00) =**	5.97e+00(8.32e+00)−	**2.0961e−13 (0.00e+00)**
CEC17-F17	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC17-F18	4.4029e+01 (0.00e+00) −	4.1209e+01 (6.94e+00) −	NaN (NaN) −	3.7526e+01 (3.65e+00) −	NaN (NaN) −	3.65e+01(7.65e−06)=	**3.6476e+01 (3.83e−03)**
CEC17-F19	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC17-F20	4.5057e+00 (8.48e−01) +	**2.4457e+00 (3.35e−01) +**	5.5446e+00 (1.22e+00) =	5.3296e+00 (1.64e+00) =	6.9080e+00 (1.71e+00) −	3.30e+00(4.62e−01)+	5.6165e+00 (7.00e−01)
CEC17-F21	1.8922e+02 (2.00e+01) −	2.5758e+02 (1.69e+01) −	**7.2781e+00 (5.17e+00) +**	2.4760e+01 (1.11e+01) +	NaN (NaN) −	8.72e+00(5.02e+00)+	3.2415e+01 (1.47e+01)
CEC17-F22	3.5412e+03 (1.73e+03) −	NaN (NaN) −	4.7374e+01 (1.68e+01) −	1.6283e+01 (1.25e+00) −	7.8046e+00 (2.89e+00) −	1.73e+02 (1.48e+02)−	**4.7969e−01 (1.19e+00)**
CEC17-F23	1.5154e+00 (3.41e−02) −	1.6313e+00 (3.07e−02) −	**1.1007e+00 (1.16e−04) +**	1.5293e+00 (2.08e−01) −	1.5713e+00 (1.70e−01) −	1.10e+00(1.05e−02)+	1.1528e+00 (3.22e−02)
CEC17-F24	1.2409e+01 (1.28e+00) −	1.6305e+01 (1.59e+00) −	5.4977e+00 (6.79e−06) −	5.2464e+00 (8.70e−01) −	4.7438e+00 (5.76e+00) −	1.52e+01(2.86e+00)−	**2.3561e+00 (2.34e−05)**
CEC17-F25	1.8209e+02 (3.21e+01) −	2.6829e+02 (2.47e+01) −	6.5344e+00 (1.26e+00) −	6.2831e+00 (9.85e−06) −	1.0138e−12 (3.03e−12) =	8.93e+01(3.43e+01)−	**5.0120e−13 (4.72e−13)**
CEC17-F26	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)=	NaN (NaN)
CEC17-F27	4.4831e+01 (5.16e+00) −	4.1943e+01 (7.39e+00) −	NaN (NaN) −	**3.6487e+01 (5.65e−03) +**	NaN (NaN) −	3.79e+01(3.13e+00)−	3.7204e+01 (2.86e+00)
CEC17-F28	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)=	NaN (NaN)
+/−/=	1/21/6	4/19/5	6/16/6	6/16/6	4/13/11	7/14/7	Base

Table 5
Friedman's test ranking of CEDE-DRL and other methods on 28 benchmark functions in IEEE CEC2017 (50D).

Algorithm	Ranking	p-value
CEDE-DRL	**2.6964**	
DeCODE	3.5357	0.146034
Cϵ-LDE	3.8214	0.051348
CORCO	3.8393	0.047761
FROFI	4.0536	0.018741
C2oDE	4.7321	0.000422
LSHADE44	5.3214	0.000005

Table 6
Wilcoxon signed-rank test results for CEDE-DRL and the other six compared methods on 28 benchmark functions in IEEE CEC2017 (50D).

CEDE-DRL VS	R^+	R^-	p-value	$\alpha = 0.1$	$\alpha = 0.05$
C2oDE	361.0	17.0	3.4E−05	YES	YES
LSHADE44	342.0	64.0	1.49E−03	YES	YES
FROFI	306.0	99.5	1.7874E−02	YES	YES
DeCODE	285.0	93.0	2.0427E−02	YES	YES
CORCO	309.5	96.5	1.4828E−02	YES	YES
Cϵ-LDE	293.5	84.5	1.1648E−02	YES	YES

include three application fields, The first to seventh are industrial chemical processes [48], the eighth to the twelfth are process synthesis and design problems [49] then the thirteenth to thirty-third are mechanical engineering problems.

Through two sets of benchmark functions and a set of real-world optimization problems, the performance of the proposed algorithm can be fully tested. In recent years, excellent algorithms are tested on the basis of these problems, which can be well compared horizontally to reflect the advantages.

CHTs Setting:

In this algorithm, we selected four CHTs to assign to four populations. which are SF, ϵ-constraint, MO-constraint and the CHT used in CORCO, introduced in the previous Section 2.3.

Parameter Settings: The maximum number of times the functions can be evaluated is limited to 20,000 × dimension. The default population size Np is 100. The number of independent runs of all experiments was set to 30, the same as for the other comparison algorithms. F was set to $\{0.6, 0.8, 1.0\}$ and CR was set to $\{0.1, 0.2, 1.0\}$, which is the same CORCO. In ϵ-constraint, the parameters T_c and cp were set to 0.8 and 5.0. The parameters of MO-constraint were set using the parameter settings in DeCODE, and the CHT of CORCO still followed the settings in the original literature. $greedy$ and γ in DQN were both set to be 0.9. The training data set size is 2000, while the algorithm updates the neural network model every 500 generations. And the sizes of the two hidden layers are both 40. The dropout probabilities for the input and hidden layers are 0.2 and 0.5, respectively. [50]

4.2. Methods for comparison

We compared with six state-of-the-art algorithms (C2oDE [51], LSHADE44 [52], FROFI [53], DeCODE [41], CORCO [42] and Cϵ-LDE [54]) to verify the effectiveness of the proposed algorithm CEDE-DRL. DeCODE and CORCO represent solving COPs using a multi-objective approach and using CHTs employed in CORCO, respectively, and LSHADE44 is the champion algorithm in the CEC2017 competition. Both C2oDE and FROFI are excellent algorithms for solving COPs in recent years. Also added is Cϵ-LDE as the latest algorithm in 2023 for comparison.

All of the algorithms are implemented in MATLAB R2020b on a desktop PC with an Intel Core i7-8550U processor @ 1.99 GHz, 8 GB RAM, and the Windows 10 64-bit operating system.

