Hindawi
International Journal of Intelligent Systems
Volume 2024, Article ID 6740701, 17 pages
https://doi.org/10.1155/2024/6740701

Research Article

Parameter Control Framework for Multiobjective Evolutionary Computation Based on Deep Reinforcement Learning

Tianwei Zhou,[1,2] **Wenwen Zhang**,[1,2] **Ben Niu**,[1,2] **Pengcheng He**,[1] **and Guanghui Yue**[3,4]

[1]*College of Management, Shenzhen University, Shenzhen 518060, China*
[2]*Great Bay Area International Institute for Innovation, Shenzhen University, Shenzhen 518060, China*
[3]*School of Biomedical Engineering, Shenzhen University Medical School, Shenzhen 518060, China*
[4]*Marshall Laboratory of Biomedical Engineering, Shenzhen University, Shenzhen 518060, China*

Correspondence should be addressed to Ben Niu; drniuben@gmail.com

Received 24 January 2023; Revised 14 November 2023; Accepted 28 November 2023; Published 3 January 2024

Academic Editor: Fabio Caraffini

To address the challenge of parameter adjustment in complex environments, this paper introduces a transfer learning-based parameter control framework via deep reinforcement learning for multiobjective evolutionary algorithms (MOEAs). To avoid the requirement for accurate Pareto front information, this framework is proposed with comprehensive global-state information, including basic problem features, the relative position of individuals, the distribution of fitness value, and the grid-IGD. Building on this framework, four reinforced multiobjective evolutionary algorithms (r-MOEAs) are proposed and tested on four DTLZ benchmarks and eight WFG benchmarks. The results of the comparative analyses reveal that compared with the original MOEAs, the four r-MOEAs exhibit faster convergence and stronger robustness. It is also confirmed that our proposed parameter control framework has the capability to learn knowledge from different experiences and improve the performance of MOEAs.

1. Introduction

Multiobjective optimization problems (MOPs) are common in practical applications, such as margin trading [1], energy system design [2], scheduling [3], and water resources management [4]. There are two significant characteristics in MOPs. Instead of one optimization objective, where only one goal is preferred, two or more optimization goals need to be considered in multiobjective optimization. Moreover, these objectives cannot be optimized at the same time. The optimization of one goal is at the cost of the degradation of the other goals with the intrinsic internal conflicts between targets. To depict these characteristics and formulate these practical requirements by mathematical expression, MOPs are modeled by the following formula:

$$\min F(x) = [f_1(x), \ldots, f_n(x)], x \in \omega,$$
$$\text{s.t.} \begin{cases} g_i(x) \le 0, & i = 1, \ldots, q, \\ h_j(x) = 0, & j = q+1, \ldots, p, \end{cases} \tag{1}$$

where $F(x)$ is the objective function with n objectives conflicting against each other and $g_i(x)$ and $h_j(x)$ are constraints for solution x. Different from the single optimum solution for single-objective optimization problems (SOPs), MOPs hold a set of optimal solutions (Pareto optimal solutions). Therefore, how to adjust parameters to find more Pareto optimal solutions becomes one of the key problems.

Addressing this issue, traditional approaches typically necessitate repeated testing for each algorithm or to transform MOPs into SOPs with one weighted objective. However, these methods are difficult to obtain Pareto optimal solutions. Meanwhile, multiobjective evolutionary algorithms (MOEAs) are capable of identifying multiple Pareto optimal solutions in a single execution. Based on this advantage, MOEAs have attracted many scholars' attention, and many famous MOEAs have been proposed since 2002, such as multiobjective particle swarm optimization (MOPSO) [5], multiobjective bacterial foraging optimization (MBFO) [6], nondominated sorting genetic algorithm II (NSGA-II) [7], multiobjective evolutionary algorithm based on decomposition(MOEA/D) [8], multiobjective differential evolution(MODE) [9], and multiobjective ant lion optimizer (MOALO) [10]. These MOEAs are still widely used in many practical applications, e.g., scheduling [11], electric power [12], and telecommunication [13].

Most single-objective or multiobjective optimization algorithms typically have problem-specific parameters. Currently, these are mainly the following methods for parameter tuning: using common parameter tuning algorithms such as parameter tuning with chess rating system (CRS-Tuning) [14], F-Race [15], and REVAC [16] or incorporating different strategies during the evolutionary iteration process to adapt the parameters to the problem or iteration process. For an MOP, uncertainty can occur in the objective function, decision variables, and function parameters [17]. Over the years, there have been several attempts to improve evolutionary algorithms by making parameters adaptive to the problem or to the iterative process. Basically, they can be organized into the following three categories which are rule-based, iteration memory-based, and learned knowledge-based. The first one is the rule-based parameter control method. This kind of method specifies a fixed way of changing certain parameters in an EA. For example, parameters are designed by raising or declining with iterations, such as [18, 19]. The second one is the iteration memory-based parameter control method. This kind of method records information such as the success rate of the policy, such as [20, 21]. The above two methods control the parameters in a single iteration, and the information retention and subsequent influence also stay in the single iteration process. The third is the learned knowledge-based parameter control method. This kind of method maintains the information learning from different problems to formulate a decision model. Every time a new problem has been solved, the model will be updated. For now, reinforcement learning (RL) and deep reinforcement learning (DRL) are used to store such experiences, and both of them belong to learning artificial intelligence methods. This type of method retains past information and learns from different problems. After training, it can select appropriate parameters for specific problems in different circumstances. In recent years, this kind of method has been discussed in [22–24]. Most references considered the single-objective problem, such as [22, 23]. While few references extended this idea towards MOEA [24]. The combination of the learned knowledge-based parameter control method and MOEA is still in its infancy, and a comprehensive framework is an urgent requirement.

For this article, we mainly focus on developing an MOEA parameter control framework based on DRL. This learned knowledge-based parameter control framework can be applied to different MOEAs and improves algorithms' efficiency and robustness on different optimization problems. The contributions of this article can be concluded as follows:

(i) First, we propose a novel learned knowledge-based parameter control framework via deep reinforcement learning for MOEAs. This framework simplifies complex parameter tuning steps for MOEA by embedding parameter control strategies in both the training stages and testing stages.

(ii) Second, we extract comprehensive global-state information based on universal information including basic problem features, the relative position of individuals, the distribution of fitness value, and the grid-IGD. This state information provides valuable insights for the agent to make effective decisions. These state features are all obtained from the current state and do not require information about accurate PFs. Moreover, feedback rewards with memories for MOEA are proposed based on the general characteristics of MOEA. Furthermore, the designed reward increases the number and quality of the Pareto front solutions.

(iii) Third, four reinforced algorithms are separately proposed by combining the most prestigious MOEAs with the designed framework. The superiority of our proposed framework is provided through the comparisons between the original MOEAs and the reinforced versions.

The remainder of this article is arranged as follows. Section 2 presents the related work. Then, we introduce RL as the preliminaries for parameterized knowledge representation in Section 3. Section 4 proposes the framework of parameter control. Section 5 illustrates the efficiency of the proposed framework through comparisons between the reinforced algorithms against corresponding original MOEAs. Section 6 summarizes this article.

2. Related Work

2.1. Rule-Based Parameter Control Method. This category can be further divided into two subcategories. The first involves parameters changing with iterations, which is a usual dynamic parameter-changing strategy. For instance, the authors of [18] proposed a framework that adjusts the parameters in MOPSO for individual particles based on knowledge extracted from the belief space. The authors of [25] proposed time variant MOPSO, where the acceleration coefficients and inertia weights vary with iterations. The second subcategory involves updates by a fixed formula, emphasizing inherent rules. The authors of [26] proposed MOEA/D-AWA with the adaptive weight vector adjustment strategy.

The shared characteristic of these two subcategories is that the scheme of parameter control is formulated before the iteration process and does not interact with the information in the iteration process. The advantage is that the randomness of the parameters is increased, and different parameter values are assigned in different iteration stages to better adapt to the problem and the iterative process. The disadvantage is that this kind of parameter control method requires a continuous trial and error to find a suitable control strategy. At the same time, for different problems, the specific strategy also needs to be adjusted to meet the requirements.

2.2. Iteration Memory-Based Parameter Control Method. This kind of method restores the information from a single run and uses this information to adjust subsequent parameters. The commonly used first-order reference indicators for MOEA are the changes in dominance relationship.

In [27], a binary space partitioning tree structure was selected to store the evaluated solutions' positions and fitness values with a fast fitness function. In this algorithm, the variational operator is parameter-free and adapted according to the current state. The author of [28] took feedback from the current state to modify the parameters. Moreover, the authors of [29] dynamically adjusted parameters based on average feedback. The author of [30] recorded the parameter values of the successful crossover and variation operators and updated the parameter values of the next generation by averaging the feedback of the successful parameter values. Moreover, an adaptive velocity strategy based on the evolutionary state for PSO was proposed in [31], which realizes adaptive control for traffic signals.

The advantage of this type of method is that it can make full use of the information during the iteration process and accordingly make real-time and specific adjustments to the parameters of the next generation or the next individual. Since the information is extracted based on the experience within the iteration process, it cannot be saved or transferred to new scenarios. For different problems, the parameters have to be rearranged to adapt to different characteristics.

2.3. Transfer Learning-Based Parameter Control Method. The transfer learning-based parameter control method restores and transfers knowledge learned from different problems. These methods take full advantage of reinforcement learning or deep reinforcement learning and learn from past information.

The authors of [24] applied q-learning on MOPSO to optimize primary control parameters, including the cognitive acceleration coefficient, inertia weight, and social acceleration coefficient. Similarly, the authors of [32] also combined q-learning with MOPSO to realize parameter control, and the distance between the previous best position and the best position of the current population is used as the state for parameter selection. Based on NSGA-II, the authors of [33] utilized q-learning to adjust the crossover and mutation probabilities with population diversity, evolutionary iteration number, and average fitness, thereby enhancing population diversity. The authors of [34] proposed a general framework of parameter control with reinforcement learning for single-objective evolutionary computation. This framework designed parameter sets in advance for each evolutionary algorithm, and q-learning will help choose one parameter-based state in each iteration. The authors of [35] combined DRL with MOEA for solving constrained multiobjective optimization problems, which took into account both population's convergence and diversity in their inputs to DRL.

The advantages of these methods can be concluded as follows. First, it can restore and summarize the past experience of adjusting parameters in different states and transfer the summarized experience to different problems. Second, for a new optimization problem, the parameters can be updated directly according to the current iteration information and past experience, improving the efficiency and accuracy of the parameter selection process.

