Deep Reinforcement Learning Based Adaptive Operator Selection for Evolutionary Multi-Objective Optimization

Ye Tian, Xiaopeng Li, Haiping Ma, Xingyi Zhang, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, and Yaochu Jin, *Fellow, IEEE*

***Abstract*—Evolutionary algorithms (EAs) have become one of the most effective techniques for multi-objective optimization, where a number of variation operators have been developed to handle the problems with various difficulties. While most EAs use a fixed operator all the time, it is a labor-intensive process to determine the best EA for a new problem. Hence, some recent studies have been dedicated to the adaptive selection of the best operators during the search process. To address the exploration versus exploitation dilemma in operator selection, this paper proposes a novel operator selection method based on reinforcement learning. In the proposed method, the decision variables are regarded as states and the candidate operators are regarded as actions. By using deep neural networks to learn a policy that estimates the Q value of each action given a state, the proposed method can determine the best operator for each parent that maximizes its cumulative improvement. An EA is developed based on the proposed method, which is verified to be more effective than the state-of-the-art ones on challenging multi-objective optimization problems.**

***Index Terms*—Multi-objective optimization, evolutionary algorithm, reinforcement learning, operator selection.**

Manuscript received –. This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0100100, in part by the National Natural Science Foundation of China under Grant 61822301, Grant 61876123, Grant 61906001, Grant 62107001, and Grant 62136008, in part by the Collaborative Innovation Program of Universities in Anhui Province under Grant GXXT-2020-013 and Grant GXXT-2020-051, in part by the Anhui Provincial Natural Science Foundation under Grant 2108085QF272, in part by the Key Program of Natural Science Project of Educational Commission of Anhui Province under Grant KJ2020A0036, in part by the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant PolyU11202418 and Grant PolyU11209219, and in part by an Alexander von Humboldt Professorship for Artificial Intelligence funded by the Federal Ministry of Education and Research, Germany. *(Corresponding author: Xingyi Zhang.)*

Y. Tian and H. Ma are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China (email: field910921@gmail.com; hpma@ahu.edu.cn).

X. Li is with the School of Computer Science and Technology, Anhui University, Hefei 230601, China (email: lxp@stu.ahu.edu.cn).

X. Zhang is with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Artificial Intelligence, Anhui University, Hefei 230601, China (email: xyzhanghust@gmail.com).

K. C. Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR (email: kctan@polyu.edu.hk).

Y. Jin is with the Faculty of Technology, Bielefeld University, Bielefeld 33619, Germany (email: yaochu.jin@uni-bielefeld.de).

I. Introduction

In the last three decades, evolutionary computation has been recognized to be effective for solving multi-objective optimization problems (MOPs), which evolves a set of solutions to approximate the Pareto fronts of MOPs in a black-box manner [1]. Since no problem-dependent information (e.g., gradient) is used, multi-objective evolutionary algorithms (MOEAs) iteratively generate new solutions via variation operators and eliminate bad solutions via environmental selection. Such a search paradigm enables MOEAs to solve different types of MOPs, and obtain a set of well-converged and diverse solutions in a single run [2].

The variation operators in evolutionary algorithms define the rules for generating new solutions (i.e., offspring) based on existing ones (i.e., parents), where existing operators exhibit quite different search dynamics as well as performance [3]. For example, the genetic algorithm generates offspring via crossover and mutation operators, where the crossover operator provides good exploration ability [4] and the mutation operator can help solutions escape from local optimums [5]. Differential evolution mutates each solution according to the difference between other solutions, which is good at handling complex variable linkages [6]. Particle swarm optimization updates solutions by learning from local and global best solutions, which provides the population with a fast convergence speed in high-dimensional search spaces [7]. Covariance matrix adaptation evolution strategy samples new solutions by learning a multivariate normal distribution model, showing high effectiveness on many real-world scenarios [8].

According to the no free lunch theorem, there does not exist an operator outperforming any others on all optimization problems [9]. This means that an operator should be carefully selected when solving specific MOPs. An operator can generally be selected from some candidates via empirical comparisons [10]. However, such a trial-and-error process is impractical for solving many real-world problems with computationally expensive objectives [11]. To address this issue, some methods have been suggested to determine the best operator

for a given problem before or during the optimization process. The former methods are known as offline algorithm recommendation, where a model is trained to learn the relations between the features of problems and the performance of multiple operators, and the best operator for a new problem can be determined by feeding the features of the new problem to the model [12], [13]. While the features of a given problem are difficult to be accurately extracted [3], the latter methods aim to adaptively select operators according to their performance at historical generations, which are known as online operator selection or hyperheuristics. These methods suggest various credit assignment strategies to measure the fitness improvement from parent to offspring solutions brought by each operator [14], [15], and suggest various selection strategies to determine the next operator for generating offspring [16], [17].

The superiority of adaptive operator selection is obvious on complex optimization problems [18], [19]. However, it suffers from the dilemma of exploration versus exploitation, where the operators with better historical performance are expected to be given higher priority for generating promising offspring, while the operators with poor historical performance are also expected to be explored to make offspring solutions escape from local optimums [17]. This becomes more serious when solving MOPs, since each solution has multiple objective values and both convergence and diversity should be considered. Therefore, this paper aims to address this dilemma via reinforcement learning, where the contributions of this work contain the following aspects:

1) A reinforcement learning based operator selection method is proposed. By regarding decision variables as states, candidate operators as actions, fitness improvement as the reward, and population evolution as the environment, an agent uses deep neural networks to learn a policy estimating the Q value of each action given a state. The Q value represents the cumulative fitness improvement brought by an operator in the future rather than in the past, hence better offspring solutions are expected to be generated at future generations.
2) An MOEA is developed by embedding the proposed operator selection method in a decomposition based MOEA with dynamical resource allocation. In the proposed MOEA, the agent iteratively updates the deep neural networks to guide the selection of operators. By adopting four different types of variation operators as the candidates, the proposed MOEA shows high versatility on MOPs with multimodal landscapes and complex variable linkages, and obtains better performance than state-of-the-art MOEAs in the experiments.

The rest of this paper is organized as follows. In Section II, existing operator selection methods are reviewed and basic concepts of reinforcement learning are introduced. In Section III, the proposed operator selection method and MOEA are elaborated. In Section IV, the experimental results are presented and analyzed. At last, conclusions are given in Section V.

II. Related Work

A. Multi-objective Optimization

In general, an unconstrained MOP is formulated as

$$\begin{aligned} &\text{Minimize:} \quad \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})) \\ &\text{Subject to:} \quad l_d \le x_d \le u_d \quad d = 1, \ldots, D \end{aligned}, \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_D)$ is a solution containing D decision variables, $\mathbf{f}(\mathbf{x})$ is its objective vector containing M objectives, $l_1, \ldots, l_D$ are the lower bounds of variables, and $u_1, \ldots, u_D$ are the upper bounds of variables. Due to the conflicting nature between the objectives, there does not exist a single solution minimizing all the objectives; instead, an MOP contains a finite or infinite number of Pareto optimal solutions trading off between the conflicting objectives [20]. Thus, the goal of solving an MOP is to find a set of solutions as a representative of all the Pareto optimal solutions, where the closeness between the found solutions and the Pareto front is known as convergence, and the spread and evenness of the found solutions are known as diversity.

Most MOEAs solve MOPs without using any problem-dependent information, which means that they can only select solutions by comparing their objective vectors and generate offspring based on existing solutions. Thus, a variety of variation operators have been suggested to generate offspring with predefined rules and formulas, where the operators of the genetic algorithm (GA) [21], differential evolution (DE) [22], particle swarm optimization (PSO) [23], and estimation of distribution algorithm (EDA) [24] are the most popular ones with MOEAs. It can be observed from many existing studies that, each of these operators holds better performance than the others on different types of MOPs, such as GA on DTLZ problems with multimodal landscapes [25], DE on UF problems with complex variable linkages [26], PSO on LSMOP problems with a large number of variables [7], and EDA on IMF problems with nonlinear variable linkages [27]. In short, no operator has the best performance on all types of MOPs.

In order to obtain good performance on various types of MOPs and solve unknown MOPs without trials, some adaptive operator selection methods have been suggested to achieve both the selection of the best operator and the optimization of an MOP in a single optimization process. In the next subsection, existing adaptive operator selection methods are reviewed.

B. Existing Adaptive Operator Selection Methods

The selection of operators during the optimization process is essentially a multi-armed bandit problem, where the goal is to maximize the total reward of a fixed number of plays on multiple arms, without knowing

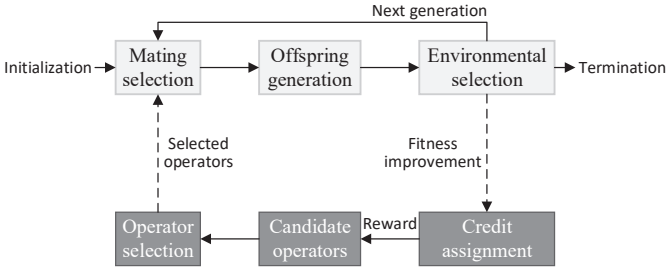


Fig. 1: General procedure of evolutionary algorithms with adaptive operator selection.

the probability distribution of the rewards got from each arm [28]. In terms of operator selection, a credit assignment strategy and an operator selection strategy should be designed as shown in Fig. 1, where the former rewards an operator (i.e., arm) according to the fitness improvement brought by the offspring solutions recently generated by the operator, and the latter decides the next operator be selected according to the rewards of all operators [17].

The idea of operator selection was originally suggested for single-objective optimization [29]. One of the most commonly used strategies of credit assignment is to reward an operator according to the improvement of objective value brought by the offspring solution [18], [30], i.e., $\max\{0, \frac{f(\mathbf{p})-f(\mathbf{o})}{f(\mathbf{p})}\}$, where $f(\mathbf{p})$ and $f(\mathbf{o})$ denotes the objective values of the parent and offspring solutions, respectively. Similarly, some work rewards an operator by comparing the objective values of the offspring solution generated by all operators [19], [31], i.e., $1-\frac{f(\mathbf{o})}{\sum_{\mathbf{o}'} f(\mathbf{o}')}$, where $f(\mathbf{o})$ and $f(\mathbf{o}')$ denotes the objective values of the offspring solution generated by the operator and others, respectively. To track the dynamics of search process, a sliding window saving the objective improvement brought by recently generated offspring solutions is suggested in [14]. The reward of each operator is set to its maximum but not average objective improvement saved in the window, since rare but large improvements are believed to be more valuable than frequent but small improvements [32].