4.3. Experimental results and comparisons

4.3.1. Experiments on the 18 benchmark test functions with 30D from IEEE CEC2010

In this section, 18 benchmark functions from the IEEE CEC2010 are used to test the performance of the proposed algorithm CEDE-DRL. Since the optimal solution of these test functions is unknown, the mean and standard deviation of the objective function values are used as comparison criteria. As shown in Table 1, we can observe that in the benchmark functions (CEC2010) with 30D (denoted as CEC10-F1∼F18), CEDE-DRL has an edge over C2oDE, LSHADE44, FROFI, DeCODE and CORCO on F2, F5, F12, F13 and F14, five test functions, it outperforms C2oDE, LSHADE44, FROFI and DeCODE on 16, 16, 14 and 10 test functions, respectively. Since the CEC2010 functions are relatively simple, the effect of CORCO on the set is similar to that of CEDE-DRL. In contrast, C2oDE, LSHADE44, FROFI and DeCODE performed better than CEDE-DRL on two, one, two and five, respectively.

Table 7
Experimental results of C2oDE, LSHADE44, FROFI, DeCODE, CORCO, Cϵ-LDE and CEDE-DRL on IEEE CEC2017 test functions (100D) with 30 independent runs.

	C2oDE	LSHADE44	FROFI	DeCODE	CORCO	Cϵ-LDE	CEDE-DRL
CEC17-F1	6.4876e−05 (4.65e−05) −	2.3332e−12 (1.13e−11) +	5.5456e+03 (4.07e+03) −	1.9483e−09 (1.40e−09) −	3.2478e−10 (3.64e−10) =	**1.91e−13 4.27e−13+**	2.3641e−10 (2.82e−10)
CEC17-F2	9.2146e−05 (9.58e−05) −	1.6874e+05 (3.78e+04) −	1.7886e+04 (7.86e+03) −	1.4839e−09 (8.43e−10) −	2.6765e−10 (1.64e−10) −	**1.01e−13(1.15e−13)+**	4.0746e−11 (4.80e−11)
CEC17-F3	1.8285e+05 (6.74e+04) −	3.4291e+06 (2.26e+06) −	2.1730e+04 (5.00e+04) −	6.1362e+03 (1.92e+03) −	**1.7703e−10 (1.17e−10)** +	1.80e+04(3.02e+04)−	4.5060e+01 (6.82e+01)
CEC17-F4	1.4731e+02 (2.49e+01) +	**1.3573e+01 (5.22e−14)** +	2.1348e+02 (6.16e+01) =	8.1690e+01 (2.09e+01) +	8.0893e+01 (2.88e+01) +	4.18e+02(4.70e+01)−	2.1366e+02 (3.83e+01)
CEC17-F5	4.4572e+01 (1.48e+01) −	5.4249e+05 (9.59e+04) −	3.4259e+03 (1.43e+03) −	5.3497e−01 (1.33e+00) −	3.7770e−01 (1.10e+00) −	9.57e−01(1.74e+00)−	**1.5946e−01 (7.97e−01)**
CEC17-F6	5.7350e+03 (2.38e+03) =	1.7305e+04 (2.14e+03) =	NaN (NaN) −	2.9964e+03 (5.12e+03) =	**3.0866e+02 (1.27e+02)** +	6.45e+02 (1.36e+02)+	7.7613e+02 (5.69e+02)
CEC17-F7	−7.8222e+02 (1.97e+02) +	−4.1980e+02 (1.19e+02) =	**−4.5650e+03 (2.53e+01)** +	−1.5934e+02 (1.48e+02) −	−8.8104e+01 (2.21e+02) −	−2.30e+03(1.73e+02)+	−6.0224e+02 (6.02e+02)
CEC17-F8	NaN (NaN) =	7.0567e−03 (1.91e−03) +	**9.1441e−04 (1.61e−04)** +	1.9566e−03 (5.88e−04) +	3.3912e−03 (8.27e−04) =	1.05e−03 (9.92e−05)+	NaN (NaN)
CEC17-F9	3.8764e−01 (1.32e+00) +	1.5376e+01 (1.64e+00) −	**0.0000e+00 (0.00e+00)** +	6.7680e−08 (2.34e−07) +	NaN (NaN) −	7.05e−01(2.92e+00)+	8.6132e+00 (2.50e+00)
CEC17-F10	1.0939e−03 (1.06e−04) −	2.2456e−03 (9.47e−04) −	3.3941e−04 (4.91e−05) −	3.7650e−04 (7.82e−05) −	3.6974e−04 (7.74e−05) −	4.01e−04(6.57e−05)−	**3.2329e−04 (1.54e−04)**
CEC17-F11	NaN (NaN) =	**−1.0305e+00 (0.00e+00)** =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC17-F12	2.4824e+02 (1.81e+01) −	4.0798e+02 (2.87e+01) −	5.7922e+00 (4.09e+00) +	2.3131e+01 (8.18e+00) =	NaN (NaN) −	**4.69e+00 (2.40e+00)+**	2.2733e+01 (7.18e+00)
CEC17-F13	6.1417e+03 (1.60e+03) −	1.2134e+04 (1.10e+04) −	2.8808e+02 (2.09e+02) −	3.9142e+01 (2.62e+00) −	3.0506e+01 (2.01e+01) −	7.88e+02 (3.42e+02)−	**1.9349e+01 (1.75e+01)**
CEC17-F14	1.1167e+00 (3.04e−02) −	1.1135e+00 (1.56e−02) −	7.8785e−01 (1.16e−02) −	7.8852e−01 (2.16e−02) −	1.1008e+00 (7.06e−02) −	7.93e−01(1.51e−02)−	**7.8420e−01 (8.55e−16)**
CEC17-F15	1.7561e+01 (1.18e+00) −	2.1457e+01 (2.21e+00) −	8.6393e+00 (1.66e−05) −	1.2409e+01 (1.28e+00) −	1.0147e+01 (1.60e+00) −	1.81e+01 (1.28e+00)−	**2.3561e+00 (3.62e−06)**
CEC17-F16	4.0665e+02 (3.14e+01) −	6.4026e+02 (2.26e+01) −	8.4822e+00 (2.90e+00) −	6.2831e+00 (1.12e−05) −	**4.4054e−13 (0.00e+00)** =	1.43e+01 (4.03e+00)−	**4.4054e−13 (0.00e+00)**
CEC17-F17	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC17-F18	5.1320e+01 (0.00e+00) −	4.3817e+01 (8.57e+00) −	NaN (NaN) −	3.9773e+01 (6.48e+00) −	NaN (NaN) −	3.73e+01(2.89e+00)−	**3.6395e+01 (6.97e−03)**
CEC17-F19	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC17-F20	1.1516e+01 (2.01e+00) =	**5.3230e+00 (6.30e−01)** +	1.1937e+01 (1.27e+00) =	1.0138e+01 (2.07e+00) +	1.5203e+01 (4.81e+00) −	6.86e+00(5.72e−01)+	1.1970e+01 (1.15e+00)
CEC17-F21	2.8513e+02 (2.14e+01) −	4.7309e+02 (1.92e+01) −	2.1928e+01 (1.29e+01) −	**6.3400e+00 (3.87e+00)** =	NaN (NaN) −	1.41e+01 (1.24e+01)−	8.7770e+00 (5.62e+00)
CEC17-F22	NaN (NaN) −	NaN (NaN) −	1.9444e+03 (8.21e+02) −	2.5101e+02 (1.99e+02) −	8.8973e+01 (5.20e+00) −	3.33e+03(1.09e+03)−	**6.3767e+01 (1.99e+01)**
CEC17-F23	1.0603e+00 (1.30e−02) −	1.1342e+00 (7.68e−03) −	7.8994e−01 (1.06e−02) +	9.9938e−01 (1.56e−01) =	1.0290e+00 (8.46e−02) −	**7.89e−01(1.08e−02)+**	8.5323e−01 (3.04e−02)
CEC17-F24	1.4922e+01 (4.44e−06) −	2.1960e+01 (1.64e+00) −	7.5084e+00 (1.79e+00) −	6.2517e+00 (1.37e+00) −	2.3561e+00 (4.64e−06) −	1.82e+01 (1.69e+00)−	**2.3561e+00 (4.56e−06)**
CEC17-F25	5.2798e+02 (2.41e+01) −	7.1848e+02 (1.82e+01) −	2.7018e+01 (9.71e+00) −	3.6317e+01 (1.81e+01) −	5.4412e+01 (3.04e+01) −	2.91e+02(4.83e+01)−	**1.2880e+01 (1.03e+01)**
CEC17-F26	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC17-F27	4.6917e+01 (3.71e+00) −	4.5122e+01 (1.00e+01) +	NaN (NaN) −	5.5482e+01 (3.50e+00) −	NaN (NaN) −	**4.20e+01(2.83e+01)+**	4.6818e+01 (7.19e+00)
CEC17-F28	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
+/−/=	3/18/8	5/17/7	5/17/7	4/16/9	3/18/8	10/13/5	Base

