# Constrained Multi-objective Optimization with Deep Reinforcement Learning Assisted Operator Selection

Fei Ming, Wenyin Gong, *Member, IEEE*, Ling Wang, *Member, IEEE*, and Yaochu Jin, *Fellow, IEEE*

***Abstract*—Solving constrained multi-objective optimization problems with evolutionary algorithms has attracted considerable attention. Various constrained multi-objective optimization evolutionary algorithms (CMOEAs) have been developed with the use of different algorithmic strategies, evolutionary operators, and constraint-handling techniques. The performance of CMOEAs may be heavily dependent on the operators used, however, it is usually difficult to select suitable operators for the problem at hand. Hence, improving operator selection is promising and necessary for CMOEAs. This work proposes an online operator selection framework assisted by Deep Reinforcement Learning. The dynamics of the population, including convergence, diversity, and feasibility, are regarded as the *state*; the candidate operators are considered as *actions*; and the improvement of the population state is treated as the *reward*. By using a Q-Network to learn a policy to estimate the Q-values of all actions, the proposed approach can adaptively select an operator that maximizes the improvement of the population according to the current state and thereby improve the algorithmic performance. The framework is embedded into four popular CMOEAs and assessed on 42 benchmark problems. The experimental results reveal that the proposed Deep Reinforcement Learning-assisted operator selection significantly improves the performance of these CMOEAs and the resulting algorithm obtains better versatility compared to nine state-of-the-art CMOEAs.**

***Index Terms*—Constrained multi-objective optimization, evolutionary algorithms, evolutionary operator selection, Deep Reinforcement Learning, Deep Q-Learning.**

## I. Introduction

CONSTRAINED multi-objective optimization problems (CMOPs) contain multiple conflicting objective functions and constraints, which widely exist in real-world applications and scientific research [1]. For example, web service location-allocation problems [2] and vehicle scheduling of urban bus lines [3].

The development of solving CMOPs by constrained multi-objective optimization evolutionary algorithms (CMOEAs) has seen prominent growth due to the wide existence of this kind of problem. Generally, a CMOEA contains three key components that affect its performance: the algorithmic strategy used to assist the selection procedures, the constraint-handling technique (CHT) to handle constraints, and the evolutionary operator to generate new solutions. In the past decade, a large number of CMOEAs have been proposed, most of which focus on enhancing algorithmic strategies [4]–[7] and CHTs [3], [8]. By contrast, little research focused on how to adaptively select the evolutionary operator in CMOEAs has been reported in the literature.

According to [9], different evolutionary operators [1] are suited to different optimization problems. For example, crossovers in Genetic Algorithms (GAs) [10] are effective in overcoming multimodal features; mutations in Differential Evolution (DE) [11] can handle complex linkages of variables because they use the weighted difference between two other solutions. The Particle Swarm Optimizer (PSO) [12] shows good performance on convergence speed and the Competitive Swarm Optimizer (CSO) [13] is good at dealing with large-scale optimization problems because of its competitive mechanism, and the multi-objective PSO (MOPSO) [14], [15] is particularly designed and effective for multi-objective situations. Given the fact that a real-world CMOP is usually subject to unknown features and challenges, using a fixed evolutionary operator will limit the applicability of a CMOEA. Although ensemble and adaptive selection of operators have received increased attention in the multi-objective optimization community [16]–[18], unfortunately, no research efforts have been dedicated to constrained multi-objective optimization. Hence, it is necessary to study the effect of adaptive operator selection in dealing with CMOPs.

Among the Reinforcement Learning community, Deep Reinforcement Learning (DRL) is an emerging topic and has been applied in several real-world applications in the multi-objective optimization field [19]–[23]. Also, its effectiveness in dealing with multi-objective optimization problems (MOPs) has been studied recently [17], [24], [25]. Since DRL uses a deep neural network to approximate the action-value function, it can handle continuous state space and thereby is suitable for operator selection for CMOPs because a population could have infinite states during the evolution. Therefore, it is promising to adopt DRL techniques to fill the research gap of operator selection in handling CMOPs. However, when applying DRL to CMOPs, the main challenge is to properly consider constraint satisfaction and feasibility in a DRL model.

This work proposes a DRL-assisted online operator selection framework for CMOPs. Specifically, the main contribu-

This work was partly supported by the National Natural Science Foundation of China under Grant Nos. 62076225 and 62073300, and the Natural Science Foundation for Distinguished Young Scholars of Hubei under Grant No. 2019CFA081. *(Corresponding authors: Wenyin Gong & Yaochu Jin.)*

F. Ming and W. Gong are with the School of Computer Science, China University of Geosciences, Wuhan 430074, China. (Email: wygong@cug.edu.cn)

L. Wang is with the Department of Automation, Tsinghua University, Beijing 100084, China. (Email: wangling@tsinghua.edu.cn)

Yaochu Jin is with the Faculty of Technology, Bielefeld University, North Rhine-Westphalia, 33619, Bielefeld, Germany (Email: yaochu.jin@uni-bielefeld.de).

[1] An evolutionary operator means the operations of an evolutionary algorithm used for generating offspring solution, such as the crossover and mutation of GA, the differential variation of DE, and the particle swarm update of PSO/CSO.

tions are as follows:

1) We propose a novel DRL model for operator selection in CMOPs. The state of the population, including the convergence, diversity, and feasibility performance, is viewed as the state, the candidate operators are regarded as actions, and the improvement of the population state is the reward. Then, we develop a Deep Q-Learning (DQL)-assisted operator selection framework for CMOEAs. A deep Q-Network (DQN) is trained to learn a policy that estimates the Q-value of an action at the current state.
2) The proposed model can contain an arbitrary number of operators and the proposed framework can be easily employed in any CMOEA. In this work, we develop an Operator Selection (OS) method and further instantiate it by using GA and DE as candidate operators in the DRL model. Then, the OS is embedded into four popular CMOEAs: CCMO [4], PPS [26], MOEA/D-DAE [27], and EMCMO [7].
3) Extensive experimental studies on four popular and challenging CMOP benchmark test suites demonstrated that the OS method can significantly improve the performance of CMOEAs. Furthermore, compared to nine state-of-the-art CMOEAs, our methods obtained better versatility on different problems.

The remainder of this article is organized as follows. Section II introduces the related work. Section III elaborates on the proposed methods. Then, experiments and analysis are presented in Section IV. Finally, conclusions and future research directions are given in Section V.

## II. Related Work and Motivations

### A. Preliminaries

Without loss of generality, a CMOP can be formulated as:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_m(\mathbf{x}))^T \\ \text{subject to} \quad & \mathbf{x} \in \mathbb{R} \\ & g_i(\mathbf{x}) \leq 0,\ i = 1, \cdots, p \\ & h_j(\mathbf{x}) = 0,\ j = p+1, \cdots, q \end{aligned}, \tag{1}$$

where $m$ denotes the number of objective functions; $\mathbf{x} = (x_1, \ldots, x_n)^T$ denotes the decision vector with $n$ dimensions (*i.e.,* the number of decision variables); $\mathbf{x} \in \mathbb{R}$, and $\mathbb{R} \subseteq \mathbb{R}^n$ represents the search space. $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ are the $i$-th inequality and $j$-th equality constraints, and $q$ denotes the number of constraints.

In a CMOP, the degree of the $j$-th constraint violation (CV) of a solution $\mathbf{x}$ is

$$\varphi_j(\mathbf{x}) = \begin{cases} \max(0, g_j(\mathbf{x})), & j = 1, \cdots, p \\ \max(0, |h_j(\mathbf{x})| - \sigma), & j = p+1, \cdots, q \end{cases}, \tag{2}$$

where $\sigma$ is a small enough positive value to relax the equality constraints into inequality ones. The overall CV of $\mathbf{x}$ is

$$\phi(\mathbf{x}) = \sum_{j=1}^{q} \varphi_j(\mathbf{x}). \tag{3}$$

$\mathbf{x}$ is feasible if $\phi(\mathbf{x}) = 0$; otherwise, it is infeasible.

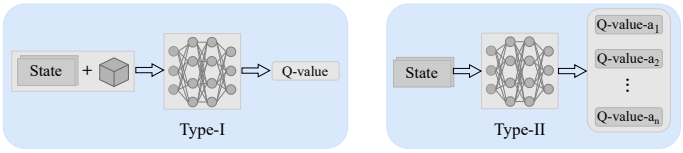


Fig. 1. An illustration of two types of working principles of the DQL technique.

### B. Adaptive Operator Selection in MOPs

In recent years, adaptive operator selection is gradually attracting research attention in the design of multi-objective optimization evolutionary algorithms (MOEAs).

Wang *et al.* [16] proposed a multi-operator ensemble method that uses multiple subpopulations for multiple operators and adjusts their sizes according to the effectiveness of each action to reward good operators and punish bad ones. Tian *et al.* [17] adopted DRL to construct an adaptive operator selection method for MOEA/D. The proposed DRL model regards the decision vectors as states and the operators as actions; then, the fitness improvement of the solution brought by the operator is taken as the reward. Alejandro *et al.* [28] suggested a fuzzy selection of operators that chooses the most appropriate operators during evolution to promote both diversity and convergence of solutions. Dong *et al.* [18] devised a test-and-apply structure to adaptively select the operator for decomposition-based MOEAs. The proposed structure contains a test phase that uses all operators to generate offspring and test the survival rate as the credit of operators. Then, the apply phase uses only the best operator for the remaining evolution. Yuan *et al.* [29] investigated the effect of different variation operators (*i.e.,* GA and DE) in MOEA. McClymont *et al.* [30] proposed a Markov chain hyper-heuristic that employs Reinforcement Learning and Markov chains to adaptively select heuristic methods, including different operators. Lin *et al.* [31] devised a one-to-one ensemble mechanism that measures the credits of operators in both the decision and objective spaces and designed an adaptive rule to guarantee that suitable operators can generate more solutions.