Remark 1. The relationship between the state and these three methods is incrementally coupled. The methods are designed from complete random to state-based. Meanwhile, the third class of methods, transfer learning-based parameter control method, has the ability to process high-dimensional evolutionary states and is more scalable and transferable in terms of problem characteristics.

3. Preliminaries

Reinforcement learning (RL) belongs to a machine learning method. Unlike supervised learning or unsupervised learning, RL interacts and learns from the environment to obtain an optimum policy that can maximize the reward. RL mainly contains the following contents: state s, action a, reward r, and transition probability p. All the above elements are also known as a Markov decision process (MDP) [36] and denoted as $\langle s, a, r, p_a \rangle$. The processes for a typical RL algorithm are listed as follows:

(i) The agent performs an action a to interact with the environment.

(ii) After action, the state transforms from s to a new state s'.

(iii) Then, the agent will be rewarded r_a according to the action a and the rule of rewarding.

(iv) With the reward r_a, the agent will recognize whether the action selection holds a positive or negative effect.

(v) If the positive reward is returned, the agent will perform that action with more probability, or else the agent will try another action to obtain better reward under the state s.

Deep learning (DL) has gained achievements in natural language processing (NLP), image classification, and many other fields. The representational power of deep learning relies heavily on multilayer neural networks with neurons [37] as the basic units. The perceptron [38] is the earliest prototype of the neural network and is known as the single-layer neural network (without hidden layers). It can only perform the simplest linear classification tasks. Improvements in computational power and data processing techniques have gradually made deep learning the most popular branch of machine learning in recent years in both academia and industry. With the birth of some famous network structures, such as the convolutional neural network (CNN) [39], generative adversarial network (GAN) [40], and recurrent neural network (RNN) [41], DL has further expanded its applications in different fields.

Deep reinforcement learning (DRL) combines RL and DL. For DRL, the NNs are embedded in RL and commonly used to restore knowledge from an environment and make a preferred decision based on the current situation. The deep q-learning network [42, 43] pioneered this kind of algorithm, and it can be applied in many areas [44], such as game [45] and design of vehicle network [46]. Then, their variant DDQN [47] was proposed to overcome the drawback that overwhelms the q value (defined by the expected return starting from state s, taking action a, and then following policy π). To reduce complexity and improve training efficiency, the continuous deep q-learning network with model-based acceleration (CDQN) was proposed in [48] by extending DRL from discrete to continuous space. Based on the advantage of CDQN, we propose the learned knowledge-based parameter control framework to realize parameter automatic tuning and improve the efficiency of MOEAs. At the same time, considering the temporal properties of the states in the evolutionary process, since RNNs have the ability to learn and perform complex transformations of data over long time scales, we choose it to handle the state during evolution. Figure 1 shows the graphical representation for RNN. In the figure, I_t represents the value of the input layer of the t_{th} generation, the S_t is the hidden layer of the t_{th} generation, and O_t represents the value of the output layer of the t_{th} generation.

The notations utilized in this paper are stated in Table 1.

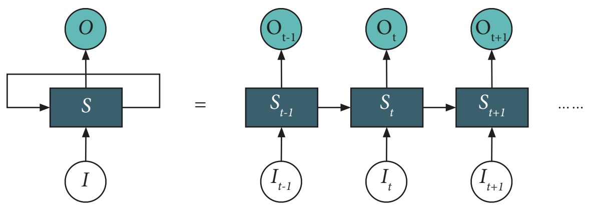

FIGURE 1: Compressed (left) and unfolded (right) basic recurrent neural network.

TABLE 1: The notations.

N	The population size
n	The dimension of objective function
$f(x)$	The objective function
$x_t^i \in \mathbb{R}^n$	The i_{th} individual in the t_{th} generation
$P_t \in \mathbb{R}^{N\times n}$	The population of the t_{th} generation
$F_t \in \mathbb{R}^N$	The fitness value of the t_{th} generation

4. Learned Knowledge-Based Parameter Control Framework via Deep Reinforcement Learning

4.1. The General Framework of Parameter Control. For MOEA, the parameter is usually set before iteration. After initialization, the population will be evaluated by the objective function, also known as a fitness value function. Then, the population starts an evolutionary process based on various designed methods. After that, the population will be evaluated again and generate a new Pareto set. Then, if the termination condition is not satisfied, the population will begin the next iteration. This represents the general process of MOEAs, although some specific MOEAs may slightly deviate from these steps. The flowchart of MOEAs is presented in Figure 2(a).

In this article, we embed DRL into the general process of MOEAs, as illustrated in Figure 2(b). The steps preceding the first termination condition assessment are similar to the general process, with the parameter for this round determined before iteration. After that, the population information is transferred to DRL. DRL will learn from this information and choose proper parameter sets for the next population P_{t+1} or individual x_t^{i+1}. Then, the chosen parameter set will be applied in the next iteration. In this process, the information transferred to DRL is defined as the state, and the chosen parameter sets are actions. Fitness evaluation and Pareto set generation are considered as the environment. Subsection 4.2 will introduce the details of the parameter control process and model the general evolutionary algorithm as an MDP.

4.2. Modeling the Parameter Control Process to MDP. This subsection clears the components needed in the parameter control process and provides evidence for every feature used in the proposed framework.

4.2.1. Environment. The environment includes an evolutionary computation process and a set of optimization problems (training functions). The optimization problems which are represented by objective functions are used to evaluate the performance of the optimizer. Note that these objective functions should have some common features such as the number of objectives and whether it is an integer or continuous programming problem. These common features could help to learn and apply.

4.2.2. State S. The state is used as the feature to describe the evolution process and provide evidence for the agent to choose the proper parameters. In real-world problems, the scope of decision space and Pareto fronts (PFs) are difficult to obtain. In this article, we select feature and process information that does not require prior knowledge about decision space and PFs for parametric decision-making.

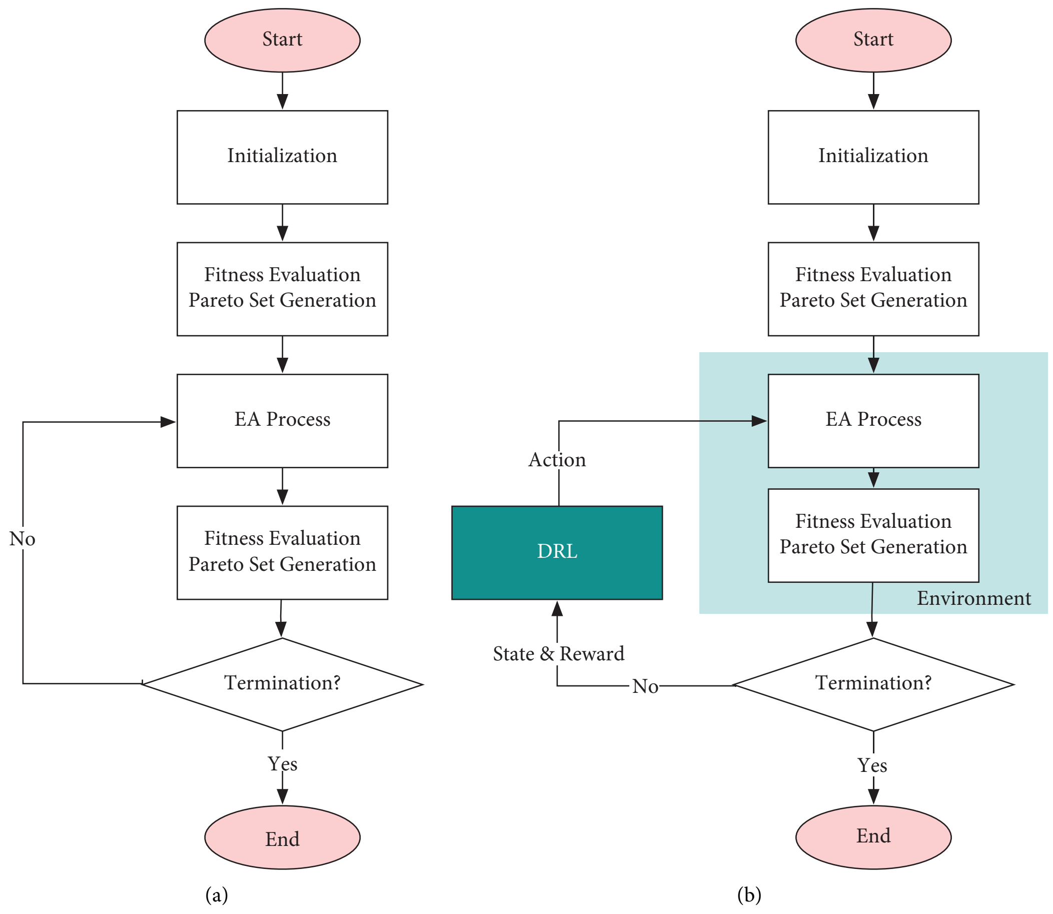

FIGURE 2: The flowchart of MOEA with and without parameter control. (a) The general flowchart of MOEA. (b) The flowchart of MOEA with parameter control.

Except for some basic information about considered problems, we choose the relative position in decision space, the distribution of the fitness value for this individual in the past n generations, and the grid-based inverted generational distance (grid-IGD) [49] of this individual.

(1) The Basic Information about Considered Problems. This kind of feature includes the number of objectives and the number of dimensions. The above information helps clarify the difficulty of the problems.

(2) The Relative Position of x_t^i. It is the basic feature of the state. The relative position in the decision space can be described by $(x_t^i - x_u)/(x_u - x_l)$, where x_u and x_l, respectively, mean the upper and lower bounds of the decision space.

(3) The Distribution of Fitness Value. This value is reflected by the distribution of fitness values of the whole population in the past n generations. This index divides $[f_{\min}, f_{\max}]$ into n equal parts and counts the number of individuals on the PFs in each part, respectively. $f_{\min}$ and $f_{\max}$ separately represent the minimum and maximum fitness values found in the current round. This feature helps clarify the scope of the fitness space.

(4) Grid-IGD. Grid-IGD is introduced to steer the evolutionary direction for unknown PF problems. Grid-IGD generated a set of reference points to estimate the PFs of the considered problem. Since grid-IGD generated representative nondominated solutions in the gird environment, it can help the agent to know the quality of the current solution set without knowing the true Pareto sets.