The credit assignment for single-objective optimization is simple since the difference between the objective values of solutions can be directly calculated. By contrast, it becomes more complex to measure the fitness improvement brought by solutions with multiple objectives, where both convergence and diversity should be taken into consideration [33]. Earlier strategies tailored for multi-objective optimization are based on Pareto dominance relation, where an operator is rewarded if the generated offspring solution dominates its parent [34] or dominates many solutions from the previous generation [15]. To explicitly consider both convergence and diversity, a Pareto metric and a density metric are integrated into the reward in [35]. With the assistance of aggregation functions and weigh vectors, the multiple objectives of an MOP can be converted into a single one to facilitate the measurement of convergence and diversity [36]. Hence, some credit assignment strategies are proposed for decomposition based MOEAs, where operators are rewarded according to the improvement of aggregation function values brought by offspring solutions [17], [37]. In another decomposition based MOEA, the reward is defined based on Pareto dominance and crowding status of an offspring solution in its neighborhood [38]. In a recently proposed MOEA for solving large-scale MOPs, the hypervolume improvement of the whole population is regarded as the reward [39].

On the contrary, the selection of operators is more tricky. If the operators are directly selected according to their recent rewards, the operators with higher rewards will be given higher priority than the others, and such a positive feedback will further improve the rewards of these operators. To reduce the greediness as well as the probability of getting trapped into local optimums, some operator selection strategies have been proposed to strike a balance between exploration and exploitation. In [19], [40], the number of times for selecting an operator is determined by the ratio of its reward to the sum of the rewards of all operators. In [41], [42], the operators are selected by using roulette wheel selection according to their rewards. In [17], the upper confidence bound algorithm is employed to select operators, which is very effective for handling the multi-armed bandit problem [28]. In [37], a Beta distribution is established for each operator and updated by Bernoulli Thompson sampling, then the operators are selected according to the rewards sampled from the distributions. In [43], the optimization process is divided into several stages, where all the operators are first selected to obtain their rewards and only the best operator is then selected at each stage. In addition, many other strategies have been adopted to select operators according to their rewards, such as the dynamic multi-armed bandit algorithm [16], fuzzy inference system [44], AdaBoost [45], and adaptive pursuit [39].

The exploration versus exploitation dilemma lies in both the credit assignment and operator selection. Firstly, due to the limited number of trials and the randomness in operators, the operators may be inaccurately rewarded by credit assignment strategies. Secondly, the operator selection strategies select operators according to their historical rewards, but an operator that performs well at previous generations may be ineffective at future generations. To alleviate the dilemma, this work aims to use deep reinforcement learning to assist both the credit assignment and operator selection, where some basic concepts of reinforcement learning are first introduced in the next subsection.

C. Reinforcement Learning

In comparison to evolutionary computation finding the optimal solutions for static problems, reinforcement learning aims to find the optimal solutions for a finite or infinite number of states of dynamic problems,

concerned with how intelligence agents ought to take actions in a dynamic environment [46]. Reinforcement learning has been employed by MOEAs for environmental selection [47], offspring generation [48], and change tracking [49], but has not been applied to the operator selection of MOEAs. In [50], a simple reinforcement learning method is applied to the operator selection for single-objective optimization, while it is totally different from the deep reinforcement learning method used in this work.

The goal of reinforcement learning is to learn a policy π taking an action at each state, where the action maximizes the expected cumulative reward in terms of the current state [51]. At each iteration t, the agent takes an action a_t at the current state s_t according to the policy, then the environment receives the action, produces a reward r_t, and transfers to the next state s_{t+1}. The process repeats until a terminal condition is fulfilled, and the tuple (s_t, a_t, r_t, s_{t+1}) is recorded as a sample to train the agent. Deep reinforcement learning currently includes policy-based methods and value-based methods. The policy-based methods handle continuous action spaces by directly learning a stochastic policy via an independent function approximator [52]. In this case, a policy is a conditional probability distribution $\pi = p(a_t|s_t;\theta)$ indicating the probability of taking action a_t at state s_t with policy parameters θ. Using policy gradient to learn the optimal policy parameters, the agent can take an action at a given state by sampling from π [53]. The value-based methods handle discrete action spaces by approximating the action-value function, which calculates the expected cumulative reward (i.e., Q value) of taking action a_t at state s_t [54]. More specifically, the action-value function is defined as $Q(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t]$, where $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$ is the discount sum of future rewards and $\gamma \in [0, 1]$ is the discount factor. Once the action-value function is approximated, the agent can take the action with the largest Q value at a given state s, i.e., $\pi(s) = \text{argmax}_{a\in\mathcal{A}} Q(s, a)$.

The classical Q-learning approximates the action-value function via a Q-table [55], where each grid stores the Q value of an action given a state. While the Q-table can only handle discrete state spaces, the deep Q-network (DQN) employs a deep neural network to approximate the action-value function with continuous state spaces [56], where the input and output of the neural network are a state and the Q values of all actions, respectively. The DQN stores the obtained tuples in an experience replay pool to form a training set, and interleaves taking actions with neural network training. The neural network is trained using gradient descent on the loss

$$\mathcal{L} = \frac{1}{|\mathcal{T}|} \sum_{t\in\mathcal{T}} \left(Q\left(s_t, a_t\right) - q_t\right)^2, \tag{2}$$

where $\mathcal{T}$ is the training set, $Q(s_t, a_t)$ is the a_t-th output neuron of the neural network with input vector s_t, and q_t is the Q value of action a_t at state s_t:

$$q_t = r_t + \gamma \max_{a'\in\mathcal{A}} Q(s_{t+1}, a'). \tag{3}$$

It is worth to note that the expected output q_t contains not only the current reward r_t but also the maximum reward $\max_{a'\in\mathcal{A}} Q(s_{t+1}, a')$ of taking the next action. Due to the recursive nature of the formula, q_t can represent the maximum cumulative reward in the future.

In the proposed operator selection method, the decision variables of solutions are regarded as states and the indexes of operators are regarded as actions. Since the states are continuous and the actions are discrete, it is desirable to use DQN in the proposed method. In the next section, the proposed method as well as MOEA is elaborated.

III. The Proposed Method

A. The Proposed Reinforcement Learning Based Operator Selection Method

The core issue in using reinforcement learning is to determine the rewards and states for a given task, so that the agent can use a deep neural network to learn the relations between states and rewards. To consider both convergence and diversity in the fitness improvement brought by offspring solutions, the proposed method employs the framework of decomposition based MOEAs [36] and consequently calculates the improvement of aggregation function values. Here the Tchebycheff approach is adopted as the aggregation function:

$$\text{Minimize } g^{tch}(\mathbf{x}, \mathbf{w}, \mathbf{z}^*) = \max_{1\le i\le M} w_i(f_i(\mathbf{x}) - z_i^*), \tag{4}$$

where $\mathbf{w} = (w_1, \ldots, w_M)$ is the weight vector corresponding to solution $\mathbf{x}$ and $\mathbf{z}^* = (z_1^*, \ldots, z_M^*)$ is the ideal point consisting of the minimum objective values in the population. An offspring solution $\mathbf{x}$ is compared to all the solutions in its parent's neighborhood once it is generated, and its fitness improvement on a neighbor $\mathbf{y}$ is calculated by

$$FI(\mathbf{x}, \mathbf{y}) = \max\left\{1 - \frac{g^{tch}(\mathbf{x}, \mathbf{w}, \mathbf{z}^*)}{g^{tch}(\mathbf{y}, \mathbf{w}, \mathbf{z}^*)}, 0\right\}, \tag{5}$$

where $\mathbf{w}$ denotes the weight vector of $\mathbf{y}$. Note that an offspring solution can replace at most n_r solutions in the neighborhood, hence its neighborhood fitness improvement $NFI_{\mathbf{x}}$ is further calculated as the sum of fitness improvement on at most n_r solutions replaced by the offspring solution. Moreover, the proposed method considers the NFI in a historical period by saving the recent tuple $(op, NFI_{\mathbf{x}})$ in a first-in first-out queue $\mathcal{R}$, where $NFI_{\mathbf{x}}$ is the neighborhood fitness improvement of offspring solution $\mathbf{x}$ generated by operator op. Thus, the reward of operator op can be calculated based on its tuples saved in $\mathcal{R}$:

$$reward_{op} = \max_{(op, NFI_{\mathbf{x}})\in\mathcal{R}} NFI_{\mathbf{x}}. \tag{6}$$

Algorithm 1: $CreditAssignment(\mathbf{x}, op, Y, \mathcal{R})$

Input: $\mathbf{x}$ (offspring), op (the operator generating $\mathbf{x}$), Y (solutions replaced by $\mathbf{x}$), $\mathcal{R}$ (queue of rewards)
Output: $\mathcal{R}$ (updated queue), $reward_{op}$ (reward of the operator)

$NFI_{\mathbf{x}} \leftarrow 0$;
foreach $\mathbf{y} \in Y$ **do**
  $FI(\mathbf{x}, \mathbf{y}) \leftarrow$ Calculate the fitness improvement by (5);
  $NFI_{\mathbf{x}} \leftarrow NFI_{\mathbf{x}} + FI(\mathbf{x}, \mathbf{y})$;
Add $(op, NFI_{\mathbf{x}})$ to $\mathcal{R}$;
if $\mathcal{R}$ *exceeds its maximum size* **then**
  Delete the oldest tuple from $\mathcal{R}$;
$reward_{op} \leftarrow$ Calculate the reward of op by (6);
return $\mathcal{R}, reward_{op}$;

Algorithm 2: $OperatorSelection(\mathbf{p}, \mathbf{w}, OP, \mathcal{T}, Q)$

Input: $\mathbf{p}$ (solution), $\mathbf{w}$ (corresponding weight vector), OP (candidate operator set), $\mathcal{T}$ (experience replay pool), Q (trained neural network)
Output: op (selected operator)

$state \leftarrow \{\mathbf{p}, \mathbf{w}\}$;
$Qvalue \leftarrow$ A $1 \times |OP|$ vector of zeros;
foreach $op \in OP$ **do**
  if $\nexists(\cdot, op, \cdot, \cdot) \in \mathcal{T}$ **then**
    return op;
  else
    $Qvalue_{op} \leftarrow Q(state, op)$;
$op \leftarrow$ Select an operator according to $Qvalue$ via roulette-wheel selection;
return op;

That is, the reward of an operator is set to its largest NFI value in the queue. The underlying idea is that rare, but large improvements are believed to be more important than frequent and small improvements [57]. Hence, the largest rather than average NFI value is considered as the reward, and the reward of each operator can be stable in a period. The pseudocode of the proposed credit assignment strategy is summarized in Algorithm 1.

After calculating the reward once an offspring solution is generated, the tuple $(\{\mathbf{p}, \mathbf{w}\}, op, reward_{op}, \{\mathbf{x}, \mathbf{w}\})$ is saved in another queue $\mathcal{T}$ for experience replay, where $\mathbf{p}$ is the main parent generating $\mathbf{x}$ and $\mathbf{w}$ is the corresponding weight vector. That is, the state consists of the decision variables of a solution and its weight vector (i.e., $s = (p_1, \ldots, p_D, w_1, \ldots, w_M)$) and the action is the operator (i.e., $a = op$). Then, $\mathcal{T}$ is used to train a deep neural network $Q(s, a)$ by (2)–(3), and the neural network can be used for selecting operators in the future. Before generating an offspring solution based on a parent $\mathbf{p}$, the Q value of each operator is calculated by the neural network with input $\{\mathbf{p}, \mathbf{w}\}$, and an operator is selected according to their Q values via roulette-wheel selection. It is worth to note that an operator can no longer be selected once it does not exist in the experience replay pool $\mathcal{T}$; in this case, the operator will be forcibly selected instead of roulette-wheel selection. The pseudocode of the proposed operator selection strategy is presented in Algorithm 2.