### C. DRL and its Applications in MOPs

As we know, Evolutionary Computation aims to find the optima from a static environment. However, Reinforcement Learning can learn an optimal coping strategy in a dynamic environment [32]. The policy enables the agent to adaptively take actions that gain the largest cumulative reward at the current environmental state [33]. Different from Reinforcement Learning, DRL uses a deep neural network to approximate the action-value function. More specifically, Q-Learning learns a policy through a discrete Q-Table that records the cumulative reward of each action at each state [34]. In contrast, DQL trains a Q-Network that approximates the action-value function to estimate the expected reward of each action [35]. The action-value function is also known as a probability model that estimates the probability of taking each action. Generally, DQL has two types of working principles. We assume that the DQN is trained, where two types are depicted in the left

half and right half of Fig 1, respectively. For the Type-I DQL model, the input includes a state and an action (shown in a cube), and the output is the Q-value of this action under this state. For the Type-II DQL model, the input contains only the state, and the DQN can output the Q-values of all actions.

Without loss of generality, DQL trains a Q-Network through the loss function

$$\mathcal{L} = \frac{1}{|\mathcal{T}|}\sum_{\mathbf{t}\in\mathcal{T}}(Q(s_t, a_t) - q_t)^2, \tag{4}$$

where $\mathcal{T}$ is the training data randomly sampled from an **experience replay** (EP); the experience replay is a data set that records historical data for the training of the Network; $Q(s_t, a_t)$ is the output of the Network with the input $(s_t, a_t)$ and $a_T$, and $q_t$ is the Q-value of taking action $a_t$ at the state $s_t$, which is formulated as

$$q_t = r_t + \gamma \max_{a'\in\mathcal{A}} Q(s_{t+1}, a'), \tag{5}$$

where $Q(s_{t+1}, a')$ is the maximum reward of taking the next action under the state where $a_t$ is performed. Therefore, $q_t$ can not only calculate the current reward of performing action $a$, but also estimate the maximum cumulative reward in the future.

Recently, DRL techniques are attracting more and more attention in the multi-objective optimization community. Researchers either employ DRL techniques to solve real-world MOPs or utilize them to enhance the performance of MOEAs.

*1) Applications in Real-world MOPs:* Li *et al.* [22] presented a DRL-assisted multi-objective bacterial foraging optimization algorithm for a five-objective renewable energy accommodation problem. Luca *et al.* [20] suggested solving the multi-objective placement problem of virtual machines in cloud data centers by a DRL framework that selects the best placement heuristic for each virtual machine. Stefan *et al.* [19] designed a DRL-based DeepCoord approach that trains an agent to adaptively learn how to best coordinate services without prior and expert knowledge in dealing with a three-objective service coordination problem.

*2) Applications in MOEAs:* Tian *et al.* [17] used the DRL technique to adaptively select the evolutionary operators in dealing with MOPs. In their methods, the decision variable of a solution is regarded as the state, the operator is regarded as the action, and the improvement of fitness of the solution is the reward. Li *et al.* [24] proposed modeling each subproblem decomposed by the reference vectors as a neural network and generating Pareto-optimal solutions directly through the trained network models. They modeled the multi-objective TSP (MOTSP) as a Pointer Network and solved it using DRL techniques. Zhang *et al.* [25] developed a concise meta-learning-based DRL technique that first trains a fine-tuned meta-model to derive submodels for the subproblems to shorten training time. Then, the authors further used their technique to solve MOTSP and multi-objective vehicle routing problems with time windows.

### D. Motivations

In the Evolutionary Computation community, evolutionary operators (also known as variation operators) that generate new candidate solutions for selection are an important part [9]. A fixed operator can not be suitable for all problems according to the No Free Lunch theorem. Therefore, adaptive operator selection is widely recognized as an interesting and key issue for Evolutionary Computation [36]. Although adaptive operator selection has attracted some attention in the multi-objective optimization community, unfortunately, there are currently no research efforts dedicated to the design of CMOEAs. Given that a CMOP inherits the features and challenges of MOPs other than its constraints, it is valuable and promising to develop adaptive operator selection methods for solving CMOPs.

During the evolution of solving a CMOP, the whole environment is dynamic because the population is different and unknown in advance at each iteration. Therefore, adopting Reinforcement Learning techniques to solve the adaptive operator selection issue is intuitively effective. However, traditional Reinforcement Learning techniques such as Q-Learning may be less effective in dealing with such problems because the environment can include an infinite number of states, that is, the search space is continuous but Q-Table can only handle discrete state space. In contrast, DRL techniques that train a deep neural network as the policy are suitable for the continuous and infinite state space.

Another advantage of DRL techniques is that the output $q_t$ of Equation (5) contains not only the current reward $r_t$ but also the maximum expected reward after taking this action. Consequently, $q_t$ can represent the maximum cumulative reward in the future due to the recursive nature of this formula [17]. As a result, DRL techniques such as DQL are more suitable for the evolutionary process of CMOEAs since both historical and future performance should be considered in the heuristic algorithms [37]. Thus, DQL is very promising for adaptive operator selection.

Meanwhile, there are some challenging issues that must be resolved to successfully apply DRL to CMOPs. Since a CMOP contains constraints, it is necessary for a DRL model to consider constraint satisfaction and feasibility in the design of the state and reward. In addition, the effectiveness of an action (*i.e.,* operator) on both handling constraints and optimizing objectives should be evaluated in training the DRL model.

To adopt DQL in CMOEAs, we need to develop a model that determines the state, action, reward, and learning procedure. Our proposed DQL model and the DQL-assisted CMOEA framework are elaborated on in the next section.

## III. Proposed Methods

### A. Proposed DRL Model

In this work, the evolutionary operators are regarded as actions, thus the actions include

$$\mathcal{A} = \{op_1, op_2, \cdots, op_i, \cdots, op_k\}, \tag{6}$$

where $op_i$ is the $i$th evolutionary operator such as GA, DE, PSO, and CSO, and $k$ is the number of candidate operators. It should be noted that in our proposed model, any number of operators can be used as candidates.

Then, we define the state of a population by considering its performance in terms of convergence, diversity, and feasibility. The average sum of objective functions of solutions in the population is used to evaluate the convergence of the population in the objective space, *i.e.,* the approximation to the CPF. Specifically, it is formulated as

$$con = \frac{\sum_{\mathbf{x}\in\mathcal{P}}\sum_{j=1}^{m} f_j(\mathbf{x})}{N}, \tag{7}$$

where $f_j(\mathbf{x})$ is the $j$-th objective function value. When the population approximates the CPF, $con$ will become smaller. The objective functions are not normalized due to the following underlying considerations. After normalization, all objectives will belong to $[0, 1]$. Although the influence of different scales can be eliminated, the $con$ fails to represent the real distribution of the population.

The average CV value of solutions in the population is used to estimate feasibility, *i.e.,* the distribution of the population in the feasible/infeasible regions. Specifically, it is formulated as

$$fea = \frac{\sum_{\mathbf{x}\in\mathcal{P}} \phi(\mathbf{x})}{N}, \tag{8}$$

where $\phi(\mathbf{x})$ is the CV of $\mathbf{x}$. If all solutions of the population are in feasible regions, $fea$ is zero. Otherwise, it will be a large value if the population is located outside the feasible regions.

The sum of scales, in all objective dimensions that the population covers, is used to estimate the diversity of the population. Specifically, it is formulated as

$$div = \frac{1}{\sum_{j=1}^{m}(f_j^{max} - f_j^{min})}, \tag{9}$$

where $f_j^{max}$ and $f_j^{min}$ represent the maximum and minimum objective function value on the $j$th objective function. If the population is stuck in a local feasible region, $div$ is very large because $f_j^{max}$ is close to $f_j^{min}$. On the contrary, if the population is well-distributed among every objective, $div$ will be small because $f_j^{max}$ is larger than $f_j^{min}$.

Then, we use three components to form a state. The state set is

$$\mathcal{S} = \{s | s = (con, fea, div)\}. \tag{10}$$

It uses $con$, $fea$, and $div$ to reflect the current state of the population. In this case, whether an operator should be selected depends on a comprehensive evaluation of the current population state and its effectiveness in enhancing these performances.

The reward is calculated by

$$r = (con + fea + div) - (con' + fea' + div'), \tag{11}$$

where $con'$, $fea'$, and $div'$ are the new state of the population for the next generation. The reward we define in this work represents the improvement of the population state, including the performance in terms of convergence, diversity, and feasibility. Therefore, the effectiveness of a selected operator can be comprehensively evaluated. It is necessary to note that these three terms are normalized when they are input to the network as training/predicting data.

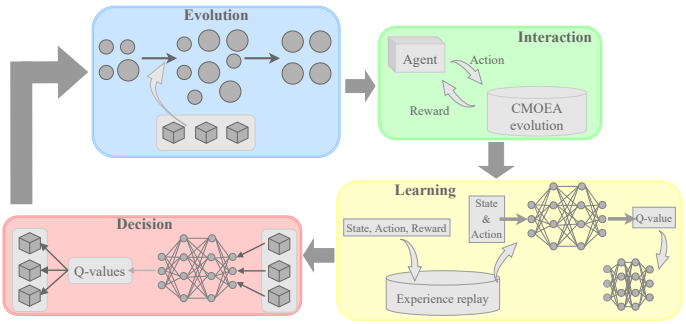


Fig. 2. The illustration of the proposed DQL model.

