Controlling sequential hybrid evolutionary algorithm by Q-learning

ZHANG, H.; Sun, J.Y.; Bäck, T.H.W.; Zhang, Q.F.; Xu, Z.B.

Research Frontier

Haotian Zhang and Jianyong Sun
Xi'an Jiaotong University, CHINA

Thomas Bäck
Leiden University, THE NETHERLANDS

Qingfu Zhang
City University of Hong Kong, HONG KONG

Zongben Xu
Xi'an Jiaotong University, CHINA

Controlling Sequential Hybrid Evolutionary Algorithm by Q-Learning

Abstract

Many state-of-the-art evolutionary algorithms (EAs) can be categorized as sequential hybrid EAs, in which various EAs are sequentially executed. The timing to switch from one EA to another is critical to the performance of the hybrid EA because the switching time determines the allocation of computational resources and thereby it helps balance exploration and exploitation. In this article, a framework for adaptive parameter control for hybrid EAs is proposed, in which the switching time is controlled by a *learned* agent rather than a manually designed scheme. First the framework is applied to an adaptive differential evolution algorithm, LSHADE, to control when to use the scheme to reduce the population. Then the framework is applied to the algorithm that won the CEC 2018 competition, i.e., the hybrid sampling evolution strategy (HSES), to control when to switch from the univariate sampling phase to the Covariance Matrix Adaptation Evolution Strategy phase. The agents for parameter control in LSHADE and HSES are trained by using Q-learning and deep Q-learning to obtain the learned algorithms Q-LSHADE and DQ-HSES. The results of experiments on the CEC 2014 and 2018 test suites show that the learned algorithms significantly outperform their counterparts and some state-of-the-art EAs.

IMAGE LICENSED BY INGRAM PUBLISHING

I. INTRODUCTION

Evolutionary computation has been studied since the 1950s [1] and many promising evolutionary algorithms (EAs) have been proposed for solving black-box optimization problems, such as the genetic algorithm (GA), [2] differential evolution (DE), [3], [4] particle swarm optimization (PSO), [5] evolution strategies (ES), [6] and evolutionary programming (EP) [7].

The successful application of an EA depends on a variety of factors, including but not limited to the incorporation of prior knowledge of the optimization problem at hand, the design of the algorithmic components (including the recombination and selection operators), and the determination of the algorithmic parameters. The algorithmic parameters of an EA significantly influence its performance. In the previous work [8], these algorithmic parameters were categorized as either structural or numerical. The structural parameters of an EA control the algorithmic procedure and hence influence the computational complexity of the algorithm. Taking a hybrid EA as an example, when to switch from one EA phase to another is an important structural parameter. Parameters such as the scaling factor (F) and the crossover rate (CR) in DEs, and the crossover and mutation probability in GAs can be classified as the numerical parameters, which are usually directly responsible for the generation of offspring.

The process of finding the optimal time-invariant parameters of EAs is commonly referred to as "parameter tuning" [9]. To find the optimal parameters, the "parameter response function," i.e., the metric that measures the performance of the considered EA, is usually optimized. Considering the intrinsic randomness of the EA, its performance should be measured by executing it several times for each set of parameter configurations, which is highly time consuming. Furthermore, information on the derivatives of the parameter response function is usually unknown. These factors make parameter tuning an expensive optimization problem [9].

Digital Object Identifier 10.1109/MCI.2022.3222057
Date of current version: 13 January 2023

Corresponding author: Jianyong Sun (e-mail: jy.sun@xjtu.edu.cn).

Time-variant (or adaptive) parameter control has been more popular in recent work on EAs than the use of time-invariant parameters. Parameter control means adaptively setting the parameters of an EA during the evolutionary search process. This has mostly involved the adaptive control of numerical parameters, such as F and CR in DE, [10], [11], [12] in the literature. By contrast, few studies have considered controlling such structural parameters as the population size [13]. Further, almost all adaptive mechanisms have been designed heuristically, which may restrict the capacity of the control mechanism because finding the optimal mechanism is challenging and time-consuming, and the ability of humans for searching is limited. In this paper, the search for the optimal control mechanism is modeled as a "learning" problem based on the recently proposed "learning-to-optimize" technology [14], [15].

The basic idea of "learning-to-optimize" is to learn useful knowledge by optimizing related problems and using this knowledge to optimize new problems efficiently. In this paper, we propose applying this idea to design an adaptive mechanism for structural parameters in sequential hybrid EAs. A sequential hybrid EA is composed of several phases in a combination of EAs. The underlying idea is to use the advantages of different EAs in different stages of search to ensure excellent algorithmic performance. The timing to switch from one phase to another is thus critical for performance.

Due to the randomness of the search process of an EA, one way to find the optimal adaptive mechanism to determine the switching time is via reinforcement learning (RL), which is based on modeling the search process of the EA as a Markov decision process (MDP). In this paper, we propose training an agent to control the switching time by using two popular RL algorithms: Q-learning and deep Q-learning [16]. Our contributions can be summarized as follows:

- An RL-based framework for sequential hybrid EAs is proposed, for the first time in the literature, in which the switching time is controlled by an RL agent that learns by using Q-learning/deep Q-learning.
- The framework is applied to the HSES [17] and a modified version of LSHADE, and the resultant algorithms are called DQ-HSES and Q-LSHADE.
- The proposed algorithms are evaluated by comparing them with three well-known EAs on the CEC 2014 and 2018 test suites. The results show that DQ-HSES and Q-LSHADE can significantly outperform their counterparts, i.e., the HSES and LSHADE, respectively. Further, the results show that DQ-HSES outperforms the compared EAs in general, which implies that the learned RL agent can significantly improve the performance of existing sequential hybrid EAs.

An RL-based framework for sequential hybrid EAs is proposed, for the first time in the literature, in which the switching time is controlled by an RL agent that learns by using Q-learning/deep Q-learning.

The remainder of this paper is organized as follows: Section II briefly introduces related work, including studies on parameter tuning/control, and provides the preliminaries on learning-to-optimize and reinforcement learning. Section III details the proposed Q-learning-based framework for the adaptive control of the structural parameters. Its applications to LSHADE and HSES are presented in Sections IV and V, respectively. Sections VI and VII show the experimental results obtained on the CEC 2014 and 2018 test suites, and Section VIII summarizes the conclusions of this paper.

II. RELATED WORK AND PRELIMINARIES

A. Parameter Tuning and Control

1) Parameter Tuning

Parameter tuning refers to the process of choosing a set of optimal parameters for an EA. Derivative-free methods of optimization, such as the Bayesian Optimization Algorithm (BOA) [18], Sequential Model based Algorithm Configuration (SMAC) [19], and Parameter Iterative Local Search (ParamILS) [20], are often used for parameter tuning in EAs. Roman et al. [21] used the BOA to tune the parameters including the spread parameter, the population size, and the offspring size in their hybrid kernel estimation of the distribution algorithm. Huang et al. [22] used the BOA to tune the numerical parameters of an EA called the scalable approach based on hierarchical decomposition [23].

2) Parameter Control

Three mechanisms are mainly used to control numerical parameters. First a numerical parameter is generated/sampled randomly in each generation. In SaDE [24], for example, F is sampled from a Gaussian distribution $\mathcal{N}(0.5, 0.3)$ in each generation for each individual. In SaNSDE [25], F is set based on either a fixed normal distribution or a Cauchy distribution.

In the second mechanism to control numerical parameters, the parameter is generated adaptively based only on information collected during the evolution process. In JADE proposed by Zhang et al. [10], the scaling factor F (resp., the crossover rate CR) is generated from a Cauchy (resp., Gaussian) distribution in which the median (resp., mean) is the Lehmer mean (resp., arithmetic mean) of successful parameters in the last generation. Some well-known DEs, such as LSHADE [12], iL-SHADE [26], and jSO [27], have the same mechanism as JADE. In the adaptive PSO proposed by Zhan et al. [28], the inertia weight and the acceleration coefficients are adaptively generated according to information collected during the evolution process.

In the third mechanism, the numerical parameters are controlled by both a handcrafted mechanism and information collected during the evolution process. In jDE [29], F and CR are either inherited from the last generation or sampled from pre-fixed uniform distributions for each individual. CoBiDE [30] samples F and

CR from bimodal Cauchy distributions with parameters that are fixed in advance in each generation for each individual if the individual does not improve fitness; otherwise, the parameters are inherited from the last generation.

Most EAs fix such structural parameters as the population size during the evolutionary procedure [13]. Few studies have investigated the adaptive control of the population size. According to a review by Piotrowski [13], research on the adaptive control of the population size can be categorized into four classes. The first class of methods is based on self-adaptation at the individual level. In [31], each individual is assigned a coefficient that is updated during the evolutionary process. The coefficients are used to update the population size. In the second class of methods to adaptively control the population size, population size is negatively correlated with the diversity of fitness in each generation [32]. In the third class of methods, the population size is set based on improvements in fitness in recent generations [33]. That is, the population size increases if the best fitness does not improve in a number of generations; otherwise decreases. In the fourth class, the population size is adaptively reduced, such as the linear population size reduction (LPSR) strategy proposed in LSHADE [12]. The LPSR strategy has also been used in several recently developed adaptive DEs, such as MPEDE [34], iL-SHADE [26], and jSO [27] Zhu et al. [35] proposed adaptively increasing or reducing the population size based on observations during the evolutionary procedure.

In Alg. 1, the involved functions, including Collect(·), Represent(·) and the Q-table or network, depend on specific EA. In the following sections, this framework is applied to two well known hybrid EAs, i.e. LSHADE [12] and HSES [17].

Structural parameters other than the population size also often appear in hybrid EAs [36], [37], [38]. The frequency of local search, the number of fitness evaluations assigned for local search, the fraction of individuals used to perform local search, and the selection of the recombination operators can be considered to be structural parameters in a hybrid EA. In the algorithm that won the CEC 2018 competition, the hybrid sampling evolution strategy (HSES) [17], the switching time from adaptive univariate sampling to CMA-ES is also a structural parameter.

The structural parameters in hybrid EAs are largely responsible for allocating computational resources to different phases of the EA to balance exploration and exploitation. The adaptive setting of the structural parameters in hybrid EAs is often handled manually. Taking EP/LS [37] as an example, the EP phase terminates when the value of the penalty function is smaller than a pre-fixed value and the value of the optimization function in the current generation is smaller than that in the previous generation. In PSO/DE [39], the computational resources are allocated according to the number of non-dominated solutions obtained. The univariate sampling phase in HSES [17] is executed for fixed generations and the CMA-ES phase is then applied. SaDE [24] summarizes historical information during the search procedure and uses it to adaptively select operators from a pool of mutation operators in each generation. cDE [40] defines a candidate pool of operators and selects the best one based on the number of success of each operator. MPEDE [34] divides the population into four parts, and assigns three of them to three DE operators and the fourth to the operator that delivers the best performance in the previous several generations. The notations used in this paper are listed in Table I.

B. Learning-to-Optimize Technology

The deep neural network is typically used in the implementation of learning-to-optimize to represent the knowledge learned for optimization. Andrychowicz et al. [14] proposed learning the descent

TABLE I The notations used in the paper.

NOTATIONS	EXPLANATION	NOTATIONS	EXPLANATION
// Reinforcement learning			
$\mathcal{S} \subseteq \mathbb{R}^D$	D-dimensional state space	a_t	the action at the t-th time step
$\mathcal{A} \subseteq \mathbb{R}^d$	d-dimensional action space	$\pi(a_t \mid s_t; \theta)$	the policy with parameter θ
μ_0	initial distribution of the state	$p(s_{t+1} \mid a_t, s_t)$	the transition probability
$r \in \mathbb{R}$	the reward	$v(s)$	the state-value function
T	the time horizon limit	$Q(s, a)$	the action-value function
s_t	the state at the t-th time step	$Q(s, a, w)$	deep Q network with parameter w
//Q-LSHADE and DQ-HSES			
LPSR	linear population size reduction	$N^{\min}, N^{\text{ini}}$	the minimum and initial population size
nfes	the number of fitness evaluations	nfes_s	the parameter used to decide when to use LPSR
HPSS	the hybrid population size strategy	LSHADE_o	the original LSHADE
LSHADE_f	the LSHADE with fixed population size	LSHADE_c	the LSHADE with HPSS

direction in the gradient descent algorithm by using a recurrent neural network called long short-term memory (LSTM) [41]. Wang et al. [42] used LSTM to learn the hyper-parameters of the commonly used training algorithm ADAM [43], and Chen et al. [44] used RNN to decide promising iterates for derivative-free problems. Li et al. [15] used RL to learn to optimize continuous optimization problems.

The application of learning-to-optimize is still in its infancy. Sharma et al. proposed using deep Q-learning to adaptively select operators from a pool of mutation operators in a hybrid DE [45]. In preliminary work for this paper [8], Q-learning was used to tune the switching time of HSES. This along with prevalent studies show that learning-to-optimize can be advantageous for the successful application of EAs in the following ways: First, it can realize the automatic tuning/control of the structural parameters to not only significantly reduce the amount of computational resources required for tuning, but also to improve the efficiency of tuning/control. Second, it can inspire the *learning* of new EAs.

C. Markov Decision Process and Reinforcement Learning

Reinforcement learning (RL) is a key technology in Artificial Intelligence. It has been applied to solve different control tasks, such as the game of GO [46], Atari games [47], and robot control [48]. It can be modeled as a Markov decision process (MDP) [49], which is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mu_0, p, r, \pi, T)$, where $\mathcal{S} \subseteq \mathbb{R}^D$ denotes the state space, $\mathcal{A} \subseteq \mathbb{R}^d$ the action space, μ_0 the initial distribution of the state, $r \in \mathbb{R}$ the reward, and T the time horizon. D (resp., d) is the number of dimensions of the state space (resp., action space), and is problem dependent. At each time step t, $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ are the current state and the action, respectively. The policy is then defined as: $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where $\pi(a_t|s_t;\theta)$ is the probability of choosing action a_t when observing s_t with θ as the parameter. $p(s_{t+1}|a_t, s_t)$ is the transition probability.

HSES [17] is the winner algorithm in the CEC 2018 competition, in which univariate sampling and CMA-ES are applied sequentially. . . . In the following discussion, how to control the switching time by applying the proposed framework is presented.

Figure 1 shows a finite-horizon MDP. Starting from an initial state s_0, an action a_0 is taken based on the policy $\pi(a_0|s_0, \theta)$. s_1 is observed according to the transition probability $p(s_1|a_0, s_0)$, and a reward r_1 is obtained. This procedure is repeated until the horizon limit T is reached. The set $\{s_0, a_0, r_1, \ldots, a_{T-1}, r_T, s_T\}$ is called a trajectory.