Table 8
Friedman's test ranking of CEDE-DRL and other methods on 28 benchmark functions in IEEE CEC2017 (100D).

Algorithm	Ranking	p-value
CEDE-DRL	**2.875**	
DeCODE	3.4286	0.337653
Cϵ-LDE	3.6429	0.183529
FROFI	3.9107	0.072828
CORCO	3.9643	0.059201
C2oDE	5.0179	0.000206
LSHADE44	5.1607	0.000075

Based on the results of the Friedman's test presented in Table 2, it can be concluded that CEDE-DRL ranked first, followed by DeCODE, CORCO, FROFI, C2oDE and LSHADE44. FROFI, C2oDE, and LSHADE44 have p-values less than 0.05, which means that they are significantly distinct from CEDE-DRL. The results show that CEDE-DRL performed slightly worse than some of the compared approaches on the CEC2010 functions.

According to Table 3, the performance of CEDE-DRL was significantly better than that of C2oDE, LSHADE44 and FROFI on the 18 CEC2010 benchmark functions with 30D, as evidenced by the p-values being less than 0.05 and the R^+ values being larger than the R^- values in all cases. Based on these findings, it can be concluded that overall, CEDE-DRL outperformed its four competitors on the aforementioned set of test functions.

4.3.2. Experiments on the 28 benchmark test functions with 50D and 100D from IEEE CEC2017

To further validate the performance of CEDE-DRL on high-dimensional COPs, the 28 test functions (denoted as CEC17-F1∼F28) with 50D and 100D from IEEE CEC2017 were adopted. Because it is more complex and higher dimensional than CEC2010, it allows for better verification performance. Tables 4 and 7 present the average and standard deviation of the fitness values for the seven algorithms on the functions from CEC2017.

Table 4 presents the fitness value statistics for the seven algorithms on the 50-dimensional functions from CEC2017. CEDE-DRL is ranked first on the eight problems (F5, F13, F15, F18, F22, F24, F25). And F16 tied for first place with CORCO. In particular, CEDE-DRL surpasses C2oDE, LSHADE44, FROFI, DeCODE, CORCO and C-ϵLDE on 21, 19, 16, 16, 13 and 14 test functions, respectively. The performance of C2oDE is better than that of CEDE-DRL only on function F5. Table 5 shows the average rankings of the seven approaches on the IEEE CEC2017 (50D) benchmark problems according to the Friedman's test. CEDE-DRL is ranked significantly lower (2.6964) than the other methods. However, the p-values of CORCO, FROFI, C2oDE and L-SHADE44 are less than 0.05, which indicates that CEDE-DRL outperformed the other methods on CEC2017. In terms of Wilcoxon's rank sum test (Table 6), CEDE-DRL provides higher $R+$ values than $R-$ values in all cases, and performs significantly better than C2oDE, LSHADE44, FROFI, DeCODE. CORCO and C-ϵLDE as the p-values are less than 0.05.