A record is formed as

$$\mathbf{t} = (s, op, r, s') = (con, fea, div, op, r, con', fea', div'), \tag{12}$$

and the EP is formed as

$$\mathcal{EP} = \{\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_{ms_{ep}}\}. \tag{13}$$

Such a data structure enables the EP to record the state, the selected operator, the reward, and the new state at each iteration. Thus, when training the network, the training data include all necessary items to approximate the action-value function in handling the operator selection issue.

Based on the above definitions and proposals, we develop a DQL model for operator selection for CMOEAs. The proposed model is shown in Fig. 2. This model contains four procedures, they are:

- *Evolution procedure:* In the evolution procedure, the CMOEA generates offspring by the selected operator (determined by the DQN) and determines solutions that can survive to the next generation.
- *Interaction procedure:* In the interaction procedure, the agent and the environment exchange information and interact with each other. The agent gives the environment an action (*i.e.,* operator) and the environment generates feedback (*i.e.,* reward) from the evolution of the CMOEA for the agent.
- *Learning procedure:* After the agent receives the feedback, it learns and improves the policy in the learning procedure. First, the record of the current iteration is added to the EP. Second, the DQN is trained based on the training data sampled from the EP. The input of the DQN is $State\&Action$, which is to say, the population state and the adopted operator at one iteration. The output of the DQN is the Q-value of adopting that operator at that population state.
- *Decision procedure:* After the DQN is trained, the agent can use it to decide which action to take in the face of a population state. The decision procedure first estimates all Q-values of all actions. Then the action with the largest Q-value is selected.

The above four procedures execute in order at each iteration to achieve the DQL-assisted operator selection for CMOEA.

### *B. Proposed DQL-assisted Framework*

Based on the DQL model, we develop a DQL-assisted adaptive operator selection framework for CMOEAs. We present the flowchart of the proposed DQL-assisted CMOEA framework in Fig. 3. The main steps are as those in a CMOEA (initialization, mating, and selection), except that the DQL-assisted framework contains two additional parts.

The blue part contains the **Operator Selection** process. In this process, the size of the EP needs to meet the requirement before the DQN is trained because we need enough data to train the Network. If the size of the EP is not enough, we randomly select an operator at the iteration to give equal weight to each operator so that the exploration can be guaranteed. Otherwise, if the size of the EP meets the requirement and the DQN is not built, the DQN is first built. If the DQN has been trained, it is directly used to adaptively select an operator for the following steps.

After we get an adaptively selected operator, the offspring are generated based on the mating selection of the CMOEA, and the environmental selection of the CMOEA is conducted to determine the population for the next generation.

Then, the **Network Update** process (in yellow) is conducted. It mainly contains two specific parts. At each iteration, the population state is detected and the record is calculated. Afterward, the EP is updated by adding the record of this iteration. After every 50 iterations, the QDN is updated. The above steps continue until the termination condition is met.

The Pseudocode of the proposed DQL-assisted framework for CMOEAs is presented in Algorithm 1. The inputs include the population size, the termination condition (maximum function evaluations), the maximum size of EP, and the required size of EP for training the DQN. The output is the same as the output solution set of the selected CMOEA. The main steps are consistent with the process in the flowchart. First, the initialization of the CMOEA is conducted to generate the initial population(s) for evolution (line 1). Then, the state of the initial population is determined (line 2) to prepare the first record. The EP set $\mathcal{EP}$ is initialized as empty then (line 3) and the index of the current generation $g$ is initialized as zero (line 4). In the main loop, the following steps are performed:

**Algorithm 1** The DQL-assisted Framework for CMOEAs

**Require:** $N$ (population size), $G_{max}$ (termination condition), $i$ (number of operators), $ms_{ep}$ (maximum size of EP), $rs_{ep}$ (required size of EP)
**Output:** $\mathcal{P}$ (the output solution set of the CMOEA)

1: Initialization of the CMOEA;
2: Determine the state of the population;
3: $\mathcal{EP} \leftarrow$ Initialize the EP;
4: $g \leftarrow 0$;
5: **while** $g < G_{max}$ **do**
6:  **if** $|\mathcal{EP}| < rs_{ep}$ **then**
7:  $i \leftarrow$ Randomly select an operator;
8:  $\mathcal{O} \leftarrow$ Generate offspring set by the CMOEA using the $i$th operator;
9:  $\mathcal{P} \leftarrow$ Select the population for the next generation by the CMOEA;
10:  $\mathbf{t} \leftarrow$ Determine the reward and new state, and form a new record;
11:  $\mathcal{EP} \leftarrow$ Update the EP with $\mathbf{t}$;
12:  **else**
13:  **if** Network is not built **then**
14:  $Q \leftarrow$ Build the DQN using $\mathcal{EP}$ by Algorithm 2;
15:  **else**
16:  $i \leftarrow$ Adaptively select an operator according to the state of the population and $Q$ using Algorithm 3;
17:  $\mathcal{O} \leftarrow$ Generate offspring set by the CMOEA using the $i$th operator;
18:  $\mathcal{P} \leftarrow$ Select the population for the next generation by the CMOEA;
19:  $\mathbf{t} \leftarrow$ Determine the reward and new state, and form a new record;
20:  $\mathcal{EP} \leftarrow$ Update the EP with $\mathbf{t}$;
21:  **end if**
22:  **end if**
23:  $g \leftarrow g + 1$;
24:  **if** $g\%50 = 0$ **then**
25:  $Q \leftarrow$ Update the DQN using $\mathcal{EP}$ by Algorithm 2;
26:  **end if**
27: **end while**
28: **return** $\mathcal{P}$

- If the $\mathcal{EP}$ has not meet the required size $rs_{ep}$, lines 7-11 are performed. First, an operator (denoted as $i$th operator) is randomly selected (line 7). Then, the offspring set $\mathcal{O}$ is generated based on the operator and the mating selection of the CMOEA (line 8). Afterward, the environmental

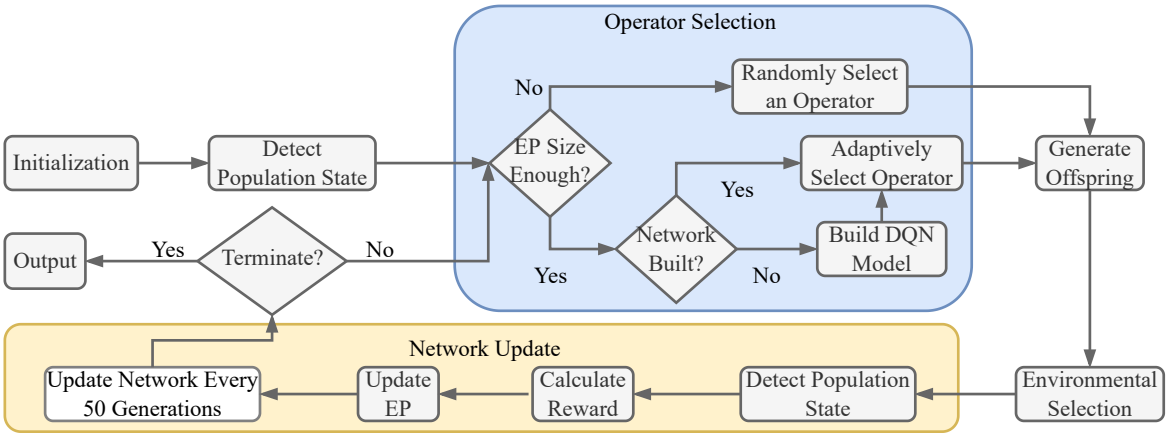


Fig. 3. The flowchart of the proposed DQL-assisted CMOEA framework.

TABLE I
DETAILED PARAMETER SETTING OF THE ADOPTED DQN ARCHITECTURE.

| Parameter | Value |
|---|---|
| Number of hidden layers | 2 |
| Number of neurons in hidden layers | 40 |
| Number of nodes in input layer | 4 |
| Number of nodes in output layer | 1 |
| Batch size | $\|\mathcal{T}\|$ |
| Maximum number of iterations | 80000 |
| Decay of learning rate | 1.00E-04 |
| Learning rate | 0.01 |
| Bias from input to hidden layer | 0.1 |
| Bias from hidden layer to output layer | 0 |
| Activation function | ReLU |

selection of the CMOEA is performed to select the population $\mathcal{P}$ for the next generation (line 9). According to this iteration, the reward and the new state are determined and used to form a new record $\mathbf{t}$ (line 10). Then $\mathcal{EP}$ is updated using the record $\mathbf{t}$ (line 11). It should be noted that $\mathcal{EP}$ is a queue that follows the *first in, first out* rule.

- Otherwise if the $\mathcal{EP}$ has met the required size $rs_{ep}$, then lines 13-20 are performed. In the beginning, the DQN $Q$ is not built, so it is initialized using data from $\mathcal{EP}$ by Algorithm 2 (line 14).
- In the later iterations where the DQN has been built, lines 16-20 are performed. First, an operator is adaptively selected according to the current state and the DQL-assisted method presented in Algorithm 3 (line 16). Then the offspring set $\mathcal{O}$ is generated by the CMOEA using the selected operator (line 17). Afterward, the population for the next generation is selected based on the environmental selection of the CMOEA (line 18). Finally, the new record is formed (line 19) and the $\mathcal{EP}$ is updated (line 20).
- Once every 50 iterations, the DQN is updated using $\mathcal{EP}$ by Algorithm 3 (line 25). This step guarantees that the DQN can approximate the recent environment.

Finally, the output solution set of the CMOEA is output as the final solution set (line 25).

### C. Train/Update the DQN

In this work, we adopt a simple Back-propagation neural network as the DQN. The detailed parameter settings of the adopted DQN architecture are listed in Table I. When the DQN is trained or updated, the procedure of Algorithm 2 is conducted.