According to the description of the proposed method, its credit assignment strategy and operator selection strategy have the following advantages in comparison to the strategies in existing methods:

1) Roughly, existing credit assignment strategies set the reward q_t to $r_t + \gamma q_{t-1}$ that considers the historical rewards, whereas the proposed strategy sets q_t to $r_t + \gamma q_{t+1}$ that considers the future rewards. Since the selected operators are expected to be effective in the future rather than in the past, the proposed strategy is more promising. Of course, the future rewards are unknown during the optimization process, hence they are deduced by training a deep neural network $Q(s, a)$.
2) Existing operator selection strategies reward operators on all the solutions equally, while the proposed strategy calculates the Q values of operators by feeding a solution and its weight vector to the neural network. That is, the proposed strategy considers the characteristic of each solution and can suggest different operators for different parents, which provides a more delicate strategy for operator selection.

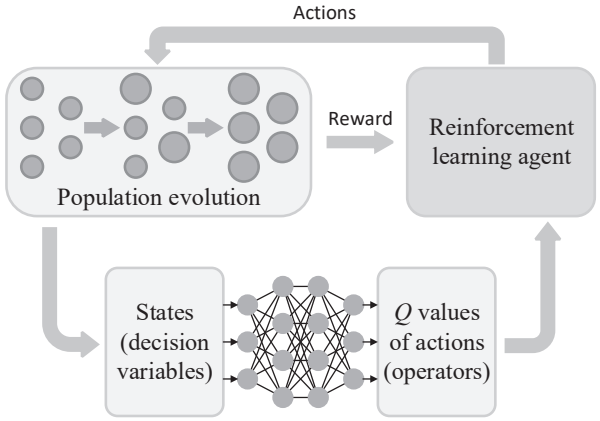


Fig. 2: Illustration of the proposed MOEA with reinforcement learning based adaptive operator selection.

B. Procedure of the Proposed MOEA/D-DQN

An MOEA is then established based on the proposed operator selection method. The proposed MOEA, termed MOEA/D-DQN, adopts the framework of the decomposition based MOEA with dynamical resource allocation, which has been verified high performance on challenging MOPs [26]. As illustrated in Fig. 2, the MOEA interacts with the reinforcement learning agent twice a

generation: Before an offspring solution is generated, the agent selects an operator (i.e., action) for the MOEA according to the neural network; after an offspring solution is generated, the MOEA sends the solutions (i.e., state) and fitness improvement (i.e., reward) to the agent for training the neural network.

Algorithm 3 details the procedure of the proposed MOEA. Following the general procedure of decomposition based MOEAs, a population is first randomly initialized (Line 1) and the same number of uniformly distributed weight vectors are generated (Line 2). The weight vectors are generated by the Das and Dennis's method [58], and the mixture uniform design [58] can also be adopted if the population size is set to an arbitrary number. Then, the neighbors of each weight vector are determined (Line 3) and their utilities are set to 1 (Line 4), which is followed by the initialization of the variables $\mathbf{z}^*, Q, \mathcal{R}, \mathcal{T}$ (Lines 5–8). At each generation, the mating pool is first constituted by the M extreme solutions (i.e., whose weight vectors are $(1,0,\ldots,0),(0,1,\ldots,0),\ldots,(0,0,\ldots,1)$) and $\frac{|P|}{5}-M$ solutions selected by 10-tournament selection according to their utilities (Lines 10-11). Then, each solution $\mathbf{p}$ in the mating pool is regarded as a main parent and solutions randomly selected from the neighbors of $\mathbf{p}$ or the whole population are regarded as other parents (Lines 12–16). Afterwards, the parents are used to generate an offspring solution by an operator selected by the neural network Q (Lines 17–18). The offspring solution is used to update the ideal point (Line 19) and at most n_r solutions in the neighbors of $\mathbf{p}$ (Lines 20–29). According to the fitness improvement brought by the offspring solution, the reward is calculated (Line 30) and the neural network is trained (Lines 31–34). After every 50 generations, the utility $\pi_{\mathbf{p}}$ of each solution $\mathbf{p}$ is updated by

$$\pi_{\mathbf{p}} = \begin{cases} 1, & \text{if } \Delta_{\mathbf{p}} > 0.001 \\ \left(0.95 + 0.05 \times \frac{\Delta_{\mathbf{p}}}{0.001}\right) \times \pi_{\mathbf{p}}, & \text{otherwise} \end{cases}, \quad (7)$$

where $\Delta_{\mathbf{p}}$ denotes the total improvement of the aggregation function values in the last 50 generations:

$$\Delta_{\mathbf{p}} = 1 - \frac{g^{tch}(\mathbf{p}, \mathbf{w}, \mathbf{z}^*)}{g^{tch}(\mathbf{p}', \mathbf{w}, \mathbf{z}^*)}, \quad (8)$$

where $\mathbf{w}$ is the weight vector of $\mathbf{p}$ and $\mathbf{p}'$ is the ancestor of $\mathbf{p}$ fifty generations ago.

In the proposed MOEA/D-DQN, four effective variation operators are adopted as candidate operators, including simulated binary crossover [59], the crossover operator in [60], and two differential evolution operators [22]. To be specific, the simulated binary crossover is the most popular crossover operator in genetic algorithm for continuous optimization, which is good at handling multimodal landscapes:

$$x_d = 0.5\left[(1+\beta)y_d^1 + (1-\beta)y_d^2\right], \quad (9)$$

where $d = 1,\ldots,D$, $\mathbf{x} = (x_1,\ldots,x_d,\ldots,x_D)$ is the offspring solution, $\mathbf{y}^1, \mathbf{y}^2$ are two parents, and

$$\beta = \begin{cases} (2\mu)^{\frac{1}{\eta+1}}, & \mu \le 0.5 \\ [2(1-\mu)]^{-\frac{1}{\eta+1}}, & \mu > 0.5 \end{cases}, \quad (10)$$

Algorithm 3: Procedure of MOEA/D-DQN

Input: OP (candidate operator set)
Output: P (final population)

1 $P \leftarrow$ Randomly initialize a population;
2 $W \leftarrow$ Generate the uniformly distributed weight vectors;
3 $B \leftarrow$ Determine the indexes of neighbors of each weight vector;
4 $\pi \leftarrow$ A $1 \times |P|$ vector of ones; `//utilities of solutions`
5 $\mathbf{z}^* \leftarrow$ Minimum objective values in P;
6 $Q \leftarrow$ Randomly initialize a deep neural network;
7 $\mathcal{R} \leftarrow \emptyset$; `//Queue of rewards`
8 $\mathcal{T} \leftarrow \emptyset$; `//Experience replay pool`
9 **while** *termination criterion is not met* **do**
10 $\quad MatingPool1 \leftarrow M$ solutions with extreme weight vectors;
11 $\quad MatingPool2 \leftarrow$ Select $\frac{|P|}{5} - M$ solutions according to π via 10-tournament selection;
12 $\quad$ **foreach** $\mathbf{p} \in MatingPool1 \cup MatingPool2$ **do**
13 $\quad\quad$ **if** $rand < 0.9$ **then**
14 $\quad\quad\quad Parent \leftarrow$ Randomly select solutions from $B_{\mathbf{p}}$;
15 $\quad\quad$ **else**
16 $\quad\quad\quad Parent \leftarrow$ Randomly select solutions from P;
17 $\quad\quad op \leftarrow OperatorSelection(\mathbf{p}, W_{\mathbf{p}}, OP, \mathcal{T}, Q)$;
18 $\quad\quad \mathbf{x} \leftarrow$ Generate an offspring solution based on $\{\mathbf{p}\} \cup Parent$ by op;
19 $\quad\quad \mathbf{z}^* \leftarrow$ Update the ideal point by $\mathbf{x}$;
20 $\quad\quad count \leftarrow n_r$; `//Maximum number of replaced solutions`
21 $\quad\quad Neighbor \leftarrow B_{\mathbf{p}}$;
22 $\quad\quad Y \leftarrow \emptyset$;
23 $\quad\quad$ **while** $count > 0$ && $Neighbor \neq \emptyset$ **do**
24 $\quad\quad\quad \mathbf{y} \leftarrow$ Randomly select a solution from $Neighbor$;
25 $\quad\quad\quad Neighbor \leftarrow Neighbor \setminus \{\mathbf{y}\}$;
26 $\quad\quad\quad$ **if** $g^{tch}(\mathbf{x}, W_{\mathbf{y}}, \mathbf{z}^*) < g^{tch}(\mathbf{y}, W_{\mathbf{y}}, \mathbf{z}^*)$ **then**
27 $\quad\quad\quad\quad$ Replace $\mathbf{y}$ in P with $\mathbf{x}$;
28 $\quad\quad\quad\quad count \leftarrow count - 1$;
29 $\quad\quad\quad\quad Y \leftarrow Y \cup \{\mathbf{y}\}$;
30 $\quad\quad [\mathcal{R}, reward_{op}] \leftarrow CreditAssignment(\mathbf{x}, op, Y, \mathcal{R})$;
31 $\quad\quad$ Add $(\{\mathbf{p}, W_{\mathbf{p}}\}, op, reward_{op}, \{\mathbf{x}, W_{\mathbf{p}}\})$ to $\mathcal{T}$;
32 $\quad\quad$ **if** $\mathcal{T}$ *exceeds its maximum size* **then**
33 $\quad\quad\quad$ Delete the oldest tuple from $\mathcal{T}$;
34 $\quad\quad$ Randomly select a batch of tuples from $\mathcal{T}$ and train Q by (2)–(3);
35 $\quad$ **if** *the generation no. is a multiplication of 50* **then**
36 $\quad\quad \pi \leftarrow$ Update the utilities by (7);
37 **return** P;

where μ is a uniformly distributed random value sampled in $[0,1]$ and η is a predefined parameter. The crossover operator in [60] is good at handling nonlinear variable linkages:

$$x_d = y_d^1 + r_1 \times \left[1 - r_2^{-(1-\frac{gen}{maxgen})^{0.7}}\right] \times (y_d^1 - y_d^2), \quad (11)$$

where r_1, r_2 are uniformly distributed random values sampled in $[0,1]$, gen is the current generation number, and $maxgen$ is the maximum number of generations. Besides, the operators of differential evolution are also effective for handling complex variable linkages, where DE/rand/1 is defined as

$$x_d = \begin{cases} y_d^1 + F \times (y_d^2 - y_d^3), & r \leq CR \\ y_d^1, & \text{otherwise} \end{cases} \quad (12)$$

and DE/rand/2 is defined as

$$x_d = \begin{cases} y_d^1 + F \times (y_d^2 - y_d^3 + y_d^4 - y_d^5), & r \leq CR \\ y_d^1, & \text{otherwise} \end{cases}, \quad (13)$$

where F, CR are predefined parameters and r is a uniformly distributed random values sampled in $[0,1]$. Note that two parents should be selected for the two crossover operators, while DE/rand/1 and DE/rand/2 require three and five parents for generating an offspring solution, respectively. In addition, the polynomial mutation is performed once an offspring solution is generated [61], which can further enhance the performance in multi-objective optimization [6], [7]:

$$x_d = x_d + (u_d - l_d)\delta, \quad (14)$$

where

$$\delta = \begin{cases} \left[2\mu + (1-2\mu)\left(1 - \frac{x_d - l_d}{u_d - l_d}\right)^{\eta+1}\right]^{\frac{1}{\eta+1}} - 1, & \mu \leq 0.5 \\ 1 - \left[2(1-\mu) + 2(\mu - 0.5)\left(1 - \frac{u_d - x_d}{u_d - l_d}\right)^{\eta+1}\right]^{\frac{1}{\eta+1}}, & \mu > 0.5 \end{cases}, \quad (15)$$

where l_d is the lower bound of the d-th decision variable, u_d is the upper bound of the d-th decision variable, μ is a uniformly distributed random value sampled in $[0,1]$, and η is a predefined parameter.