4.2.3. *Action A and Policy π.*

For an MDP, the agent can sample or choose an action from the policy π defined as a probability distribution $p(A_t \mid S_t; \theta^r)$ under the state S_t, where θ^r is the parameter for the policy. In this article, the action A_t^i is the proper parameter set in the t_{th} generation for the i_{th} individual. For different MOEAs, the parameters that need to be adaptively modified are different. For A_t^i, the number and scope for each parameter are defined before training and should keep the same while testing on real problems.

The policy π is a distribution among actions under different states. It can be described by the following formula:

$$\pi(a \mid s) = \mathbb{P}[A = a \mid S = s]. \qquad (2)$$

4.2.4. Reward R. The reward for MDP can be described by formula (3), in which R_s^a means the expected reward gained under the state s with the action a:

$$R_{s,t+1}^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]. \tag{3}$$

For multiobjective optimization problems, when a solution transfers from an nondominated solution to a dominated one, this situation will be considered a success and should be given a reward. At the same time, if the number of dominated solutions in the archive at this iteration increases, this situation should also be awarded. We take into consideration these two factors and design feedback reward with memory. The rewards not only consider the situation for now but also compare it with history memory. Thus, the reward proposed in this article is described by the following formula:

$$R_t^i = \begin{cases} 10, & \text{if the } i_{\text{th}} \text{ individual becomes dominated solution in the } t_{\text{th}} \text{ generation,} \\ 5, & \text{else if the } i_{\text{th}} \text{ individual keeps dominating in the } \mathrm{t}_{\text{th}} \text{ generation,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The reward designed in (3) encourages the agent to keep the individual nondominated and try to evolve more nondominated solutions.

4.2.5. Transmission Probability P. The transmission probability P represents the probability transferred from the state s to a new state s', and it can be described by the following formula:

$$P_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]. \tag{5}$$

In this article, since the state space of MOPs is too large to measure, P is hard to forecast. Thus, we need to choose model-free reinforcement learning methods to make decisions under different circumstances.

4.3. Embedding Continuous q-Learning with Normalized Advantage Functions

4.3.1. Training Phase. The model-free RL method is suitable for problems where the environment is unknown or the environment is difficult to accurately describe and explore. It has been developed to policy value functions and large neural networks, which makes it possible to directly pass raw representations as input to neural networks to access policies for complex problems. In this article, based on the feature of state space and continuity features of the parameters, we choose CDQN as the parameter controller, identifying the environment and making parametric decisions.

With the embedded CDQN, Algorithm 1 presents the pseudocode for the proposed framework at the training phase. Lines 1–7 finish the process of initialization, including CDQN and the evolutionary algorithm. Then, the parameter sets a_t^i will be chosen by μ_{model}, and the action that maximizes the expected reward is always given by $\mu(x \mid \theta^{\mu})$. Then, the evolutionary algorithm will update the individual x_t^i according to a_t^i. Then, the state and reward will be updated. After that, the target network will be updated according to updated θ^Q, which is in lines 16-17. In this process, the termination condition can be decided by the demand for testing or training. For example, the termination condition could be defined by the number of iterations, the maximum number of evaluations, or the length of time.

4.3.2. Testing Phase. Since the knowledge from the training phase will be utilized in the testing phase, the procedure for the testing phase will load the parameters of CDQN obtained in the training stage. The pseudocode is summarized in Algorithm 2. The parameters for CDQN obtained from the training stage will be loaded before the start moment of the testing stage. However, compared with the training stage, the parameter will not be updated through the iterations.

4.4. Reinforced MOEAs. In this subsection, we will apply the proposed framework to four classical MOEAs to realize adaptive parameter control, namely, reinforced-NSGA-II (R-NSGA-II), reinforced-MOEA/D (R-MOEA/D), reinforced-MOPSO (R-MOPSO), and reinforced-MODE (R-MODE). The rationale for the four algorithms will be presented, followed by a description of the parameters incorporated into framework tuning. The framework of four reinforced algorithms is summarized in Figure 3.

4.4.1. Reinforced-NSGA-II. Different from NSGA-II [7], where crossover probability η_c and mutation probability η_m are mainly set before iteration, the proposed R-NSGA-II sets the above two parameters based on the proposed framework to realize adaptively tuning. In R-NSGA-II, the evolutionary process is mainly according to the mutation operator and crossover operator, in which the mutation operator changes the components of the individual according to the crossover probability η_c, and the crossover operator is randomly selected by the individuals after mutation according to the crossover probability η_m. After each iteration, the retained individuals are selected by sorting based on the nondominant rank value and the degree of crowding distance. This algorithm divides the population into a group of Pareto nondominant sets. An individual in a nondominant set is not dominated by any individual in the current or later nondominated set. The method is to select all nondominant individuals that are not dominated by any other individual each time, delete a nondominant set

```
     Input: The population size $N$, the times of iteration $T$, the times of episode $M$, the discounting rate $\tau$
     Output: Trained parameters of network $\theta$.
(1)  Initialize normalized $Q$ network $Q(s,a|\theta^{Q})$ randomly
(2)  Initialize target network $Q'$ with weight $\theta^{Q'} \leftarrow \theta^{Q}$
(3)  Initialize replay buffer
(4)  for episode = $1, M$ do
(5)    Initialize a random process $\mathcal{N}$ for action exploration
(6)    Initialize the population $P$ randomly
(7)    Receive initial observation state $s_1^1 \sim p(x_1^1)$
(8)    for $t = 1, T$ do
(9)      for $i = 1, N$ do
(10)       Select action $a_t^i = \mu(s_t^i|\theta^{\mu}) + \mathcal{N}_t^i$
(11)       Apply $a_t^i$ to update the individual $x_t^i$
(12)       Evaluate population and selection
(13)       Store transition $(s_t^i, a_t^i, r_t^i, s_t^{i+1})$ in $R$
(14)       Sample a random mini-batch of $m$ transitions from $R$
(15)       Set $y_t^i = r_t^i + \gamma V'(s_t^{i+1}|\theta^{Q'})$
(16)       Update $\theta^{Q}$ by minimizing the loss $L = 1/N \sum (y_t^i - Q(s_t^i, a_t^i|\theta^{Q}))^2$
(17)       Update the target network $\theta_{t+1}^{Q'} \leftarrow \tau\theta_t^{Q} + (1-\tau)\theta_t^{Q'}$
(18)       $i = i + 1$
(19)     end for
(20)     $t = t + 1$
(21)   end for
(22) end for
```

Algorithm 1: Pseudocode for the proposed framework of the training phase

```
     Input: The population size $N$, the times of iteration $T$, trained parameter of network $\theta$
     Output: The Pareto sets of MOP
(1)  Initialize normalized Q network with fixed weight $\theta$
(2)  Initialize target network $Q'$ with fixed weight $\theta^{Q'} \leftarrow \theta^{Q}$
(3)  Initialize replay buffer
(4)  for episode = $1, M$ do
(5)    Initialize a random process $\mathcal{N}$ for action exploration
(6)    Initialize the population $P$ randomly
(7)    Receive initial observation state $s_1^1 \sim p(x_1^1)$
(8)    for $t = 1, T$ do
(9)      for $i = 1, N$ do
(10)       Select action $a_t^i = \mu(s_t^i|\theta^{\mu}) + \mathcal{N}_t^i$
(11)       Apply $a_t^i$ to update the individual
(12)       Evaluate population and selection
(13)       $i = i + 1$
(14)     end for
(15)     $t = t + 1$
(16)   end for
(17)   episode = episode + 1
(18) end for
```

Algorithm 2: Pseudocode for the proposed framework of the testing phase

from the population, and then repeat the process of the remaining until termination condition meets. Then, the population is arranged by crowd distance, which is the sum of the distance between adjacent individuals in each dimension.

4.4.2. Reinforced-MOEA/D. Reinforced-MOEA/D is constructed based on MOEA/D [8] and the proposed framework. In R-MOEA/D, the multiobjective problem is divided into a set of single-objective subproblems or several multiobjective subproblems. Then, the framework is utilized to adjust the parameters in the simulated binary crossover (SBX) operators and polynomial mutation (PM) operators. After that, Pareto front is approached by optimizing the subproblems in collaborative ways with the neighborhood relationship between subproblems. In SBX, the two offspring are created using the following equations:

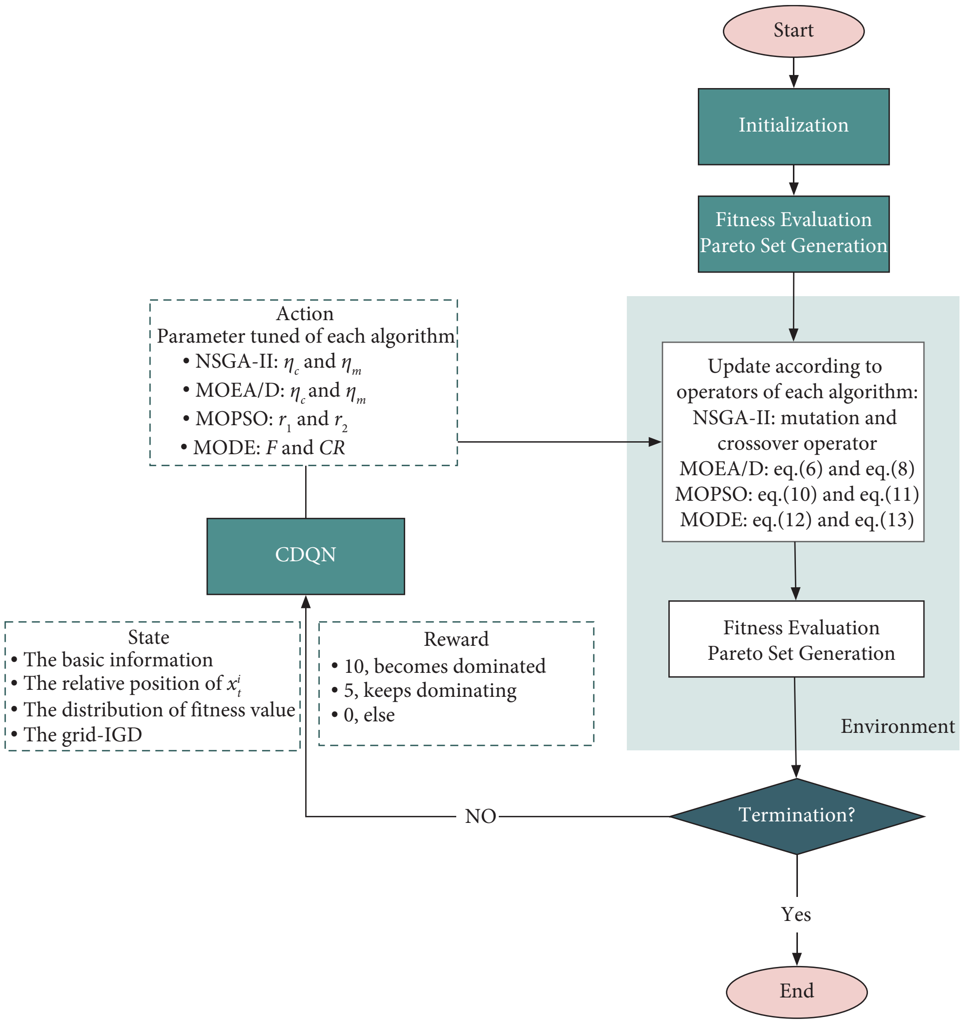
Start
Initialization
Fitness Evaluation
Pareto Set Generation
Action
Parameter tuned of each algorithm
• NSGA-II: η_c and η_m
• MOEA/D: η_c and η_m
• MOPSO: r_1 and r_2
• MODE: F and CR
Update according to operators of each algorithm:
NSGA-II: mutation and crossover operator
MOEA/D: eq.(6) and eq.(8)
MOPSO: eq.(10) and eq.(11)
MODE: eq.(12) and eq.(13)
CDQN
Fitness Evaluation
Pareto Set Generation
State
• The basic information
• The relative position of x_t^i
• The distribution of fitness value
• The grid-IGD
Reward
• 10, becomes dominated
• 5, keeps dominating
• 0, else
Environment
Termination?
NO
Yes
End