The aim of RL is to find an optimal policy π^* such that the expectation of the cumulative reward, i.e., $R(\tau) = [\sum_{t=0}^{T-1} \gamma^t r_{t+1}]$, is maximized, where γ is a constant that controls the time decay. Many RL methods have been developed to handle different environments, such as Q-learning for discrete action and state space, deep Q-Learning (DQL) for discrete action and continuous state space, and the policy gradient for continuous action and state space [16]. Note that Q-learning is applicable only to discrete state spaces. Deep Q-learning (DQL) has been proposed to deal with a continuous state space [47]. A deep neural network is applied to regress the discrete state into a continuous one. The details of Q-learning and DQL have been provided in *Supplementary Materials*.

III. THE FRAMEWORK

A sequential hybrid EA is composed of various EA phases. Each EA phase is equipped with some computational budgets. The timing to switch from one EA phase to another is important to a hybrid EA's performance. The switching time can be considered as a structural parameter. This paper focuses on hybrid EAs with two EA phases and presents a general framework in which an intelligent agent is employed to control the switching time.

The proposed framework is summarized in Algorithm 1. First, new solutions are generated by using the first EA (i.e., $\mathsf{EA}_1(\cdot)$) where Θ is the number of fitness evaluations used for implementing $\mathsf{EA}_1(\cdot)$. That is, the algorithm judges whether to switch after every Θ evaluations. At each time t, $\mathcal{G}$ takes the current population and the algorithmic parameters Γ_{t-1} as input.[1] It outputs new population $\mathbf{X}^t$ and its function values $\mathbf{F}^t$. Second, information collected so far by function $\mathsf{Collect}(\cdot)$ (line 7) is summarized by function $\mathsf{Represent}(\cdot)$ (line 8) to obtain the current state s_t. At last, the action a_t is taken based on the learned network $Q(s_t, a, w)$ in line 9 (or Q-table $Q(s_t, a)$), where a represents the action which takes value from its domain $\mathcal{A}$. In this paper $\mathcal{A} = \{0, 1\}$. Depending on a_t, the search process decides whether to switch from EA_1 to EA_2 or not (line 11 to 14). If switching happens, EA_2 will be implemented with left computing resources maxNFEs $- t\Theta$ where maxNFEs is the maximum number of fitness evaluation for this hybrid algorithm. Thus T is always set smaller than $\lfloor \frac{maxNFEs}{\Theta} \rfloor$, so that guaranteeing some computing resources left for EA_2. Notice that a DQN $Q(s_t, a_t, w)$ is used in Algorithm 1 to represent knowledge. Substituting $Q(s_t, a_t, w)$ with a Q-table $Q(s_t, a_t)$ can obtain a framework based on Q-learning.

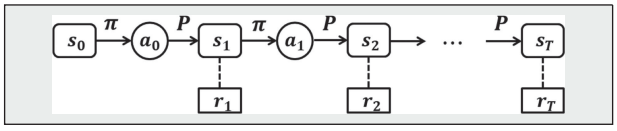

FIGURE 1 Illustration of a finite-horizon Markov decision process.

[1] Γ_t can be either time-invariant or variant. Time-variant parameters might be updated by some adaptive schemes, but not by Q-learning.

In Algorithm 1, the involved functions, including Collect$(\cdot)$, Represent$(\cdot)$ and the Q-table or network, depend on specific EA. In the following sections, this framework is applied to two well known hybrid EAs, i.e., LSHADE [12] and HSES [17].

Algorithm 1. The Algorithmic Framework based on Adaptive Structural Parameter Control

Input: an optimization function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{\tilde{d}}$, an initial population $\mathbf{X}^0 \in \mathbb{R}^{\tilde{d} \times N}$, maximum number of fitness evaluation maxNFEs, a Q network $Q(\cdot,\cdot,w)$ and horizon T.
Output: an optimal solution $\mathbf{x}^*$

Initialization: $\mathbf{X}^0, \Gamma_0, \mathcal{F} \leftarrow \emptyset$ and $\mathcal{X} \leftarrow \emptyset$;
Set $\mathcal{G} \leftarrow \mathrm{EA}_1(\cdot); \Theta \leftarrow \mathrm{NFE}_s$;
for $t = 1 \rightarrow T$ **do**
  $[\mathbf{X}^t, \mathbf{F}^t, \Gamma_t] \leftarrow \mathcal{G}(\mathbf{X}^{t-1}, f, \Theta; \Gamma_{t-1})$;
  $\mathcal{F} \leftarrow \mathcal{F} \bigcup \mathbf{F}^t$;
  $\mathcal{X} \leftarrow \mathcal{X} \bigcup \mathbf{X}^t$;
  $\mathcal{U}^t \leftarrow \mathrm{Collect}(\mathcal{X}, \mathcal{F})$;
  $s_t \leftarrow \mathrm{Represent}(\mathcal{U}^t)$;
  $a_t \leftarrow \arg\max_{a \in \mathcal{A}}\{Q(s_t, a, w)\}$;
  if $a_t == 1$ or $t == T$ **then**
    $\mathcal{G} \leftarrow \mathrm{EA}_2(\cdot)$;
    Exit;
  else
    $\mathcal{G} \leftarrow \mathrm{EA}_1(\cdot)$;
  end
end
$[\mathbf{X}^*, \mathbf{F}^*, \Gamma_{t+1}] \leftarrow \mathcal{G}(\mathbf{X}^t, f, \mathrm{maxNFEs} - t\Theta; \Gamma_t)$;
return $\mathbf{x}^* \leftarrow \arg\min \mathbf{F}^*$.

IV. APPLYING THE FRAMEWORK TO LSHADE

A. Hybrid Population Size Strategy

The population size is an essential parameter for EA, which is often fixed during the search procedure in most EAs. LSHADE [12] applies a simple scheme to control the population size, called linear population size reduction (LPSR) strategy. The scheme takes the population size as a linear function of the number of fitness evaluations (nfes) with negative slope. The LPSR strategy can be applied as follows:

$$N_{g+1} = \left[\frac{N^{\min} - N^{\mathrm{ini}}}{\mathrm{maxNFEs}} \cdot \mathrm{nfes} + N^{\mathrm{ini}}\right] \tag{1}$$

where $N^{\min}, N^{\mathrm{ini}}$ are pre-fixed constants representing the minimum population size and initial population size, respectively; and $[a]$ rounds a to its nearest integer. Some DE variants, such as iL-SHADE [26] and jSO [27], also apply the LPSR.

In the algorithms employing LPSR, LPSR is applied from the very beginning of the evolutionary search. However, it might be more appropriate to apply it whenever necessary. In light of this idea, our goal is to control when to apply the LPSR strategy. Thus the update formula of the population size is proposed as follows:

$$N_{g+1} = \begin{cases} \left[\frac{N^{\min} - N^{\mathrm{ini}}}{\mathrm{maxNFEs} - \mathrm{nfes}_s} \cdot (\mathrm{nfes} - \mathrm{nfes}_s) + N^{\mathrm{ini}}\right] & \text{if } \mathrm{nfes} \geq \mathrm{nfes}_s, \\ N_g & \text{otherwise,} \end{cases} \tag{2}$$

where nfes_s is the pre-defined parameter used to decide when to use LPSR.

It is seen that when $\mathrm{nfes} < \mathrm{nfes}_s$, the population size is fixed, while LPSR is used otherwise. Figure 2 shows how different nfes_s values affect the population size. It is seen that when $\mathrm{nfes}_s = 0$ (resp. $\mathrm{nfes}_s = \mathrm{maxNFEs}$), it degenerates to LPSR (resp. the pre-fixed population size). Different nfes_s could result in significantly different performance. The proposed population size updating strategy is named as hybrid population size strategy (HPSS).

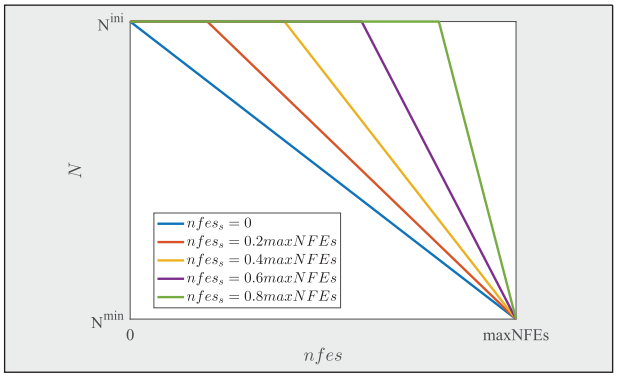

FIGURE 2 The effect of nfes_s on the population size reduction.

B. Q-LSHADE

This section presents how to apply the proposed framework (Algorithm 1) to LSHADE with HPSS. Note that LSHADE equipped with or without LPSR are regarded as two EAs. For clarification, LSHADE with fixed population size is denoted as LSHADE$_f$, and the original LSHADE is denoted as LSHADE$_o$. Then the proposed framework is used to control when to switch from LSHADE$_f$ to LSHADE$_o$.

The evolution procedure of LSHADE is summarized in *Supplementary Materials*. It can be written in a concise form as follows:

$$[\mathbf{X}^t, \{\tilde{f}^k_{\mathrm{best}}\}_{k=0}^{G_{\max}}, \Gamma_t] = \mathrm{LSHADE}_o(\mathbf{X}^{t-1}, f, \mathrm{NFEs}; \Gamma_{t-1}),$$

where NFEs is the number of fitness evaluations used by the algorithm, $G_{\max}$ is the number of generation conducted in NFEs, $\{\tilde{f}_{\text{best}}^k\}_{k=0}^{G_{\max}}$ is the union set of the best solutions found in each generation, and Γ_t is the algorithmic parameter including the means of F and CR, H and p. The form of LSHADE$_f$ is the same as LSHADE$_o$.

Further the LSHADE with HPSS is named as LSHADE$_c$, which is the combination of LSHADE$_f$ and LSHADE$_o$ and it has the similar form:

$$[\mathbf{X}^t, \{\tilde{f}_{\text{best}}^k\}_{k=0}^{G_{\max}}, \Gamma_t] = \text{LSHADE}_c(\mathbf{X}^{t-1}, f, \text{NFEs}; \Gamma_{t-1}, \mathsf{nfes}_s),$$

where nfes_s is the number of fitness evaluations assigned for LSHADE$_f$.

By applying the proposed framework, the resultant algorithm is named as Q-LSHADE and summarized in Algorithm 2. In Algorithm 2, LSHADE with fixed population size (i.e., LSHADE$_f$) is carried out for $0.2 \cdot$ maxNFEs (line 5). The best function value f_{best} is updated afterwards (line 6). The state is then computed according to the obtained best solutions found in each generation (line 7). The agent, which is a Q-table, is used to output an action (line 11) to decide whether to use LPSR or not. In case $a_t = 1$, LSHADE$_o$ is run for $(1 - 0.2t) \cdot$ maxNFEs. Otherwise, LSHADE$_f$ is run for another $0.2 \cdot$ maxNFEs. Here T is the horizon limit which is set to be 4.

C. Training the Agent

The evolution process of Q-LSHADE is modeled as an MDP and the Q-learning algorithm is used to train the agent. In the following, the MDP components for Q-LSHADE are presented, including state, action, transition probability and reward. f_{best}^k represents the minimum function value obtained up to the k-th generation.

State: s_t is a concatenation of s_t^1 and s_t^2, i.e., $s_t = [s_t^1, s_t^2]$, which are defined as follows. Denote

$$g_t = \left\lfloor \frac{0.2 \cdot t \cdot \text{maxNFEs}}{N} \right\rfloor + 1$$

where N is the initial population size, and $\lfloor x \rfloor$ means the greatest integer less than or equal to x. For $t = \{1, 2, 3, 4\}$,

$$s_t^1 = \frac{\log\left(f_{\text{best}}^{g_t-50}\right) - \log\left(f_{\text{best}}^{g_t}\right)}{\left|\log\left(f_{\text{best}}^{g_t-50}\right)\right|}, s_t^2 = \frac{\log\left(f_{\text{best}}^{0}\right) - \log\left(f_{\text{best}}^{g_t}\right)}{\left|\log\left(f_{\text{best}}^{0}\right)\right|} \quad (3)$$

where s_t^1 is used to measure the difference between the best function values in adjacent 50 generations; s_t^2 measures the descent rate from the first population. Eq (3) realizes the Represent $(\cdot)$ function in line 7 of Algorithm 2.

Action: The action space $\mathcal{A}$ is $\{0, 1\}$. That is, the agent can either choose to implement LPSR ($a_t = 1$) or not ($a_t = 0$) at the t-th time step.

Transition Probability: Given a horizon limit T, when $t < T - 1$ and $a_t = 0$, the next state s_{t+1} is defined as above. However, in case at some $t < T$, $a_t = 1$, and $t = T - 1, a_t = 0$, s_{t+1} will be the "terminal state" since in both cases Q-LSHADE will implement the LPSR: $t < T$, $a_t = 1$ implies Q-LSHADE implements LPSR at t-th time step. $t = T - 1, a_t = 0$ implies that Q-LSHADE does not implement LPSR at $t = T - 1$. However, the time horizon is set as T, which implies Q-LSHADE implements LPSR at $t = T$. Thus $t = T - 1, a_t = 0$ implies Q-LSHADE uses LPSR at $t = T$.

Reward: The agent will have no effect on the search after terminal state. If s_{t+1} is a terminal state, the following cases are considered.

- If $t = T - 1$ and $a_t = 0$, the reward r_{t+1} will be the negative logarithm of the minimum function value found by LSHADE$_c$ with $\mathsf{nfes}_s = 0.2 \cdot T \cdot$ maxNFEs.
- If $t < T$ and $a_t = 1$, reward r_{t+1} will be the negative logarithm of the minimum function value found by LSHADE$_c$ with $\mathsf{nfes}_s = 0.2 \cdot t \cdot$ maxNFEs.

If s_{t+1} is not a terminal state, the reward is set zero since the algorithm's performance is not observed before terminal state.