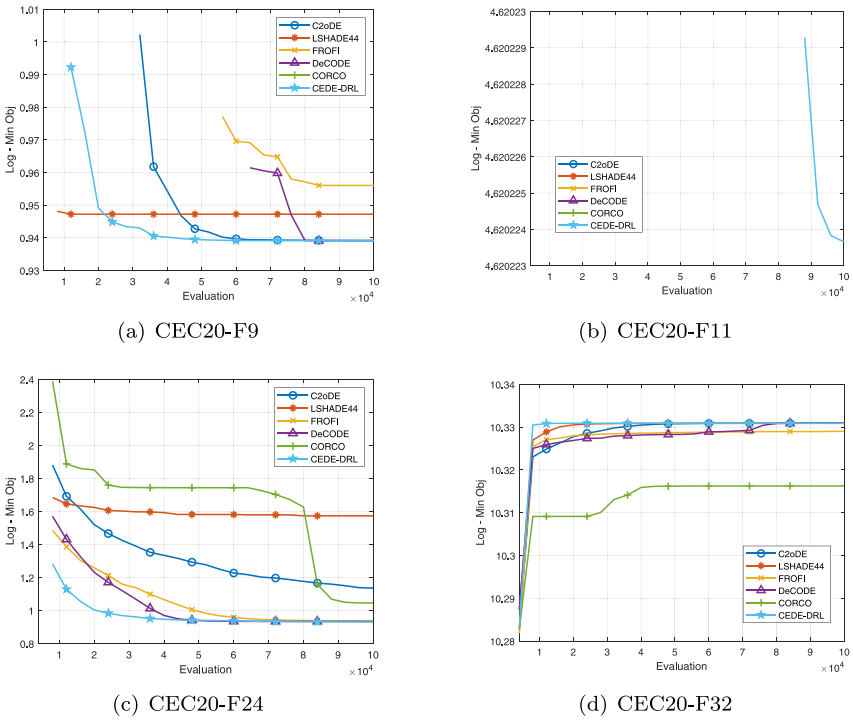
In the case of D = 100, from the data of the mean objective function value and standard deviation given in Table 7. It is obvious to see that CEDE-DRL is superior to C2oDE, LSHADE44, FROFI, DeCODE, CORCO and Cϵ-LDE on 18, 17, 16, 18 and 13 test functions, respectively. In contrast, C2oDE, LSHADE44, FROFI, DeCODE and CORCO outperforms CEDE-DRL on no more than five test function. The Friedman's test in Table 8 shows that CEDE-DRL is undoubtedly the smallest (2.875), followed by DeCODE (3.4286), Cϵ-LDE (3.6429), FROFI (3.9107), CORCO (3.9643), C2oDE (5.0179) and LSHADE44 (5.1607). The p-values of C2oDE and LSHADE44 are less than 0.05.

Table 9 of the Wilcoxon's rank sum test shows CEDE-DRL performed significantly better than C2oDE, LSHADE44, FROFI, DeCODE, CORCO as the $p-$ values are less than 0.05 and all the $R+$ values are larger than the $R-$ values.

In conclusion, after comparing with advanced algorithms, we can see that CEDE-DRL performs well on the 28 benchmark problems of IEEE CEC2017 for 50D and 100D with better search ability and robustness. In addition, we found that CEDE-DRL has more significant impact on performance as the dimensionality and complexity of the test function increases.

4.3.3. Experiments on the 33 real-world COPs

To further verify the effectiveness of CEDE-DRL on real-world constrained optimization problems, it was evaluated on the 33 test functions from IEEE CEC2020 with other comparison algorithms. From the Table 10 we can intuitively see that CEDE-DRL is the best performing of the six compared algorithms for solving 33 real-world problems. CEDE-DRL is eligible for first place on 20 functions (1, 3, 8, 9, 10, 11, 13, 14, 15, 16, 18, 19, 20, 21, 24, 27, 28, 29, 30, 32). the results of the Friedman's test and the Wilcoxon's rank sum tests reported in Tables 11 and 12 indicate that CEDE-DRL achieves the first rank (2.4394)



(a) CEC20-F9 (b) CEC20-F11

(c) CEC20-F24 (d) CEC20-F32

Fig. 3. Minimum objective value convergence of CEDE-DRL and other five methods on the functions from the IEEE CEC2020 competition, the minimum value of subfigures (a), (b), (c) are positive and subfigure (d) is negative.

Table 9
Wilcoxon signed-rank test results for CEDE-DRL and the other six compared methods on 28 benchmark functions in IEEE CEC2017 (100D).

CEDE-DRL VS	R^+	R^-	p-value	$\alpha = 0.1$	$\alpha = 0.05$
C2oDE	329.5	76.5	3.828E−03	YES	YES
LSHADE44	306.0	100.0	1.8431E−02	YES	YES
FROFI	273.0	105.0	4.2345E−02	YES	YES
DeCODE	282.0	96.0	2.4628E−02	YES	YES
CORCO	283.5	94.5	1.6803E−02	YES	YES
Cϵ-LDE	223.0	155.0	4.07182E−01	NO	NO

regarding the Friedman's test, followed by DeCODE (2.6515). Furthermore, CEDE-DRL performs significantly better than C2oDE, LSHADE44, FROFI and CORCO as the p-values are less than 0.05. Fig. 3 shows the convergence of the results of the comparison algorithms on some functions. The horizontal coordinate is the number of function evaluations and the vertical coordinate is the logarithm of the minimum objective function value. As can be seen from the subfigures (a), (c) and (d), compared with the other five algorithms, CEDE-DRL can find the minimum value and converge faster. The subfigure (b) shows that only CEDE-DRL can find a feasible solution to this problem and converge quickly.

To sum up, CEDE-DRL still has a strong performance in real-world application problems compared to today's advanced algorithms.

4.3.4. Experiments of component ablation

To verify the effectiveness of the algorithms in combining different populations, we separated out the four combined CHTs for the experiment of component ablation. In order to ensure fairness, we conducted comparative experiments on all five test algorithms in CEC2017 (50D). From Table 13, it can be seen that among the 28 functions, CEDE-DRL can achieve the smallest results among the 13 problems of F1, F2, F4, F5, F6, F7, F13, F15, F16, F18, F20, F22, F23. Fig. 4 shows the average objective value convergence of CEDE-DRL and other four CHTs on three 50-D CEC2017 (F4, F13, F21) benchmark problems. It is intuitively clear from the convergence that CEDE-DRL that the combination of multiple populations is the most effective compared to the population evolution of one CHT alone.