Lastly, since the neural network Q is trained after the generation of each offspring solution, the strong correlation between the loss in (2) and the estimated output in (3) makes it difficult to converge [56]. Hence, two neural networks Q and $\hat{Q}$ rather than a single one are in fact used in the proposed method. The primary network Q is trained as usual, while the target network $\hat{Q}$ is used to estimate the output in (3). The target network is not trained, but directly copies the weights from the primary network after every ten training steps. The use of two neural networks has been verified to be effective in reinforcement learning [56], [62].

C. Computational Complexity of MOEA/D-DQN

According to Algorithm 3, the time complexity of the proposed MOEA/D-DQN is mainly determined by five steps in generating each offspring solution, i.e., operator selection, offspring generation, population update, credit assignment, and neural network training. The time complexity of operator selection is $O(D^2)$ for a deduction procedure of the neural network, where D is the number of decision variables indicating the size of layers. The time complexity of offspring generation is $O(D)$ for all the variation operators. The time complexity of population update is $O(M)$ for calculating the aggregation function. The time complexity of credit assignment is $O(N)$ for calculating the neighborhood fitness improvement. The time complexity of neural network training is $O(D^2)$ for a backpropagation procedure. As a consequence, the total time complexity of MOEA/D-DQN in one generation is $O(ND^2)$.

IV. Experimental Studies

In order to verify the performance of the proposed MOEA/D-DQN, it is compared to several classical or state-of-the-art MOEAs in the experiments, including MOEA/D [36], MOEA/D-DRA [26], IM-MOEA [63], SMEA [64], MOEA/D-FRRMAB [17], and MOEA/D-DYTS [37]. MOEA/D is a classical decomposition based MOEA, while MOEA/D-DRA is a variant of MOEA/D based on dynamical resource allocation. IM-MOEA generates offspring by Gaussian process based inverse models, and SMEA generates offspring by differential evolution with self-organizing map based mating selection. MOEA/D-FRRMAB and MOEA/D-DYTS are two MOEAs with adaptive operator selection, which select operators via a fitness-rate-rank-based multiarmed bandit method and dynamic Thompson sampling, respectively. In short, the framework of the proposed method is the same as the compared MOEAs, while the main contribution of MOEA/D-DQN lies in the reinforcement learning based operator selection method.

A. Experimental Settings

All the parameters of the compared MOEAs are set as suggested in their original papers, and the experimental results are collected from PlatEMO [65]. The detailed parameter settings of the compared MOEAs are listed in Table I. For the neural network in MOEA/D-DQN, a fully connected neural network with seven layers is adopted, where the input layer size is $D + M$ (i.e., sum of the number of decision variables and the number of objectives), the output layer size is the same as the number of candidate operators, and the sizes of the five hidden layers are 128, 256, 128, 64, and 32. The neural network is trained by Adam [66] with a learning rate of 0.01 for one epoch each time, where the size of $\mathcal{R}$ is the same as the population size and the size of $\mathcal{T}$ is 512. The batch size is set to 16, which means that each time 16 tuples $(\{\mathbf{p}, \mathbf{w}\}, op, reward_{op}, \{\mathbf{x}, \mathbf{w}\})$ are randomly selected from $\mathcal{T}$ and used as the training samples, where $\mathbf{p}$ is the main parent generating offspring

TABLE I: Detailed Settings of the Compared MOEAs.

General Settings	
Population size	100 for ZDT problems, 105 for DTLZ and WFG problems, 600 for UF1–7, 1000 for UF8–10, 50 for neural network training.
Maximum number of function evaluations	10000 for ZDT problems, 30000 for DTLZ and WFG problems, 300000 for UF problems, 10000 for neural network training.
Variation Operators	
Simulated binary crossover	Used in MOEA/D and MOEA/D-DQN, crossover probability: 1, distribution index: 20.
Differential evolution	Used in MOEA/D-DRA, SMEA, MOEA/D-FRRMAB, MOEA/D-DYTS, and MOEA/D-DQN, $F = 0.5$, $CR = 1$.
Gaussian process based inverse models	Used in IM-MOEA.
Polynomial mutation	Used in all MOEAs, mutation probability: $\frac{1}{\#variables}$, distribution index: 20.
Aggregation Function	
Tchebycheff approach	Used in MOEA/D, MOEA/D-DRA, MOEA/D-FRRMAB, MOEA/D-DYTS, and MOEA/D-DQN, neighborhood size: 20, maximum number of replaced parents: 2.
Other Settings	
IM-MOEA	Number of divisions: 10.
SMEA	Initial learning rate: 0.7, size of neighborhood mating pools: 5.
MOEA/D-FRRMAB	Scaling factor: 5, Sliding window size: $\frac{1}{2} \times population\_size$, Decaying factor: 1.
MOEA/D-DYTS	Update threshold of sampling: 100.

$\mathbf{x}$, $\mathbf{w}$ is the corresponding weight vector, op is the selected operator, and $reward_{op}$ is the reward calculated by (6).

Four test suites are involved in the experiments, including ZDT [67], DTLZ [68], WFG [69], and UF [70] having 31 benchmark MOPs in total. For the ZDT and UF problems, the number of decision variables is set to 30 and the number of objectives is set to 2, except for ZDT4 and ZDT6 having 10 decision variables and UF8–UF10 having 3 objectives. For the DTLZ and WFG problems, the number of objectives is set to 3, the number of decision variables is set to 12 for DTLZ2–DTLZ6 and WFG1–WFG9, 7 for DTLZ1, and 22 for DTLZ7. Besides, the neural network training problem is also involved in the experiments, whose definition can be found in [7]. Three datasets taken from the UCI machine learning repository [71] are adopted for training, resulting in three MOPs (denoted as NN1–NN3) with 321, 401, and 1241 decision variables.

The inverted generational distance (IGD) [72] is adopted to evaluate the obtained solutions for benchmark MOPs, which is calculated according to approximately 10000 reference points sampled by the methods suggested in [58]. The hypervolume (HV) [73] is adopted to evaluate the obtained solutions for neural network training, where the reference point is set to $(1, 1)$. The mean and standard deviation of the indicator values obtained by each MOEA on each MOP for 30 independent runs are recorded. Furthermore, the Wilcoxon rank sum test with a significance level of 0.05 is used to perform statistical analysis [74].

B. Comparison Between MOEA/D-DQN and Other MOEAs

The statistical results of the seven compared MOEAs on four test suites are listed in Table II. In general, the proposed MOEA/D-DQN shows the best performance on 18 out of 34 MOPs, which is followed by MOEA/D, SMEA, IM-MOEA, MOEA/D-DRA, MOEA/D-FRRMAB, and MOEA/D-DYTS gaining 6, 4, 3, 1, 1, and 1 best result, respectively. For MOEA/D, MOEA/D-DRA, IM-MOEA, and SMEA using a single variation operator, it can be found that they exhibit quite different performance on different test suites; for example, the simulated binary crossover based MOEA/D performs well on WFG problems while the Gaussian process based IM-MOEA performs well on UF problems. By contrast, MOEA/D-DQN strikes a balance between the performance on different test suites, which is significantly better than the four MOEAs on most MOPs. As for MOEA/D-FRRMAB and MOEA/D-DYTS based on adaptive operator selection, they are also underperformed by the proposed MOEA/D-DQN, which verifies the superiority of the proposed operator selection method.

For further analysis, Figs. 3 and 4 plot the populations with the median IGD values obtained by the seven compared MOEAs on ZDT1 and UF1. It can be found that MOEA/D and IM-MOEA have better convergence performance on ZDT1 than on UF1, which means that their variation operators are good at handling the unimodal landscapes of ZDT1 but less effective for handling the complex variable linkages of UF1. On the contrary, MOEA/D-DRA, MOEA/D-FRRMAB, and MOEA/D-DYTS have better convergence performance on UF1 than on ZDT1, since the differential evolution operators are good at handling complex variable linkages but converge slowly on simple landscapes. In contrast to the above MOEAs, the proposed MOEA/D-DQN has good convergence performance on both ZDT1 and UF1, due to its effectiveness in selecting suitable operators to handle different MOPs. Although MOEA/D-DQN is as good as MOEA/D on ZDT1 and as good as MOEA/D-DRA and MOEA/D-FRRMAB on UF1, it can be observed from Fig. 5 that MOEA/D-DQN holds a faster convergence speed than all the other MOEAs, which implies that the ensemble of multiple operators and the reinforcement learning based operator selection can improve the performance of MOEAs. In addition, Fig. 6 shows the mean IGD values obtained by the compared MOEAs on

TABLE II: IGD/HV Values Obtained by Seven MOEAs on Benchmark Suites/Neural Network Training, Where the Best Result in Each Row is Highlighted and "+", "−", and "=" Indicate that the Result is Significantly Better, Significantly Worse, and Statistically Similar to the Proposed MOEA/D-DQN.