The inputs include the experience replay $\mathcal{EP}$ and the required size of training data $s_{tr}$. The output is a DQN $Q$. First, $s_{tr}$ records are sampled from $\mathcal{EP}$ and form a set $\mathcal{T}$ as the training data (line 1). Then, the former four items (*i.e.,* the state $s_t$ and the action $a_t$) are used as the input of the DQN (lines 2-3), while the fifth item (*i.e.,* the reward $r_t$) is used as the output of the DQN (line 4). The last three items are used to train DQN as the $s_{t+1}$ in Equation (5) (line 5). An important step is that we need to normalize the inputs and output to eliminate the influence of different scales so that the DQN can be accurately estimated (line 6). Then, the normalized training data is used to train the DQN using Equation (4) as the loss function (line 7).

**Algorithm 2** Train/Update Network

**Require:** $\mathcal{EP}$ (experience replay), $s_{tr}$ (required size of training data)
**Output:** $Q$ (DQN)
1: $\mathcal{T} \leftarrow$ Randomly sample $s_{tr}$ records as the training data;
2: $s_t \leftarrow \{t_1, t_2, t_3\}, \mathbf{t} \in \mathcal{T}$;
3: $a_t \leftarrow t_4, \mathbf{t} \in \mathcal{T}$;
4: $r_t \leftarrow t_5, \mathbf{t} \in \mathcal{T}$;
5: $s_{t+1} \leftarrow \{t_6, t_7, t_8\}, \mathbf{t} \in \mathcal{T}$;
6: Normalize all items of $\mathcal{T}$;
7: $Q \leftarrow$ Train the DQN using Equation (4) as loss function;
8: **return** $Q$

**Algorithm 3** Select Operator

**Require:** $\varepsilon$ (possibility of greedy), $s$ (current population state), $\mathcal{A}$ (operator set), $Q$ (DQN)
**Output:** $a$ (the selected operator)
1: $k \leftarrow$ Generate a random number in $[0, 1]$;
2: **if** $k \leq \varepsilon$ **then**
3: $s \leftarrow$ Normalize all items of the state data $s$;
4: $i = argmax_{a \in \mathcal{A}} Q(s, a)$;
5: **else**
6: $i \leftarrow$ Generate a random number in $\{1, 2, \cdots, k\}$;
7: **end if**
8: **return** the $i$th operator $a$ in $\mathcal{A}$

### D. Proposed OS Method

The OS method in this work is similar to the common method of selecting the action in Reinforcement Learning. The detailed Pseudocode is presented in Algorithm 3. The inputs include the possibility of a greedy $\varepsilon$ to control the possibility of selecting the operator by DQN or in a random method. They also include the current population state $s$, the operator set $\mathcal{A}$, and the DQN $Q$. The output is the selected operator $a$. First, a random number $k$ between $[0, 1]$ is generated (line 1). If $k$ is less than the possibility of greedy, the DQN is used to select an operator (lines 3-4). When DQN is used, all operators are tested through the DQN, and the one with the maximum reward is adopted. If $k$ is greater than $\varepsilon$, the index of the selected operator is randomly generated (line 6) to guarantee exploration. In this work, we use GA and DE operators (*i.e.,* their crossover and mutation strategies) as the candidate actions to instantiate the framework. The reasons for selecting these two operators are two-fold:

- First, GA and DE are the two most commonly used operators in existing CMOEAs;
- Second, GA can handle multimodality well and has a strong capability for convergence, while DE can handle linkages well and is good at exploring the decision space [9].

It is worth noting that before the $\mathcal{EP}$ meets the required size (*i.e.,* line 6 in Algorithm 1), the operator is randomly selected to enhance exploration. Also, parameter $\varepsilon$ enables some random selection to enhance the exploration.

### *E. Computational Complexity*

According to Algorithms 1 to Algorithm 3, the computational complexity of our methods is determined by three components, *i.e.,* the selected CMOEA, the training of DQN, and the calculations of population state. The complexity of training the DQN (a Backpropagation neural network) is $O(4u^2)$ if we use $u$ to represent the number of neurons. The time consumption of calculating the state is $O(mnN)$. The time complexity of a CMOEA is usually $O(mN^2)$ to $O(N^3)$ (*e.g.,* CCMO [4]). Therefore, the overall computational complexity is determined by the selected CMOEA.

### *F. Remarks*

The differences between our proposed operator selection method and existing adaptive operator selection methods are two-fold:

- Our method can apply to CMOPs because the design of the DQL model and the algorithmic framework both consider constraints and the feasibility of solutions, but existing methods are not applicable to CMOPs.
- Our method is based on DQL, which can evaluate the improvement of the population brought by the selected operator in the future, while existing methods can only evaluate the improvement according to historical evolution.

Although Tian *et al.* [17] also proposed a DQL-assisted operator selection method, our methods are different from theirs in three aspects:

- First, this work handles CMOPs, while the topic of [17] is unconstrained MOPs. Handling CMOPs is more difficult than MOPs due to the existence of constraints.
- Second, our proposed DQL model is different from the model in [17]. In [17], the decision variable is regarded as the state, and the improvement of a single solution (offspring *vs* parent solution) is regarded as the reward.
- Third, the method in [17] is a specific algorithm for MOPs, while our proposed framework can be embedded into any CMOEAs.

The differences between our methodology and existing applications of DRL in solving MOPs are as follows:

- First, this work proposes a universal adaptive operator selection framework that can be used in any algorithm for CMOPs. By contrast, most existing applications focus on using DRL to solve a specific real-world application problem.
- Second, most existing applications of DRL in solving MOPs can only solve discrete MOPs, while our method can solve any form of CMOP as long as the embedded operators and the adopted CMOEA can handle discrete or continuous decision variables.

## IV. Experimental Studies

This section presents the experimental studies. Section IV-A presents the detailed experimental settings. Section IV-B compares the DQL-assisted operator selection with the original CMOEA and random operator selection. Section IV-C gives the comparison studies between our method and state-of-the-art CMOEAs. Section IV-D shows the parameter analyses of the DRL-assisted operator selection technique and the adopted neural network. Section IV-E studies the effectiveness of using two performance indicators, instead of the proposed simplistic assessments in Equations (7) and (9), to assess the population state.

### *A. Experimental Settings*

*1) Benchmark Problems:* In the experimental studies, we selected four challenging CMOP benchmark test suites to test our methods. The benchmarks include CF [38], DAS-CMOP [39], DOC [40], and LIR-CMOP [41]. The main difficulties and challenges of these benchmarks are summarized in Table S-II in the Supplementary file.

*2) Algorithms in Comparison:* In this work, we embed our proposed DQL-assisted operator selection framework into four existing CMOEAs; they are CCMO [4], MOEA/D-DAE [27], EMCMO [7], and PPS [26]. In the comparison studies, we selected nine state-of-the-art CMOEAs for comparison, including methods using different operators. They are c-DPEA [42], ToP [40], CMOEA-MS [5], BiCo [43], MFOSPEA2 [6], ShiP-A [44], DSPCMDE [45], NSGA-II-ToR [8], and CCEA [46].

*3) Parameter Settings and Genetic Operators:* For algorithms that use GA as the operator, the simulated binary crossover (SBX) and polynomial mutation (PM) [47] were used with the following parameter settings:

- Crossover probability was $p_c = 1$; distribution index was $\eta_c = 20$;
- Mutation probability was $p_m = 1/n$; distribution index was $\eta_m = 20$.

For algorithms that use DE as the operator, the parameters $CR$ and $F$ in the DE operator were set to 1 and 0.5, respectively.

The evolutionary settings and parameters of our methods are presented in Table S-III in the Supplementary file. The explanations for the settings are also presented in Section IV in the Supplementary file. All other parameter settings of the comparison methods were the same as in their original literature (*i.e.,* the default settings in PlatEMO).

*4) Performance Indicators:* Inverted generational distance based on modified distance calculation (IGD+) [48] and hypervolume (HV) [48] were adopted as indicators to evaluate the performance of different algorithms. We used two Pareto-compliant indicators to achieve a sound and fair comparison [49]. Detailed information on these three indicators could be found in the Supplementary file. The value $NaN$ means an algorithm cannot find a feasible solution or the final solution set is far from the CPF.

*5) Statistical Analysis:* Each algorithm is executed 30 independent runs on each test instance. The mean and standard deviation values of IGD+ and HV were recorded. The Wilcoxon rank-sum test with a significance level of $0.05$ was used to perform the statistical analysis using the KEEL software [50]. "$+$", "$-$", and "$=$" were used to show that the result of other algorithms was significantly better than, significantly worse than, and statistically similar to those obtained by our methods

TABLE II
AVERAGE RANKINGS AND THE P-VALUES BY THE FRIEDMAN TEST ON THE EFFECTIVENESS OF DQL-ASSISTED OS.

| | HV ranking | p-value | IGD+ ranking | p-value |
|---|---|---|---|---|
| CCMO | 2.5476 | **0.000001** | 2.5714 | **0.000001** |
| RandOS-CCMO | 1.9762 | **0.021947** | 1.9524 | **0.029096** |
| DRLOS-CCMO | 1.4762 | | 1.4762 | |
| | HV ranking | p-value | IGD+ ranking | p-value |
| EMCMO | 2.5000 | **0.000000** | 2.5238 | **0.000000** |
| RandOS-EMCMO | 2.1071 | **0.001063** | 2.1429 | **0.000208** |
| DRLOS-EMCMO | 1.3929 | | 1.3333 | |
| | HV ranking | p-value | IGD+ ranking | p-value |
| MOEA/D-DAE | 2.3214 | **0.001873** | 2.2857 | **0.003220** |
| RandOS-MOEA/D-DAE | 2.0357 | 0.071814 | 2.0714 | **0.049535** |
| DRLOS-MOEA/D-DAE | 1.6429 | | 1.6429 | |
| | HV ranking | p-value | IGD+ ranking | p-value |
| PPS | 1.9881 | **0.010346** | 2.0476 | **0.012090** |
| RandOS-PPS | 2.2857 | 0.230062 | 2.2500 | 0.113631 |
| DRLOS-PPS | 1.7262 | | 1.7024 | |

to the Wilcoxon test, respectively. All $NaN$ values of HV and IGD+ are replaced by zero and 100, respectively.