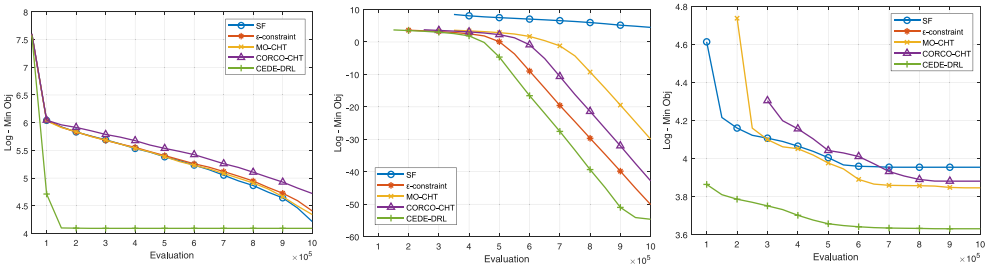
5. Conclusion

In this work, we have proposed a DQN assisted co-evolutionary DE called CEDE-DRL. To ensure the diversity of the training set in DRL, four populations are designed to evolve simultaneously. To handle different types of COPs, we use four differently biased CHTs assigned to each of the four populations. DQN is applied to the population selection to improve the overall optimality finding ability of the algorithm. Also the elimination of individual archiving is introduced in the mutation strategy to ensure that the algorithm does not converge to a local optimum prematurely. Three benchmark sets were chosen to test the algorithm: 30-dimensional CEC2010, 50-dimensional and 100-dimensional CEC2017. In addition, the first 33 problems of CEC2020 were added to test the ability of the algorithm in practical applications. The results of the comparison with the five state-of-the-art algorithms show that the proposed algorithm possesses stronger optimization capabilities and robustness when dealing with constraint problems. However, the problem to be solved is that it is difficult to find the minimum feasible solution for a small number of complex COPs. Also the state of agent settings of DQN can be divided more carefully so that the algorithm can achieve better performance.

In the future we plan to invoke Double DQN to alleviate the overestimation problem in DQN and to improve CHTs to reduce the complexity of the algorithm. At the same time, we also plan to apply the developed algorithm to more problems with constraints encountered in actual production problems.

Table 10
Experimental results of C2oDE, LSHADE44, FROFI, DeCODE, CORCO and CEDE-DRL on IEEE CEC2020 test functions with 30 independent runs.

	C2oDE	LSHADE44	FROFI	DeCODE	CORCO	CEDE-DRL
CEC20-F1	NaN (NaN) −	3.8484e+02 (1.01e+02) −	NaN (NaN) −	NaN (NaN) −	NaN (NaN) −	**3.1944e+02 (1.29e+02)**
CEC20-F2	7.0490e+03 (2.59e−08) −	8.8176e+03 (5.41e+03) −	NaN (NaN) −	NaN (NaN) −	**7.0490e+03 (2.95e−08)+**	NaN (NaN)
CEC20-F3	−3.8453e+03 (3.06e+02) −	−4.1816e+03 (2.72e+02) −	NaN (NaN) −	−4.5291e+03 (7.89e−04) −	NaN (NaN) −	**−4.5291e+03 (5.03e−08)**
CEC20-F4	−3.5321e−01 (6.69e−03) =	−5.1283e−04 (4.09e−04) −	0.0000e+00 (0.00e+00) −	**−3.5604e−01 (2.61e−02) +**	NaN (NaN) −	−1.1651e−01 (1.13e−01)
CEC20-F5	−2.5143e−04 (5.30e−04) +	1.1997e+01 (3.32e+01) +	−0.0000e+00 (0.00e+00) +	**−5.9835e+01 (1.21e+02) +**	7.3498e−11 (3.44e−10) =	2.8000e+01 (4.58e+01)
CEC20-F6	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC20-F7	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN) =	NaN (NaN)
CEC20-F8	2.0000e+00 (5.09e−10) −	**2.0000e+00 (0.00e+00) =**	2.0001e+00 (1.25e−04) −	**2.0000e+00 (0.00e+00) =**	2.0139e+00 (1.20e−02) −	**2.0000e+00 (0.00e+00)**
CEC20-F9	2.5579e+00 (1.36e−04) −	2.5785e+00 (1.62e−02) −	2.6013e+00 (3.77e−02) −	2.5577e+00 (1.42e−12) −	NaN (NaN) −	**2.5577e+00 (7.80e−13)**
CEC20-F10	1.0767e+00 (1.23e−04) −	1.1085e+00 (5.19e−02) −	1.1032e+00 (1.26e−02) −	1.0765e+00 (1.18e−12) −	1.0766e+00 (1.15e−04) −	**1.0765e+00 (2.74e−14)**
CEC20-F11	NaN (NaN) −	1.1342e+02 (1.08e+01) −	NaN (NaN) −	1.1078e+02 (6.38e+00) −	NaN (NaN) −	**1.0152e+02 (3.73e+00)**
CEC20-F12	**2.9266e+00 (1.49e−03) +**	3.5484e+00 (4.86e−01) −	2.9553e+00 (1.95e−02) +	7.0860e+00 (2.37e+00) −	3.0668e+00 (1.68e−01) −	2.9646e+00 (1.94e−01)
CEC20-F13	2.6898e+04 (8.34e+00) −	2.6932e+04 (1.17e+02) −	2.6915e+04 (2.10e+01) −	2.6887e+04 (1.83e−08) −	2.8579e+04 (1.03e+03) −	**2.6887e+04 (1.11e−11)**
CEC20-F14	6.3095e+04 (3.65e+03) −	7.5750e+04 (1.18e+04) −	6.3673e+04 (3.03e+03) −	6.1069e+04 (3.33e+03) −	1.3074e+05 (3.22e+04) −	**5.3639e+04 (6.59e−03)**
CEC20-F15	2.9967e+03 (1.21e+00) −	3.0382e+03 (2.47e+01) −	3.0156e+03 (5.65e+00) −	2.9944e+03 (1.02e−05) −	3.3448e+03 (8.74e+01) −	**2.9944e+03 (8.17e−09)**
CEC20-F16	3.4924e−02 (8.93e−03) −	1.1505e+01 (9.74e+00) −	4.9593e−02 (4.89e−03) −	3.2213e−02 (1.42e−08) −	3.2749e−02 (1.22e−03) −	**3.2213e−02 (2.29e−12)**
CEC20-F17	1.2680e−02 (1.65e−05) −	1.2907e−02 (1.58e−04) −	1.2772e−02 (4.76e−05) −	**1.2665e−02 (4.88e−18) +**	1.4821e−02 (1.75e−03) −	1.2665e−02 (1.11e−08)
CEC20-F18	6.0597e+03 (4.23e−03) −	6.1508e+03 (1.30e+02) −	6.1734e+03 (5.78e+01) −	6.3973e+03 (6.39e+01) −	7.5040e+03 (2.53e+03) −	**6.0597e+03 (9.28e−13)**
CEC20-F19	1.6702e+00 (3.76e−06) −	**1.6702e+00 (1.36e−16) =**	1.7221e+00 (1.70e−02) −	**1.6702e+00 (4.53e−17) =**	1.6709e+00 (1.43e−03) −	**1.6702e+00 (0.00e+00)**
CEC20-F20	2.6390e+02 (6.20e−05) −	**2.6390e+02 (0.00e+00) =**	2.6408e+02 (1.27e−01) −	2.6390e+02 (1.99e−02) −	2.6793e+02 (1.83e+00) −	**2.6390e+02 (0.00e+00)**
CEC20-F21	**2.3524e−01 (1.13e−16) =**	**2.3524e−01 (1.13e−16) =**	2.3558e−01 (2.91e−04) −	2.3524e−01 (1.22e−13) −	2.4177e−01 (8.36e−03) −	**2.3524e−01 (1.13e−16)**
CEC20-F22	5.2869e−01 (2.11e−03) =	5.4357e−01 (1.15e−02) −	5.2618e−01 (3.49e−04) +	**5.2615e−01 (1.21e−03) +**	5.2694e−01 (2.92e−03) +	5.2860e−01 (3.94e−03)
CEC20-F23	1.6091e+01 (7.34e−03) −	1.6364e+01 (2.71e−01) −	NaN (NaN) −	**1.6070e+01 (3.77e−14) +**	1.6070e+01 (2.40e−05) −	1.6070e+01 (5.10e−07)
CEC20-F24	3.1121e+00 (2.68e−01) −	4.8201e+00 (5.97e−01) −	2.5500e+00 (2.10e−03) −	2.5438e+00 (9.89e−07) =	2.8449e+00 (6.27e−02) −	**2.5364e+00 (8.09e−03)**
CEC20-F25	2.3593e+03 (2.91e+02) −	3.9053e+03 (6.34e+02) −	2.9792e+03 (3.60e+02) −	**1.6161e+03 (5.59e−03) +**	3.2896e+03 (4.38e+02) −	1.6164e+03 (3.02e−01)
CEC20-F26	6.4045e+01 (1.91e+01) −	5.8943e+01 (1.74e+01) −	NaN (NaN) −	5.5837e+01 (1.93e+01) −	**4.2353e+01 (6.50e+00) =**	4.4331e+01 (2.07e+01)
CEC20-F27	5.2902e+02 (1.93e+00) −	5.2696e+02 (2.94e+00) −	5.3925e+02 (3.33e+00) −	5.2475e+02 (1.70e−01) −	5.3597e+02 (3.98e+01) −	**5.2457e+02 (3.87e−02)**
CEC20-F28	1.7010e+04 (2.28e+01) −	1.7398e+04 (3.52e+02) −	1.7315e+04 (1.57e+02) −	1.6958e+04 (6.44e−04) −	2.1020e+04 (8.11e+03) −	**1.6958e+04 (3.71e−12)**
CEC20-F29	2.9649e+06 (1.15e−05) −	**2.9649e+06 (1.43e−09) =**	3.0326e+06 (4.88e+04) −	2.9649e+06 (1.95e−06) −	**2.9649e+06 (1.14e−09) =**	**2.9649e+06 (1.26e−09)**
CEC20-F30	2.6625e+00 (3.86e−03) −	2.6841e+00 (5.81e−02) −	2.7274e+00 (7.66e−02) −	**2.6586e+00 (4.53e−16) =**	8.0341e+00 (3.83e+00) −	**2.6586e+00 (4.53e−16)**
CEC20-F31	9.7273e−13 (1.63e−12) −	**0.0000e+00 (0.00e+00) +**	1.9511e−15 (9.31e−15) +	6.7562e−15 (2.55e−14) +	3.5324e−15 (1.72e−14) +	5.7121e−14 (1.74e−13)
CEC20-F32	−3.0666e+04 (1.38e−02) −	**−3.0666e+04 (3.71e−12) =**	−3.0608e+04 (2.01e+01) −	−3.0666e+04 (1.84e−04) −	−3.0220e+04 (3.58e+02) −	**−3.0666e+04 (3.71e−12)**
CEC20-F33	2.6393e+00 (5.05e−07) −	**2.6393e+00 (7.90e−16) +**	2.6393e+00 (1.09e−15) =	2.6393e+00 (8.50e−16) =	2.6393e+00 (7.85e−16) =	2.6393e+00 (4.43e−15)
+/−/=	2/26/5	3/22/8	4/26/3	7/19/7	3/24/6	Base