FIGURE 3: The flowchart of the reinforced framework.

$$u^{ik}_{a} = \frac{1}{2}\left[(1+\beta)p^{ik}_{a} + (1-\beta)p^{ik}_{b}\right],$$
$$u^{ik}_{b} = \frac{1}{2}\left[(1-\beta)p^{ik}_{a} + (1+\beta)p^{ik}_{b}\right], \tag{6}$$

where u^{ik}_{a} and u^{ik}_{b} are the offspring after the SBX, p^{ik}_{a} and p^{ik}_{b} are randomly selected parent individuals, β is the random number of expansion factors, and the value of β is determined by the following equation:

$$\beta = \begin{cases} (2r)^{1/\eta_c+1}, & \text{if } r \le 0.5, \\ \left[\dfrac{1}{2(1-r)}\right]^{1/\eta_c+1}, & \text{otherwise,} \end{cases} \tag{7}$$

where r is a random value between [0, 1] and η_c represents the distribution index in SBX tuned by the proposed framework. When η_c is larger, the offspring will be more similar to their parent. Conversely, when the value of η_c is smaller, the offspring will tend to be different from their parents. The formula for polynomial mutation is shown as follows:

$$v_t^i = p_t^i + \delta \times (u_t - l_t), \tag{8}$$

where p_t^i is the individual before the mutation, v_t^i is the individual after the mutation, and u_t and l_t denote the upper and lower bounds of the individual, respectively:

$$\delta = \begin{cases} \left[2r + (1-2r)(1-\delta_1)^{\eta_m+1}\right]^{1/\eta_m+1} - 1, & \text{if } r \le 0.5, \\ 1 - \left[2(1-r) + 2(r-0.5)(1-\delta_2)^{\eta_m+1}\right]^{1/\eta_m+1}, & \text{if } r > 0.5, \end{cases} \tag{9}$$

where r is a random value between [0, 1], η_m represents the distribution index in PM tuned adaptively by the parameter control framework, $\delta_1 = (p_t^i - l_t)/(u_t - l_t)$, and $\delta_2 = (u_t - p_t^i)/(u_t - l_t)$.

4.4.3. Reinforced-MOPSO. R-MOPSO is proposed by combining MOPSO [5] with the proposed parameter control framework. In R-MOPSO, the position and velocity of each particle are separately described by the following equations:

$$V_t = w \times P_t + r_1 \times \left(P_t^{\text{best}} - \text{POP}_t^i\right) + r_2 \times \left(\text{REP}(h) - \text{POP}_t^i\right), \tag{10}$$

$$P_{t+1} = P_t + V_t, \tag{11}$$

where w is the inertia weight, $r_1, r_2 \in [0,1]$ are dynamic parameters adjusted by the proposed framework, $P_{\text{bests}}(i)$ is the best position of the particle i, $\text{REP}(h)$ is a value that is taken from the repository, and the index h is selected by the following method: assigning a fitness value to those hypercubes containing more than one particle and dividing any number (greater than zero) by the number of particles that they contain.

4.4.4. Reinforced-MODE. R-MODE is composed of the original MODE [9] and the proposed parameter control framework. Similar to MODE, the main idea of R-MODE is to balance the degree of exploration and exploitation in the evolution process by selecting the learning particles and the learning ratios. The mutation operator and the crossover operator are utilized to create the offspring. The individual formulated through mutation can be expressed by the following equation:

$$v_t^i = p_t^i + F \times \sum_{k=1}^{2} \left(p_t^{i_a^k} - p_t^{i_b^k}\right), \tag{12}$$

where F is the scaling factor of disturbance which is adaptively tuned in the parameter control framework, $p^{i_a^k}$ and $p^{i_b^k}$ are randomly selected from parent individuals, and v_t^i is the generated particle after mutation.

Binomial crossover is one of the frequently used crossover operators and can be described by the following equation:

$$u_{t+1}^i = \begin{cases} v_t^i, & \text{if } r\,\text{and}_i[0,1] \le \text{CR}, \\ x_t^i, & \text{otherwise}, \end{cases} \tag{13}$$

where $\text{rand}_i[0,1]$ is a uniformly distributed random number. When it is less than the crossover rate CR, the individual generated by the mutation operator will be chosen; otherwise, the original individual x_t^i is retained.

For MODE, F and CR are the parameters related to the algorithm's efficiency. We apply the framework to adaptively tune the parameters according to the state.

5. Experimental Study

In this section, the implementation details are presented first. The comparison results against classical MOEAs and their reinforced ones are presented afterwards. (The code will be published in https://github.com/velvet999 after the paper is accepted.)

5.1. Test Functions. ZDT [7], DTLZ [50], and walking fish group (WFG) [51] benchmarks are used to train and test the proposed framework. Specifically in this article, we choose ZDT1-ZDT4 and ZDT6 as the training sets and DTLZ1-DTLZ4 and WFG1-WFG8 as the testing sets.

5.2. Measure Metrics. Two widely used performance indicators, the inverted generational distance (IGD) [52] and hypervolume (HV) [53], are used to evaluate the quality of the obtained nondominated solution set, which can be able to account for both convergence (closeness to the true Pareto front) and the distribution of the achieved nondominated solutions.

5.2.1. Inverted Generational Distance. IGD is an integrated performance evaluation index that can evaluate the distribution and convergence of solutions simultaneously. IGD mainly evaluates the distribution performance and convergence performance of the algorithm by calculating the point (individual) to the individual settings from the real Pareto front side to the algorithm. The better the comprehensive performances of the algorithm, the smaller the value of IGD:

$$\text{IGD} = \frac{\sum_{i=1}^{n} |d_i|}{n}. \tag{14}$$

5.2.2. Hypervolume. The HV indicator (or s-metric) is a performance metric that indicates the quality of a nondominated approximation set, where it is described as the "size of the space covered or size of dominated space":

$$\text{HV}\left(f^{\text{ref}}, x\right) = \Lambda\left(\cup_{x_n \in P} \left[f_1(x_n), f_1^{\text{ref}}\right] \times \cdots \times \left[f_m(x_n), f_m^{\text{ref}}\right]\right), \tag{15}$$

where $\mathrm{HV}(f^{\mathrm{ref}}, x)$ resolves the size of the space covered by an approximation set x, $f_m^{\mathrm{ref}} \in \mathbb{R}$ refers to a chosen reference point of the m_{th} dimension, $f_m(x_n)$ is the fitness value of the individual n of the m_{th} dimension, and $\Lambda(.)$ refers to the Lebesgue measure.

5.3. Algorithms and Parameter Settings.

In this subsection, the general parameters are stated first, and then, we list the specific parameters used in the experiment for each comparison function. After that, the parameters of the proposed framework used in the experiment are provided.

5.3.1. Common Parameter Settings

(1) Number of variables and objectives: The number of the objectives for both DTLZ and WFG is 3, which is a common setting in multiobjective experiments. The number of variables of DTLZ is 30 and that of WFG is 10. The different number of variables can also test the adaptability and robustness of the algorithm.

(2) Statistical approach: Due to the heuristic characteristics of evolutionary algorithms, each algorithm independently performed 30 iterations on each function to overcome randomness. The Mann–Whitney–Wilcoxon rank-sum test [54] is employed for this purpose, and its significance level is 5%.

(3) Population size and the number of evaluations: The number of evaluations and population size are the same. The population size N is 100, and the maximum number of evaluations (MAXNFE) is 10,000.

5.3.2. Parameter Settings for Classical MOEAs.

All of the code and details of comparison algorithms can be found in pymoo [55] which is a multiobjective optimization tool in Python.

For NSGA-II, the crossover probability is specified as 1.0, with the mutation probability $1/n$ where n is the variable number.

For MOEA/D, SBX is chosen for crossover, and its probability is 0.9; polynomial mutation is used with a distribution index of 20 and a probability of 0.2.

For MOPSO, c_1 and c_2 are set to 1.49618 and $\omega = 0.729844$. The polynomial mutation with a mutation index $\mu_m = 20$ and probability $1/N$, where N is the population size.

For MODE, the crossover rate is 0.7, the mutation rate is 1/30, and the child variability factor is 0.7.

5.3.3. Parameter Settings for r-MOEAs.

Since the parameter of r-MOEAs is randomly generated at first and adaptively adjusted by the agent during the process, Figure 4 gives the detail of CDQN, which is the parameter controller in the proposed framework.