The training process is summarized in Algorithm 3. In the algorithm, to handle the scale problem (i.e., different

Algorithm 2. Pseudo Code of Q-LSHADE

Input: Initial parameters Γ_0, the maximum number of fitness evaluation maxNFEs, objective function f and the agent Q
Output: f_{best}

1: Set $f_{\text{best}} \leftarrow +\infty$, $a_0 \leftarrow 0$;
2: Initialize population $\mathbf{X}^0 = \{x_i^0, 1 \le i \le N\}$ randomly;
3: Set $\mathcal{G} \leftarrow$ LSHADE$_f$; Initialize Γ_0;
4: **for** $t = 1 \rightarrow T$ **do**
5: $[\mathbf{X}^t, \{\tilde{f}_{\text{best}}^k\}_{k=0}^{G_{\max}}, \Gamma_t] \leftarrow \mathcal{G}(\mathbf{X}^{t-1}, f, 0.2 \cdot \text{maxNFEs}, \Gamma_{t-1})$;
6: $f_{\text{best}} \leftarrow \min\{f_{\text{best}}, f(\mathbf{X}_i^t), 1 \le i \le N\}$;
7: $s_t \leftarrow \text{Represent}(\{\log(\tilde{f}_{\text{best}}^k)\}_{k=0}^{G_{\max}})$;
8: **if** $Q(s_t, 0) == Q(s_t, 1)$ **then**
9: Take action a_t randomly;
10: **else**
11: $a_t \leftarrow \arg\max_{a \in \{0,1\}}\{Q(s_t, a)\}$;
12: **end**
13: **if** $a_t == 1$ or $t == T$ **then**
14: $\mathcal{G} \leftarrow$ LSHADE$_o$;
15: Exit;
16: **else**
17: $\mathcal{G} \leftarrow$ LSHADE$_f$;
18: **end**
19: **end**
20: $[\mathbf{X}^t, \{\tilde{f}_{\text{best}}^k\}_{k=0}^{G_{\max}}, \Gamma_{t+1}] \leftarrow \mathcal{G}(\mathbf{X}^t, f, (1 - 0.2t)\text{maxNFEs}, \Gamma_t)$;
21: **return** $f_{\text{best}} = \min\{f_{\text{best}}, \{f(x_i), x_i \in \mathbf{X}^t\}\}$.

functions have different ranges of values), $Q(s,a)$ is used to aggregate the Q-tables $q(s,a)$ trained for each function.

In Algorithm 3, for each training function f_l, T trajectories Tr_m^l, $m = 1, 2, \ldots, T$ are first generated (line 4 to line 17). Each trajectory Tr_m^l is generated as follows: first LSHADE$_c$ is implemented with $\mathsf{nfes}_s = 0.2m \cdot \mathrm{maxNFEs}$ for 51 times. The best function values in each generation $\{\tilde{f}_{\mathrm{best}}^{i,k}\}_{i=1}^{51}, k = 0, \ldots, G_{\max}$ (line 7) are recorded. The mean of the logarithm of these best function values, i.e., $\frac{1}{51}\sum_{i=1}^{51} \log(\tilde{f}_{\mathrm{best}}^{i,k}), k = 0, \ldots, G_{\max}$ is computed (line 9). The state, action and reward are then obtained by summarizing these values according to the equations defined beforehand. With these trajectories, the Q-table $q(s,a)$ is updated from line 24 to line 28. Finally $Q(s,a)$ is updated by aggregating the trained Q-tables $q(s,a)$ (line 34 and line 37). After training, $Q(s,a)$ is used for Algorithm 2.

Algorithm 3. The Training Process for Q-LSHADE

Input: Training functions $f_1, \ldots, f_L$, the maximal number of epochs maxE, the horizon T and the learning rate α
Output: Q-table $Q(s,a)$

1: Initialize $Q(s,a) \leftarrow 0$ for all $s \in \mathcal{S}, a \in \{0,1\}$;
2: **for** $l = 1 \rightarrow L$ **do**
3: Initialize $q(s,a) \leftarrow 0$ for all $s \in \mathcal{S}, a \in \{0,1\}$;
//Create trajectories for each training function:
4: **for** $m = 1 \rightarrow T$ **do**
5: **for** $i = 1 \rightarrow 51$ **do**
6: Randomly initialize $\mathbf{x}^0$;
7: $\{\tilde{f}_{\mathrm{best}}^{i,k}\}_{k=0}^{G_{\max}} \leftarrow$ LSHADE$_c(\mathbf{x}^0, f_l, \mathrm{maxNFEs}, \Gamma, 0.2m \cdot \mathrm{maxNFEs})$;
8: **end**
9: $\bar{f}_{\mathrm{best}}^{k} \leftarrow \frac{1}{51}\sum_{i=1}^{51} \log(\tilde{f}_{\mathrm{best}}^{i,k}),\ k = 0, \ldots, G_{\max}$;
10: calculate $\{s_t\}_{t=1}^{m}$ according to Eq. (3) by using $\{\bar{f}_{\mathrm{best}}^{k}\}_{k=0}^{G_{\max}}$;
11: set $a_t \leftarrow 0,\ t < m$ and $a_m \leftarrow 1$;
12: **if** $m < T$ **then**
13: $\{r_t\}_{t=1}^{m} \leftarrow 0$ and $r_{m+1} \leftarrow -\min(\{\bar{f}_{\mathrm{best}}^{k}\}_{k=0}^{G_{\max}})$;
14: $\mathrm{Tr}_m^l \leftarrow \{s_1, a_1, r_2, \ldots, s_m, a_m, r_{m+1}\}$;
15: **else**
16: $\{r_t\}_{t=1}^{m-1} \leftarrow 0$ and $r_m \leftarrow -\min(\{\bar{f}_{\mathrm{best}}^{k}\}_{k=0}^{G_{max}})$;
17: $\mathrm{Tr}_m^l \leftarrow \{s_1, a_1, r_2, \ldots, s_{m-1}, a_{m-1}, r_m\}$;
18: **end**
19: **end**
20: // Update the Q table $q(s,a)$
21: **for** $e = 1 \rightarrow$ maxE **do**
22: **for** $t = 1 \rightarrow T-1$ **do**
23: **if** $t == T-1$ **then**
24: $q(s_t, 0) \leftarrow (1-\alpha)q(s_t,0) + \alpha(r_{t+1})$ where $\{s_t\} \in \mathrm{Tr}_T^l, r_{t+1} \in \mathrm{Tr}_T^l$;
25: **else**
26: $q(s_t, 0) \leftarrow (1-\alpha)q(s_t,0) + \alpha \max_{a_{t+1}} q(s_{t+1}, a_{t+1})$, where $\{s_t\} \in \mathrm{Tr}_T^l$;
27: **end**
28: $q(s_t, 1) \leftarrow (1-\alpha)q(s_t,1) + \alpha(r_{t+1})$ where $\{s_t\} \in \mathrm{Tr}_T^l, r_{t+1} \in \mathrm{Tr}_t^l$;
29: **end**
30: **end**
31: // Aggregate the Q-tables for the training functions
32: **for** $s \in \mathcal{S}$ **do**
33: **if** $q(s,1) > q(s,0)$ **then**
34: $Q(s,1) \leftarrow Q(s,1) + 1$;
35: **end**
36: **if** $q(s,1) < q(s,0)$ **then**
37: $Q(s,0) \leftarrow Q(s,0) + 1$;
38: **end**
39: **end**
40: **end**
41: **return** $Q(s,a)$.

Remark. Notice that from line 13 to line 17, the recorded trajectories are divided into two cases ($m < T$ and $m = T$). The two cases are corresponding to the two cases of reward. For $m < T$, it records the trajectory generated by LSHADE$_c$ using LPSR at $0.2m \cdot \mathrm{maxNFEs}$. For $m = T$, it records the trajectory generated by LSHADE$_c$ using LPSR at $0.2T \cdot \mathrm{maxNFEs}$. Thus for $m = T$, the trajectory only records actions until a_{T-1}, since $a_{T-1} = 0$ implies LSHADE$_c$ using LPSR at $0.2T \cdot \mathrm{maxNFEs}$, which can result in a terminal state.

V. CONTROLLING THE SWITCHING TIME IN HSES

In this section, the proposed framework is applied to control the switching time in HSES.

A. HSES

HSES [17] is the winner algorithm in the CEC 2018 competition, in which univariate sampling and CMA-ES are applied sequentially. The pseudo-code of HSES can be found in *Supplementary Materials*. In HSES, the timing to switch from univariate sampling to CMA-ES is very important to the performance of HSES. In HSES, the CMA-ES is implemented after using univariate sampling 100 generations. In the following discussion, how to control the switching time by applying the proposed framework is presented.

Algorithm 4. The Pseudo-Code of DQ-HSES

Input: an optimization function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{\widetilde{d}}$, initial population $\mathbf{X}^0 \in \mathbb{R}^{\widetilde{d} \times N}$, and the maximum number of evaluations maxNFEs

Output: an optimal solution $\mathbf{x}^*$

1: Set Idx $\in \mathbb{R}^{\widetilde{d}} \leftarrow \vec{0}, t \leftarrow 0, a_0 \leftarrow 0$;
2: **while** $a_t \neq 1$ and $t < T$ **do**
3: $\quad \mathbf{X}^{t+1} \leftarrow \text{UniSampling}(\mathbf{X}^t, f, \text{Idx}; 10)$;
4: $\quad t \leftarrow t + 1$;
5: $\quad s_t \leftarrow \text{Represent}(\mathbf{X}^t, \log(f(\mathbf{X}^t)))$;
6: $\quad a_t \leftarrow \arg\max_a \text{DQN}(s_t, a, w)$;
7: **end**
8: $[\mathbf{X}^{t_2}, \mathbf{x}_2^*] \leftarrow \text{CMA-ES}(\mathbf{X}^t; \theta)$;
9: Idx $\leftarrow \text{Detect}(\mathbf{X}^{t_2})$;
10: $[\mathbf{X}^{t_3}, \mathbf{x}_3^*] \leftarrow \text{UniSampling}(\mathbf{X}^{t_2}, f, \text{Idx})$;
11: **return** $\mathbf{x}^* \leftarrow \mathbf{x}_3^*$.

B. DQ-HSES

Here we present how to use a Deep Q Network (DQN) to control the switching time in HSES. Algorithm 4 summarizes the pseudo-code of the developed algorithm (named as DQ-HSES).

In DQ-HSES, whether to switch from univariate sampling to CMA-ES is judged in every 10 generations. In Algorithm 4, univariate sampling is first carried out for 10 generations (line 3). The obtained function values are summarized to obtain the current state (line 5). The DQN with the learned parameter w determines an action in line 6, which takes a value of 0 or 1. If the action is 1, the switch happens; otherwise the univariate sampling is carried out again. The rest of the components, including the parameter settings, are the same as in HSES. Notice that in the original HSES, the fitness evaluation number of CMA-ES is limited to maxNFEs/2. In our experiment, we guarantee the evaluations for the first univariate sampling is not larger than maxNFEs/2 by setting T for the first univariate sampling procedure. The flow chart of DQ-HSES is shown in Figure 3, in which the RL agent is used to control when to escape from the univariate sampling phase.

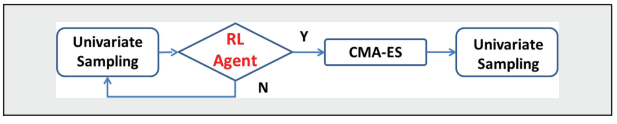

FIGURE 3 The flow chart of the proposed approach DQ-HSES.

C. Training the Switcher Agent

To find the optimal weights (denoted as w^*) for the DQN, which is called the switcher agent, deep Q-learning is used by modeling the evolution process of DQ-HSES as a finite-horizon MDP with continuous state and discrete action space. f_{best}^k represents the minimum function value obtained up to the k-th generation.

State: s_t is a concatenation of s_t^1 and s_t^2 which are defined as follows. When $t > 1$,

$$s_t^1 \triangleq \frac{\log\left(f_{\text{best}}^{10(t-2)}\right) - \log\left(f_{\text{best}}^{10t}\right)}{\left|\log\left(f_{\text{best}}^{10(t-2)}\right)\right|}, \qquad s_t^2 \triangleq \frac{\log\left(f_{\text{best}}^{0}\right) - \log\left(f_{\text{best}}^{10t}\right)}{\left|\log\left(f_{\text{best}}^{0}\right)\right|} \tag{4}$$

In case $t = 1$, $s_1^1 \triangleq \frac{\log(f_{\text{best}}^0) - \log(f_{\text{best}}^{10})}{|\log(f_{\text{best}}^0)|}$. In the definition, s_t^1 is used to measure the difference between the best function values in adjacent 20 generations; s_t^2 measures the descent rate from the first population.

The state s_t is then modified as a combination of various bins including $[10s_t^1, 10s_t^2]$, $[10s_t^1, 10s_t^2]$, $[s_t^1, 200s_t^2]$, $[s_t^1, 200s_t^2]$ for 10D, 30D, 50D and 100D problems, respectively.

Action: The action space $\mathcal{A}$ is $\{0, 1\}$. That is, the agent can either choose to switch (the action takes 1) to CMA-ES or not switch (the action takes 0).

Transition Probability: In case at some $t < T$, $a_t = 1$, and $t = T - 1$, $a_t = 0$, s_{t+1} will be the "terminal state". The horizon T constrains the maximal computational resources used for the first univariate sampling.

Reward: Three cases are considered when defining the reward. For the terminal state s_{t+1},

- ❏ if $t = T - 1, a_t = 0$ the reward r_{t+1} will be the negative logarithm of the minimum function value found by HSES by switching at the $10T-$th generation.
- ❏ if $t < T$ and $a_t = 1$, the reward r_{t+1} will be the negative logarithm of the minimum function value found by HSES by switching at the $(10t)$-th generation.

For non-terminal state, its reward is set to zero.

1) The Training Details

Given the above tuple definitions, the DQL training procedure can be summarized in Algorithm 5. Algorithm 5 differs from the classical DQL in two aspects. First, in classical DQL, the network takes a state as input. Its output is a vector which has the same dimension as the action space. Each component of the output vector corresponds to the Q-value for each action [50]. This means that the weights in the DQN except the last layer are shared for all actions. The shared weights can extract some common features of the state, but such structure could sacrifice the DQN's learning capacity. To address this problem, two networks $Q^1(S, w)$ and $Q^2(S, w)$ are used to replace $Q(S, 0, w)$ and $Q(S, 1, w)$, respectively.

Second, a DQN is trained for each training function rather than a DQN for all the functions. This is to eliminate the interferences among different learned DQNs due to the diverse ranges of different training functions.