(a) CEC17-F4 (b) CEC17-F13 (c) CEC17-F21

Fig. 4. Minimum objective value convergence of CEDE-DRL and other four CHTs on the functions from the IEEE CEC2017 competition, the minimum value of subfigures (a), (c) are positive and subfigure (b) is negative.

Table 11
Friedman's test ranking of CEDE-DRL and other methods on 33 benchmark functions in IEEE CEC2020.

Algorithm	Ranking	*p*-value
CEDE-DRL	**2.4394**	
DeCODE	2.6515	0.64511
C2oDE	3.0455	0.188206
LSHADE44	4.0303	0.000552
FROFI	4.3333	0.000039
CORCO	4.5	0.000008

Table 12
Wilcoxon signed-rank test results for CEDE-DRL and the other five compared methods on 33 benchmark functions in IEEE CEC2020.

CEDE-DRL VS	R^+	R^-	*p*-value	α=0.1	α=0.05
C2oDE	372.5	155.5	4.1531E−02	YES	YES
LSHADE44	497.5	63.5	9.8E−05	YES	YES
FROFI	481.5	46.5	4.6E−05	YES	YES
DeCODE	333.5	227.5	3.39108E−01	NO	NO
CORCO	485.0	76.0	2.49E−04	YES	YES

Table 13
Ablation experiment.