Problem	MOEA/D	MOEA/D-DRA	IM-MOEA	SMEA	MOEA/D-FRRMAB	MOEA/D-DYTS	MOEA/D-DQN
ZDT1	2.6800e-2 (1.58e-2) -	4.6123e-2 (1.82e-2) -	6.2559e-2 (7.72e-3) -	7.2358e-1 (1.01e-1) -	3.8123e-2 (2.14e-2) -	3.8144e-2 (2.08e-2) -	6.9980e-3 (1.76e-3)
ZDT2	5.9006e-2 (7.55e-2) -	1.1258e-1 (4.76e-2) -	6.2866e-2 (1.08e-2) -	1.4689e+0 (1.80e-1) -	1.2262e-1 (5.89e-2) -	1.1662e-1 (4.88e-2) -	2.4188e-2 (2.15e-2)
ZDT3	3.1487e-2 (2.29e-2) -	3.1938e-1 (5.43e-2) -	7.6586e-2 (1.45e-2) -	7.8261e-1 (9.28e-2) -	1.9540e-1 (4.85e-2) -	2.0511e-1 (4.72e-2) -	1.7122e-2 (5.58e-3)
ZDT4	1.7288e-1 (1.12e-1) +	1.1806e+1 (6.89e+0) -	1.0835e-2 (1.47e-3) +	2.8495e+1 (4.86e+0) -	7.8472e+0 (4.44e+0) -	3.9875e+0 (3.34e+0) -	2.5453e-1 (9.70e-2)
ZDT6	1.1214e-2 (2.47e-3) -	5.3427e-2 (1.06e-1) -	1.5028e+0 (9.01e-2) -	9.8493e-2 (1.98e-1) -	4.2009e-3 (3.96e-3) =	3.5353e-3 (9.44e-4) =	3.8825e-3 (2.74e-3)
DTLZ1	2.0982e-2 (3.27e-4) -	1.2916e+0 (1.53e+0) -	3.4132e+0 (7.67e-1) -	2.0141e+0 (1.26e+0) -	3.4192e-1 (6.12e-1) -	6.3591e-1 (9.08e-1) -	1.8507e-2 (8.58e-4)
DTLZ2	5.4467e-2 (1.28e-6) -	7.5716e-2 (6.94e-4) -	9.4179e-2 (5.15e-3) -	8.0869e-2 (3.18e-3) -	7.5370e-2 (6.65e-4) -	7.5319e-2 (5.90e-4) -	4.7007e-2 (1.53e-4)
DTLZ3	9.7856e-1 (1.26e+0) =	2.5743e+1 (2.40e+1) -	6.0011e+1 (1.01e+1) -	1.4375e+1 (1.81e+1) -	1.8153e+1 (1.89e+1) -	1.7439e+1 (2.28e+1) -	1.2903e+0 (3.04e+0)
DTLZ4	3.8444e-1 (3.21e-1) -	2.2549e-1 (1.76e-1) -	7.9127e-2 (3.12e-3) -	1.1164e-1 (6.26e-3) -	1.2233e-1 (7.09e-2) -	1.6551e-1 (9.04e-2) -	4.7086e-2 (2.36e-4)
DTLZ5	3.3764e-2 (6.54e-5) -	1.4166e-2 (1.45e-4) +	2.4473e-2 (4.64e-3) +	2.2885e-2 (2.25e-3) +	1.4252e-2 (1.58e-4) +	1.4401e-2 (1.19e-4) +	2.6538e-2 (5.78e-4)
DTLZ6	3.3881e-2 (2.56e-5) -	1.4338e-2 (6.68e-5) +	4.7806e+0 (1.13e-1) -	8.4479e-3 (1.45e-3) +	1.4528e-2 (5.14e-5) +	1.4537e-2 (3.79e-5) +	2.9008e-2 (4.72e-5)
DTLZ7	1.9743e-1 (1.64e-1) -	2.2761e-1 (6.01e-2) -	3.4424e-1 (4.58e-2) -	4.7891e-1 (1.78e-1) -	1.9837e-1 (3.99e-2) -	2.1575e-1 (7.33e-2) -	1.1006e-1 (3.56e-4)
WFG1	3.4553e-1 (3.64e-2) +	1.3536e+0 (8.07e-2) -	1.3058e+0 (6.48e-2) -	1.6100e+0 (5.58e-2) -	1.5443e+0 (7.38e-2) -	1.4408e+0 (1.09e-1) -	7.1675e-1 (8.33e-2)
WFG2	2.6195e-1 (1.71e-2) -	3.4125e-1 (2.29e-2) -	2.6466e-1 (1.52e-2) -	2.7646e-1 (1.44e-2) -	3.6394e-1 (3.00e-2) -	3.5163e-1 (2.96e-2) -	2.0742e-1 (1.96e-2)
WFG3	2.0213e-1 (5.61e-2) -	1.8917e-1 (3.15e-2) -	2.1991e-1 (1.76e-2) -	2.6856e-1 (3.86e-2) -	2.1493e-1 (3.59e-2) -	1.9109e-1 (4.03e-2) -	1.2825e-1 (4.77e-3)
WFG4	2.6556e-1 (7.53e-3) -	3.8005e-1 (1.12e-2) -	3.3302e-1 (7.61e-3) -	3.7257e-1 (2.00e-2) -	3.9968e-1 (1.25e-2) -	3.9868e-1 (1.57e-2) -	2.3703e-1 (5.38e-3)
WFG5	2.5422e-1 (3.39e-3) -	3.3639e-1 (4.02e-3) -	3.4015e-1 (1.63e-2) -	2.7850e-1 (8.20e-3) -	3.3880e-1 (4.34e-3) -	3.3789e-1 (5.03e-3) -	2.2353e-1 (3.01e-3)
WFG6	2.9751e-1 (1.92e-2) +	4.3393e-1 (2.83e-2) =	3.5758e-1 (1.53e-2) +	3.5166e-1 (3.43e-2) +	4.3816e-1 (3.03e-2) -	4.3740e-1 (4.00e-2) =	4.2542e-1 (4.02e-2)
WFG7	3.5619e-1 (5.14e-2) +	3.6897e-1 (1.03e-2) =	3.5277e-1 (1.18e-2) +	3.2754e-1 (1.56e-2) +	3.6693e-1 (9.62e-3) =	3.6391e-1 (7.96e-3) =	3.6882e-1 (1.13e-2)
WFG8	3.3073e-1 (1.16e-2) +	4.7517e-1 (4.77e-2) =	4.2368e-1 (1.53e-2) +	4.0988e-1 (1.51e-2) +	4.4389e-1 (2.41e-2) +	4.5188e-1 (4.59e-2) +	4.8352e-1 (5.64e-2)
WFG9	3.2760e-1 (6.22e-2) +	3.4008e-1 (1.98e-2) =	3.3717e-1 (1.39e-2) =	3.2822e-1 (5.93e-2) =	3.8344e-1 (4.80e-2) =	3.4880e-1 (3.77e-2) +	3.5458e-1 (3.56e-2)
UF1	1.4922e-1 (5.41e-2) -	1.9429e-3 (7.06e-4) -	5.4924e-2 (1.06e-2) -	3.1349e-2 (3.10e-3) -	2.3837e-3 (1.94e-3) =	2.7501e-2 (7.77e-2) =	1.3666e-3 (1.05e-4)
UF2	4.5377e-2 (3.63e-2) -	3.8904e-3 (1.98e-3) -	2.0652e-2 (4.63e-3) -	3.0338e-2 (2.61e-3) -	3.0706e-3 (9.33e-4) -	2.0981e-3 (7.41e-4) =	2.0228e-3 (4.02e-4)
UF3	3.0328e-1 (3.32e-2) -	2.7810e-2 (4.57e-2) =	7.0425e-2 (7.56e-3) -	1.2291e-1 (3.10e-2) -	1.8076e-2 (1.59e-2) =	2.5324e-2 (2.19e-2) =	2.1607e-2 (1.97e-2)
UF4	5.6160e-2 (5.72e-3) -	6.1483e-2 (4.47e-3) -	5.9624e-2 (2.70e-3) -	5.1042e-2 (2.55e-3) -	5.1265e-2 (2.86e-3) -	4.9417e-2 (2.48e-3) -	3.3923e-2 (1.36e-3)
UF5	4.8038e-1 (1.27e-1) =	5.3079e-1 (1.07e-1) =	5.7316e-1 (1.02e-1) =	1.0703e+0 (9.55e-2) -	5.2543e-1 (1.38e-1) =	6.3341e-1 (1.26e-1) -	5.2563e-1 (1.62e-1)
UF6	3.9102e-1 (1.26e-1) =	3.9160e-1 (7.58e-2) =	1.4211e-1 (2.47e-2) +	3.7554e-1 (2.18e-2) =	4.4067e-1 (6.41e-2) -	4.5274e-1 (4.32e-2) -	3.7661e-1 (7.47e-2)
UF7	4.7546e-1 (1.77e-1) -	2.3092e-1 (2.48e-1) -	3.3065e-2 (4.82e-2) -	1.3341e-2 (9.09e-4) +	2.2646e-1 (2.61e-1) -	3.1521e-1 (2.59e-1) -	3.1031e-2 (1.06e-1)
UF8	2.8741e-1 (2.30e-1) -	1.1058e-1 (5.11e-2) =	1.5769e-1 (5.96e-2) -	1.2721e-1 (9.70e-3) -	9.6619e-2 (2.88e-2) =	1.0052e-1 (1.29e-2) -	9.3945e-2 (7.22e-2)
UF9	2.6355e-1 (3.35e-2) -	1.7194e-1 (3.57e-2) -	1.5937e-1 (6.44e-2) -	1.1807e-1 (1.11e-2) -	1.6370e-1 (5.13e-2) -	1.8932e-1 (6.28e-3) -	7.7286e-2 (6.56e-2)
UF10	7.3227e-1 (1.69e-1) -	4.7879e-1 (7.97e-2) +	2.3738e-1 (5.84e-4) +	2.5258e+0 (1.61e-1) -	8.5939e-1 (1.29e-1) -	6.6767e-1 (8.06e-2) -	5.3962e-1 (1.80e-1)
NN1 (Statlog_Australian)	6.2055e-1 (1.20e-2) -	7.6301e-1 (1.86e-2) -	6.9280e-1 (2.36e-2) -	7.9543e-1 (1.37e-2) =	7.2571e-1 (2.56e-2) -	7.3659e-1 (2.52e-2) -	8.1633e-1 (1.16e-2)
NN2 (Climate)	6.3505e-1 (1.59e-2) -	8.0321e-1 (2.71e-2) -	7.0199e-1 (2.56e-2) -	8.2303e-1 (1.61e-2) -	7.6123e-1 (2.47e-2) -	7.7099e-1 (2.83e-2) -	8.4754e-1 (7.12e-3)
NN3 (Bench_Sonar)	5.0728e-1 (1.79e-2) -	6.7753e-1 (2.30e-2) -	5.4119e-1 (2.67e-2) -	7.6268e-1 (1.50e-2) +	6.7969e-1 (2.37e-2) -	6.7220e-1 (2.44e-2) -	7.2155e-1 (1.52e-2)
+/-/=	6/25/3	3/23/8	7/25/2	7/24/3	3/24/7	4/24/6	—

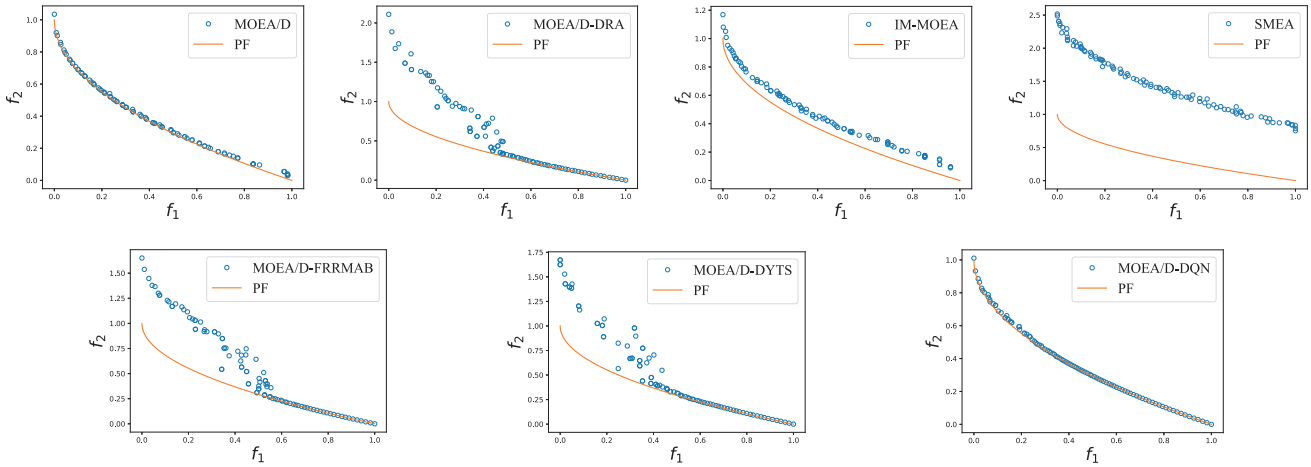


Fig. 3: Populations with the median IGD values obtained by seven MOEAs on ZDT1.