Notice that in line 3, the trajectory Tr_m^l is generated similarly to that in the training process for Q-LSHADE (Algorithm 3). The difference is implementing

Algorithm 5. The training process for the DQN

Input: Training functions $f_1, \ldots, f_L$, the maximal number of epochs maxE, the horizon limit T and the learning rate α
Output: DQNs $\{Q_l^1(\cdot, w_1^*), Q_l^2(\cdot, w_2^*)\}_{l=1}^{L}$
1: Randomly Initialize $\{Q_l^1(\cdot, w_1), Q_l^2(\cdot, w_2)\}_{i=1}^{L}$
2: **for** $l = 1 \rightarrow L$ **do**
3: Create T trajectories $\mathrm{Tr}_m^l, 1 \leq m \leq T$;
4: **for** $e = 1 \rightarrow$ maxE **do**
5: **for** $t = 1 \rightarrow T - 1$ **do**
6: //Compute the target
7: **if** $t == T - 1$ **then**
8: $d_t^1 \leftarrow r_{t+1}$, where $r_{t+1} \in \mathrm{Tr}_T^l$;
9: **else**
10: $d_t^1 \leftarrow \gamma \max\{Q_l^1(s_{t+1}, w_1), Q_l^2(s_{t+1}, w_2)\}$, where $\{s_t\} \in \mathrm{Tr}_T^l$;
11: **end**
12: $d_t^2 \leftarrow r_{t+1}$, where $r_{t+1} \in \mathrm{Tr}_t^l$;
13: //update the parameters of the DQNs
14: $w_1 \leftarrow w_1 - \alpha \nabla_{w_1} \|Q_l^1(s_t, w_1) - d_t^1\|^2$;
15: $w_2 \leftarrow w_2 - \alpha \nabla_{w_2} \|Q_l^2(s_t, w_2) - d_t^2\|^2$;
16: **end**
17: **end**
18: $w_1^* \leftarrow w_1, w_2^* \leftarrow w_2$;
19: **return** $Q_l^1(\cdot, w_1^*), Q_l^2(\cdot, w_2^*)$.
20: **end**

HSES with $G_1 = 10\ m$ for 51 times and recording the best function values in each generation (G_1 is the number of generation assigned to the first univariate sampling in HSES), i.e., $\{\tilde{f}_{\mathrm{best}}^{i,k}\}_{i=1}^{51}, k = 0, \ldots, G_{\max}$, instead of using LSHADE$_c$ in Algorithm 3.

2) DQ-HSES

By embedding the learned DQNs into HSES, the resultant algorithm is named as DQN based HSES (dubbed as DQ-HSES). DQ-HSES is summarized in Algorithm 6. In the algorithm, the univariate sampling is first implemented for 10 generations. The learned DQNs are used to determine whether switching to CMA-ES or not (line 4 to 24). The rest of the algorithm after switching is the same as HSES. Likewise, the parameter settings for CMA-ES and univariate sampling are the same as in HSES.

Line 5–21 show how to use the L learned DQNs to determine an action for an optimization function. Here the concept of 'boosting' [51] is borrowed from machine learning. The core idea is to combine several weak agents for a strong agent. In our study, the L learned DQNs can be seen as weak agents. They vote to decide the action.

To implement the boosting mechanism, Vote_0 and Vote_1 are used to respectively record the number of votes for selecting action 0 and 1 by the learned DQNs (line 5 to line 15). The action is selected based on the greatness of the two values. If they are equal, a random action is selected.

To avoid over-fitting, the value of DQNs is set to be zero when a new state is far from the states ever met in the training set (line 8). The function $\mathsf{Judge}(s_t, \mathrm{Tr}_T^l, c, T)$, summarized in Algorithm 7, is used to make this decision. The Euclidean distance between s and the states in the training set Tr_T^l is firstly computed (line 3). If the minimum distance is larger than a fixed constant c, zero returns; otherwise one returns. The thought behind is that if a new state s_t is too far from the states in the training set, the prediction could be unstable and implausible since the training data is not comprehensive. In the experiment, c is set as 2.4,1.6,16.7,18.2 for 10D, 30D, 50D and 100D problems, respectively.

VI. EXPERIMENTAL RESULTS ON Q-LSHADE

The performance of Q-LSHADE is first investigated against its counterpart LSHADE on the CEC 2018 test suite[2] as the benchmark. The test suite contains 29 test functions that can be classified into four categories: unimodal functions F_1 and F_3, multi-modal functions $F_4 - F_{10}$, hybrid functions $F_{11} - F_{20}$, and composition functions $F_{21} - F_{30}$. In this section, Q-LSHADE is tested on 10D test functions and its performance is compared with LSHADE.

F_{13} and F_{16} are taken as the training functions. The training parameters are $\alpha = 0.005$ and maxE $= 100,000$. The other parameters of Q-LSHADE and LSHADE are the same as in the original reference [12]. maxNFEs is set to 10,000D. Because Q-learning works only for MDP with a finite and discrete state space, the space is divided into $[0, 10^{-6}]$, $(10^{-6}, 10^{-5}]$, $(10^{-5}, 10^{-3}]$, $(10^{-3}, 10^{-1}]$, $(10^{-1}, 1]$, $(1, +\infty)$ for s^1, and [0,0.1], (0.1,0.25], (0.25,0.4], (0.4,0.6], (0.6,1.5], $(1.5, +\infty)$ for s^2.

Table II summarizes the statistics obtained by Q-LSHADE and LSHADE on the benchmark functions over 51 runs. All the statistics in Table II and subsequent tables are computed based on the error values (i.e., the difference between the obtained optimum and the known global optimum). When the error values are smaller than or equal to 10^{-8}, they are assumed to be zero. The Wilcoxon rank-sum hypothesis test between LSHADE and Q-LSHADE at the 5% significance level is also listed, where †/§ implies that LSHADE performs significantly better/worse than Q-LSHADE and ≈ means that there is no significant difference between the algorithms. Furthermore, the value of BM is listed for recording the number of functions for which the algorithm obtains the best mean value.

Table II shows that Q-LSHADE significantly outperforms LSHADE on four functions and is worse than it on two functions. The value of BM obtained by

[2] https://github.com/P-N-Suganthan/CEC2018

Algorithm 6. Deep Q network based HSES (DQ-HSES)

Input: an optimization function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{\widetilde{d}}$, initial population $\mathbf{X}^0 \in \mathbb{R}^{\widetilde{d} \times N}$, the learned DQNs $\{Q_l^1(\cdot, w_1), Q_l^2(\cdot, w_2)\}_{l=1}^{L}$ and the maximum number of evaluations maxNFEs and a threshold c;
Output: an optimal solution $\mathbf{x}^*$

Set Idx $\leftarrow \vec{0}, t \leftarrow 0$;
while $t < T$ **do**
  $[\mathbf{X}^{t+1}, \mathbf{x}_1^*] \leftarrow \text{UniSampling}(\mathbf{X}^t, f, \text{Idx}; 10)$;
  Compute the state s_t by Eq. (4);
  set $\text{Vote}_0 \leftarrow 0, \text{Vote}_1 \leftarrow 0$;
  for $l = 1 \rightarrow L$ **do**
    if $\text{Judge}(s_t, \text{Tr}_T^l, c, T) == 0$ **then**
      $Q_l^1(s_t, w_1) \leftarrow 0, Q_l^2(s_t, w_2) \leftarrow 0$;
    end
    if $Q_l^1(s_t, w_1) > Q_l^2(s_t, w_2)$ **then**
      $\text{Vote}_0 \leftarrow \text{Vote}_0 + 1$;
    end
    else if $Q_l^1(s_t, w_1) < Q_l^2(s_t, w_2)$ **then**
      $\text{Vote}_1 \leftarrow \text{Vote}_1 + 1$;
    end
  end
  if $\text{Vote}_1 \neq \text{Vote}_0$ **then**
    $a_t \leftarrow (\text{Vote}_1 > \text{Vote}_0)$;
  else
    Take a_t randomly;
  end
  if $a_t == 1$ **then**
    Exit;
  end
  $t \leftarrow t + 1$;
end
$[\mathbf{X}^{t_2}, \mathbf{x}_2^*] \leftarrow \text{CMA} - \text{ES}(\mathbf{X}^t; \theta)$;
$\text{Idx} \leftarrow \text{Detect}(\mathbf{X}^{t_2})$;
$[\mathbf{X}^{t_3}, \mathbf{x}_3^*] \leftarrow \text{UniSampling}(\mathbf{X}^{t_2}, f, \text{Idx})$;
return $\mathbf{x}^* \leftarrow \mathbf{x}_3^*$.

Algorithm 7. Judge

Input: s, Tr_T^l, c, and time horizon T
Outut: output

$d_{min} \leftarrow +\infty$;
for $i = 1 \rightarrow T - 1$ **do**
  $d_{\min} \leftarrow \min\{d_{\min}, \|s - s_i\|_2^2\}, s_i \in \text{Tr}_T^l$;
end
if $d_{\min} > c$ **then**
  return output $\leftarrow 0$.
else
  return output $\leftarrow 1$.
end

Q-LSHADE is larger than that of LSHADE. Furthermore, the values of *BM* and the results of the hypothesis test on the remaining functions, i.e., functions excluding F_{13} and F_{16}, are presented in the brackets in Table II. When only the functions excluding the training functions are considered, Q-LSHADE still performs significantly better than LSHADE on four functions and worse on two functions. The value of *BM* obtained by Q-LSHADE is still greater than that of LSHADE. Thus the proposed learning-to-optimize based framework can learn a suitable agent to control the structural parameter.

VII. EXPERIMENTAL RESULTS ON DQ-HSES

In this section, DQ-HSES is compared with its counterpart HSES and some well-known EAs on the CEC 2018 competition test suite. Furthermore, to verify the capability of DQ-HSES for generalization, it is tested on the CEC 2014[3] test suite. Note that the original HSES is the winner of the CEC 2018 competition. The authors of a review [52] comprehensively studied various DEs, and concluded that SHADE, cDE, and CoBiDE with the rand/1/bin and curr-to-pbest/1/bin operators are the best three methods. Therefore, **cDE**, **CoBiDE**, and **jSO** (an advanced version of SHADE) are chosen as the baselines.

CoBiDE [30] is an adaptive DE that adapts the scaling factor F and crossover rate CR as follows: First, the initial values of F and CR for each individual are sampled from the following distributions:

$$F = \begin{cases} randc(0.65, 0.1) \text{ if } randu[0,1] \geq 0.5 \\ randc(1, 0.1) \text{ if } randu[0,1] < 0.5 \end{cases} \quad (5)$$

$$CR = \begin{cases} randc(0.95, 0.1) \text{ if } randu[0,1] \geq 0.5 \\ randc(0.1, 0.1) \text{ if } randu[0,1] < 0.5 \end{cases} \quad (6)$$

During the evolution, the successful values of F and CR are inherited, while their failed values are re-initialized using Eq. (5) and Eq. (6), respectively.

cDE [40] is also an adaptive DE. Its values of F and CR are sampled from a candidate pool composed of $F_{pool} = \{0.5, 0.8, 1\}$ and $CR_{pool} = \{0, 0.5, 1\}$. The candidate pool contains nine candidate pairs in total. They are denoted by $O_1 = \{0.5, 0\}, \ldots, O_9 = \{1, 1\}$. In the tth generation, the probability of selection of each candidate pair is $s_{t,k} = \frac{N_k^{succ} + c}{\sum_i N_i^{succ} + c}$, where N_k^{succ} is the number of individuals with better fitness values when the pair O_k is used and $c > 0$ is a constant used to avoid a singularity.

jSO [27] is an advanced version of LSHADE [12] and iL-SHADE [26], which are two variants of SHADE [11]. It took second place in the CEC 2017 competition.

To train the DQNs, six functions from CEC 2013[4], $F_2, F_6 - F_{10}$, and five functions from CEC 2018: $F_1, F_{10}, F_{15}, F_{18}$, and F_{29} are chosen as training

[3]The CEC 2014 test suite can be found at https://github.com/P-N-Suganthan/CEC2014

[4]The CEC 2013 test suite can be found at https://github.com/P-N-Suganthan/CEC2013

functions. The functions chosen from CEC 2018 cover all four categories in the suite. Further, both the CEC 2013 and the CEC 2018 test suites contain functions that are variations of some basic functions, such as the Bent Cigar function, Rosenbrock's function, and Rastrigin's function. The CEC 2013 test suite uses the Rotated Bent Cigar function and the CEC 2018 test suite uses the Shifted and Rotated Bent Cigar function. The two functions hold the same properties: unimodal, non-separable, and smooth-but-narrow ridge. This renders the functions in the CEC 2013 test suite more suitable for training the agent. A multi-layer perceptron (MLP) network with one hidden layer is used to be the agent. The other hyper-parameters are shown in Table III.

TABLE II Results obtained by LSHADE and Q-LSHADE on the CEC 2018 test suite in 10D, averaged over 51 runs.

	$10D$			
	Q-LSHADE		LSHADE	
	MEAN	STD	MEAN	STD
F_1	**0.00e+00**	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00
F_3	**0.00e+00**	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00
F_4	**0.00e+00**	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00
F_5	**2.34e+00**	7.66e-01	2.44e+00$^{\approx}$	9.81e-01
F_6	**0.00e+00**	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00
F_7	**1.20e+01**	7.98e-01	1.23e+01$^{\approx}$	1.18e+00
F_8	**2.38e+00**	9.54e-01	2.53e+00$^{\approx}$	9.40e-01
F_9	**0.00e+00**	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00
F_{10}	**1.94e+01**	3.12e+01	2.80e+01$^{\approx}$	4.30e+01
F_{11}	**2.21e-02**	1.57e-01	2.49e-02§	1.78e-01
F_{12}	**1.03e+01**	3.34e+01	3.14e+01$^{\approx}$	5.29e+01
F_{13}	**3.09e+00**	2.44e+00	**3.09e+00**$^{\approx}$	2.49e+00
F_{14}	**5.47e-01**	7.17e-01	7.93e-01$^{\approx}$	9.20e-01
F_{15}	**5.02e-02**	1.18e-01	1.88e-01§	2.14e-01
F_{16}	**3.38e-01**	2.13e-01	3.50e-01$^{\approx}$	2.23e-01
F_{17}	1.76e-01	2.11e-01	**7.15e-02**†	1.04e-01
F_{18}	**1.30e-01**	1.90e-01	2.47e-01§	2.63e-01
F_{19}	2.56e-02	1.67e-02	**5.76e-03**†	8.77e-03
F_{20}	**0.00e+00**	0.00e+00	1.22e-02$^{\approx}$	6.11e-02
F_{21}	**1.47e+02**	5.18e+01	1.51e+02$^{\approx}$	5.23e+01
F_{22}	9.84e+01	1.10e+01	**9.80e+01**$^{\approx}$	1.40e+01
F_{23}	3.03e+02	1.44e+00	**3.02e+02**$^{\approx}$	1.47e+00
F_{24}	**2.94e+02**	8.46e+01	3.03e+02$^{\approx}$	7.51e+01
F_{25}	**4.11e+02**	2.10e+01	4.13e+02$^{\approx}$	2.18e+01
F_{26}	**3.00e+02**	0.00e+00	**3.00e+02**$^{\approx}$	0.00e+00
F_{27}	**3.89e+02**	2.19e-01	**3.89e+02**$^{\approx}$	1.78e-01
F_{28}	**3.00e+02**	6.11e+01	3.30e+02§	9.20e+01
F_{29}	**2.34e+02**	2.88e+00	**2.34e+02**$^{\approx}$	3.36e+00
F_{30}	**3.96e+02**	9.56e+00	1.64e+04$^{\approx}$	1.14e+05
BM	**25(23)**		13(12)	
$\dagger/\approx/\S$	2/22/4 (2/20/4)			

TABLE III The hyper-parameter settings for algorithm 5.