### B. On the Effectiveness of DQL-assisted OS

In the first part of the experiments, we compare the CMOEAs embedded using the DQL-assisted operator selection method with the original CMOEAs using a fixed operator and the CMOEAs using random operator selection to verify the effectiveness of the DQL-assisted operator selection method. The results in terms of HV and IGD+ on the four benchmarks are presented in Tables S-V to S-XII in the Supplementary file.

For the CFs, it can be found that DRLOS-CCMO and DRLOS-EMCMO significantly outperformed the corresponding methods using a fixed operator or random operator. However, CCMO and EMCMO perform better or at least competitively on the three-objective CF8-10, revealing that for these large objective space instances, the operators of GA can enhance convergence. As for MOEA/D-DAE and PPS, it can be determined that using random and adaptively selected operators outperformed using a fixed operator. For DAS-CMOPs, the superiority of CCMO and EMCMO on DAS-CMOP4-8 reveals that these instances prefer the GA operator. On the contrary, DAS-CMOP1-3 and DAS-CMOP9 prefer the DE operator. However, an adaptive operator selection method can better handle DAS-CMOP1-3 and DAS-CMOP9 than PPS using the DE operator, revealing that the DQL-assisted operator selection can learn to decide which operator to use according to the population state during evolution. For DOCs, the results are similar to those of CFs. DRLOS-CCMO and DRLOS-EMCMO performed significantly better than other methods, revealing that DOC prefers the DE operator and the adaptive selection method can accurately determine DE as the operator. Since PPS uses DE as the operator, the performance is not significantly influenced by operator selection on DOCs. The results among RandOS and DRLOS also reveal that using DRL to adaptively select operators during evolution performs better than random selection. For LIR-CMOPs, it is apparent that the proposed OS method can significantly improve the performances of all CMOEAs. Nevertheless, the results show that LIR-CMOP13-14 prefer GA as operator.

To better understand the performance, we conduct the Friedman test with Holm correction at a significance level of 0.05 on all results. The average rankings and p-values are summarized in Table II. It can be found that the DQL-assisted operator selection method outperforms the fixed operator and random operator selection. For these four CMOEAs, using our proposed DQL-assisted operator selection method can improve their performance.

In summary, our proposed DQL-assisted OS method can improve the performance of these CMOEAs. The DQL-assisted operator selection method can better determine the operator based on the population state compared to using a fixed or randomly selected operator.

### C. Comparison Studies

After the verification of the proposed DQL-assisted OS method, we further compare the DRLOS-EMCMO algorithm with nine state-of-the-art CMOEAs to study the algorithmic performance of these CMOEAs. The statistical results on four benchmarks are presented in Tables III to IV in this file and Tables S-XIII to S-XVIII in the Supplementary file.

In general, DRLOS-EMCMO outperforms other CMOEAs on these challenging CMOPs. But it performed worse in some CMOPs. Among the selected CMOPs, some instances rely on a particular operator. For example, the results show that DAS-CMOP4-8 and LIR-CMOP13-14 need the GA operator because of their multimodal feature. Besides, some instances of the CF test suite need the DE operator due to the linkage between variables. However, since the computational resources are used for all operators in DRLOS-EMCMO in the learning stage, DRLOS-EMCMO has fewer resources for the specifically needed operator in dealing with these problems, and thus, it performs worse. Additionally, it can be found that LIR-CMOP1-4, the CMOPs with an extremely small feasible region, require some constraint relaxation techniques such as penalty function in c-DPEA [42] and ShiP-A [3], or $\varepsilon$-constrained in DSPCMDE [45] and CCEA [46]. However, EMCMO does not contain a constraint relaxation technique, and thus, it performs worse on LIR-CMOP1-4. Except for these specific problems, DRLOS-EMCMO generally performs better than other CMOEAs.

We depict the final solutions sets obtained by DRLOS-EMCMO and other CMOEAs on CF1, DAS-CMOP9, DOC2, and LIR-CMOP8 in Figs S-1 to S-3 in the Supplementary file and Fig 4 in this file. The results clearly show that DRLOS-EMCMO can approximate the CPF and obtain an even distribution in every instance. For DOC2, it can be found that only DRLOS-EMCMO can converge to two segments of the CPF, revealing that using GA performs worse than using an adaptively selected operator. As for LIR-CMOP8, it can be found DSPCMDE, adopting DE as the operator, also converges to the CPF. However, some dominant and extreme solutions remain, revealing that when GA can be adaptively selected, convergence can be further enhanced.

TABLE III
STATISTICAL RESULTS OF IGD+ OBTAINED BY DRLOS-EMCMO AND OTHER METHODS ON DOC BENCHMARK PROBLEMS. THE BEST RESULT IN EACH ROW IS HIGHLIGHTED.

| Problem | cDPEA | ToP | CMOEA_MS | BiCo | MFOSPEA2 | ShiP_A | DSPCMDE | NSGAIIToR | CCEA | DRLOS-EMCMO |
|---|---|---|---|---|---|---|---|---|---|---|
| DOC1 | 5.5922e-1 (5.87e-1) − | 3.3066e-3 (1.48e-4) − | 3.9980e+0 (3.22e+0) − | 2.5299e-2 (4.21e-2) − | 1.2071e-1 (2.31e-1) − | 2.5551e+0 (2.06e+0) − | 3.9738e+2 (4.57e+2) − | 6.3475e+1 (2.25e+1) − | 1.6477e+0 (1.59e+0) − | 2.6377e-3 (1.82e-4) |
| DOC2 | NaN (NaN) | NaN (NaN) | NaN (NaN) | NaN (NaN) | NaN (NaN) | NaN (NaN) | 2.4608e-1 (7.76e-2) − | NaN (NaN) | NaN (NaN) | 4.3113e-2 (7.17e-2) |
| DOC3 | 7.4298e+2 (2.24e+2) ≈ | 1.8914e+2 (1.74e+2) + | 5.9087e+2 (2.60e+2) ≈ | 4.9838e+2 (2.67e+2) ≈ | 7.3066e+2 (2.74e+2) ≈ | NaN (NaN) | 1.1650e+2 (1.66e+2) + | NaN (NaN) | 5.8119e+2 (1.29e+2) ≈ | 6.0270e+2 (4.85e+2) |
| DOC4 | 6.0555e-1 (3.81e-1) − | 1.0471e-1 (7.36e-2) ≈ | 7.7869e-1 (5.33e-1) − | 2.5349e-1 (1.82e-1) − | 3.9220e-1 (3.13e-1) − | 8.6957e-1 (8.10e-1) − | 4.4231e-2 (8.64e-2) − | 1.6298e+1 (6.10e+0) − | 8.7862e-1 (5.61e-1) − | 4.1216e-2 (4.26e-2) |
| DOC5 | NaN (NaN) | 3.8442e+1 (6.64e+1) − | 9.5192e+1 (2.16e+1) − | NaN (NaN) | NaN (NaN) | NaN (NaN) | NaN (NaN) | NaN (NaN) | NaN (NaN) | 2.1808e+1 (5.00e+1) |
| DOC6 | 3.5707e+0 (2.89e+0) − | 5.9406e+0 (2.48e+0) − | 2.4328e+0 (2.55e+0) − | 9.1093e-1 (8.16e-1) − | 9.6498e-1 (1.13e+0) − | 1.8291e+0 (1.77e+0) − | 2.7957e-3 (1.78e-4) ≈ | 2.3044e+1 (6.45e+0) − | 2.6678e+0 (2.51e+0) − | 9.4061e-3 (3.24e-2) |
| DOC7 | 7.3217e+0 (2.11e+0) − | 1.3820e+0 (7.42e-1) − | 3.8020e+0 (1.32e+0) − | 5.2266e+0 (2.01e+0) − | 5.3131e+0 (2.15e+0) − | NaN (NaN) | 2.7176e-1 (8.13e-1) − | NaN (NaN) | 5.5487e+0 (2.08e+0) − | 1.4072e-1 (2.67e-1) |
| DOC8 | 7.1094e+1 (4.52e+1) − | 5.9654e+1 (2.73e+1) − | 1.6705e+2 (7.52e+1) − | 6.4607e+1 (5.86e+1) − | 7.8331e+1 (6.07e+1) − | 8.9302e+1 (5.55e+1) − | 2.1322e-1 (5.80e-2) + | 3.7281e+2 (8.39e+1) − | 8.9903e+1 (5.45e+1) − | 2.7264e-1 (8.88e-2) |
| DOC9 | 1.5307e-1 (1.16e-1) ≈ | 2.2487e-1 (7.38e-2) − | 7.9311e-2 (1.06e-1) − | 1.4714e-1 (1.01e-1) − | 1.1725e-1 (9.59e-2) ≈ | 1.3416e-1 (1.12e-1) − | 8.8190e-2 (1.05e-2) − | 6.3222e-1 (1.41e-1) − | 1.5036e-1 (1.24e-1) ≈ | 5.0283e-2 (9.22e-3) |
| +/ − / ≈ | 0/5/2 | 1/6/1 | 0/7/1 | 0/6/1 | 0/5/2 | 0/5/0 | 2/5/1 | 0/5/0 | 0/5/2 | |

TABLE IV
STATISTICAL RESULTS OF HV OBTAINED BY DRLOS-EMCMO AND OTHER METHODS ON LIR-CMOP BENCHMARK PROBLEMS. THE BEST RESULT IN EACH ROW IS HIGHLIGHTED.