ZDT1 and UF1 with more decision variables, where the number of function evaluations is also set proportionally to the number of decision variables. It can be found that the performance of MOEA/D-DQN fluctuates slightly with the increase of number of decision variables, and is better than the compared MOEAs on all the test instances besides UF1 with 50 and 150 decision variables. As a consequence, the effectiveness of the proposed MOEA on different types of MOPs is demonstrated.

To investigate the efficiency of the compared MOEAs, Table III presents the runtimes obtained by the compared MOEAs. It can be found that the single-operator based MOEA/D, MOEA/D-DRA, and IM-MOEA are more efficient than the multi-operator based MOEA/D-FRRMAB, MOEA/D-DYTS, and MOEA/D-DQN. Besides, SMEA is inefficient due to its HV based environmental selection. In short, the efficiency of the proposed MOEA/D-DQN is similar to existing MOEAs with adaptive operator selection.

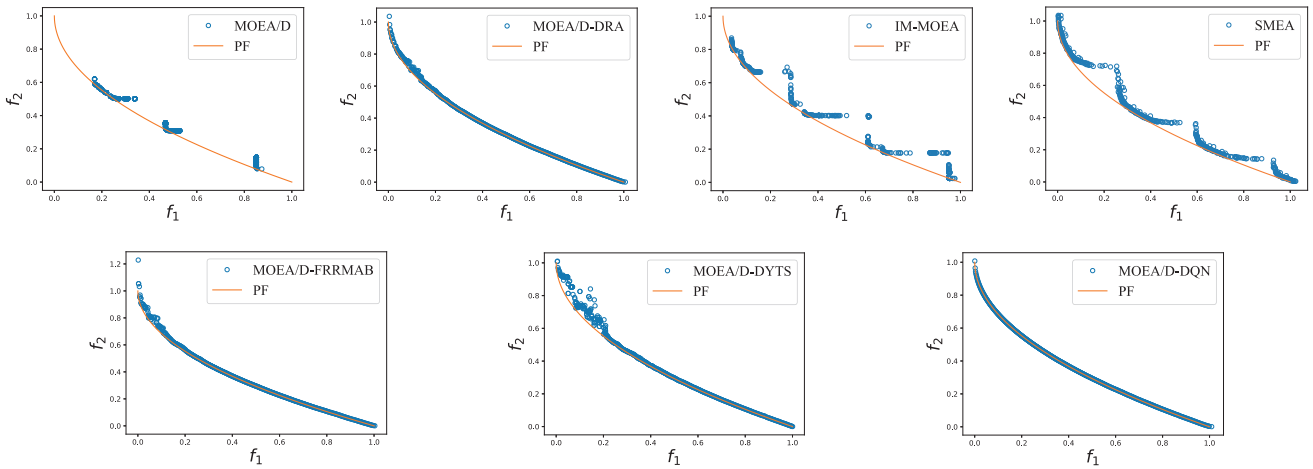


Fig. 4: Populations with the median IGD values obtained by seven MOEAs on UF1.

TABLE III: Average Runtimes (in Second) Obtained by Seven MOEAs on Benchmark Suites, Where the Best Result in Each Row is Highlighted.

Problem	MOEA/D	MOEA/D-DRA	IM-MOEA	SMEA	MOEA/D-FRRMAB	MOEA/D-DYTS	MOEA/D-DQN
ZDT problems	2.4919	3.4711	2.8350	7.6200	3.4580	5.5583	9.0468
DTLZ problems	7.2857	5.5611	9.1036	130.7161	8.7838	18.8286	27.4248
WFG problems	8.0473	5.1461	8.2802	248.3478	7.8594	15.2994	29.3160
UF problems	91.5787	87.8362	21.3048	4428.4500	127.5550	241.0180	295.9573
Neural network training	6.9808	5.2872	11.3552	11.7115	7.2369	10.7930	11.6806

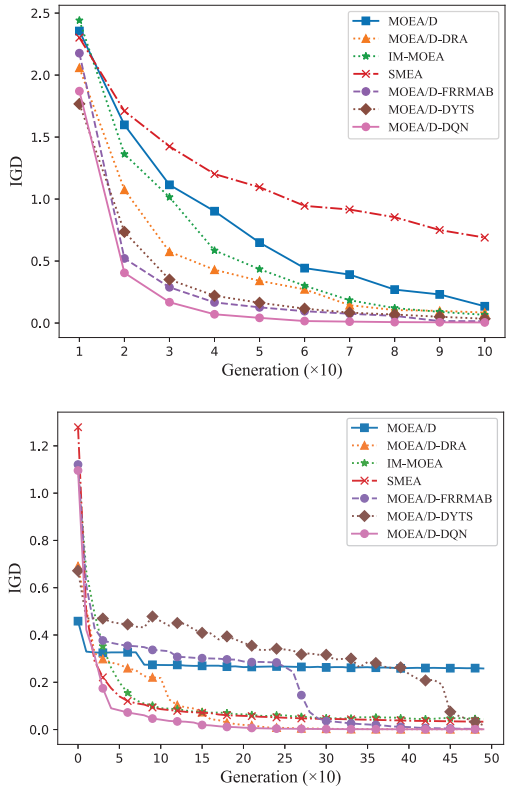


Fig. 5: Convergence profiles of mean IGD values obtained by seven MOEAs on ZDT1 (up) and UF1 (down).

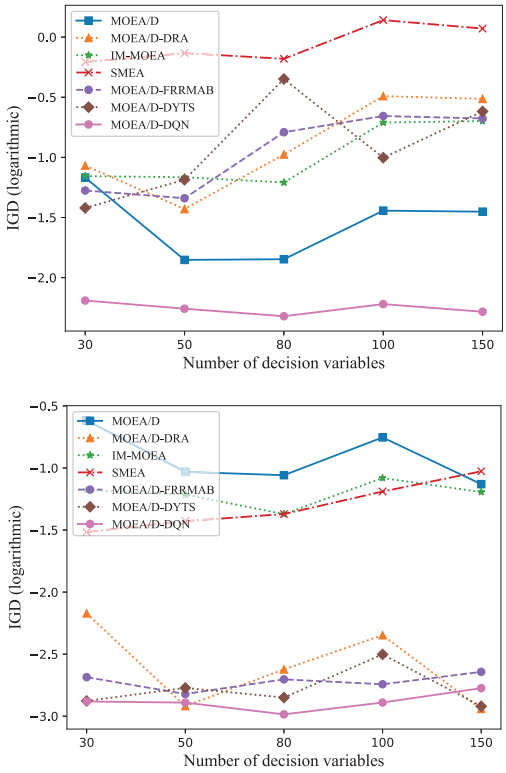


Fig. 6: Mean IGD values obtained by seven MOEAs on ZDT1 (up) and UF1 (down) with different numbers of decision variables.

TABLE IV: IGD/HV Values Obtained by MOEA/D-DQN with Multiple and a Single Operator on Benchmark Suites/Neural Network Training, Where the Best Result in Each Row is Highlighted and "+", "−", and "=" Indicate that the Result is Significantly Better, Significantly Worse, and Statistically Similar to the Original MOEA/D-DQN.