HIDDEN LAYERS	1	LEARNING RATE α	0.0001
Hidden units	200	MDP horizon T	20
Activation functions	Sigmoid, Sigmoid	Maximal epoch *maxE*	5000
Optimization method	Gradient descent		

A. Results of DQ-HSES

In this section, first the results of comparison between DQ-HSES and HSES on the CEC 2014 and 2018 test suites are shown, and then the results of comparisons between DQ-HSES and cDE, and CoBiDE and jSO are presented. Then the learned DQNs in different dimensions are visualized.

1) DQ-HSES vs. HSES on CEC 2018

DQ-HSES and HSES are run for 200, 100, 51, and 20 times for all the problems in $10D$, $30D$, $50D$, and $100D$. The results are summarized in Tables IV and V. The Wilcoxon rank-sum hypothesis test is applied between HSES and DQ-HSES at the 5% significance level, where $\dagger/\S$ implies that HSES performs significantly better/worse than DQ-HSES and $\approx$ means that there is no significant difference between the algorithms.

Tables IV and V show that DQ-HSES performs better than HSES on eight functions and worse on one function on all $10D$ problems, while the results for 30D problems are nine and three; for $50D$, five and three; for $100D$, two and one.

The values of *BM* and the results of the hypothesis test on functions excluding the training functions are also presented in parentheses in the tables. DQ-HSES performs better than HSES on eight functions and worse on one function for $10D$ problems. For $30D$ problems, it performs better on seven functions and worse on two functions, and DQ-HSES performs better than HSES on five functions and worse on three functions for $50D$ problems. For $100D$ problems, it performs better on two functions and worse on one function.

On the whole, DQ-HSES obtains greater values of *BM* than HSES. Thus we conclude that the learned agent improves the performance of HSES.

TABLE IV **Results obtained by HSES and DQ-HSES on the test functions of CEC 2018 in 10D, 30D, and 50D.**

	$10D$			
	HSES		DQ-HSES	
	Mean	Std	Mean	Std
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_5	**8.05e-01**$^{\approx}$	8.42e-01	8.15e-01	8.79e-01
F_6	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_7	1.12e+01§	6.83e-01	**1.10e+01**	5.61e-01
F_8	**6.16e-01**$^{\approx}$	7.63e-01	7.01e-01	7.96e-01
F_9	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_{10}	1.00e+02$^{\approx}$	1.43e+02	**8.88e+01**	1.30e+02
F_{11}	**1.64e-01**†	3.96e-01	7.61e-01	3.50e+00
F_{12}	**1.71e+01**$^{\approx}$	4.58e+01	2.26e+01	5.52e+01
F_{13}	3.45e+00$^{\approx}$	2.63e+00	**3.17e+00**	2.58e+00
F_{14}	6.51e+00§	3.59e+01	**6.00e+00**	1.20e+01
F_{15}	**5.56e-01**$^{\approx}$	6.18e-01	8.17e-01	1.73e+00
F_{16}	**4.36e+00**$^{\approx}$	2.01e+01	4.41e+00	2.03e+01
F_{17}	1.52e+01§	1.07e+01	**1.18e+01**	9.95e+00
F_{18}	**5.33e-01**$^{\approx}$	4.62e-01	7.02e-01	1.61e+00
F_{19}	1.01e+00§	2.72e+00	**4.31e-01**	8.89e-01
F_{20}	1.04e+01§	1.02e+01	**6.98e+00**	9.06e+00
F_{21}	**1.95e+02**$^{\approx}$	2.44e+01	1.98e+02	1.76e+01
F_{22}	**1.00e+02**$^{\approx}$	0.00e+00	**1.00e+02**	0.00e+00
F_{23}	3.01e+02§	3.41e+00	**2.99e+02**	2.13e+01
F_{24}	3.24e+02§	2.78e+01	**3.21e+02**	3.90e+01
F_{25}	**4.46e+02**$^{\approx}$	1.03e+00	**4.46e+02**	1.18e+00
F_{26}	3.01e+02§	1.86e+01	**3.00e+02**	8.82e+00
F_{27}	**3.97e+02**$^{\approx}$	1.84e+00	**3.97e+02**	1.93e+00
F_{28}	5.94e+02$^{\approx}$	3.11e+01	**5.92e+02**	3.80e+01
F_{29}	**2.63e+02**$^{\approx}$	9.82e+00	2.64e+02	1.10e+01
F_{30}	**4.13e+02**$^{\approx}$	2.37e+01	**4.13e+02**	2.34e+01
BM	18 (14)		**20(18)**	
$\dagger/\approx/\S$	1/20/8 (1/15/8)			

	$30D$			
	HSES		DQ-HSES	
	Mean	Std	Mean	Std
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**3.43e+00**$^{\approx}$	6.39e+00	4.11e+00	9.37e+00
F_5	7.97e+00§	3.09e+00	**6.95e+00**	2.11e+00
F_6	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_7	4.11e+01§	4.34e+00	**3.53e+01**	1.31e+00
F_8	8.00e+00§	2.38e+00	**6.48e+00**	2.06e+00
F_9	**0.00e+00**$^{\approx}$	0.00e+00	2.63e-02	2.22e-01
F_{10}	9.79e+02§	3.62e+02	**8.43e+02**	3.36e+02
F_{11}	**1.41e+01**$^{\approx}$	2.12e+01	1.61e+01	2.29e+01
F_{12}	4.15e+01§	1.14e+02	**2.38e+01**	9.60e+01
F_{13}	3.21e+01$^{\approx}$	1.60e+01	**3.01e+01**	1.53e+01
F_{14}	**1.46e+01**$^{\approx}$	1.12e+01	1.55e+01	9.35e+00
F_{15}	**5.24e+00**†	2.77e+00	1.16e+01	1.17e+01
F_{16}	**2.59e+02**$^{\approx}$	1.79e+02	2.93e+02	1.81e+02
F_{17}	**4.45e+01**†	7.63e+01	6.33e+01	9.21e+01
F_{18}	2.07e+01$^{\approx}$	5.82e+00	**1.89e+01**	6.85e+00
F_{19}	3.87e+00$^{\approx}$	1.87e+00	**3.78e+00**	2.04e+00
F_{20}	**1.49e+02**$^{\approx}$	5.85e+01	1.58e+02	5.88e+01
F_{21}	2.08e+02§	4.10e+00	**2.07e+02**	3.05e+00
F_{22}	**1.00e+02**$^{\approx}$	0.00e+00	**1.00e+02**	0.00e+00
F_{23}	3.53e+02$^{\approx}$	8.45e+00	**3.52e+02**	8.79e+00
F_{24}	4.20e+02§	4.33e+00	**4.19e+02**	5.30e+00
F_{25}	**3.86e+02**$^{\approx}$	3.00e-02	**3.86e+02**	2.25e-02
F_{26}	9.11e+02$^{\approx}$	1.27e+02	**9.03e+02**	1.37e+02
F_{27}	**5.17e+02**†	8.03e+00	5.25e+02	9.05e+00
F_{28}	3.16e+02§	3.80e+01	**3.15e+02**	3.73e+01
F_{29}	4.70e+02§	7.03e+01	**4.62e+02**	7.25e+01
F_{30}	**2.05e+03**$^{\approx}$	4.06e+01	2.06e+03	4.91e+01
BM	15(13)		**19(15)**	
$\dagger/\approx/\S$	3/17/9 (2/15/7)			

	$50D$			
	HSES		DQ-HSES	
	Mean	Std	Mean	Std
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**3.63e+01**$^{\approx}$	4.52e+01	4.17e+01	4.95e+01
F_5	1.36e+00§	9.94e-01	**7.60e-01**	8.57e-01
F_6	**6.43e-07**$^{\approx}$	7.97e-07	1.25e-06	2.16e-06
F_7	**5.51e+01**$^{\approx}$	6.10e-01	**5.51e+01**	6.75e-01
F_8	1.50e+00$^{\approx}$	1.24e+00	**1.46e+00**	1.11e+00
F_9	**0.00e+00**†	0.00e+00	1.61e+01	2.51e+01
F_{10}	**3.42e+02**$^{\approx}$	3.27e+02	4.86e+02	4.31e+02
F_{11}	**2.24e+01**$^{\approx}$	2.31e+00	2.25e+01	2.01e+00
F_{12}	1.39e+02$^{\approx}$	1.28e+02	**1.16e+02**	1.25e+02
F_{13}	**4.65e+01**$^{\approx}$	2.45e+01	4.76e+01	1.55e+01
F_{14}	1.67e+01$^{\approx}$	7.87e+00	**1.49e+01**	9.14e+00
F_{15}	**1.72e+01**$^{\approx}$	2.01e+00	1.76e+01	4.61e-01
F_{16}	**4.88e+02**†	1.87e+02	6.42e+02	2.60e+02
F_{17}	**2.55e+02**†	1.25e+02	5.05e+02	3.39e+02
F_{18}	**2.09e+01**$^{\approx}$	1.65e-01	**2.09e+01**	1.34e-01
F_{19}	**1.01e+01**$^{\approx}$	7.17e+00	1.02e+01	7.64e+00
F_{20}	5.38e+01§	5.93e+01	**4.19e+01**	5.44e+01
F_{21}	**2.04e+02**$^{\approx}$	1.10e+00	**2.04e+02**	8.95e-01
F_{22}	**1.00e+02**$^{\approx}$	0.00e+00	**1.00e+02**	1.48e-06
F_{23}	4.24e+02$^{\approx}$	8.04e+00	**4.23e+02**	7.72e+00
F_{24}	**4.88e+02**$^{\approx}$	2.76e+00	**4.88e+02**	2.85e+00
F_{25}	5.48e+02$^{\approx}$	2.18e+01	**5.45e+02**	2.97e+01
F_{26}	6.27e+02§	1.50e+02	**5.81e+02**	1.64e+02
F_{27}	5.89e+02§	1.63e+01	**5.69e+02**	2.69e+01
F_{28}	5.03e+02$^{\approx}$	9.55e+00	**5.01e+02**	1.36e+01
F_{29}	**4.86e+02**$^{\approx}$	1.51e+02	5.02e+02	1.74e+02
F_{30}	6.04e+05§	1.86e+04	**5.96e+05**	1.13e+04
BM	**18**(13)		**18(16)**	
$\dagger/\approx/\S$	3/21/5 (3/16/5)			

2) DQ-HSES vs. HSES on CEC 2014

To validate the capability of DQ-HSES for generalization, more experiments on the CEC 2014 test suite are conducted. Note that no function from CEC 2014 is used for training. Table VI shows the results obtained on the benchmark in 10D, 30D, and 50D over 51 runs. It is clear that for 10D problems, DQ-HSES performs significantly better than HSES on seven functions and worse on two functions. DQ-HSES performs significantly better than HSES on 10 functions and worse on two functions for 30D problems. For 50D problems, it performs significantly better on seven functions and worse on one function.

In general, DQ-HSES obtains greater values of *BM* than HSES. Thus we may conclude that the learned agent can generalize well on unseen optimization problems. The curves of optimization of some functions from the CEC 2014 test

TABLE V Results obtained by HSES and DQ-HSES on functions of CEC 2018 in 100D, averaged over 20 runs.

	$100D$			
	HSES		DQ-HSES	
	MEAN	STD	MEAN	STD
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	1.28e-08$^{\approx}$	3.58e-08	**0.00e+00**	0.00e+00
F_4	**6.15e+00**$^{\approx}$	2.19e+01	1.37e+01	4.15e+01
F_5	**3.73e+00**$^{\approx}$	1.43e+00	4.47e+00	2.17e+00
F_6	**8.47e-08**$^{\approx}$	2.03e-08	9.33e-08	2.36e-08
F_7	1.10e+02$^{\approx}$	1.59e+00	**1.09e+02**	1.04e+00
F_8	4.32e+00§	2.12e+00	**2.78e+00**	1.63e+00
F_9	2.78e+00$^{\approx}$	5.57e+00	**5.00e-01**	8.24e-01
F_{10}	1.29e+03$^{\approx}$	3.79e+02	**1.25e+03**	3.13e+02
F_{11}	**2.04e+01**$^{\approx}$	3.21e+01	3.82e+01	4.06e+01
F_{12}	**7.64e+02**$^{\approx}$	2.45e+02	8.60e+02	2.75e+02
F_{13}	**4.39e+01**$^{\approx}$	5.23e+00	4.73e+01	1.14e+01
F_{14}	2.04e+01$^{\approx}$	4.84e+00	**1.92e+01**	6.82e+00
F_{15}	1.05e+02$^{\approx}$	2.30e+01	**9.85e+01**	2.19e+01
F_{16}	**1.09e+03**$^{\approx}$	3.89e+02	1.55e+03	9.10e+02
F_{17}	7.11e+02$^{\approx}$	2.84e+02	**5.88e+02**	3.28e+02
F_{18}	4.94e+00$^{\approx}$	7.42e+00	**3.57e+00**	5.96e+00
F_{19}	2.30e+01§	1.59e+01	**1.42e+01**	1.99e+00
F_{20}	5.41e+02$^{\approx}$	2.52e+02	**4.93e+02**	2.08e+02
F_{21}	**2.21e+02**$^{\approx}$	4.11e+00	2.23e+02	3.33e+00
F_{22}	**1.00e+02**†	3.51e-07	4.37e+02	8.29e+02
F_{23}	5.45e+02$^{\approx}$	5.92e+00	**5.43e+02**	8.13e+00
F_{24}	**8.44e+02**$^{\approx}$	5.35e+00	**8.44e+02**	8.21e+00
F_{25}	7.47e+02$^{\approx}$	4.72e+01	**7.36e+02**	3.28e+01
F_{26}	2.35e+03$^{\approx}$	1.00e+02	**2.34e+03**	1.37e+02
F_{27}	**6.38e+02**$^{\approx}$	8.01e+00	6.39e+02	9.81e+00
F_{28}	**4.72e+02**$^{\approx}$	1.01e+02	5.08e+02	8.39e+01
F_{29}	**1.19e+03**$^{\approx}$	2.76e+02	1.34e+03	3.35e+02
F_{30}	2.67e+03$^{\approx}$	1.28e+02	**2.65e+03**	1.22e+02
BM	14(12)		**17(13)**	
$\dagger/\approx/\S$	1/26/2 (1/21/2)			

TABLE VI Results Obtained by HSES and DQ-HSES on the CEC 2014 Test Functions.