| Problem | cDPEA | ToP | CMOEA_MS | BiCo | MFOSPEA2 | ShiP_A | DSPCMDE | NSGAIIToR | CCEA | DRLOS-EMCMO |
|---|---|---|---|---|---|---|---|---|---|---|
| LIRCMOP1 | 1.6448e-1 (1.08e-2) + | 1.0538e-1 (8.84e-3) − | 1.0477e-1 (1.34e-2) − | 1.3536e-1 (6.52e-3) ≈ | 1.3108e-1 (1.60e-2) − | 2.2967e-1 (2.04e-3) + | 1.9200e-1 (2.37e-2) + | 9.6272e-2 (5.01e-3) − | 1.6415e-1 (1.47e-2) + | 1.4210e-1 (1.90e-2) |
| LIRCMOP2 | 2.8144e-1 (1.58e-2) + | 2.1555e-1 (1.42e-2) − | 2.2253e-1 (2.32e-2) − | 2.5522e-1 (1.01e-2) ≈ | 2.5272e-1 (1.53e-2) ≈ | 3.5009e-1 (2.21e-3) + | 3.2355e-1 (1.62e-2) + | 2.0338e-1 (6.34e-3) − | 2.9124e-1 (1.40e-2) + | 2.6388e-1 (2.75e-2) |
| LIRCMOP3 | 1.4625e-1 (1.43e-2) + | 9.2802e-2 (5.34e-3) − | 9.9956e-2 (1.62e-2) − | 1.2583e-1 (6.96e-3) + | 1.1792e-1 (1.17e-2) ≈ | 1.9042e-1 (6.25e-3) + | 1.6138e-1 (2.11e-2) + | 8.7510e-2 (4.22e-3) − | 1.5215e-1 (1.21e-2) + | 1.1822e-1 (1.61e-2) |
| LIRCMOP4 | 2.4589e-1 (1.43e-2) + | 1.8034e-1 (9.33e-3) − | 1.8730e-1 (1.51e-2) − | 2.1935e-1 (1.32e-2) ≈ | 2.1831e-1 (1.16e-2) ≈ | 2.8717e-1 (1.10e-2) + | 2.6399e-1 (2.49e-2) + | 1.7739e-1 (6.98e-3) − | 2.5082e-1 (1.34e-2) + | 2.1807e-1 (1.98e-2) |
| LIRCMOP5 | 1.5599e-1 (2.41e-2) − | 0.0000e+0 (0.00e+0) − | 1.0881e-1 (6.93e-2) − | 0.0000e+0 (0.00e+0) − | 1.4741e-1 (2.38e-2) − | 1.4847e-1 (1.58e-2) − | 2.6549e-1 (3.49e-2) − | 0.0000e+0 (0.00e+0) − | 8.9994e-2 (7.55e-2) − | 2.8301e-1 (1.38e-2) |
| LIRCMOP6 | 1.0447e-1 (1.26e-2) − | 3.8214e-3 (1.45e-2) − | 6.4431e-2 (5.03e-2) − | 0.0000e+0 (0.00e+0) − | 1.0871e-1 (1.33e-2) − | 9.8962e-2 (9.44e-3) − | 1.6168e-1 (4.32e-2) − | 0.0000e+0 (0.00e+0) − | 5.1631e-2 (4.62e-2) − | 1.9379e-1 (1.07e-3) |
| LIRCMOP7 | 2.5030e-1 (7.64e-3) − | 1.4326e-2 (5.45e-2) − | 2.4401e-1 (7.28e-3) − | 1.6982e-1 (1.13e-1) − | 2.5122e-1 (8.92e-3) − | 2.4627e-1 (9.69e-3) − | 2.8984e-1 (1.05e-2) − | 0.0000e+0 (0.00e+0) − | 2.4328e-1 (7.28e-3) − | 2.9240e-1 (4.63e-3) |
| LIRCMOP8 | 2.3549e-1 (9.02e-3) − | 1.3858e-2 (5.27e-2) − | 2.2602e-1 (9.02e-3) − | 5.9252e-2 (1.00e-1) − | 2.3840e-1 (1.01e-2) − | 2.3424e-1 (1.20e-2) − | 2.9288e-1 (4.64e-4) − | 0.0000e+0 (0.00e+0) − | 2.2442e-1 (4.88e-3) − | 2.9365e-1 (9.30e-4) |
| LIRCMOP9 | 3.7822e-1 (6.59e-2) − | 2.9975e-1 (7.81e-2) − | 2.3427e-1 (6.54e-2) − | 1.2413e-1 (4.59e-2) − | 3.2135e-1 (6.48e-2) − | 3.4557e-1 (5.83e-2) − | 3.4121e-1 (2.58e-2) − | 3.6661e-2 (1.23e-2) − | 2.2556e-1 (5.96e-2) − | 4.4138e-1 (2.17e-2) |
| LIRCMOP10 | 5.4129e-1 (5.30e-2) − | 4.5018e-1 (1.02e-1) − | 3.6909e-1 (1.67e-1) − | 6.4856e-2 (3.07e-2) − | 5.0681e-1 (9.87e-2) − | 3.5948e-1 (1.05e-1) − | 5.9874e-1 (2.98e-2) − | 5.3776e-2 (2.94e-2) − | 1.2244e-1 (9.89e-2) − | 6.8631e-1 (1.23e-2) |
| LIRCMOP11 | 6.3446e-1 (3.25e-2) − | 3.8783e-1 (8.40e-2) − | 3.9226e-1 (1.19e-1) − | 2.2469e-1 (8.90e-2) − | 6.2055e-1 (3.97e-2) − | 4.5297e-1 (1.03e-1) − | 6.2135e-1 (5.89e-2) − | 6.0264e-2 (2.87e-2) − | 3.3593e-1 (1.63e-1) − | 6.8144e-1 (9.55e-3) |
| LIRCMOP12 | 5.2337e-1 (4.27e-2) − | 4.7168e-1 (4.51e-2) − | 4.2431e-1 (7.04e-2) − | 3.4807e-1 (1.07e-1) − | 5.1668e-1 (3.92e-2) − | 4.9940e-1 (3.07e-2) − | 5.1211e-1 (5.02e-2) − | 7.5184e-2 (2.28e-2) − | 4.1243e-1 (5.49e-2) − | 5.7470e-1 (1.69e-2) |
| LIRCMOP13 | 5.5469e-1 (1.72e-3) + | 2.3424e-3 (1.26e-2) − | 5.4982e-1 (3.00e-2) + | 1.1278e-4 (1.46e-4) − | 1.1147e-4 (1.02e-4) − | 5.3448e-1 (3.24e-3) ≈ | 5.0437e-1 (3.10e-3) − | 0.0000e+0 (0.00e+0) − | 5.5860e-1 (9.25e-4) + | 5.3441e-1 (3.20e-3) |
| LIRCMOP14 | 5.5445e-1 (1.37e-3) + | 2.4647e-3 (1.23e-2) − | 5.5565e-1 (1.14e-3) + | 3.9193e-4 (3.00e-4) − | 4.6467e-4 (2.92e-4) − | 5.4340e-1 (2.85e-3) − | 5.3045e-1 (3.44e-3) − | 5.3460e-5 (1.79e-4) − | 5.5810e-1 (7.14e-4) + | 5.4856e-1 (1.82e-3) |
| +/ − / ≈ | 6/8/0 | 0/14/0 | 2/12/0 | 1/10/3 | 0/11/3 | 4/9/1 | 4/10/0 | 0/14/0 | 6/8/0 | |

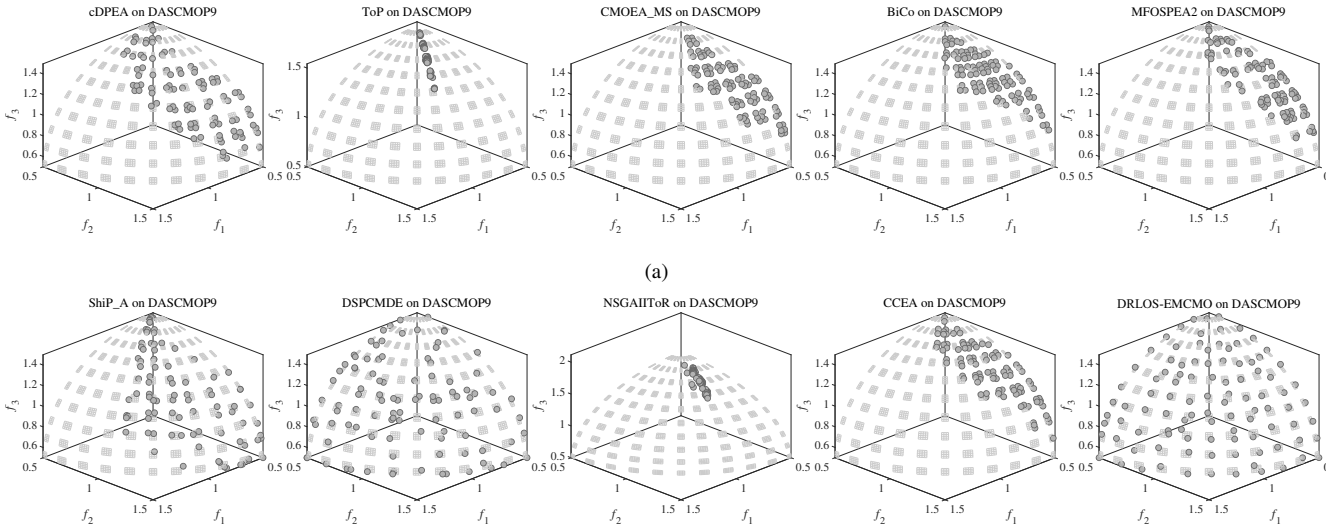


Fig. 4. The final solution sets obtained by DRLOS-EMCMO and other methods on DAS-CMOP9 with the median IGD+ value among 30 runs.