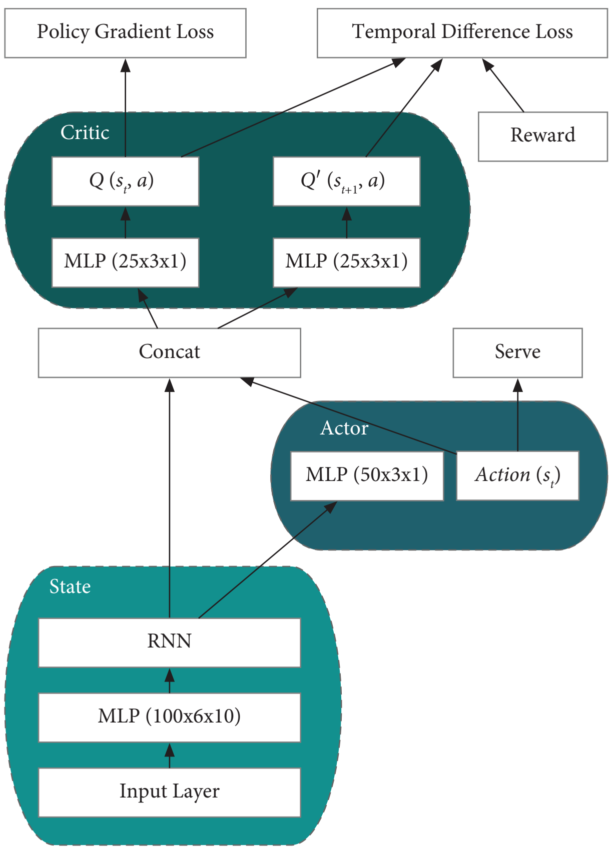

Figure 4: The structure of the neural network used in CDQN.

5.4. Comparison.

In this section, we present the results of the DTLZ benchmark generated by the comparison algorithms and the proposed framework-embedded algorithms. The statistical results of the IGD metric on 4 DTLZ test problems and 8 WFG test problems are summarized in Table 2, while those of HV are summarized in Table 3. The mean values, standard deviation values, and results of the Mann–Whitney–Wilcoxon rank-sum test (in parenthesis) are provided. For each testing problem, the Mann–Whitney–Wilcoxon rank-sum test is performed on the results obtained by one algorithm's original version and reinforced version instead of the results in the whole table. For example, the rank-sum test of NSGA-II is performed between the values of metrics obtained by NSGA-II and R-NSGA-II.

As Table 2 shows, the average IGDs obtained by reinforced algorithms are smaller than those by their original version. Since IGD is used to measure the quality of solution and uniformity of distribution of solutions, this indicates that the solution obtained by the reinforced version is more close to the real Pareto front. R-MOEA/D achieves the best performance on DTLZ1 and 4th place among 4 original and 4 reinforced algorithms, while R-NSGA-II achieves the best on DTLZ3. For the WFG benchmark, R-MOEA/D also achieves the best among 8 problems on 5 problems, while there are 8 questions in total. The other reinforced

Table 2: Means and standard deviation of IGD obtained by NSGA-II, MOEA/D, and their reinforced versions on DTLZ and WFG.

Problems	NSGA-II	R-NSGA-II	MOEA/D	R-MOEA/D
DTLZ1	3.35E − 02 ± 2.99E − 03(−)	8.63E − 03 ± 6.04E − 03	2.56E + 00 ± 2.83E − 01(−)	2.03E + 00 ± 2.53E − 01
DTLZ2	2.92E − 02 ± 1.62E − 02(−)	2.63E − 02 ± 1.46E − 02	3.13E − 02 ± 1.76E − 02(−)	**6.27E − 03 ± 3.52E − 03**
DTLZ3	8.57E + 00 ± 6.99E − 01(−)	**7.37E + 00 ± 7.34E − 01**	2.41E + 01 ± 3.83E − 01(−)	2.14E + 01 ± 2.17E + 00
DTLZ4	9.70E − 01 ± 1.63E − 02(−)	8.25E − 01 ± 4.94E − 02	7.11E − 02 ± 5.55E − 17(−)	**6.02E − 02 ± 3.55E − 03**
WFG1	1.56E + 00 ± 2.18E − 01(−)	1.33E + 00 ± 4.58E − 02	1.59E + 00 ± 1.72E − 01(−)	1.38E + 00 ± 1.59E − 01
WFG2	1.56E + 00 ± 1.55E − 01(−)	1.38E + 00 ± 4.71E − 02	5.37E − 01 ± 4.10E − 02(−)	4.80E − 01 ± 1.70E − 02
WFG3	3.49E − 01 ± 3.08E − 01(−)	7.94E − 02 ± 6.89E − 02	9.66E − 01 ± 3.14E − 01(−)	1.59E + 00 ± 3.93E − 01
WFG4	3.43E − 01 ± 2.92E − 02(−)	3.03E − 01 ± 1.14E − 02	3.00E − 01 ± 3.00E − 02(−)	**2.44E − 01 ± 2.82E − 02**
WFG5	3.52E − 01 ± 4.71E − 02(−)	2.91E − 01 ± 1.14E − 02	3.16E − 01 ± 1.89E − 02(−)	**2.76E − 01 ± 8.88E − 03**
WFG6	3.57E − 01 ± 2.56E − 02(−)	3.35E − 01 ± 2.42E − 02	3.39E − 01 ± 2.18E − 02(−)	**3.02E − 01 ± 1.17E − 02**
WFG7	4.62E − 01 ± 3.19E − 02(−)	4.05E − 01 ± 1.41E − 02	2.70E − 01 ± 2.07E − 02(−)	**2.38E − 01 ± 8.83E − 03**
WFG8	6.04E − 01 ± 1.12E − 01(−)	5.31E − 01 ± 9.67E − 02	6.02E − 01 ± 1.11E − 01(−)	**5.26E − 01 ± 9.60E − 02**
+\≈\−	0\0\12		0\0\12	
Problems	MOPSO	R-MOPSO	MODE	R-MODE
DTLZ1	3.33E − 02 ± 2.48E − 03(−)	**8.48E − 03 ± 5.08E − 03**	1.44E − 02 ± 9.89E − 03(−)	1.12E − 02 ± 5.04E − 03
DTLZ2	3.31E − 02 ± 1.90E − 02(−)	2.98E − 02 ± 1.71E − 02	2.86E − 02 ± 1.93E − 02(−)	1.43E − 02 ± 1.52E − 02
DTLZ3	2.49E + 01 ± 5.85E + 00(−)	2.29E + 01 ± 5.39E + 00	2.72E + 01 ± 2.96E + 00(−)	2.38E + 01 ± 3.18E + 00
DTLZ4	7.19E − 01 ± 1.39E − 01(−)	6.05E − 01 ± 1.22E − 01	7.59E − 01 ± 1.68E − 01(−)	6.52E − 01 ± 1.50E − 01
WFG1	1.89E + 00 ± 1.22E − 01(−)	**1.14E + 00 ± 7.60E − 02**	3.27E + 00 ± 3.15E − 01(−)	1.41E + 00 ± 9.79E − 02
WFG2	2.91E − 01 ± 8.44E − 02(−)	**2.47E − 01 ± 6.48E − 02**	2.59E + 00 ± 3.04E − 01(−)	1.97E + 00 ± 1.89E − 01
WFG3	2.35E − 02 ± 1.13E − 02(−)	**5.19E − 03 ± 2.55E − 03**	3.53E − 01 ± 1.21E − 01(−)	6.21E − 01 ± 1.53E − 01
WFG4	7.59E − 01 ± 1.44E − 01(−)	4.37E − 01 ± 8.81E − 02	1.26E + 00 ± 2.56E − 01(−)	5.45E − 01 ± 1.08E − 01
WFG5	1.35E + 00 ± 1.95E − 01(−)	8.29E − 01 ± 1.17E − 01	2.36E + 00 ± 3.26E − 01(−)	1.04E + 00 ± 1.44E − 01
WFG6	7.01E − 01 ± 1.01E − 01(−)	4.02E − 01 ± 5.91E − 02	8.76E − 01 ± 1.29E − 01(−)	5.04E − 01 ± 7.31E − 02
WFG7	8.04E − 01 ± 1.04E − 01(−)	4.67E − 01 ± 6.36E − 02	1.33E + 00 ± 2.05E − 01(−)	5.83E − 01 ± 7.66E − 02
WFG8	1.08E + 00 ± 2.53E − 01(−)	6.26E − 01 ± 1.51E − 01	1.81E + 00 ± 4.47E − 01(−)	7.84E − 01 ± 1.84E − 01
+\≈\−	0\0\12		0\0\12	

Bold value represents the best performance for each problem.

Table 3: Means and standard deviation of HV obtained by NSGA-II, MOEA/D, and their reinforced versions on DTLZ and WFG.