	$10D$			
	HSES		DQ-HSES	
	Mean	Std	Mean	Std
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_2	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**3.47e+01**$^{\approx}$	6.22e-03	**3.47e+01**	6.15e-03
F_5	1.72e+01$^{\approx}$	6.95e+00	**1.60e+01**	8.01e+00
F_6	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_7	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_8	**4.09e-01**$^{\approx}$	6.34e-01	5.07e-01	6.09e-01
F_9	6.82e-01$^{\approx}$	8.78e-01	**6.63e-01**	7.87e-01
F_{10}	**5.43e+01**$^{\approx}$	8.68e+01	7.25e+01	8.79e+01
F_{11}	6.05e+01$^{\approx}$	1.06e+02	**4.69e+01**	9.53e+01
F_{12}	3.29e-02$^{\approx}$	4.58e-02	**2.26e-02**	5.35e-02
F_{13}	1.09e-02§	4.69e-03	**9.30e-03**	4.47e-03
F_{14}	3.54e-01§	7.17e-02	**3.12e-01**	8.87e-02
F_{15}	9.02e-01§	2.13e-01	**7.53e-01**	3.11e-01
F_{16}	1.63e+00§	4.62e-01	**1.09e+00**	4.76e-01
F_{17}	8.53e+01$^{\approx}$	1.91e+02	**6.04e+01**	1.39e+02
F_{18}	**3.58e-01**$^{\approx}$	2.29e-01	4.48e-01	3.84e-01
F_{19}	6.93e-01§	5.17e-01	**4.87e-01**	4.47e-01
F_{20}	**6.74e-01**$^{\approx}$	1.93e-01	1.12e+00	1.59e+00
F_{21}	**1.59e+01**†	5.29e+01	2.39e+01	6.41e+01
F_{22}	**1.99e+01**$^{\approx}$	3.00e+01	2.01e+01	3.42e+01
F_{23}	**3.29e+02**$^{\approx}$	0.00e+00	**3.29e+02**	0.00e+00
F_{24}	1.10e+02§	1.73e+01	**1.05e+02**	4.82e+00
F_{25}	1.96e+02$^{\approx}$	1.00e+01	**1.94e+02**	1.27e+01
F_{26}	**1.00e+02**$^{\approx}$	5.26e-03	**1.00e+02**	4.88e-03
F_{27}	**2.61e+02**$^{\approx}$	1.13e+02	2.72e+02	1.00e+02
F_{28}	**3.60e+02**$^{\approx}$	1.26e+02	3.62e+02	1.28e+02
F_{29}	2.20e+02§	1.31e+01	**2.17e+02**	1.99e+01
F_{30}	**5.37e+02**†	4.18e+01	5.77e+02	8.71e+01
BM	17		**21**	
$\dagger/\approx/\S$	2/21/7			

	$30D$			
	HSES		DQ-HSES	
	Mean	Std	Mean	Std
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_2	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_5	1.99e+01§	3.55e-04	**1.99e+01**	2.61e-02
F_6	9.33e-01$^{\approx}$	1.31e+00	**8.93e-01**	8.58e-01
F_7	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_8	7.60e+00§	3.29e+00	**5.22e+00**	1.60e+00
F_9	7.60e+00§	2.93e+00	**6.35e+00**	2.41e+00
F_{10}	4.02e+02§	2.34e+02	**2.08e+02**	1.89e+02
F_{11}	8.16e+02§	3.64e+02	**6.75e+02**	2.64e+02
F_{12}	**1.24e-02**$^{\approx}$	1.34e-02	1.68e-02	2.14e-02
F_{13}	5.06e-02§	1.33e-02	**4.12e-02**	1.07e-02
F_{14}	3.40e-01$^{\approx}$	7.09e-02	**3.39e-01**	5.58e-02
F_{15}	**2.75e+00**†	4.98e-01	3.89e+00	8.39e-01
F_{16}	1.01e+01§	9.29e-01	**9.09e+00**	9.36e-01
F_{17}	4.04e+01$^{\approx}$	7.52e+01	**3.13e+01**	7.33e+01
F_{18}	**5.47e+00**$^{\approx}$	3.66e+00	9.30e+00	1.04e+01
F_{19}	2.98e+00§	8.07e-01	**2.44e+00**	5.89e-01
F_{20}	2.34e+00§	1.75e+00	**1.69e+00**	6.51e-01
F_{21}	**1.02e+01**$^{\approx}$	3.17e+01	4.73e+01	1.31e+02
F_{22}	**1.69e+02**$^{\approx}$	7.56e+01	1.84e+02	9.88e+01
F_{23}	**3.15e+02**$^{\approx}$	4.29e-06	**3.15e+02**	5.74e-06
F_{24}	2.24e+02§	7.20e-01	**2.23e+02**	6.22e-01
F_{25}	**2.08e+02**$^{\approx}$	1.12e+00	**2.08e+02**	1.67e+00
F_{26}	**1.30e+02**$^{\approx}$	4.14e+01	1.38e+02	4.49e+01
F_{27}	**3.04e+02**$^{\approx}$	1.18e+01	3.07e+02	2.80e+01
F_{28}	8.93e+02$^{\approx}$	2.56e+01	**8.87e+02**	2.74e+01
F_{29}	**2.69e+02**†	7.67e+01	6.32e+02	1.79e+02
F_{30}	**1.60e+03**$^{\approx}$	4.27e+02	1.72e+03	3.98e+02
BM	16		**21**	
$\dagger/\approx/\S$	2/18/10			

	$50D$			
	HSES		DQ-HSES	
	Mean	Std	Mean	Std
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_2	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_5	2.00e+01§	4.98e-05	**1.99e+01**	1.10e-04
F_6	**2.78e-05**$^{\approx}$	7.41e-05	4.00e-05	1.23e-04
F_7	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_8	**1.44e+00**$^{\approx}$	1.08e+00	**1.44e+00**	1.02e+00
F_9	8.66e-01§	8.80e-01	**8.18e-01**	8.35e-01
F_{10}	2.66e+02$^{\approx}$	2.21e+02	**2.11e+02**	1.59e+02
F_{11}	5.19e+02$^{\approx}$	2.49e+02	**4.77e+02**	2.81e+02
F_{12}	2.68e-02$^{\approx}$	4.14e-02	**1.42e-02**	1.46e-02
F_{13}	7.02e-02$^{\approx}$	9.79e-03	**6.99e-02**	1.26e-02
F_{14}	**3.83e-01**$^{\approx}$	5.01e-02	3.89e-01	4.59e-02
F_{15}	**5.30e+00**†	1.12e+00	7.56e+00	2.51e+00
F_{16}	1.90e+01§	1.00e+00	**1.75e+01**	9.78e-01
F_{17}	9.81e+02§	1.25e+03	**1.94e+02**	6.19e+02
F_{18}	8.51e-01§	4.80e-01	**7.71e-01**	5.15e-01
F_{19}	7.30e+00$^{\approx}$	1.63e+00	**7.28e+00**	1.61e+00
F_{20}	**2.39e+00**$^{\approx}$	6.55e-01	2.43e+00	6.05e-01
F_{21}	1.36e+03$^{\approx}$	4.50e+02	**1.24e+03**	3.68e+02
F_{22}	1.68e+02§	5.25e+01	**1.58e+02**	5.66e+01
F_{23}	**3.44e+02**$^{\approx}$	2.64e-05	**3.44e+02**	2.34e-05
F_{24}	**2.67e+02**$^{\approx}$	1.51e+00	**2.67e+02**	1.45e+00
F_{25}	**2.16e+02**$^{\approx}$	1.21e+00	**2.16e+02**	3.25e+00
F_{26}	**1.11e+02**$^{\approx}$	2.35e+01	1.17e+02	3.18e+01
F_{27}	**3.01e+02**$^{\approx}$	8.48e+00	**3.01e+02**	8.41e+00
F_{28}	**1.22e+03**$^{\approx}$	4.35e+01	1.23e+03	6.44e+01
F_{29}	5.00e+02§	1.75e+01	**4.77e+02**	3.25e+01
F_{30}	8.87e+03$^{\approx}$	4.08e+02	**8.80e+03**	3.25e+02
BM	16		**24**	
$\dagger/\approx/\S$	1/22/7			

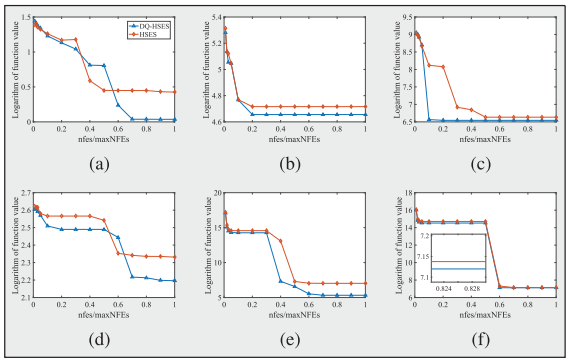

FIGURE 4 The curves of optimization of six functions on the CEC 2014 test suite. (a) F_{16} in 10D, (b) F_{24} in 10D, (c) F_{11} in 30D, (d) F_{16} in 30D, (e) F_{17} in 50D, and (f) F_{21} in 50D.

are presented to show the process of optimization of DQ-HSES in comparison with that of HSES in Figure 4.

For each function, HSES and DQ-HSES are run 11 times and the logarithm of the mean of the function values obtained during the search process are plotted. The figures show that DQ-HSES performs significantly better than HSES, and usually requires fewer fitness evaluations (nfes) to reach a better solution than it.

TABLE VII Results on 10D test functions of CEC 2018.

	cDE		CoBiDE		jSO		DQ-HSES	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_5	4.62e+00§	1.84e+00	3.57e+00§	1.53e+00	1.75e+00§	7.60e-01	**8.15e-01**	8.79e-01
F_6	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_7	1.55e+01§	2.74e+00	1.34e+01§	2.00e+00	1.17e+01§	6.06e-01	**1.10e+01**	5.61e-01
F_8	5.73e+00§	2.11e+00	3.76e+00§	1.71e+00	1.95e+00§	7.43e-01	**7.01e-01**	7.96e-01
F_9	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_{10}	1.87e+02§	1.10e+02	9.17e+01§	9.56e+01	**3.58e+01**$^{\approx}$	5.54e+01	8.88e+01	1.30e+02
F_{11}	5.54e-01†	9.14e-01	9.75e-02†	2.98e-01	**0.00e+00**†	0.00e+00	7.61e-01	3.50e+00
F_{12}	1.39e+02§	1.60e+02	1.67e-01†	1.31e-01	**2.66e+00**†	1.67e+01	2.26e+01	5.52e+01
F_{13}	5.05e+00§	2.97e+00	**1.07e+00**†	1.83e+00	2.96e+00†	2.35e+00	3.17e+00	2.58e+00
F_{14}	6.48e-01†	9.73e-01	**0.00e+00**†	0.00e+00	5.85e-02†	2.36e-01	6.00e+00	1.20e+01
F_{15}	3.09e-01†	4.85e-01	**9.31e-03**†	3.52e-02	2.20e-01†	2.00e-01	8.17e-01	1.73e+00
F_{16}	2.96e+00†	1.68e+01	**2.24e-01**†	1.52e-01	5.68e-01†	2.64e-01	4.41e+00	2.03e+01
F_{17}	1.32e+00†	8.12e+00	1.57e+00†	6.18e-01	**5.02e-01**†	3.48e-01	1.18e+01	9.95e+00
F_{18}	1.13e+00§	3.92e+00	**6.37e-03**†	2.77e-02	3.08e-01†	1.95e-01	7.02e-01	1.61e+00
F_{19}	**4.19e-03**†	8.08e-03	8.99e-03†	1.33e-02	1.07e-02†	1.25e-02	4.31e-01	8.89e-01
F_{20}	9.90e-02†	1.97e-01	**0.00e+00**†	0.00e+00	3.42e-01†	1.28e-01	6.98e+00	9.06e+00
F_{21}	1.79e+02†	4.90e+01	1.37e+02†	5.15e+01	**1.32e+02**†	4.83e+01	1.98e+02	1.76e+01
F_{22}	9.28e+01†	2.65e+01	**7.86e+01**†	4.16e+01	1.00e+02$^{\approx}$	0.00e+00	1.00e+02	0.00e+00
F_{23}	3.07e+02§	2.56e+00	3.04e+02§	1.58e+00	3.01e+02§	1.58e+00	**2.99e+02**	2.13e+01
F_{24}	3.26e+02§	4.62e+01	**2.74e+02**†	1.02e+02	2.96e+02$^{\approx}$	7.93e+01	3.21e+02	3.90e+01
F_{25}	4.17e+02†	2.34e+01	**3.99e+02**†	9.14e+00	4.05e+02†	1.74e+01	4.46e+02	1.18e+00
F_{26}	3.19e+02§	1.30e+02	**3.00e+02**$^{\approx}$	0.00e+00	**3.00e+02**$^{\approx}$	0.00e+00	**3.00e+02**	8.82e+00
F_{27}	3.89e+02†	2.01e+00	**3.88e+02**†	9.02e-01	3.89e+02†	2.25e-01	3.97e+02	1.93e+00
F_{28}	4.04e+02†	1.39e+02	**3.05e+02**†	3.97e+01	3.39e+02†	9.65e+01	5.92e+02	3.80e+01
F_{29}	2.36e+02†	6.43e+00	**2.29e+02**†	2.77e+00	2.34e+02†	2.95e+00	2.64e+02	1.10e+01
F_{30}	8.06e+04§	2.45e+05	3.95e+02†	3.50e+00	**3.94e+02**†	4.49e-02	4.13e+02	2.34e+01
BM	6(5)		**18(14)**		12(10)		10(9)	
†/ ≈ /§	13/5/11(11/1/9)		18/6/5(15/5/4)		16/9/4(13/7/4)			

3) DQ-HSES vs. the Baselines

Tables VII to X shows the results obtained by cDE, CoBiDE, jSO, and DQ-HSES on the CEC 2018 test functions in 10D, 30D, 50D, and 100D. cDE, CoBiDE, and jSO are implemented 51 times. The values of *BM* and the results of the hypothesis test on the functions excluding the training functions are presented in the parentheses in the tables. They show the following:

- ❏ On 10D functions, DQ-HSES performs better than cDE on 11 functions and worse on 13 functions, better than CoBiDE on five functions and worse on 18, and better than jSO on four functions and worse on 16 functions.
- ❏ On 30D functions, DQ-HSES performs better than cDE on 23 functions and worse on two functions, better than CoBiDE on 15 functions and worse on seven functions, and better than jSO on 10 functions and worse on nine functions.
- ❏ On 50D functions, DQ-HSES performs better than cDE on 26 functions and worse on three functions, better than CoBiDE on 26 functions and worse on three functions, and better than jSO on 16 functions and worse on seven functions.
- ❏ On 100D functions, DQ-HSES performs better than cDE on 28 functions and is not worse than it on any function, is better than CoBiDE on 29 functions and not worse than it on any function, and is better than jSO on 22 functions and worse on two functions.
- ❏ DQ-HSES delivers the best performance on all functions excluding the training functions, on 30D, 50D, and 100D functions.