TABLE V
AVERAGE RANKINGS AND THE P-VALUES BY THE FRIEDMAN TEST ON THE COMPARISON STUDIES.

| | HV ranking | p-value | IGD+ ranking | p-value |
|---|---|---|---|---|
| c-DPEA | 4.2738 | **0.027929** | 4.2381 | **0.020103** |
| ToP | 7.1548 | **0.000000** | 7.2024 | **0.000000** |
| CMOEA-MS | 6.4286 | **0.000000** | 6.4762 | **0.000000** |
| BiCo | 6.6071 | **0.000000** | 6.3690 | **0.000000** |
| MFOSPEA2 | 4.8929 | **0.001717** | 5.0357 | **0.000413** |
| ShiP-A | 4.5833 | **0.007658** | 4.4881 | **0.006876** |
| DSPCMDE | 3.8690 | **0.040718** | 3.9981 | **0.034346** |
| NSGA-II-ToR | 9.3333 | **0.000000** | 9.5238 | **0.000000** |
| CCEA | 5.5357 | **0.000040** | 5.4762 | **0.000027** |
| DRLOS-EMCMO | 2.8214 | | 2.7024 | |

Similarly for DAS-CMOP9, DSPCMDE finds fewer segments of the CPF compared to DRLOS-EMCMO, demonstrating that when GA can be used during evolution, the population can more effectively converge to the CPF and is less likely to get trapped in local optima.

Additionally, we depict the convergence profiles of DRLOS-EMCMO and other CMOEAs on CF4, DAS-CMOP1, DOC7, and LIR-CMOP6 in Fig 5. It can be determined that DRLOS-EMCMO can not only achieve faster convergence speed but also obtain a better final indicator value.

In summary, DRLOS-EMCMO outperforms these CMOEAs using a fixed operator, revealing that the DQL-assisted operator selection can achieve better versatility on different CMOPs.

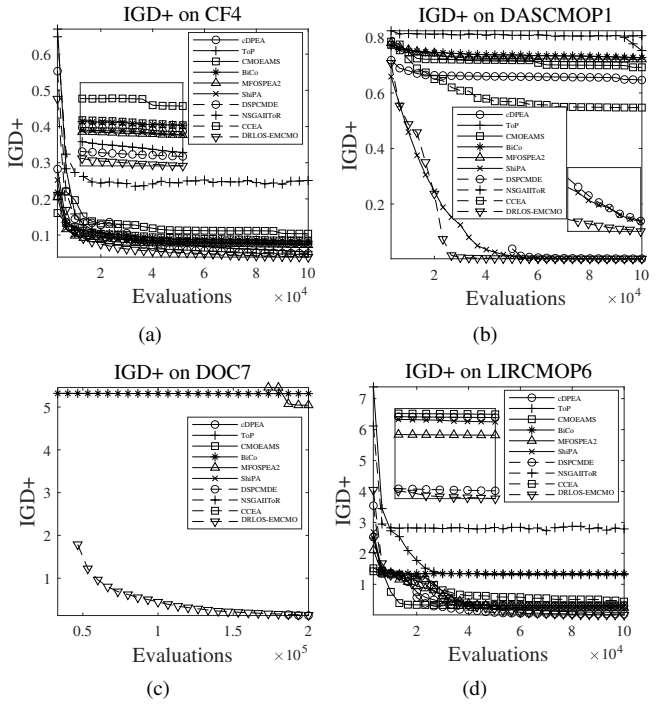


Fig. 5. The convergence profiles on IGD+ of DRLOS-EMCMO and other methods on CF4, DAS-CMOP1, DOC7, and LIR-CMOP6 with the median IGD+ values among 30 runs.

### *D. Parameter Analyses*

In the first part of parameter analyses, we change the required size of EP and the batch size to test its influence on training the DQL. Also, we change the greedy threshold to test the robustness of our method. The statistical results are presented in Tables S-XIX to S-L in the Supplementary file. It can be found that these two parameters have little influence on the performance in terms of all benchmark problems, demonstrating that our method is not parameter sensitive. Therefore current parameter settings are applicable when applying the DQL-assisted operator selection method to a new CMOEA.

Then in the second part of parameter analyses, we conduct different settings of important parameters in the DQN, including the decay of learning rate, learning rate, number of iterations in training, and number of nodes in each hidden layer, to test the influence of these parameter settings. Detailed explanations of variants are reported in Table S-IV in the Supplementary file. In this part, only DRLOS-EMCMO is adopted to save space. Statistical results of HV and IGD+ are reported in Tables S-LI to S-LVIII in the Supplementary file. On all benchmark problems, different parameter settings have little or no significant influence on DRLOS-EMCMO, revealing that the DQN adopted in this work is not parameter sensitive.

### *E. Ablation Studies on Assessing Population State Using Indicators*

In this part, we adopt two indicators, HV [48] and Spacing [51], to estimate the population state, respectively. These two indicators are selected because they do not need prior knowledge of the true PF. Compared to the items used in Equations (7) and (9), HV and Spacing are two more sophisticated indicators that can provide a more concise estimation of the population state. The HV indicator can evaluate convergence and diversity, while the Spacing indicator can evaluate diversity.

The results are reported in Tables S-LIX and S-LX in the Supplementary file, according to which the following conclusions can be found.

- For CCMO and EMCMO, using simplistic evaluations as in Equations (7) and (9) performs better when dealing with DSA-CMOP1-3 and DOCs, revealing that a sophisticated evaluation can easily lead to local optima because these instances need not only operators of GA but also other operators. In contrast, using HV and Spacing indicators performs better on DAS-CMOP4-9 which prefers GA. The indicators provide a more concise estimation of the dynamics of the population, allowing the algorithm to determine how to select operators of GA.
- For MOEA/D-DAE, the choice of simplistic or sophisticated evaluations has no significant influence on the performance.
- For PPS, using sophisticated evaluations performs significantly worse on instances with no preference for operators of GA and performs better on some instances with a preference for operators of GA, which further demonstrates that a simplistic evaluation has better versatility.

## V. Conclusions and Future Work

In this article, we propose a DQL-assisted online operator selection method for CMOPs, filling the research gap in operator selection in CMOPs and introducing DRL techniques to CMOPs. We develop a DQL model that uses population convergence, diversity, and feasibility as the state, the operators as actions, and the improvement of the population state as the reward. Based on this DQL model, we develop a versatile and easy-to-use DQL-assisted operator selection framework that can contain any number of operators and be embedded into any CMOEA. We embedded the proposed method into four existing CMOEAs. Experimental studies have demonstrated that the proposed adaptive operator selection method is effective, and the resulting algorithm outperformed nine state-of-the-art CMOEAs.

Nevertheless, some issues must be addressed regarding the current study of this paper. According to the experimental results, DRL-assisted CMOEAs perform worse when a CMOP prefers specific operators. Therefore, it is necessary to improve learning efficiency so that fewer computational resources can train a more accurate DRL model to determine the desired operators. Moreover, increasing the maximum number of function evaluations in the experiments may improve the numerical performance of DRL-assisted CMOEAs, which is also true when the proposed method is applied to real-world problems.

In the future, the following directions are worth trying:

- Some other operators can be embedded to extend this framework to solve other kinds of MOPs. For example,

the CSO [13] or other enhanced operators [52] can be embedded to solve large-scale CMOPs [53]. In addition, some advanced learning-based optimizers such as switching particle swarm optimizers [54], [55] can be used as actions to enhance performance.

- Advanced neural networks [56] can be employed as the DQN to see if they can enhance the performance of the proposed DQL-assisted operator selection method.
- Moreover, hyperparameter adaptation [57] is necessary for future study since there are many hyperparameters that need to be adjusted.

The codes of the methods in this work can be obtained from the authors upon request.

## REFERENCES


[1] A. Kumar, G. Wu, M. Z. Ali, Q. Luo, R. Mallipeddi, P. N. Suganthan, and S. Das, "A benchmark-suite of real-world constrained multi-objective optimization problems and some baseline results," *Swarm and Evolutionary Computation*, vol. 67, p. 100961, 2021.

[2] B. Tan, H. Ma, Y. Mei, and M. Zhang, "Evolutionary multi-objective optimization for web service location allocation problem," *IEEE Transactions on Services Computing*, vol. 14, no. 2, pp. 458–471, 2021.

[3] Z. Ma and Y. Wang, "Shift-based penalty for evolutionary constrained multiobjective optimization and its application," *IEEE Transactions on Cybernetics*, pp. 1–13, 2021, doi:10.1109/TCYB.2021.3069814.

[4] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 102–116, 2021.

[5] Y. Tian, Y. Zhang, Y. Su, X. Zhang, K. C. Tan, and Y. Jin, "Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9559–9572, 2022.

[6] R. Jiao, B. Xue, and M. Zhang, "A multiform optimization framework for constrained multiobjective optimization," *IEEE Transactions on Cybernetics*, pp. 1–13, 2022, doi:10.1109/TCYB.2022.3178132.

[7] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, and C. Yue, "An evolutionary multitasking optimization framework for constrained multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 2, pp. 263–277, 2022.

[8] Z. Ma, Y. Wang, and W. Song, "A new fitness function with two rankings for evolutionary constrained multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 8, pp. 5005–5016, 2021.

[9] Y. Tian, X. Zhang, C. He, K. Tan, and Y. Jin, "Principled design of translation, scale, and rotation invariant variation operators for metaheuristics," *Chinese Journal of Electronics*, 07 2022.