	SF	ϵ-constraint	MO-CHT	CORCO-CHT	CEDE-DRL
CEC17-F1	2.5550e−19 −	4.4897e−19 −	3.3981e−19 −	5.1844e−19 −	**1.5604e−21**
CEC17-F2	2.7193e−19 −	3.4854e−19 −	3.5767e−19 −	1.8552e−18 −	**1.5188e−22**
CEC17-F3	5.4968e+06 −	**1.8372e−15** =	6.7947e+05 −	3.1696e+05 −	2.9929e+00
CEC17-F4	6.7683e+01 =	8.1986e+01 −	7.6810e+01 −	1.1236e+02 −	**6.4591e+01**
CEC17-F5	2.1771e−25 −	1.3954e−25 −	4.9500e−25 −	2.8470e−25 −	**9.2847e−26**
CEC17-F6	8.4151e+03 −	6.6111e+02 −	7.0328e+02 −	6.6008e+02 −	**2.0429e+02**
CEC17-F7	−2.9704e+01 −	−1.1829e+03 −	−8.5688e+01 −	−1.3505e+01 −	**−1.9823e+03**
CEC17-F8	**−1.3363e−04** +	4.0172e−03 −	−1.2490e−04 +	−2.5279e−05 +	6.6445e−04
CEC17-F9	**−2.0371e−03** +	NaN =	NaN =	NaN =	−2.0089e−03
CEC17-F10	**−4.8050e−05** +	1.0342e−03 −	−4.4902e−05 +	4.1184e−05 −	−2.1988e−05
CEC17-F11	**−1.4790e+00** +	NaN =	NaN =	NaN =	NaN
CEC17-F12	**3.9816e+00** +	7.7324e+00 +	3.9816e+00 +	3.9816e+00 +	6.4562e+00
CEC17-F13	9.0122e+01 −	1.7507e−22 −	9.8085e−14 −	2.5802e−19 −	**1.7842e−24**
CEC17-F14	1.5901e+00 −	1.1000e+00 +	**1.1000e+00** +	1.5907e+00 −	1.1000e+00
CEC17-F15	2.1371e+01 −	3.4557e+00 −	9.2677e+00 −	1.0838e+01 −	**2.4818e+00**
CEC17-F16	3.1714e+02 −	**2.0961e−13** =	6.2831e+00 −	1.0446e+01 −	**2.0961e−13**
CEC17-F17	NaN =	NaN =	NaN =	NaN =	NaN
CEC17-F18	3.6484e+01 −	3.6501e+01 −	3.6494e+01 −	3.6486e+01 −	**3.6474e+01**
CEC17-F19	NaN =	NaN =	NaN =	NaN =	NaN
CEC17-F20	6.7150e+00 =	6.1008e+00 =	6.9870e+00 −	6.0868e+00 =	**5.6165e+00**
CEC17-F21	5.2138e+01 −	**3.3117e+01** +	4.6785e+01 =	4.8457e+01 −	3.2415e+01
CEC17-F22	1.8054e+04 −	3.1151e+00 −	1.4363e+01 −	1.5198e+03 −	**4.7969e−01**
CEC17-F23	1.5477e+00 −	1.6356e+00 −	1.1963e+00 −	1.5764e+00 −	**1.1528e+00**
CEC17-F24	1.8850e+01 −	**2.3562e+00** +	9.1106e+00 −	1.0681e+01 −	2.3561e+00
CEC17-F25	3.2351e+02 −	**3.9120e−13** +	1.0210e+01 −	2.0892e+01 −	5.0120e−13
CEC17-F26	NaN =	NaN =	NaN =	NaN =	NaN
CEC17-F27	3.6850e+01 +	4.1689e+01 −	3.6498e+01 +	**3.6486e+01** +	3.7204e+01
CEC17-F28	NaN =	NaN =	NaN =	NaN =	NaN
+/−/=	6/16/6	5/14/9	5/16/7	3/18/7	Base

CRediT authorship contribution statement

Zhenzhen Hu: Resources, Project administration, Software, Data curation, Writing – original draft, Writing – review & editing. **Wenyin Gong:** Funding acquisition, Conceptualization, Methodology, Writing – review & editing. **Witold Pedrycz:** Supervision, Conceptualization, Writing – review & editing. **Yanchi Li:** Resources, Data curation, Writing – review & editing.

Declaration of competing 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.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under Grant No. 62076225.

All authors approved the version of the manuscript to be published.

References

[1] J.C. Seck-Tuoh-Mora, N. Hernandez-Romero, P. Lagos-Eulogio, J. Medina-Marin, N.S. Zuñiga-Peña, A continuous-state cellular automata algorithm for global optimization, Expert Syst. Appl. 177 (2021) 114930.

[2] W. Gong, Z. Cai, D. Liang, Adaptive ranking mutation operator based differential evolution for constrained optimization, IEEE Trans. Cybern. 45 (4) (2015) 716–727.

[3] J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization, J. Appl. Mech. 41 (8) (2006) 8–31.

[4] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.

[5] P. Jiang, Y. Cheng, J. Yi, J. Liu, An efficient constrained global optimization algorithm with a clustering-assisted multiobjective infill criterion using Gaussian process regression for expensive problems, Inform. Sci. 569 (2021) 728–745.

[6] L. Ma, M. Huang, S. Yang, R. Wang, X. Wang, An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization, IEEE Trans. Cybern. 52 (7) (2022) 6684–6696.

[7] H. Zhao, C. Zhang, B. Zhang, P. Duan, Y. Yang, Decomposition-based subproblem optimal solution updating direction-guided evolutionary many-objective algorithm, Inform. Sci. 448–449 (2018) 91–111.

[8] Y. Chen, X. Sun, D. Gong, Y. Zhang, J. Choi, S. Klasky, Personalized search inspired fast interactive estimation of distribution algorithm and its application, IEEE Trans. Evol. Comput. 21 (4) (2017) 588–600.

[9] Y. Wang, B.-C. Wang, H.-X. Li, G.G. Yen, Incorporating objective function information into the feasibility rule for constrained evolutionary optimization, IEEE Trans. Cybern. 46 (12) (2016) 2938–2952.

[10] R. Storn, K. Price, Differential evolution a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.

[11] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948.

[12] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.

[13] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.

[14] Z. Cheng, H. Song, J. Wang, H. Zhang, T. Chang, M. Zhang, Hybrid firefly algorithm with grouping attraction for constrained optimization problem, Knowl.-Based Syst. 220 (2021) 106937.

[15] A. Eiben, M. Horvath, W. Kowalczyk, M. Schut, Reinforcement learning for online control of evolutionary algorithms, in: S. Brueckner, S. Hassas, M. Jelasity, D. Yamins (Eds.), International Workshop on Engineering Self-Organising Applications, 2006, pp. 151–160.

[16] S. Lü, S. Han, W. Zhou, J. Zhang, Recruitment-imitation mechanism for evolutionary reinforcement learning, Inform. Sci. 553 (2021) 172–188.

[17] J. Sun, X. Liu, T. Bäck, Z. Xu, Learning adaptive differential evolution algorithm from optimization experiences by policy gradient, IEEE Trans. Evol. Comput. 25 (4) (2021) 666–680.