In general, except for the test functions in 10D, DQ-HSES obtains the largest *BM* values.

TABLE VIII Results on 30D test functions of CEC 2018.

	cDE		CoBiDE		jSO		DQ-HSES	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
F_1	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_3	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	**0.00e+00**	0.00e+00
F_4	5.68e+01§	1.08e+01	4.20e+01§	2.82e+01	5.86e+01§	7.77e-01	**4.11e+00**	9.37e+00
F_5	5.06e+01§	6.17e+00	3.99e+01§	9.75e+00	8.55e+00§	2.09e+00	**6.95e+00**	2.11e+00
F_6	**0.00e+00**$^{\approx}$	0.00e+00	3.91e-08§	4.48e-08	**0.00e+00**$^{\approx}$	2.71e-08	**0.00e+00**	0.00e+00
F_7	9.20e+01§	6.45e+00	7.11e+01§	1.01e+01	3.89e+01§	1.45e+00	**3.53e+01**	1.31e+00
F_8	5.81e+01§	9.17e+00	3.91e+01§	1.07e+01	9.09e+00§	1.83e+00	**6.48e+00**	2.06e+00
F_9	7.53e-01§	1.44e+00	**0.00e+00**†	0.00e+00	**0.00e+00**$^{\approx}$	0.00e+00	2.63e-02	2.22e-01
F_{10}	2.32e+03§	2.87e+02	1.82e+03§	4.56e+02	1.52e+03§	2.77e+02	**8.43e+02**	3.36e+02
F_{11}	2.02e+01§	1.19e+01	1.62e+01§	9.81e+00	**3.03e+00**†	2.64e+00	1.61e+01	2.29e+01
F_{12}	2.02e+04§	1.44e+04	2.83e+03§	5.20e+03	1.70e+02§	1.01e+02	**2.38e+01**	9.60e+01
F_{13}	5.27e+01§	2.30e+01	2.50e+01$^{\approx}$	8.67e+00	**1.48e+01**†	4.83e+00	3.01e+01	1.53e+01
F_{14}	3.41e+01§	9.23e+00	**1.08e+01**†	4.70e+00	2.18e+01§	1.24e+00	1.55e+01	9.35e+00
F_{15}	1.86e+01§	4.81e+00	6.85e+00$^{\approx}$	2.79e+00	**1.08e+00**†	6.91e-01	1.16e+01	1.17e+01
F_{16}	4.87e+02§	1.36e+02	3.78e+02§	1.47e+02	**7.89e+01**†	8.47e+01	2.93e+02	1.81e+02
F_{17}	9.68e+01§	2.90e+01	4.38e+01†	3.29e+01	**3.29e+01**†	8.07e+00	6.33e+01	9.21e+01
F_{18}	3.16e+01§	5.46e+00	**1.85e+01**†	8.65e+00	2.04e+01$^{\approx}$	2.87e+00	1.89e+01	6.85e+00
F_{19}	1.53e+01§	2.31e+00	4.77e+00§	1.44e+00	4.50e+00§	1.73e+00	**3.78e+00**	2.04e+00
F_{20}	1.00e+02†	5.48e+01	4.31e+01†	5.79e+01	**2.93e+01**†	5.85e+00	1.58e+02	5.88e+01
F_{21}	2.57e+02§	7.19e+00	2.42e+02§	9.26e+00	2.09e+02§	1.95e+00	**2.07e+02**	3.05e+00
F_{22}	2.07e+02§	5.37e+02	**1.00e+02**$^{\approx}$	0.00e+00	**1.00e+02**$^{\approx}$	0.00e+00	**1.00e+02**	0.00e+00
F_{23}	3.98e+02§	6.56e+00	3.88e+02§	8.96e+00	**3.50e+02**$^{\approx}$	3.29e+00	3.52e+02	8.79e+00
F_{24}	4.80e+02§	8.44e+00	4.64e+02§	1.18e+01	4.26e+02§	2.46e+00	**4.19e+02**	5.30e+00
F_{25}	**3.86e+02**$^{\approx}$	4.74e-01	**3.86e+02**$^{\approx}$	4.72e-01	**3.86e+02**$^{\approx}$	7.72e-03	**3.86e+02**	2.25e-02
F_{26}	1.55e+03§	2.03e+02	1.36e+03§	2.90e+02	9.20e+02$^{\approx}$	4.29e+01	**9.03e+02**	1.37e+02
F_{27}	**4.96e+02**†	1.24e+01	**4.96e+02**†	1.04e+01	4.97e+02†	7.00e+00	5.25e+02	9.05e+00
F_{28}	3.23e+02§	4.63e+01	3.27e+02§	4.77e+01	**3.08e+02**†	3.02e+01	3.15e+02	3.73e+01
F_{29}	5.19e+02§	4.89e+01	**4.29e+02**†	4.68e+01	4.33e+02$^{\approx}$	1.36e+01	4.62e+02	7.25e+01
F_{30}	2.12e+03§	1.45e+02	2.05e+03$^{\approx}$	8.41e+01	**1.97e+03**†	1.89e+01	2.06e+03	4.91e+01
BM	5(4)		9(6)		**15(13)**		**15(13)**	
†/ ≈ /§	2/4/23(2/3/19)		7/7/15(5/5/14)		9/10/10(8/7/9)			

The average performance score (APS) [52] [53], is also used as a metric to compare the performances of the algorithms. Considering the comparison of M algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_M$ on K functions $F_1, \ldots, F_K$, if algorithm $\mathcal{A}_j$ outperforms $\mathcal{A}_i$ on the k-th function F_k with statistical significance, $\delta^k_{i,j}$ is set to be 1, otherwise 0. The performance score of algorithm $\mathcal{A}_i$ on function F_k is defined as: $PS^k(\mathcal{A}_i) = \sum_{j \in \{1,\ldots,M\} \setminus \{i\}} \delta^k_{i,j}$. $PS^k(\mathcal{A}_i)$ represents the number of algorithms that outperform $\mathcal{A}_i$ on function F_k. The APS can then be computed as $APS(\mathcal{A}_i) = \frac{1}{K}\sum_{k=1}^{K} PS^k(\mathcal{A}_i)$. A smaller APS represents better performance.

Table XI shows the APS values obtained by cDE, CoBiDE, jSO, HSES, Q-HSES [8] (the conference version of this paper), and DQ-HSES

TABLE IX Results on 50D test functions of CEC 2018.

	cDE		CoBiDE		jSO		DQ-HSES	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
F_1	6.38e-03§	1.68e-02	1.11e+04§	4.31e+03	**0.00e+00**≈	0.00e+00	**0.00e+00**	0.00e+00
F_3	6.72e-08§	1.05e-07	6.56e+00§	2.73e+00	**0.00e+00**≈	0.00e+00	**0.00e+00**	0.00e+00
F_4	5.71e+01§	4.47e+01	7.10e+01§	3.07e+01	5.62e+01§	4.87e+01	**4.17e+01**	4.95e+01
F_5	1.57e+02§	1.39e+01	2.54e+02§	1.38e+01	1.64e+01§	3.46e+00	**7.60e-01**	8.57e-01
F_6	2.14e-03§	3.65e-03	1.98e+00§	2.60e-01	**1.09e-06**≈	2.62e-06	1.25e-06	2.16e-06
F_7	2.32e+02§	1.55e+01	3.44e+02§	1.71e+01	6.64e+01§	3.47e+00	**5.51e+01**	6.75e-01
F_8	1.57e+02§	1.34e+01	2.58e+02§	1.29e+01	1.69e+01§	3.13e+00	**1.46e+00**	1.11e+00
F_9	2.82e+00†	4.09e+00	2.27e+02§	1.01e+02	**0.00e+00**†	0.00e+00	1.61e+01	2.51e+01
F_{10}	6.26e+03§	3.41e+02	8.54e+03§	3.75e+02	3.13e+03§	3.67e+02	**4.86e+02**	4.31e+02
F_{11}	2.12e+02§	6.41e+01	1.07e+02§	7.78e+00	2.79e+01§	3.32e+00	**2.25e+01**	2.01e+00
F_{12}	6.12e+04§	5.62e+04	6.27e+03§	1.10e+03	1.68e+03§	5.22e+02	**1.16e+02**	1.25e+02
F_{13}	6.58e+03§	9.15e+03	2.44e+02§	2.00e+01	**3.05e+01**†	2.12e+01	4.76e+01	1.55e+01
F_{14}	3.47e+02§	8.84e+01	9.18e+01§	7.09e+00	2.49e+01§	1.87e+00	**1.49e+01**	9.14e+00
F_{15}	3.87e+02§	2.07e+02	8.52e+01§	8.08e+00	2.38e+01§	2.48e+00	**1.76e+01**	4.61e-01
F_{16}	1.44e+03§	2.25e+02	1.34e+03§	1.60e+02	**4.50e+02**†	1.37e+02	6.42e+02	2.60e+02
F_{17}	9.91e+02§	2.16e+02	8.75e+02§	1.14e+02	**2.82e+02**†	8.61e+01	5.05e+02	3.39e+02
F_{18}	5.77e+03§	8.02e+03	5.67e+01§	4.32e+00	2.42e+01§	2.01e+00	**2.09e+01**	1.34e-01
F_{19}	1.70e+02§	4.30e+01	4.68e+01§	4.33e+00	1.41e+01§	2.26e+00	**1.02e+01**	7.64e+00
F_{20}	8.70e+02§	1.80e+02	6.37e+02§	1.24e+02	1.40e+02§	7.73e+01	**4.19e+01**	5.44e+01
F_{21}	3.63e+02§	1.44e+01	4.57e+02§	1.30e+01	2.19e+02§	3.76e+00	**2.04e+02**	8.95e-01
F_{22}	5.47e+03§	2.87e+03	5.51e+03§	4.48e+03	1.48e+03≈	1.75e+03	**1.00e+02**	1.48e-06
F_{23}	5.89e+02§	1.58e+01	6.83e+02§	1.32e+01	4.30e+02§	6.23e+00	**4.23e+02**	7.72e+00
F_{24}	6.55e+02§	1.93e+01	7.40e+02§	2.01e+01	5.07e+02§	4.12e+00	**4.88e+02**	2.85e+00
F_{25}	5.15e+02†	4.30e+01	4.82e+02†	5.91e+00	**4.80e+02**†	2.79e+00	5.45e+02	2.97e+01
F_{26}	2.68e+03§	1.77e+02	3.57e+03§	1.50e+02	1.12e+03§	5.61e+01	**5.81e+02**	1.64e+02
F_{27}	6.37e+02§	6.21e+01	5.49e+02†	1.56e+01	**5.11e+02**†	1.10e+01	5.69e+02	2.69e+01
F_{28}	4.84e+02†	2.43e+01	**4.58e+02**†	5.73e-02	4.59e+02†	6.83e+00	5.01e+02	1.36e+01
F_{29}	6.84e+02§	1.67e+02	8.51e+02§	9.93e+01	**3.62e+02**≈	1.31e+01	5.02e+02	1.74e+02
F_{30}	7.76e+05§	1.50e+05	6.28e+05§	1.30e+04	6.01e+05≈	2.98e+04	**5.96e+05**	1.13e+04
BM	0(0)		1(0)		10(8)		**20(16)**	
†/ ≈ /§	3/0/26(3/0/21)		3/0/26(3/0/21)		7/6/16(7/4/13)			

TABLE X Results on 100D test functions of CEC 2018.