[10] J. Holland, "Adaptation in natural and artificial systems," *An Introductory Analysis with Application to Biology, Control and Artificial Intelligence*, 01 1994.

[11] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 01 1997.

[12] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[13] X. Zhang, X. Zheng, R. Cheng, J. Qiu, and Y. Jin, "A competitive mechanism based multi-objective particle swarm optimizer with fast convergence," *Information Sciences*, vol. 427, pp. 63–76, 2018.

[14] L. Hu, Y. Yang, Z. Tang, Y. He, and X. Luo, "Fcan-mopso: An improved fuzzy-based graph clustering algorithm for complex networks with multi-objective particle swarm optimization," *IEEE Transactions on Fuzzy Systems*, pp. 1–16, 2023.

[15] L. Hu, K. C. C. Chan, X. Yuan, and S. Xiong, "A variational bayesian framework for cluster analysis in a complex network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 11, pp. 2115–2128, 2020.

[16] C. Wang, R. Xu, and X. Zhang, "An evolutionary algorithm based on multi-operator ensemble for multi-objective optimization," in *Intelligent Computing Theories and Application*, D.-S. Huang, V. Bevilacqua, and P. Premaratne, Eds. Cham: Springer International Publishing, 2019, pp. 14–24.

[17] Y. Tian, X. Li, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–14, 2022, doi:10.1109/TETCI.2022.3146882.

[18] L. Dong, Q. Lin, Y. Zhou, and J. Jiang, "Adaptive operator selection with test-and-apply structure for decomposition-based multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 68, p. 101013, 2022.

[19] S. Schneider, R. Khalili, A. Manzoor, H. Qarawlus, R. Schellenberg, H. Karl, and A. Hecker, "Self-learning multi-objective service coordination using deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3829–3842, 2021.

[20] L. Caviglione, M. Gaggero, M. Paolucci, and R. Ronco, "Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters," *Soft Computing*, vol. 25, pp. 1–20, 10 2021.

[21] W. Liu, R. Wang, T. Zhang, K. Li, W. Li, and H. Ishibuchi, "Hybridization of evolutionary algorithm and deep reinforcement learning for multi-objective orienteering optimization," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2022, doi:10.1109/TEVC.2022.3199045.

[22] Y. Li, G. Hao, Y. Liu, Y. Yu, Z. Ni, and Y. Zhao, "Many-objective distribution network reconfiguration via deep reinforcement learning assisted optimization algorithm," *IEEE Transactions on Power Delivery*, vol. 37, no. 3, pp. 2230–2244, 2022.

[23] F. Zhao, S. Di, and L. Wang, "A hyperheuristic with q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3337–3350, 2023.

[24] 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.

[25] Z. Zhang, Z. Wu, H. Zhang, and J. Wang, "Meta-learning-based deep reinforcement learning for multiobjective optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2022, doi:10.1109/TNNLS.2022.3148435.

[26] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman, "Push and pull search for solving constrained multi-objective optimization problems," *Swarm and Evolutionary Computation*, vol. 44, pp. 665–679, 2019.

[27] Q. Zhu, Q. Zhang, and Q. Lin, "A constrained multiobjective evolutionary algorithm with detect-and-escape strategy," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 938–947, 2020.

[28] A. Santiago, B. Dorronsoro, H. J. Fraire, and P. Ruiz, "Micro-genetic algorithm with fuzzy selection of operators for multi-objective optimization: $\mu$fame," *Swarm and Evolutionary Computation*, vol. 61, p. 100818, 2021.

[29] Y. Yuan, H. Xu, and B. Wang, "An experimental investigation of variation operators in reference-point based many-objective optimization," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 775–782.

[30] 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*, ser. GECCO '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 2003–2010.

[31] A. Lin, P. Yu, S. Cheng, and L. Xing, "One-to-one ensemble mechanism for decomposition-based multi-objective optimization," *Swarm and Evolutionary Computation*, vol. 68, p. 101007, 2022.

[32] R. Sutton and A. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998.

[33] S. Fayaz, M. Zaman, and M. Butt, *Machine Learning: An Introduction to Reinforcement Learning*, 07 2022, pp. 1–22.

[34] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 05 1992.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, 02 2015.

[36] 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, 03 2019.

[37] W. Gong, Z. Cai, and D. Liang, "Adaptive ranking mutation operator based differential evolution for constrained optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 716–727, 2015.

[38] Q. Zhang, A. Zhou, S. Zhao, P. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the cec 2009 special session and competition," *Mechanical Engineering*, 01 2008.
[39] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, and E. Goodman, "Difficulty adjustable and scalable constrained multiobjective test problem toolkit," *Evolutionary Computation*, vol. 28, no. 3, pp. 339–378, 2020.
[40] Z. Liu and Y. Wang, "Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 870–884, 2019.
[41] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. Yugen, J. Mo, C. Wei, and E. Goodman, "An improved epsilon constraint-handling method in moea/d for cmops with large infeasible regions," *Soft Computing*, vol. 23, 12 2019.
[42] M. Ming, A. Trivedi, R. Wang, D. Srinivasan, and T. Zhang, "A dual-population-based evolutionary algorithm for constrained multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 739–753, 2021.
[43] Z.-Z. Liu, B.-C. Wang, and K. Tang, "Handling constrained multiobjective optimization problems via bidirectional coevolution," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 163–10 176, 2022.
[44] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for pareto-based algorithms in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 348–365, 2014.
[45] K. Yu, J. Liang, B. Qu, Y. Luo, and C. Yue, "Dynamic selection preference-assisted constrained multiobjective differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 5, pp. 2954–2965, 2022.
[46] J. Yuan, H.-L. Liu, and Z. He, "A constrained multi-objective evolutionary algorithm using valuable infeasible solutions," *Swarm and Evolutionary Computation*, vol. 68, p. 101020, 2022.
[47] R. Agrawal, K. Deb, and R. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 06 2000.
[48] H. Ishibuchi, R. Imada, N. Masuyama, and Y. Nojima, "Comparison of hypervolume, igd and igd+ from the viewpoint of optimal distributions of solutions," in *Evolutionary Multi-Criterion Optimization*, 01 2019, pp. 332–345.
[49] H. Ishibuchi, L. M. Pang, and K. Shang, "Difficulties in fair performance comparison of multi-objective evolutionary algorithms [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 17, no. 1, pp. 86–101, 2022.
[50] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009.
[51] J. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," p. 203, 05 1995.
[52] Y. Tian, H. Chen, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Integrating conjugate gradients into evolutionary algorithms for large-scale continuous multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 10, pp. 1801–1817, 2022.
[53] Y. Tian, Y. Feng, X. Zhang, and C. Sun, "A fast clustering based evolutionary algorithm for super-large-scale sparse multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 4, pp. 1048–1063, 2023.
[54] H. Li, J. Li, P. Wu, Y. You, and N. Zeng, "A ranking-system-based switching particle swarm optimizer with dynamic learning strategies," *Neurocomputing*, vol. 494, pp. 356–367, 2022.
[55] N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone, and X. Liu, "A dynamic neighborhood-based switching particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9290–9301, 2022.
[56] D. Guo, X. Wang, K. Gao, Y. Jin, J. Ding, and T. Chai, "Evolutionary optimization of high-dimensional multiobjective and many-objective expensive problems assisted by a dropout neural network," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2084–2097, 2022.
[57] X. Luo, Y. Yuan, S. Chen, N. Zeng, and Z. Wang, "Position-transitional particle swarm optimization-incorporated latent factor analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3958–3970, 2022.

**Ming Fei** received the B.Sc. degree from China University of Geosciences, Wuhan, China, in 2019. He is currently pursuing the Ph.D degree with the School of Computer, China University of Geosciences, Wuhan, China.

His current research interests include evolutionary multiobjective optimization methods and their applications.

**Wenyin Gong (Member, IEEE)** received the B.Eng., M.Eng., and Ph.D. degrees in computer science from China University of Geosciences, Wuhan, China, in 2004, 2007, and 2010, respectively.

He is currently a Professor with School of Computer Science, China University of Geosciences, Wuhan, China. His research interests include evolutionary algorithms, evolutionary optimization, and their applications. He has published over 80 research papers in journals and international conferences.

He served as a referee for over 30 international journals, such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, *IEEE Computational Intelligence Magazine*, *ACM Transactions on Intelligent Systems and Technology*, *Information Sciences*, *European Journal of Operational Research*, *Applied Soft Computing*, *Journal of Power Sources*, etc. Professor Gong currently serves as Associate Editor of *Expert Systems with Applications*, *International Journal of Bio-Inspired Computation*, *Memetic Computing*, etc.

**Ling Wang (Member, IEEE)** received the B.Sc. in automation and Ph.D. in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively.

Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a full professor in 2008. His current research interests include intelligent optimization and production scheduling. He has authored five academic books and more than 300 refereed papers.

Professor Wang is a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, the Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China. Professor Wang now is the Editor-in-Chief of *International Journal of Automation and Control*, and the Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Swarm and Evolutionary Computation*, *Expert Systems with Applications*, etc.

**Yaochu Jin (Fellow, IEEE)** received the B.Sc., M.Sc., and Ph.D. degrees in automatic control from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree in neuroinformatics from Ruhr University Bochum, Bochum, Germany, in 2001.

He is currently an Alexander von Humboldt Professor of artificial intelligence with the Faculty of Technology, Bielefeld University, Bielefeld, Germany, endowed by the German Federal Ministry of Education and Research. He is also the Distinguished Chair and a Professor of computational intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K. His main research interests include multiobjective and data-driven evolutionary optimization, evolutionary multiobjective learning, trustworthy AI, and evolutionary developmental AI.

Dr. Jin is a member of Academia Europaea. He was named by the Web of Science as “a Highly Cited Researcher” from 2019 to 2022 consecutively.