[18] Z. Li, L. Shi, C. Yue, Z. Shang, B. Qu, Differential evolution based on reinforcement learning with fitness ranking for solving multimodal multiobjective problems, Swarm Evol. Comput. 49 (2019) 234–244.

[19] J. Sun, X. Liu, T. Back, Z. Xu, Learning adaptive differential evolution algorithm from optimization experiences by policy gradient, IEEE Trans. Evol. Comput. 25 (4) (2021) 666–680.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, M.G. Bellemare, A. Graves, M. Riedmiller, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[21] K. Li, T. Zhang, R. Wang, Deep reinforcement learning for multi-objective optimization, IEEE Trans. Cybern. 51 (6) (2020) 3103–3114.
[22] Y. Tian, X. Li, H. Ma, X. Zhang, K.C. Tan, Y. Jin, Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization, IEEE Trans. Emerg. Top. Comput. Intell. (2022) 1–14.
[23] O. Sigaud, Combining evolution and deep reinforcement learning for policy search: a survey, ACM Trans. Evol. Learn. (2022).
[24] M. Sharma, A. Komninos, M. López-Ibáñez, D. Kazakov, Deep reinforcement learning based parameter control in differential evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 709–717.
[25] Z. Tan, K. Li, Differential evolution with mixed mutation strategy based on deep reinforcement learning, Appl. Soft Comput. 111 (2021) 107678.
[26] S. Handoko, N. Thien, Z. Yuan, H. Lau, Reinforcement learning for adaptive operator selection in memetic search applied to quadratic assignment problem, in: GECCO 2014 - Companion Publication of the 2014 Genetic and Evolutionary Computation Conference, 2014.
[27] B. Ghasemishabankareh, X. Li, M. Ozlen, Cooperative coevolutionary differential evolution with improved augmented Lagrangian to solve constrained optimisation problems, Inform. Sci. 369 (2016) 441–456.
[28] C. Huang, X. Zhou, X. Ran, Y. Liu, W. Deng, W. Deng, Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem, Inform. Sci. 619 (2023) 2–18.
[29] W.-F. Gao, G.G. Yen, S.-Y. Liu, A dual-population differential evolution with coevolution for constrained optimization, IEEE Trans. Cybern. 45 (5) (2015) 1108–1121.
[30] S.Y. Chong, P. Tino, X. Yao, Relationship between generalization and diversity in coevolutionary learning, IEEE Trans. Comput. Intell. AI Games 1 (3) (2009) 214–232.
[31] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Engrg. 186 (2) (2000) 311–338.
[32] F. zhuo Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, Appl. Math. Comput. 186 (1) (2007) 340–356.
[33] K. Zielinski, R. Laur, Constrained single-objective optimization using differential evolution, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 223–230.
[34] S. Venkatraman, G. Yen, A generic framework for constrained optimization using genetic algorithms, IEEE Trans. Evol. Comput. 9 (4) (2005) 424–435.
[35] T. Takahama, S. Sakai, Constrained optimization by the ϵ constrained differential evolution with gradient-based mutation and feasible elites, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 1–8.
[36] Z. Fan, Y. Fang, W. Li, Y. Yuan, Z. Wang, X. Bian, LSHADE44 with an improved ϵ constraint-handling method for solving constrained single-objective optimization problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1–8.
[37] S. Domínguez-Isidro, E. Mezura-Montes, A cost-benefit local search coordination in multimeme differential evolution for constrained numerical optimization problems, Swarm Evol. Comput. 39 (2018) 249–266.
[38] Z. Cai, Y. Wang, A multiobjective optimization-based evolutionary algorithm for constrained optimization, IEEE Trans. Evol. Comput. 10 (6) (2006) 658–675.
[39] Y. Zhou, M. Zhu, J. Wang, Z. Zhang, Y. Xiang, J. Zhang, Tri-goal evolution framework for constrained many-objective optimization, IEEE Trans. Syst. Man Cybern.: Syst. 50 (8) (2020) 3086–3099.
[40] C. Peng, H.-L. Liu, F. Gu, A novel constraint-handling technique based on dynamic weights for constrained optimization problems, Soft Comput. 22 (2018) 3919–3935.
[41] B.-C. Wang, H.-X. Li, Q. Zhang, Y. Wang, Decomposition-based multiobjective optimization for constrained evolutionary optimization, IEEE Trans. Syst. Man Cybern.: Syst. 51 (1) (2021) 574–587.
[42] Y. Wang, J.-P. Li, X. Xue, B.-c. Wang, Utilizing the correlation between constraints and objective function for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 24 (1) (2020) 29–43.
[43] D. Guo, X. Wang, K. Gao, Y. Jin, J. Ding, T. Chai, Evolutionary optimization of high-dimensional multiobjective and many-objective expensive problems assisted by a dropout neural network, IEEE Trans. Syst. Man Cybern.: Syst. 52 (4) (2022) 2084–2097.
[44] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, et al., KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (2009) 307–318.
[45] R. Mallipeddi, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real- Parameter Optimization, 24, Nanyang Technological University, 2010.
[46] G. Wu, R. Mallipeddi, P. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization, Technical Report, National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, 2016.
[47] A. Kumar, G. Wu, M.Z. Ali, R. Mallipeddi, P.N. Suganthan, S. Das, A test-suite of non-convex constrained optimization problems from the real-world and some baseline results, Swarm Evol. Comput. 56 (2020) 100693.
[48] C.A. Floudas, Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications, Oxford University Press, 1995.
[49] R. Angira, B.V. Babu, Optimization of process synthesis and design problems: A modified differential evolution approach, Chem. Eng. Sci. 61 (14) (2006) 4707–4721.
[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[51] B.-C. Wang, H.-X. Li, J.-P. Li, Y. Wang, Composite differential evolution for constrained evolutionary optimization, IEEE Trans. Syst. Man Cybern.: Syst. 49 (7) (2018) 1482–1495.
[52] R. Polakova, L-SHADE with competing strategies applied to constrained optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1683–1689.
[53] Y. Wang, B.-C. Wang, H.-X. Li, G.G. Yen, Incorporating objective function information into the feasibility rule for constrained evolutionary optimization, IEEE Trans. Cybern. 46 (12) (2016) 2938–2952.
[54] W. chuan Wang, L. Xu, K. wing Chau, C. jun Liu, Q. Ma, D. mei Xu, Ce-LDE: A lightweight variant of differential evolution algorithm with combined e constrained method and levy flight for constrained optimization problems, Expert Syst. Appl. 211 (2023) 118644.