	cDE		CoBiDE		jSO		DQ-HSES	
	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
F_1	1.93e+03§	2.99e+03	2.46e+07§	5.12e+06	**0.00e+00**≈	0.00e+00	**0.00e+00**	0.00e+00
F_3	6.77e+01§	6.37e+01	1.31e+04§	2.51e+03	2.39e-06§	2.72e-06	**0.00e+00**	0.00e+00
F_4	2.14e+02§	3.15e+01	3.05e+02§	2.33e+01	1.89e+02§	2.89e+01	**9.71e+00**	3.66e+01
F_5	5.53e+02§	3.34e+01	7.74e+02§	3.32e+01	4.39e+01§	5.60e+00	**4.03e+00**	2.17e+00
F_6	4.06e-01§	5.09e-01	1.64e+01§	1.74e+00	2.02e-04§	6.19e-04	**9.03e-08**	2.41e-08
F_7	7.36e+02§	4.11e+01	9.04e+02§	3.86e+01	1.44e+02§	6.70e+00	**1.09e+02**	1.18e+00
F_8	5.54e+02§	3.55e+01	7.72e+02§	2.30e+01	4.21e+01§	5.52e+00	**3.12e+00**	1.84e+00
F_9	1.89e+03§	1.21e+03	6.08e+03§	1.64e+03	**4.59e-02**≈	1.14e-01	3.35e-01	6.81e-01
F_{10}	1.80e+04§	4.42e+02	2.22e+04§	5.39e+02	9.70e+03§	6.81e+02	**1.34e+03**	3.97e+02
F_{11}	1.18e+03§	2.39e+02	7.01e+02§	4.18e+01	1.13e+02§	4.32e+01	**3.04e+01**	4.01e+01
F_{12}	5.35e+05§	3.20e+05	5.37e+06§	1.49e+06	1.84e+04§	8.35e+03	**8.51e+02**	2.76e+02
F_{13}	4.92e+03§	5.63e+03	4.79e+03§	6.90e+02	1.44e+02§	3.80e+01	**4.57e+01**	9.86e+00
F_{14}	4.69e+03§	7.02e+03	3.53e+02§	1.81e+01	6.43e+01§	1.08e+01	**1.96e+01**	6.32e+00
F_{15}	3.65e+03§	5.62e+03	6.77e+02§	4.02e+01	1.62e+02§	3.80e+01	**1.01e+02**	1.98e+01
F_{16}	4.61e+03§	3.03e+02	5.16e+03§	3.11e+02	1.85e+03§	3.48e+02	**1.28e+03**	8.44e+02
F_{17}	3.55e+03§	2.52e+02	3.42e+03§	1.88e+02	1.27e+03§	2.38e+02	**5.41e+02**	3.01e+02
F_{18}	4.97e+04§	1.77e+04	5.32e+02§	6.24e+01	1.67e+02§	3.64e+01	**4.00e+00**	6.74e+00
F_{19}	4.91e+03§	7.31e+03	3.01e+02§	2.13e+01	1.04e+02§	2.00e+01	**2.03e+01**	2.28e+01
F_{20}	3.43e+03§	2.18e+02	3.18e+03§	2.00e+02	1.37e+03§	2.42e+02	**5.14e+02**	2.44e+02
F_{21}	7.81e+02§	3.91e+01	1.00e+03§	2.74e+01	2.63e+02§	6.42e+00	**2.22e+02**	3.18e+00
F_{22}	1.90e+04§	5.78e+02	2.34e+04§	6.05e+02	1.02e+04§	2.18e+03	**2.92e+02**	6.76e+02
F_{23}	8.67e+02§	2.46e+01	1.18e+03§	1.80e+01	5.71e+02§	1.07e+01	**5.45e+02**	7.67e+00
F_{24}	1.38e+03§	3.72e+01	1.62e+03§	2.64e+01	9.02e+02§	7.89e+00	**8.44e+02**	6.92e+00
F_{25}	7.58e+02≈	5.41e+01	8.71e+02§	2.72e+01	**7.36e+02**≈	3.53e+01	7.41e+02	3.91e+01
F_{26}	8.56e+03§	4.62e+02	1.07e+04§	3.17e+02	3.26e+03§	8.01e+01	**2.35e+03**	1.21e+02
F_{27}	8.28e+02§	9.80e+01	8.46e+02§	4.38e+01	**5.85e+02**†	2.16e+01	6.39e+02	9.95e+00
F_{28}	5.54e+02§	2.74e+01	6.75e+02§	2.15e+01	5.26e+02≈	2.73e+01	**4.80e+02**	9.88e+01
F_{29}	3.40e+03§	3.31e+02	4.05e+03§	2.45e+02	**1.25e+03**≈	1.91e+02	1.28e+03	2.93e+02
F_{30}	7.35e+03§	3.53e+03	1.48e+04§	2.00e+03	**2.32e+03**†	1.18e+02	2.68e+03	1.26e+02
BM	0(0)		0(0)		6(4)		**24(20)**	
†/ ≈ /§	0/1/28(0/1/23)		0/0/29(0/0/24)		2/5/22(2/3/19)			

TABLE XI The APS values obtained by cDE, CoBiDE, jSO, HSES, Q-HSES, and DQ-HSES.

DIM.	cDE	CoBiDE	jSO	HSES	Q-HSES	DQ-HSES
10D	2.250	**0.875**	0.958	1.792	1.750	1.667
30D	3.958	2.750	1.375	1.333	0.833	**0.792**
50D	4.167	4.083	1.583	0.875	**0.667**	0.708
100D	4.083	4.792	2.333	0.375	0.208	**0.167**
Avg.	3.615	3.125	1.562	1.094	0.865	**0.833**
W. Avg.	3.779	3.812	1.779	0.858	0.625	**0.604**

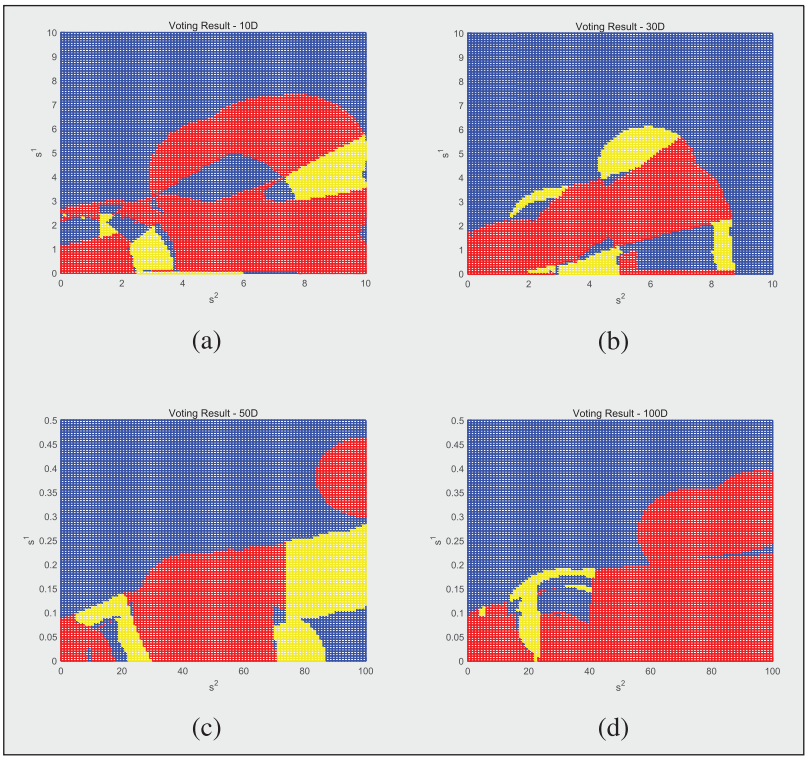

FIGURE 5 The visualization of the results of voting for 10D (a), 30D (b), 50D (c), and 100D (d) functions. Red region means not switching to CMA-ES; Yellow region means switching to CMA-ES; Blue region means selecting a random action.

TABLE XII The average CPU times of HSES and DQ-HSES (in seconds).

	10D	30D	50D	100D
HSES	4.06	31.32	99.83	608.79
DQ-HSES	5.77	32.99	98.46	623.31

on the CEC 2018 test suite excluding the training functions. It is clear that CoBiDE obtains the minimum APS on 10D functions, and DQ-HSES obtains the minimum APS on 30D and 100D functions. The last two rows of the table show the average APS and the weighted-average APS on the functions over the four sets of dimensions with weights $0.1, 0.2, 0.3$, and 0.4 for 10D, 30D, 50D, and 100D functions, respectively, as used in the CEC 2017 competition [27]. It is clear that DQ-HSES achieves the minimum average and weighted-average APS values. Moreover, Q-HSES has a higher score than HSES but lower than DQ-HSES, which implies that using Q-learning can improve the performance of HSES while DQN can deliver even better results. Thus we may conclude that DQ-HSES is competitive with the state-of-the-art EAs.

4) Visualization of DQNs

Figure 5 shows the results of voting of DQNs on four dimensions. It shows the following: 1) The results of voting are different in different dimensions. 2) For each dimension, the result is too irregular to be manually designed. This verifies the advantage of using a trained agent.

5) Time Complexity Analysis

Table XII shows the average running times of HSES and DQ-HSES on $F_1 - F_{30}$ for each set of dimensional functions. It is clear that DQ-HSES generally requires slightly longer than HSES except on 50D problems. This shows that the switcher agent can better allocate computational resources without sacrificing too much time.

VIII. CONCLUSION

In this article, we propose a framework for the adaptive control of the structural parameters of sequential hybrid EAs based on Q-learning. By way of case studies, we apply the framework to adaptively control the switching time of the strategy to set the population size in LSHADE as well as that of HSES from the univariate sampling phase to the CMA-ES phase. The experimental results on the CEC 2018 test suite show that the learned algorithm based on LSHADE, Q-LSHADE, outperforms LSHADE. The learned algorithm based on HSES (the winner of the CEC 2018 competition), DQ-HSES, also outperforms it as well as some well-known EAs. The results verify the effectiveness of the proposed framework and indicate that learned knowledge is useful for solving new optimization problems. In the future work, we intend to study how to adaptively design EAs instead of only controlling their algorithmic parameters. Moreover, we will examine ways to learn local search algorithms to solve combinatorial optimization problems.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants 11991023, 62076197, and 61903294, in part by the Major Project of National Science Foundation of China under Grant U1811461, in part by the Key Project of National Science Foundation of China under Grant 11690011, and in part by the Project of National Science Foundation of China under Grant 61721002.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, NY, USA:Oxford Univ. Press, 1996.

[2] D. E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.

[3] K. V. Price, "An introduction to differential evolution," in *New Ideas in Optimization*.U.K., GBR: McGraw-Hill Ltd., 1999, pp. 79–108.
[4] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[5] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, 1995, pp. 39–43.
[6] N. Hansen, S. D. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, Mar. 2003.
[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer, 1996.
[8] H. Zhang, J. Sun, and Z. Xu, "Adaptive structural hyper-parameter configuration by Q-learning," in *Proc. IEEE Congr. Evol. Comput.*, 2020, pp. 1–8.
[9] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020.
[10] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optimal external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–458, Oct. 2009.
[11] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 71–78.
[12] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 1658–1665.
[13] A. P. Piotrowski, "Review of differential evolution population size," *Swarm Evol. Comput.*, vol. 32, pp. 1–24, 2017.
[14] M. Andrychowicz et al., "Learning to learn by gradient descent by gradient descent," in *Proc. Conf. Workshop Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.
[15] K. Li and J. Malik, "Learning to optimize," in *Proc. Int. Conf. Learn. Representations*, 2017.
[16] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[17] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," in *Proc. IEEE Congr. Evol. Comput.*, 2018, pp. 1–7.
[18] P. I. Frazier, "A tutorial on Bayesian optimization," 2018, *arXiv:1807.02811*.
[19] F. Hutter, H. H. Hoos, and K. Leytonbrown, "Sequential model-based optimization for general algorithm configuration," in *Proc. Int. Conf. Learn. Intell. Optim.*, 2011, pp. 507–523.
[20] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stútzle, "ParamILS: An automatic algorithm configuration framework," *J. Artif. Intell. Res.*, vol. 36, pp. 267–306, 2009.
[21] I. Roman, J. Ceberio, A. Mendiburu, and J. A. Lozano, "Bayesian optimization for parameter tuning in evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 4839–4845.
[22] C. Huang, B. Yuan, Y. Li, and X. Yao, "Automatic parameter tuning using bayesian optimization method," in *Proc. IEEE Congr. Evol. Comput.*, 2019, pp. 2090–2097.
[23] K. Tang, J. Wang, X. Li, and X. Yao, "A scalable approach to capacitated arc routing problems based on hierarchical decomposition," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3928–3940, Nov. 2017.
[24] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[25] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1110–1116.
[26] J. Brest, M. S. Maucec, and B. Boskovic, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 1188–1195.
[27] J. Brest, M. S. Maucec, and B. BOskovic, "Single objective realparameter optimization: Algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 1311–1318.
[28] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst. Man, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.
[29] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.
[30] Y. Wang, H. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Appl. Soft Comput.*, vol. 18, pp. 232–247, 2014.
[31] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, pp. 673–686, 2006.
[32] V. Tirronen and F. Neri, "Differential evolution with fitness diversity self-adaptation," in *Nature Inspired Algorithms for Optimisation*. Berlin, Germany: Springer, 2009, pp. 199–234.
[33] H. Wang, S. Rahnamayan, and Z. Wu, "Adaptive differential evolution with variable population size for solving high-dimensional problems," in *Proc. IEEE Congr. Evol. Comput.*, 2011, pp. 2626–2632.
[34] G. Wu, R. Mallipeddi, P. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, pp. 329–345, 2016.
[35] W. Zhu, Y. Tang, J. an Fang, and W. Zhang, "Adaptive population tuning scheme for differential evolution," *Inf. Sci.*, vol. 223, pp. 164–191, 2013.
[36] A. Ajith, G. Crina, and I. Hisao, *Hybrid Evolutionary Algorithms*, vol. 75. Berlin, Germany:Springer, 2007.
[37] T. A. A. Victoire and A. E. Jeyakumar, "A modified hybrid EP-SQP approach for dynamic dispatch with valve-point effect," *Int. J. Elect. Power Energy Syst.*, vol. 27, no. 8, pp. 594–601, 2005.
[38] P. Attaviriyanupap, H. Kita, E. Tanaka, and J. Hasegawa, "A hybrid and for dynamic economic dispatch with nonsmooth fuel cost function," *IEEE Trans. Power Syst.*, vol. 17, no. 2, pp. 411–416, May 2002.
[39] H. Wang, Y. Fu, M. Huang, G. Huang, and J. Wang, "A hybrid evolutionary algorithm with adaptive multi-population strategy for multi-objective optimization problems," *Soft Comput.*, vol. 21, pp. 5975–5987, 2017.
[40] J. Tvrdík, "Competitive differential evolution," in *Proc. MENDEL*, 2006, pp. 7–12.
[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[42] S. Wang, J. Sun, and Z. Xu, "HyperAdam: A learnable task-adaptive adam for network training," in *Proc. Nat. Conf. Artif. Intell.*, 2019, pp. 5297–5304.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[44] Y. Chen et al., "Learning to learn without gradient descent by gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 748–756.
[45] M. Sharma, A. Komninos, M. Lopezibanez, and D. Kazakov, "Deep reinforcement learning based parameter control in differential evolution," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 709–717.
[46] D. Silver et al., "Mastering the game of with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[47] V. Mnih et al., "Playing atari with deep reinforcement learning," *Comput. Sci.*, pp. 1–9, 2013.
[48] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2015.
[49] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1994.
[50] Y. Tang and A. Kucukelbir, "Variational deep Q network," in *Proc. Workshop Conf. Workshop Neural Inf. Process. Syst.*, 2017.
[51] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany:Springer, 2006.
[52] R. Tanabe and A. Fukunaga, "Reviewing and benchmarking parameter control methods in differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 3, pp. 1170–1184, Mar. 2020.
[53] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, 2011.