Problem	MOEA/D-OP1	MOEA/D-OP2	MOEA/D-OP3	MOEA/D-OP4	MOEA/D-DQN
ZDT1	1.1554e-2 (4.48e-3) -	3.6240e-2 (1.75e-2) -	4.4362e-2 (2.00e-2) -	5.2718e-2 (1.96e-2) -	6.9980e-3 (1.76e-3)
ZDT2	2.0159e-2 (1.23e-2) =	6.0830e-2 (5.62e-2) -	1.0700e-1 (5.54e-2) -	1.4857e-1 (5.73e-2) -	2.4188e-2 (2.15e-2)
ZDT3	4.3307e-2 (2.58e-2) -	2.2010e-1 (6.31e-2) -	2.2983e-1 (5.18e-2) -	2.0123e-1 (4.23e-2) -	1.7122e-2 (5.58e-3)
ZDT4	1.2382e-1 (6.48e-2) +	6.2584e-1 (3.03e-1) -	1.2818e+0 (1.04e+0) -	1.7178e+1 (6.35e+0) -	2.5453e-1 (9.70e-2)
ZDT6	4.5689e-2 (7.93e-2) -	3.1665e-3 (8.09e-5) =	1.0660e-2 (2.45e-2) -	9.2326e-3 (8.38e-3) -	3.8825e-3 (2.74e-3)
DTLZ1	2.0782e-2 (1.89e-4) -	1.0673e+0 (1.60e+0) -	2.9318e-1 (6.21e-1) -	1.3359e+0 (2.13e+0) -	1.8507e-2 (8.58e-4)
DTLZ2	5.4495e-2 (9.26e-5) -	7.5482e-2 (3.47e-3) -	6.8332e-2 (3.38e-3) -	8.1626e-2 (4.28e-3) -	4.7007e-2 (1.53e-4)
DTLZ3	6.7471e-1 (1.28e+0) =	3.4119e+1 (3.35e+1) -	3.6550e+1 (4.51e+1) -	2.2049e+1 (3.30e+1) -	1.2903e+0 (3.04e+0)
DTLZ4	3.0991e-1 (2.92e-1) -	7.2420e-2 (4.91e-3) -	1.5007e-1 (1.38e-1) -	1.0033e-1 (3.19e-2) -	4.7086e-2 (2.36e-4)
DTLZ5	3.3019e-2 (4.74e-4) -	3.0277e-2 (1.11e-3) -	2.9569e-2 (1.16e-3) -	2.7432e-2 (1.66e-3) -	2.6538e-2 (5.78e-4)
DTLZ6	3.3856e-2 (4.15e-5) -	3.3576e-2 (1.34e-4) -	3.3299e-2 (2.57e-4) -	3.3060e-2 (2.80e-4) -	2.9008e-2 (4.72e-5)
DTLZ7	1.5409e-1 (1.57e-3) -	1.5373e-1 (2.81e-3) -	5.2182e-1 (3.28e-1) -	5.0073e-1 (3.31e-1) -	1.1006e-1 (3.56e-4)
WFG1	2.5873e-1 (2.10e-2) +	6.8505e-1 (1.46e-1) =	1.1416e+0 (1.50e-1) -	1.4677e+0 (3.40e-2) -	7.1675e-1 (8.33e-2)
WFG2	2.3869e-1 (2.12e-2) -	2.9602e-1 (2.49e-2) -	2.8355e-1 (2.16e-2) -	2.8629e-1 (2.15e-2) -	2.0742e-1 (1.96e-2)
WFG3	1.5931e-1 (4.76e-3) -	1.6717e-1 (1.43e-2) -	1.8733e-1 (2.05e-2) -	2.2936e-1 (2.09e-2) -	1.2825e-1 (4.77e-3)
WFG4	2.6389e-1 (6.64e-3) -	3.4070e-1 (1.65e-2) -	3.2749e-1 (1.48e-2) -	3.6940e-1 (1.69e-2) -	2.3703e-1 (5.38e-3)
WFG5	2.5097e-1 (3.38e-3) -	2.7192e-1 (7.79e-3) -	2.5955e-1 (6.21e-3) -	2.6298e-1 (6.24e-3) -	2.2353e-1 (3.01e-3)
WFG6	2.9117e-1 (1.93e-2) +	3.9338e-1 (1.15e-2) +	3.7678e-1 (2.51e-2) +	3.6522e-1 (2.01e-2) +	4.2542e-1 (4.02e-2)
WFG7	2.8538e-1 (1.65e-2) +	3.7268e-1 (2.80e-2) =	3.3623e-1 (1.86e-2) +	3.7822e-1 (2.27e-2) =	3.6882e-1 (1.13e-2)
WFG8	3.3973e-1 (8.30e-3) +	5.0260e-1 (3.83e-2) -	4.5396e-1 (3.01e-2) +	4.8381e-1 (2.27e-2) =	4.8352e-1 (5.64e-2)
WFG9	3.0634e-1 (4.95e-2) +	3.5093e-1 (2.58e-2) =	3.2316e-1 (4.28e-2) =	3.4944e-1 (3.40e-2) =	3.5458e-1 (3.56e-2)
UF1	1.3930e-1 (1.06e-1) -	1.4971e-2 (9.89e-3) -	2.9363e-2 (5.85e-2) =	1.7860e-3 (2.32e-3) =	1.3666e-3 (1.05e-4)
UF2	1.7842e-2 (1.13e-2) -	2.5884e-3 (8.69e-4) -	2.6555e-3 (2.01e-3) =	2.8186e-3 (9.14e-4) -	2.0228e-3 (4.02e-4)
UF3	3.4550e-1 (4.64e-2) -	1.5582e-1 (1.01e-1) -	8.3464e-2 (4.50e-2) -	1.6469e-2 (1.74e-2) =	2.1607e-2 (1.97e-2)
UF4	4.8002e-2 (6.43e-3) -	5.4258e-2 (2.88e-3) -	5.3660e-2 (3.31e-3) -	5.2840e-2 (3.21e-3) -	3.3923e-2 (1.36e-3)
UF5	5.1969e-1 (1.35e-1) =	5.0021e-1 (1.33e-1) =	5.3475e-1 (1.38e-1) =	4.7151e-1 (1.45e-1) =	5.2563e-1 (1.62e-1)
UF6	4.3553e-1 (6.92e-2) -	4.1098e-1 (1.14e-1) =	4.0028e-1 (8.26e-2) =	4.3869e-1 (1.12e-1) -	3.7661e-1 (7.47e-2)
UF7	5.3382e-1 (1.62e-1) -	2.3914e-1 (2.68e-1) -	3.8103e-1 (2.14e-1) -	1.7435e-1 (2.63e-1) -	3.1031e-2 (1.06e-1)
UF8	2.5450e-1 (1.92e-1) -	2.3313e-1 (6.28e-2) -	2.2645e-1 (6.56e-2) -	2.9513e-1 (1.32e-1) -	9.3945e-2 (7.22e-2)
UF9	2.2444e-1 (7.01e-2) -	1.9439e-1 (1.07e-2) -	1.6755e-1 (5.29e-2) -	1.8379e-1 (4.63e-2) -	7.7286e-2 (6.56e-2)
UF10	7.6045e-1 (2.01e-1) -	6.5741e-1 (8.38e-2) -	7.0186e-1 (9.09e-2) -	6.9640e-1 (9.21e-2) -	5.3962e-1 (1.80e-1)
NN1 (Statlog_Australian)	6.4620e-1 (1.55e-2) -	7.9567e-1 (2.29e-2) -	8.0524e-1 (2.06e-2) -	8.0065e-1 (1.88e-2) -	8.1633e-1 (1.16e-2)
NN2 (Climate)	5.9942e-1 (1.88e-2) -	8.2200e-1 (2.22e-2) -	8.3815e-1 (2.11e-2) -	8.4409e-1 (3.55e-2) =	8.4754e-1 (7.12e-3)
NN3 (Bench_Sonar)	4.6028e-1 (2.19e-3) -	7.1667e-1 (2.65e-2) -	7.4554e-1 (2.14e-2) +	7.5682e-1 (2.32e-2) +	7.2155e-1 (1.52e-2)
+/-/=	6/25/3	1/27/6	4/25/5	2/25/7	—

C. Further Investigations of MOEA/D-DQN

To further verify the effectiveness of the proposed operator selection method, MOEA/D-DQN is compared to its variants with a single operator, so that the influence brought by the difference between other strategies can be totally eliminated. Table IV presents the statistical results of MOEA/D-DQN and its four variants, where OP1 denotes simulated binary crossover, OP2 denotes the crossover operator in MOEA/D-M2M, OP3 denotes DE/rand/1, and OP4 denotes DE/rand/2. It is obvious that the proposed MOEA/D-DQN still exhibits the best overall performance, and is competitive to different variants on different test suites. For example, MOEA/D-DQN is competitive to MOEA/D-OP1 on ZDT and WFG problems and competitive to MOEA/D-OP3 and MOEA/D-OP4 on UF problems. These observations are consistent with existing studies, where simulated binary crossover is more suitable for ZDT and WFG problems without variable linkages while differential evolution is more suitable for UF problems with complex variable linkages [25], [75].

Moreover, Fig. 7 depicts the ratio of operators selected by MOEA/D-DQN during the optimization process on ZDT1 and UF1. It can be found that MOEA/D-DQN prefers OP1 on ZDT1 while preferring OP1, OP2, and OP3 on UF1, and the other operators are also selected with a small probability. That is, MOEA/D-DQN does not select a single operator but assembles multiple operators for solving each problem. Although a single operator can lead to satisfactory performance on specific problems (e.g., differential evolution on UF problems), as evidenced by Table IV (e.g., the results on UF problems), using reinforcement learning to assemble multiple operators can further improve the performance. In short, the effectiveness of the proposed operator selection method is confirmed.

Lastly, the influence of the neighborhood size used in MOEA/D on the performance of MOEA/D-DQN is investigated. For this aim, Fig. 8 presents the mean IGD values obtained by MOEA/D-DQN with different neighborhood sizes on ZDT1 and UF1. It can be observed

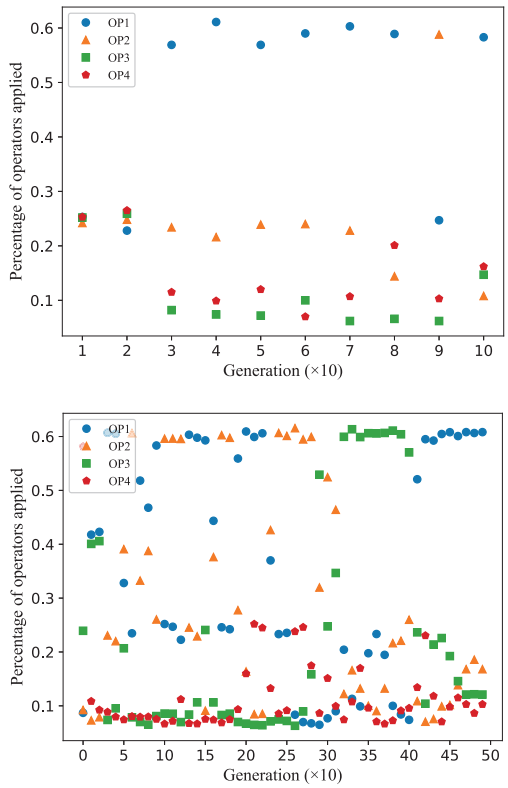


Fig. 7: Ratio of operators selected by MOEA/D-DQN during the optimization process of solving ZDT1 (up) and UF1 (down).

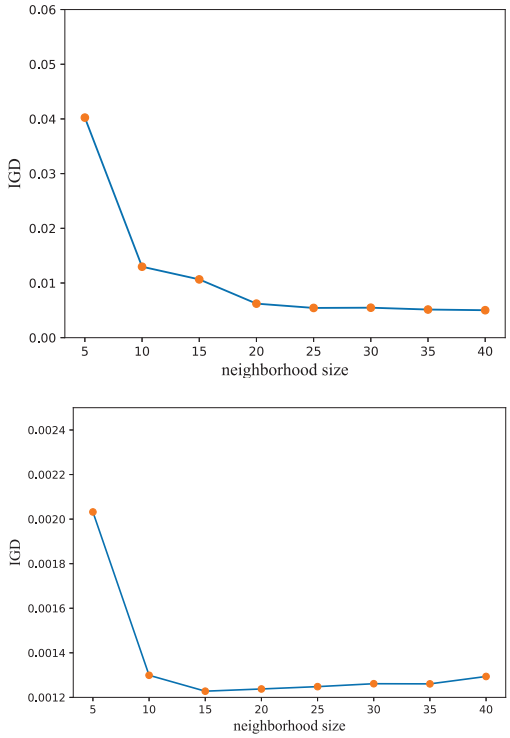


Fig. 8: Mean IGD values obtained by MOEA/D-DQN with different neighborhood sizes on ZDT1 (up) and UF1 (down).

that the obtained IGD values are very similar when the neighborhood size is larger than 15, which means that the performance of MOEA/D-DQN is not sensitive to relatively large neighborhood sizes, and it is reasonable to set the neighborhood size to 20.