Problems	NSGA-II	R-NSGA-II	MOEA/D	R-MOEA/D
DTLZ1	5.51E − 01 ± 3.20E − 01(−)	4.21E − 01 ± 2.10E − 01	4.49E − 01 ± 3.64E − 01(−)	2.15E − 01 ± 2.64E − 01
DTLZ2	5.55E − 01 ± 7.05E − 04(−)	5.20E − 01 ± 7.05E − 04	5.32E − 01 ± 3.89E − 03(−)	**6.32E − 01 ± 3.29E − 03**
DTLZ3	0.00E + 00 ± 0.00E + 00(≈)	**1.34E − 01 ± 2.20E − 03**	0.00E + 00 ± 0.00E + 00(≈)	0.00E + 00 ± 0.00E + 00
DTLZ4	3.86E − 01 ± 3.49E − 01(−)	4.66E − 01 ± 2.10E − 01	5.05E − 01 ± 9.09E − 01(−)	6.75E − 01 ± 1.09E − 01
WFG1	5.83E − 01 ± 5.87E − 02(−)	6.46E − 01 ± 3.22E − 02	6.95E − 01 ± 4.64E − 02(−)	**9.24E − 01 ± 4.55E − 02**
WFG2	8.22E − 01 ± 3.56E − 02(−)	9.12E − 01 ± 5.18E − 03	9.06E − 01 ± 5.31E − 03(−)	1.19E + 00 ± 5.31E − 03
WFG3	2.83E − 01 ± 3.54E − 02(−)	3.58E − 01 ± 7.89E − 03	3.77E − 01 ± 8.98E − 03(−)	**5.66E − 01 ± 8.62E − 03**
WFG4	4.92E − 01 ± 6.72E − 03(−)	5.32E − 01 ± 3.14E − 03	5.01E − 01 ± 4.65E − 03(−)	6.62E − 01 ± 4.23E − 03
WFG5	4.79E − 01 ± 5.55E − 03(−)	5.07E − 01 ± 4.15E − 03	4.80E − 01 ± 5.29E − 03(−)	**7.11E − 01 ± 5.29E − 03**
WFG6	4.43E − 01 ± 1.90E − 02(−)	4.79E − 01 ± 1.75E − 02	4.48E − 01 ± 1.83E − 02(−)	6.58E − 01 ± 1.65E − 02
WFG7	4.23E − 01 ± 2.68E − 02(−)	5.37E − 01 ± 2.92E − 03	5.12E − 01 ± 5.42E − 03(−)	**7.01E − 01 ± 4.99E − 03**
WFG8	4.10E − 01 ± 1.51E − 02(−)	4.45E − 01 ± 3.72E − 03	4.25E − 01 ± 5.00E − 03(−)	5.99E − 01 ± 4.65E − 03
+\≈\−	0\1\11		0\1\11	
Problems	MOPSO	R-MOPSO	MODE	R-MODE
DTLZ1	0.00E + 00 ± 0.00E + 00(−)	0.00E + 00 ± 0.00E + 00	6.75E − 01 ± 1.20E − 01(−)	**5.91E − 01 ± 2.21E − 01**
DTLZ2	4.30E − 01 ± 3.53E − 02(−)	5.30E − 01 ± 2.13E − 02	2.15E − 01 ± 4.05E − 04(−)	5.85E − 01 ± 6.15E − 04
DTLZ3	0.00E + 00 ± 0.00E + 00(≈)	0.00E + 00 ± 0.00E + 00	0.00E + 00 ± 0.00E + 00(≈)	0.00E + 00 ± 0.00E + 00
DTLZ4	4.33E − 01 ± 7.90E − 02(−)	**8.75E − 01 ± 4.10E − 02**	3.19E − 01 ± 1.88E − 01(−)	6.69E − 01 ± 1.38E − 01
WFG1	1.73E − 01 ± 3.01E − 02(−)	2.30E − 01 ± 2.92E − 02	3.83E − 01 ± 8.41E − 02(−)	5.14E − 01 ± 4.83E − 01
WFG2	8.39E − 01 ± 1.69E − 02(−)	1.10E + 00 ± 1.64E − 02	8.81E − 01 ± 2.75E − 02(−)	**1.32E + 00 ± 1.30E + 00**
WFG3	1.78E − 01 ± 3.49E − 02(−)	2.48E − 01 ± 3.32E − 02	3.27E − 01 ± 2.28E − 02(−)	4.78E − 01 ± 4.30E − 01
WFG4	3.69E − 01 ± 2.22E − 02(−)	5.02E − 01 ± 2.06E − 02	4.95E − 01 ± 2.89E − 02(−)	**7.92E − 01 ± 7.13E − 01**
WFG5	1.25E − 01 ± 1.91E − 02(−)	1.72E − 01 ± 1.73E − 02	4.83E − 01 ± 1.59E − 02(−)	6.49E − 01 ± 6.49E − 01
WFG6	3.58E − 01 ± 2.63E − 02(−)	5.01E − 01 ± 2.56E − 02	4.29E − 01 ± 4.24E − 02(−)	**6.96E − 01 ± 6.89E − 01**
WFG7	3.25E − 01 ± 4.41E − 02(−)	4.77E − 01 ± 4.05E − 02	4.91E − 01 ± 4.37E − 02(−)	5.46E − 01 ± 5.07E − 01
WFG8	2.50E − 01 ± 1.69E − 02(−)	3.67E − 01 ± 1.70E − 02	4.11E − 01 ± 2.75E − 02(−)	**7.16E − 01 ± 6.51E − 01**
+\≈\−	0\1\11		0\1\11	

Bold value represents the best performance for each problem.

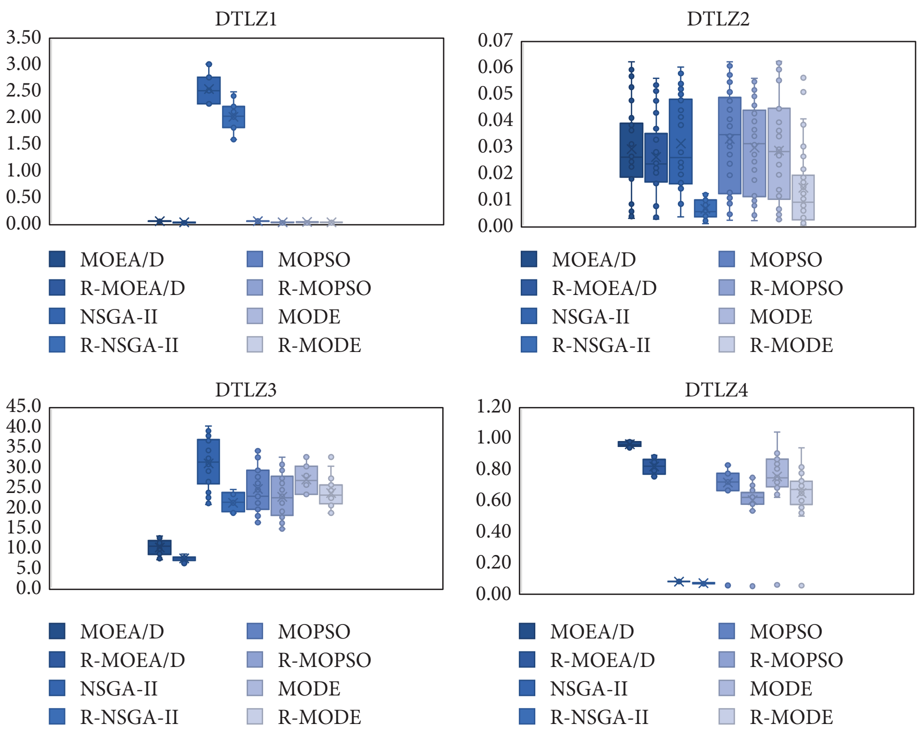

Figure 5: The boxplot of IGD obtained by four original algorithms and their reinforced version on DTLZ.

algorithms may not achieve the best on the problems, but they can also obtain better results than their original ones. For DTLZ3, a multimodal problem, all algorithms do not perform exceptionally well, indicating that while the framework can enhance the performance of algorithms, this improvement has limitations on problems that the algorithm inherently struggles with.

Figures 5 and 6 show the boxplot of metrics IGD obtained by 30 runs. It can be observed that IGD obtained by reinforced algorithms owns more stability and superiority than their classical ones. For all problems, the reinforced algorithm obtained the best results. From the figures, we can clearly observe that the scope of IGD obtained by the reinforced algorithm is smaller than that by the original ones, which shows that the reinforced algorithm is more stable during the 30 runs.

HV can measure the convergence and diversity of an algorithm simultaneously. The means and standard deviation of HV obtained by four classical algorithms and their reinforced version on DTLZ and WFG over 30 runs are presented in Table 3. The reinforced algorithms get promoted on most problems compared with their original ones.

In addition to verifying the transferability of the framework on different problems, we further validated its performance on the same problem but with the different number of variables. We choose NSGA-II, MOPSO, and their reinforced algorithms to run on WFG of 5 (6 for WFG2 and WFG3), 20, and 30 variables. The results are summarized in Table 4. From Table 4, we can see that the reinforced algorithms are more likely to achieve success on the same problem even with different variables.

Meanwhile, we also applied the Friedman test [56] to the results. Table 5 and Figure 7 summarize the average ranking of each of the eight algorithms on all problems from the two test suites, where differences in their performance are detected. The lower the ranking, the better the performance of an algorithm. It is worth noting that the Mann–Whitney–Wilcoxon rank-sum test is used to compare the performance of only two algorithms at a time, while the Friedman test is applied to rank all algorithms based on their overall performance. The reinforced algorithms show better performance than their classical ones clearly.

5.5. Further Analysis. From the boxplot, it is not hard to observe that the algorithm with the embedded framework is better than the original algorithm on both the mean value and the standard deviation value. This further validates the generality and flexibility of the proposed framework. From this perspective, the algorithm framework is a meaningful innovation. Deep reinforcement learning can make the right choice under complex circumstances during the iteration process. Compared with the rule-based parameter control method or iteration memory-based parameter control method, this framework with deep reinforcement learning holds more scalability and flexibility and can be further adjusted according to specific problems and algorithms. At the same time, it can be further concluded that with the designed framework, automatically selected parameters will improve both the convergence and robustness of the algorithm.

While training does require a certain amount of time and computational resources, early offline training has an enhancing effect on the results in subsequent applications. In

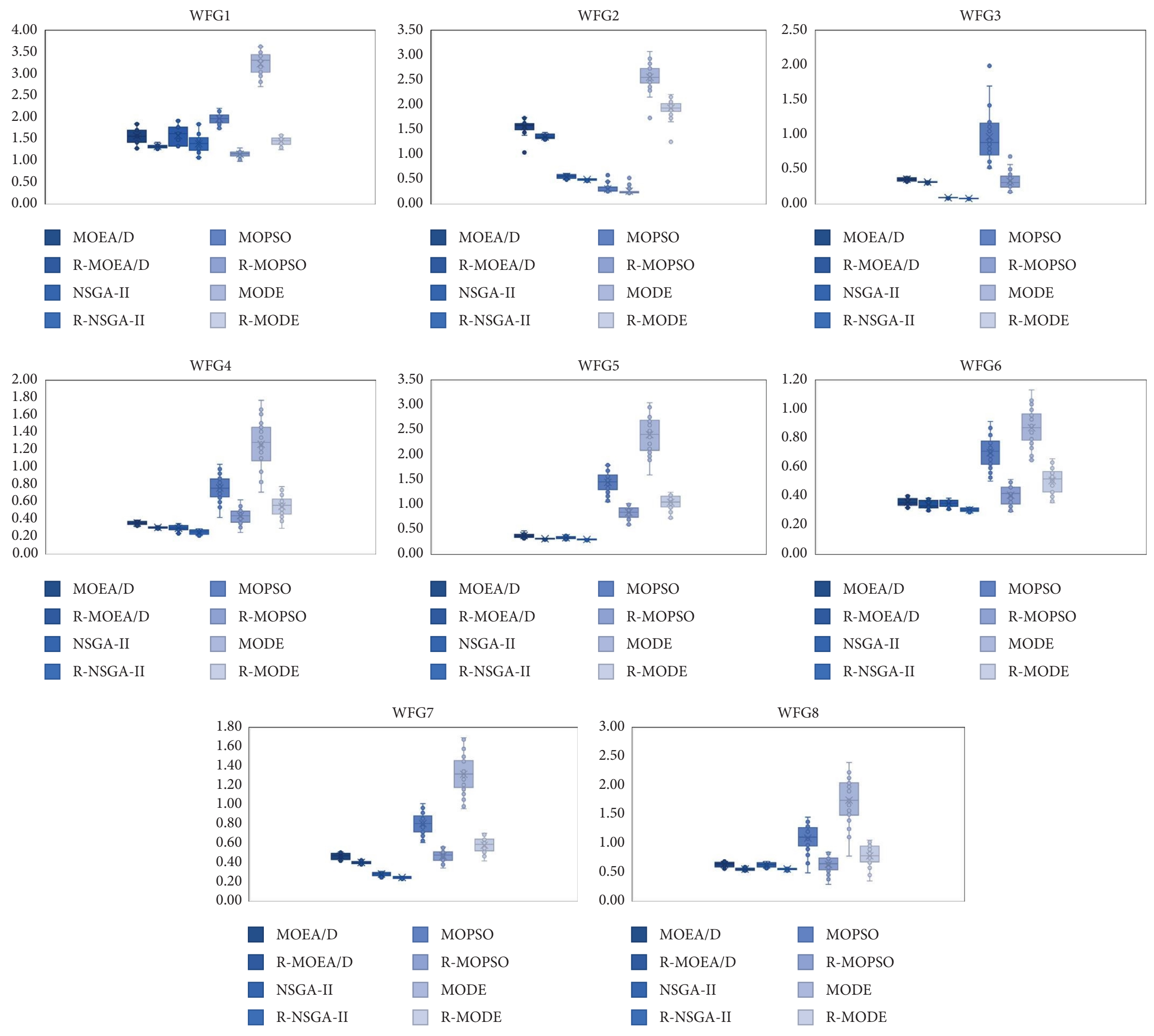

Figure 6: The boxplot of IGD obtained by four original algorithms and their reinforced version on WFG.

Table 4: Means and standard deviation of IGD obtained by NSGA-II, MOPSO, and their reinforced versions on WFG of different variables.

Function	Variables	NSGA-II	R-NSGA-II	MOPSO	R-MOPSO
	5	$2.84E-01 \pm 2.84E-02(+)$	$3.13E-01 \pm -1.61E-01$	$1.35E+00 \pm 2.51E-01(-)$	$1.11E+00 \pm -3.28E-02$
WFG1	20	$8.27E-01 \pm 6.87E-02(-)$	$6.61E-01 \pm -2.02E-01$	$1.85E+00 \pm 5.55E-02(-)$	$1.76E+00 \pm -5.85E-02$
	30	$1.06E+00 \pm 7.45E-02(+)$	$1.17E+00 \pm -1.56E-01$	$1.88E+00 \pm 8.15E-02(-)$	$1.80E+00 \pm -8.06E-02$
	6	$2.16E-01 \pm 1.03E-02(-)$	$1.30E-01 \pm -2.89E-01$	$2.02E-01 \pm 9.47E-03(+)$	$4.82E-01 \pm -6.29E-02$
WFG2	20	$2.40E-01 \pm 1.32E-02(-)$	$1.92E-01 \pm -9.00E-02$	$2.50E-01 \pm 1.12E-02(+)$	$7.13E-01 \pm -6.49E-02$
	30	$2.75E-01 \pm 5.82E-02(-)$	$2.48E-01 \pm -5.90E-02$	$2.74E-01 \pm 1.27E-02(\approx)$	$2.66E-01 \pm -4.70E-02$
	6	$9.18E-02 \pm 1.35E-02(-)$	$8.26E-02 \pm -2.10E-02$	$2.50E-01 \pm 9.47E-02(-)$	$1.41E-01 \pm -1.52E-02$
WFG3	20	$1.83E-01 \pm 2.41E-02(+)$	$2.01E-01 \pm -7.05E-02$	$3.52E-01 \pm 6.74E-02(+)$	$5.17E-01 \pm -5.91E-02$
	30	$2.40E-01 \pm 3.15E-02(-)$	$1.68E-01 \pm -3.67E-02$	$4.10E-01 \pm 8.75E-02(\approx)$	$4.00E-01 \pm -3.52E-02$
	5	$2.69E-01 \pm 8.19E-03(-)$	$2.42E-01 \pm -2.93E-02$	$3.60E-01 \pm 3.63E-02(-)$	$3.41E-01 \pm -3.57E-02$
WFG4	20	$2.89E-01 \pm 8.91E-03(-)$	$1.74E-01 \pm -1.96E-02$	$4.21E-01 \pm 5.20E-02(+)$	$4.78E-01 \pm -9.39E-03$
	30	$3.02E-01 \pm 1.17E-02(-)$	$1.81E-01 \pm -2.06E-02$	$4.18E-01 \pm 4.32E-02(+)$	$4.81E-01 \pm -2.81E-02$
	5	$2.79E-01 \pm 8.30E-03(-)$	$2.51E-01 \pm -3.26E-02$	$3.64E-01 \pm 3.49E-02(-)$	$2.39E-01 \pm -3.20E-02$
WFG5	20	$2.95E-01 \pm 1.10E-02(-)$	$1.77E-01 \pm -6.80E-02$	$9.76E-01 \pm 5.96E-02(-)$	$3.86E-01 \pm -3.63E-02$
	30	$3.09E-01 \pm 1.08E-02(\approx)$	$3.08E-01 \pm -3.46E-02$	$1.10E+00 \pm 6.34E-02(-)$	$4.64E-01 \pm -4.64E-02$

Table 4: Continued.

Function	Variables	NSGA-II	R-NSGA-II	MOPSO	R-MOPSO
WFG6	5	$2.67E-01 \pm 8.35E-03(-)$	$1.60E-01 \pm -7.32E-02$	$3.20E-01 \pm 4.22E-02(-)$	$2.86E-01 \pm -1.57E-02$
	20	$3.37E-01 \pm 1.48E-02(-)$	$2.02E-01 \pm -2.61E-02$	$3.77E-01 \pm 3.89E-02(+)$	$4.87E-01 \pm -4.15E-02$
	30	$3.48E-01 \pm 1.33E-02(-)$	$2.78E-01 \pm -2.17E-02$	$4.08E-01 \pm 3.66E-02(-)$	$3.02E-01 \pm -3.08E-02$
WFG7	5	$2.75E-01 \pm 8.09E-03(-)$	$1.37E-01 \pm -7.69E-02$	$3.12E-01 \pm 2.85E-02(-)$	$2.63E-01 \pm -2.91E-02$
	20	$2.89E-01 \pm 1.16E-02(+)$	$3.18E-01 \pm -2.63E-02$	$4.38E-01 \pm 6.28E-02(+)$	$6.52E-01 \pm -3.29E-02$
	30	$3.21E-01 \pm 3.23E-02(-)$	$2.89E-01 \pm -1.84E-02$	$4.51E-01 \pm 4.36E-02(+)$	$6.13E-01 \pm -2.93E-02$
WFG8	5	$4.16E-01 \pm 7.84E-02(-)$	$3.74E-01 \pm -4.67E-02$	$7.63E-01 \pm 4.97E-02(-)$	$6.29E-01 \pm -4.44E-02$
	20	$3.60E-01 \pm 1.39E-02(-)$	$1.80E-01 \pm -2.89E-01$	$5.71E-01 \pm 3.87E-02(-)$	$4.83E-02 \pm -2.98E-02$
	30	$3.63E-01 \pm 1.38E-02(-)$	$3.27E-01 \pm -3.25E-02$	$5.53E-01 \pm 3.48E-02(-)$	$2.62E-02 \pm -3.89E-02$
+\≈\−		4\1\19		8\2\14	

Table 5: Average rankings of IGD and HV by the Friedman test.

Algorithm	IGD ranking	HV ranking
MOEA/D	5.25	5.31
R-MOEA/D	3.43	3.39
NSGA-II	5.70	5.77
R-NSGA-II	3.57	3.62
MOPSO	5.55	5.52
R-MOPSO	3.72	3.57
MODE	5.15	5.06
R-MODE	3.64	3.72

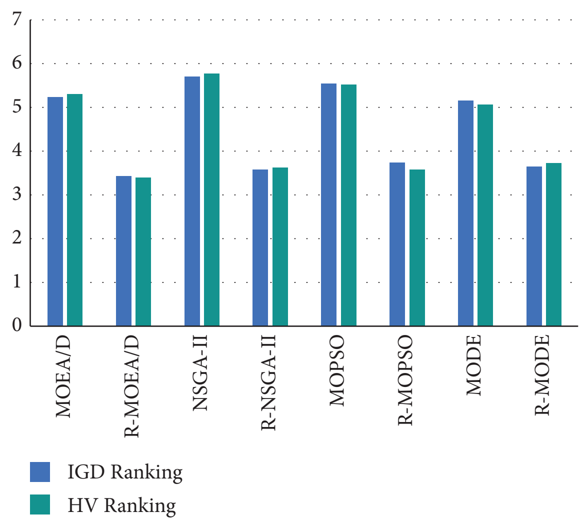

Figure 7: Average rankings of all algorithms obtained by the Friedman test on all the test functions.

practical application scenarios, such as drones and irregular flight recovery, it is feasible to obtain past data, and the time requirement for offline training is not rigorous. Therefore, at the training level, it is feasible to train the model based on our proposed reinforced algorithms. While at the application level, DRL can assist the evolutionary algorithm in choosing the right parameters for better results in specific problem optimization.

As for computation time in the testing phase, with the embedding of DRL, the computation time has increased. But in some practical scenarios, such as power system optimization [57] and supply chain management [58], accuracy is more important than timeliness. At the same time, with the increase in computing power, it also provides more chances for pursuing accuracy.

6. Conclusion

This paper presents a novel parameter control framework for MOEAs. The framework utilizes the ability of deep reinforcement learning to choose proper parameters under high-dimensional state features. We clear every component of the Markov decision process including the environment, state, action, reward, and transmission probability and employ a classic and recognized deep reinforcement learning algorithm to process the state and make choices in continuous space.

We introduced four reinforced MOEAs based on classical MOEAs with the proposed framework to verify the universality and validity of the designed framework. R-MOEA/D, R-MOPSO, R-NSGA-II, and R-MODE are trained and compared with their original algorithms. The experimental results demonstrate that the proposed framework can adapt to different algorithms, improving their efficiency and robustness on various testing problems after training. As observed from the boxplot, the efficiency of the improved algorithm is always better than that of their original algorithm. That further proves the universality of the proposed framework.