V. Conclusions

In this paper, a reinforcement learning based adaptive operator selection method has been proposed for evolutionary multi-objective optimization. By using deep neural networks to learn the relations between decision variables and Q values (i.e., cumulative fitness improvements) of candidate operators during the optimization process, the proposed method can suggest promising operators to evolve the population for different MOPs. The proposed method has been embedded in a decomposition based MOEA, which has shown competitive performance to state-of-the-art MOEAs and operator selection methods on a variety of benchmark problems. Besides, the experimental results have also verified that different operators are suitable for different types of problems, and the proposed method can generally select the most suitable one for each problem.

This work has shown the bright prospect of reinforcement learning in assisting evolutionary algorithms for solving static optimization problems. While the proposed method uses a discrete action space aiming to select suitable operators with predefined parameters, it is reasonable to additionally consider a continuous action space to adaptively tune the parameters of operators in the future. Moreover, since the proposed method only considers the decision variables of a single solution as a state, the involvement of more information about the whole population is highly desirable, where the suitable operators for different types of problems can be selected more accurately.

References

[1] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[2] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys*, vol. 54, no. 8, p. 174, 2022.

[3] Y. Tian, S. Peng, X. Zhang, T. Rodemann, K. C. Tan, and Y. Jin, "A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 5–18, 2020.

[4] S. Yang, Y. Tian, C. He, X. Zhang, K. C. Tan, and Y. Jin, "Gradient guided evolutionary approach to training deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[5] 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.

[6] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2008.

[7] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, 2020.

[8] T. Rodemann, "Industrial portfolio management for many-objective optimization algorithms," in *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, 2018.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] T. M. Lindauer, H. H. Hoos, F. Hutter, and T. Schaub, "AutoFolio: An automatically configured algorithm selector," *Journal of Artificial Intelligence Research*, vol. 53, pp. 745–778, 2015.

[11] C. He, Y. Tian, H. Wang, and Y. Jin, "A repository of real-world datasets for data-driven evolutionary multiobjective optimization," *Complex & Intelligent Systems*, vol. 6, pp. 189–197, 2020.

[12] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evolutionary Computation*, vol. 27, no. 1, pp. 3–45, 2018.

[13] Y. Tian, S. Peng, T. Rodemann, X. Zhang, and Y. Jin, "Automated selection of evolutionary multi-objective optimization algorithms," in *Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence*, 2019, pp. 3225–3232.

[14] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Extreme value based adaptive operator selection," in *Proceedings of the 2008 International Conference on Parallel Problem Solving from Nature*, 2008, pp. 175–184.

[15] K. McClymont and E. C. Keedwell, "Markov chain hyper-heuristic (MCHH) an online selective hyper-heuristic for multi-objective continuous problems," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011, pp. 2003–2010.

[16] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, 2008, pp. 913–920.

[17] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 114–130, 2013.

[18] D. Molina, A. LaTorre, and F. Herrera, "SHADE with iterative local search for large-scale global optimization," in *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, 2018.

[19] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2020.

[20] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media, 2007.

[21] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[22] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[23] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[24] P. L. naga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA: Kluwer, 2001.

[25] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, 2018.

[26] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 203–208.

[27] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3129–3142, 2021.

[28] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[29] L. Davis, *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, 1991.

[30] Q. Lin, Z. Liu, Q. Yan, Z. Du, C. A. C. Coello, Z. Liang, W. Wang, and J. Chen, "Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm," *Information Sciences*, vol. 339, pp. 332–352, 2016.

[31] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2013.

[32] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag, "Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms," in *Proceedings of the 2009 International Conference on Learning and Intelligent Optimization*, 2009, pp. 176–190.

[33] N. Hitomi and D. Selva, "A classification and comparison of credit assignment strategies in multiobjective adaptive operator selection," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 294–314, 2016.

[34] V. L. Huang, A. K. Qin, P. N. Suganthan, and M. F. Tasgetiren, "Multi-objective optimization based on self-adaptive differential evolution algorithm," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 3601–3608.

[35] K. Li, Á. Fialho, and S. Kwong, "Multi-objective differential evolution with adaptive control of parameters and operators," in *Proceedings of the 2011 International Conference on Learning and Intelligent Optimization*, 2011, pp. 473–487.

[36] 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.

[37] L. Sun and K. Li, "Adaptive operator selection based on dynamic Thompson sampling for MOEA/D," in *Proceedings of the 2020 International Conference on Parallel Problem Solving from Nature*, 2020, pp. 271–284.

[38] W. Lin, Q. Lin, J. Ji, Z. Zhu, C. A. C. Coello, and K.-C. Wong, "Decomposition-based multiobjective optimization with bicriteria assisted adaptive operator selection," *Swarm and Evolutionary Computation*, vol. 60, p. 100790, 2021.

[39] C. Huang, L. Li, C. He, R. Cheng, and X. Yao, "Operator-adapted evolutionary large-scale multiobjective optimization for voltage transformer ratio error estimation," in *Proceedings of the 2021 International Conference on Evolutionary Multi-Criterion Optimization*, 2021, pp. 672–683.

[40] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708–711, 2007.

[41] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.

[42] 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.

[43] Z. Yan, Y. Tan, W. Zheng, L. Meng, and H. Zhang, "Leader recommend operators selection strategy for a multiobjective evolutionary algorithm based on decomposition," *Information Sciences*, vol. 550, pp. 166–188, 2021.

[44] A. Santiago, B. Dorronsoro, A. J. Nebro, J. J. Durillo, O. Castillo, and H. J. Fraire, "A novel multi-objective evolutionary algorithm with fuzzy logic based adaptive selection of operators: FAME," *Information Sciences*, vol. 471, pp. 233–251, 2019.

[45] C. Wang, R. Xu, J. Qiu, and X. Zhang, "AdaBoost-inspired multi-operator ensemble strategy for multi-objective evolutionary algorithms," *Neurocomputing*, vol. 384, pp. 243–255, 2020.

[46] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. The MIT Press, Cambridge, Massachusetts, 1998.

[47] K. V. Moffaert, M. M. Drugan, and A. Nowé, "Hypervolume-based multi-objective reinforcement learning," in *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*, 2013, pp. 352–366.

[48] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3103–3114, 2021.

[49] F. Zou, G. G. Yen, L. Tang, and C. Wang, "A reinforcement learning approach for dynamic multi-objective optimization," *Information Sciences*, vol. 546, pp. 815–834, 2021.

[50] S. D. Handoko, D. T. Nguyen, Z. Yuan, and H. C. Lau, "Reinforcement learning for adaptive operator selection in memetic search applied to quadratic assignment problem," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 193–194.

[51] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.
[52] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *Advances in Neural Information Processing Systems*, 1999, pp. 1057–1063.
[53] J. Sun, X. Liu, T. Bäck, and Z. Xu, "Learning adaptive differential evolution algorithm from optimization experiences by policy gradient," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 666–680, 2021.
[54] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proceedings of the 1998 AAAI Conference on Artificial Intelligence*, 1998, pp. 761–768.
[55] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[56] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[57] A. Fialho, R. Ros, M. Schoenauer, and M. Sebag, "Comparison-based adaptive strategy selection with bandits in differential evolution," in *Proceedings of the 2010 International Conference on Parallel Problem Solving from Nature*, 2010, pp. 194–203.
[58] Y. Tian, X. Xiang, X. Zhang, R. Cheng, and Y. Jin, "Sampling reference points on the Pareto fronts of benchmark multi-objective optimization problems," in *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, 2018.
[59] K. Deb, K. Sindhya, and T. Okabe, "Self-adaptive simulated binary crossover for real-parameter optimization," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 2007, pp. 1187–1194.
[60] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450–455, 2013.
[61] K. Deb and M. Goyal, "A combined genetic adaptive search (geneas) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.
[62] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the 2016 AAAI Conference on Artificial Intelligence*, 2016.
[63] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 838–856, 2015.
[64] H. Zhang, A. Zhou, S. Song, Q. Zhang, X. Gao, and J. Zhang, "A self-organizing multiobjective evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 792–806, 2016.
[65] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.
[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[67] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.
[68] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization*. Springer, 2005, pp. 105–145.
[69] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.
[70] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, Tech. Rep., 2008.
[71] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[72] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.
[73] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
[74] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
[75] M. Li, S. Yang, and X. Liu, "Pareto or non-Pareto: Bi-criterion evolution in multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 645–665, 2015.

Ye Tian received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively.

He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. His current research interests include evolutionary computation and its applications. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the 2022 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award.

Xiaopeng Li received the B.Sc. degree from Jinan University, Jinan, China, in 2019, where he is currently pursuing the M.Sc. degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

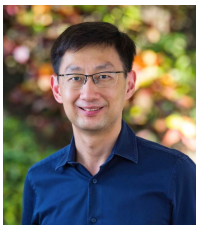
His current research interests include evolutionary optimization and reinforcement learning.

Haiping Ma received the B.E. degree from Anhui University, Hefei, China, in 2008 and the Ph.D. degree from University of Science and Technology of China, Hefei, China, in 2013.

She is currently a Lecturer with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. Her current research interests include data mining, multi-objective optimization methods, and their applications.

Xingyi Zhang (SM'18) received the B.Sc. degree from Fuyang Normal College, Fuyang, China, in 2003, and the M.Sc. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2009, respectively.

He is currently a Professor with the School of Artificial Intelligence, Anhui University, Hefei, China. His current research interests include unconventional models and algorithms of computation, multi-objective optimization, and membrane computing. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award.



Kay Chen Tan (SM'08-F'14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor of the Department of Computing at the Hong Kong Polytechnic University, Hong Kong. He has published over 200 refereed articles and six books, and holds one U.S. patent on surface defect detection.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He has served as the Editor-in-Chief of IEEE Transactions on Evolutionary Computation from 2015-2020 and IEEE Computational Intelligence Magazine from 2010-2013, and currently serves as the Editorial Board Member of over 10 journals. He is currently an IEEE Distinguished Lecturer Program (DLP) speaker and Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Yaochu Jin (SM'02-F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Germany, in 2001.

He is an Alexander von Humboldt Professor for Artificial Intelligence, Chair of Nature Inspired Computing and Engineering, Faculty of Technology, Bielefeld University, Germany. He is also a Distinguished Chair, Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, U.K. He was a "Finland Distinguished Professor" of University of Jyväskylä, Finland, "Changjiang Distinguished Visiting Professor", Northeastern University, China, and "Distinguished Visiting Scholar", University of Technology Sydney, Australia. His main research interests include evolutionary optimization, evolutionary learning, trustworthy machine learning, and evolutionary developmental systems.

Prof. Jin is presently the Editor-in-Chief of Complex & Intelligent Systems. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS in 2016-2021 and an IEEE Distinguished Lecturer in 2013–2015 and 2017–2019, the Vice President for Technical Activities of the IEEE Computational Intelligence Society (2015–2016). He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, and the 2015, 2017, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award. He was named by the Web of Science as a Highly Cited Researcher from 2019 to 2021 consecutively. He is a Member of Academia Europaea.