Regarding future studies, the applicability of the parameter control framework for different kinds of problems, such as integer optimization, will also be studied. Moreover, some real-world problems will be considered as the training and testing sets. Since real-world problems are more complex, it will be a challenge for state feature design.

Data Availability

Data are available on request. Please contact Wenwen Zhang at velvet999@126.com.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 62103286, 72334004, 71971143, 62371305, 62001302, 72004072, and

72374143, the Social Science Youth Foundation of the Ministry of Education of China under Grant No. 21YJC630181, the Guangdong Provincial Philosophy and Social Sciences Planning Project under Grant Nos. GD22XGL22 and GD22CGL35, the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2021A1515011348, the Guangdong Province Innovation Team Intelligent Management and Interdisciplinary Innovation under Grant No. 2021WCXTD002, and the Special Projects in Key Fields of Ordinary Colleges and Universities in Guangdong Province under Grant No. 2022ZDZX2054.

References

[1] Q. He, Z. He, S. Duan, and Y. Zhong, "Multi-objective interval portfolio optimization modeling and solving for margin trading," *Swarm and Evolutionary Computation*, vol. 75, 2022 https://www.sciencedirect.com/science/article/pii/S2210650222001110, Article ID 101141.

[2] B. Jiang, H. Lei, W. Li, and R. Wang, "A novel multi-objective evolutionary algorithm for hybrid renewable energy system design," *Swarm and Evolutionary Computation*, vol. 75, 2022 https://www.sciencedirect.com/science/article/pii/S2210650222001535, Article ID 101186.

[3] W. Yu, L. Zhang, and N. Ge, "An adaptive multiobjective evolutionary algorithm for dynamic multiobjective flexible scheduling problem," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 12335–12366, 2022.

[4] R. C. Narayanan, N. Ganesh, R. Čep, P. Jangir, J. S. Chohan, and K. Kalita, "A novel many-objective sine–cosine algorithm (maosca) for engineering applications," *Mathematics*, vol. 11, no. 10, p. 2301, 2023.

[5] C. Coello Coello and M. Lechuga, "Mopso: a proposal for multiple objective particle swarm optimization," *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1051–1056, 2002.

[6] B. Niu, H. Wang, J. Wang, and L. Tan, "Multi-objective bacterial foraging optimization," *Neurocomputing*, vol. 116, pp. 336–345, 2013, https://www.sciencedirect.com/science/article/pii/S0925231212006844.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[8] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[9] F. Xue, A. Sanderson, and R. Graves, "Pareto-based multi-objective differential evolution," *The 2003 Congress on Evolutionary Computation*, vol. 2, pp. 862–869, 2003.

[10] S. Mirjalili, P. Jangir, and S. Saremi, "Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems," *Applied Intelligence*, vol. 46, no. 1, pp. 79–95, 2017.

[11] G. Wang, X. Li, L. Gao, and P. Li, "Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified MOEA/D," *Swarm and Evolutionary Computation*, vol. 62, 2021 https://www.sciencedirect.com/science/article/pii/S2210650221000195, Article ID 100858.

[12] E. Jiang, L. Wang, and J. Wang, "Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 646–663, 2021.

[13] R. Massobrio, J. Toutouh, S. Nesmachnow, and E. Alba, "Infrastructure deployment in vehicular communication networks using a parallel multiobjective evolutionary algorithm," *International Journal of Intelligent Systems*, vol. 32, no. 8, pp. 801–829, 2017.

[14] N. Veček, M. Mernik, B. Filipič, and M. Črepinšek, "Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms," *Information Sciences*, vol. 372, pp. 446–469, 2016, https://www.sciencedirect.com/science/article/pii/S0020025516306430.

[15] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952, http://www.jstor.org/stable/2334029.

[16] V. Nannen and A. E. Eiben, "Relevance estimation and value calibration of evolutionary algorithm parameters," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07*, pp. 975–980, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, October 2007.

[17] Z. He, G. G. Yen, and J. Lv, "Evolutionary multiobjective optimization with robustness enhancement," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 494–507, 2020.

[18] W. Daneshyari and G. G. Yen, "Cultural mopso: a cultural framework to adapt parameters of multiobjective particle swarm optimization," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1325–1332, Hong Kong, China, June 2008.

[19] G. Ochoa, "Setting the mutation rate: scope and limitations of the 1/l heuristic," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, GECCO'02*, pp. 495–502, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, January 2002.

[20] H. Han, W. Lu, and J. Qiao, "An adaptive multiobjective particle swarm optimization based on multiple adaptive methods," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2754–2767, 2017.

[21] Z. Guo, O. K. Ersoy, and X. Yan, "A multi-objective differential evolutionary algorithm with angle-based objective space division and parameter adaption for solving sodium gluconate production process and benchmark problems," *Swarm and Evolutionary Computation*, vol. 55, 2020 https://www.sciencedirect.com/science/article/pii/S2210650219302433, Article ID 100670.

[22] D. Wu and G. G. Wang, "Employing reinforcement learning to enhance particle swarm optimization methods," *Engineering Optimization*, vol. 54, no. 2, pp. 329–348, 2022.

[23] Z. Tan and K. Li, "Differential evolution with mixed mutation strategy based on deep reinforcement learning," *Applied Soft Computing*, vol. 111, 2021 https://www.sciencedirect.com/science/article/pii/S1568494621005998, Article ID 107678.

[24] Y. Liu, H. Lu, S. Cheng, and Y. Shi, "An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning," in *Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 815–822, Wellington, New Zealand, June 2019.

[25] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, "Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients," *Information Sciences*, vol. 177, no. 22, pp. 5033–5049, 2007, https://www.sciencedirect.com/science/article/pii/S0020025507003155.

[26] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evolutionary Computation*, vol. 22, no. 2, pp. 231–264, 2014.

[27] C. K. Chow and S. Y. Yuen, "An evolutionary algorithm that makes decision based on the entire previous search history," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 741–769, 2011.

[28] L. Davis, "Handbook of genetic algorithms," *Artificial Intelligence*, vol. 100, pp. 325–330, 1998.

[29] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[30] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4560–4569, 2013, https://www.sciencedirect.com/science/article/pii/S0096300312010831.

[31] X. Li, K. Mao, F. Lin, and X. Zhang, "Particle swarm optimization with state-based adaptive velocity limit strategy," *Neurocomputing*, vol. 447, pp. 64–79, 2021, https://www.sciencedirect.com/science/article/pii/S092523122100463X.

[32] C. El Hatri and J. Boumhidi, "Q-learning based intelligent multi-objective particle swarm optimization of light control for traffic urban congestion management," in *Proceedings of the 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, pp. 794–799, Tangier, Morocco, October 2016.

[33] L. Chen, P. Cheng, Y. Wang, and W. Ye, "Combining multi-objective evolutionary approach and machine learning to optimize pci configuration in large-scale lte networks," in *Proceedings of the 2022 5th International Conference on Communication Engineering and Technology (ICCET)*, pp. 32–39, Shanghai, China, February 2022.

[34] Q. Liu, H. Qiu, B. Niu, and H. Wang, "General parameter control framework for evolutionary computation," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 11432–11464, 2022.

[35] M. Zuo, D. Gong, Y. Wang, X. Ye, B. Zeng, and F. Meng, "Process knowledge-guided autonomous evolutionary optimization for constrained multiobjective problems," *IEEE Transactions on Evolutionary Computation*, pp. 1–15, 2023.

[36] R. A. Howard, *Dynamic Programming And Markov Processes*, Assistant Professor of Electrical Engineering Massachusetts Institute of Technology, Washington DC, USA, 1960.

[37] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc, New York, NY, USA, 2014https://proceedings.neurips.cc/paper_files/paper/2014/file/375c71349b295fbe2dcdca9206f20a06-Paper.pdf.

[38] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[39] L. Atlas, T. Homma, and R. Marks, "An artificial neural network for spatio-temporal bipolar patterns: application to phoneme classification," in *Neural Information Processing Systems*, 1987.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, , 2014, https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[41] J. J. Hopfield, "Hopfield network," *Scholarpedia*, vol. 2, no. 5, p. 1977, 2007.

[42] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," 2013, https://arxiv.org/abs/1312.5602.

[43] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[44] X. Hou, J. Wang, Z. Fang, Y. Ren, K.-C. Chen, and L. Hanzo, "Edge intelligence for mission-critical 6g services in space-air-ground integrated networks," *IEEE Network*, vol. 36, no. 2, pp. 181–189, 2022.

[45] C. Berner, G. Brockman, B. Chan et al., "Dota 2 with large scale deep reinforcement learning," 2019, http://arxiv.org/abs/1912.06680.

[46] W. Wei, J. Wang, Z. Fang, J. Chen, Y. Ren, and Y. Dong, "3u: joint design of uav-usv-uuv networks for cooperative target hunting," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 4085–4090, 2023.

[47] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[48] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, https://arxiv.org/abs/1509.02971.

[49] X. Cai, Y. Xiao, M. Li, H. Hu, H. Ishibuchi, and X. Li, "A grid-based inverted generational distance for multi/many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 21–34, 2021.

[50] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 1, pp. 825–830, Honolulu, HI, USA, May 2002.

[51] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005*, vol. 3, pp. 280–295, Springer, Guanajuato, Mexico, 2005.

[52] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion," in *Proceedings of the 2006 IEEE international conference on evolutionary computation*, pp. 892–899, IEEE, Vancouver, BC, Canada, July 2006.

[53] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature-PPSN VIII*, X. Yao, E. K. Burke, J. A. Lozano et al., Eds., pp. 832–842, Springer Berlin Heidelberg, Berlin, Germany, 2004.

[54] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[55] J. Blank and K. Deb, "Pymoo: multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.

[56] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[57] B. Zeng, W. Zhang, P. Hu, J. Sun, and D. Gong, "Synergetic renewable generation allocation and 5g base station placement for decarbonizing development of power distribution system: a multi-objective interval evolutionary optimization approach," *Applied Energy*, vol. 351, 2023 https://www.sciencedirect.com/science/article/pii/S0306261923011959, Article ID 121831.

[58] R. Kiani Mavi, S. A. Hosseini Shekarabi, N. Kiani Mavi, S. Arisian, and R. Moghdani, "Multi-objective optimisation of sustainable closed-loop supply chain networks in the tire industry," *Engineering Applications of Artificial Intelligence*, vol. 126, 2023 https://www.sciencedirect.com/science/article/pii/S0952197623013003, Article ID 107116.