Adaptive Structural Hyper-Parameter Configuration by Q-Learning

Haotian Zhang, Jianyong Sun and Zongben Xu
School of Mathematics and Statistics,
National Engineering Laboratory for Big Data Analytics,
Xi'an Jiaotong University, Xi'an, China
zht570795275@stu.xjtu.edu.cn, jy.sun@xjtu.edu.cn, zb.xu@xjtu.edu.cn

***Abstract*—Tuning hyper-parameters for evolutionary algorithms is an important issue in computational intelligence. Performance of an evolutionary algorithm depends not only on its operation strategy design, but also on its hyper-parameters. Hyper-parameters can be categorized in two dimensions as structural/numerical and time-invariant/time-variant. Particularly, structural hyper-parameters in existing studies are usually tuned in advance for time-invariant parameters, or with hand-crafted scheduling for time-invariant parameters. In this paper, we make the first attempt to model the tuning of structural hyper-parameters as a reinforcement learning problem, and present to tune the structural hyper-parameter which controls computational resource allocation in the CEC 2018 winner algorithm by Q-learning. Experimental results show favorably against the winner algorithm on the CEC 2018 test functions.**

***Index Terms*—Reinforcement learning, evolutionary algorithm, hyper-parameter tuning, Q-learning**

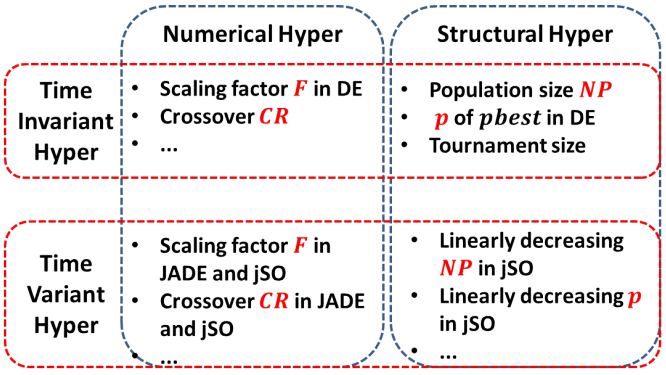

Fig. 1. Categories of hyper-parameters in EAs.

I. Introduction

Evolutionary algorithm (EA) is an important research area in computation intelligence. Over several decades, fruitful research studies have been conducted. Example EAs, such as differential evolution (DE) [1], [2], particle swarm optimization (PSO) [3], CMA-ES [4] and many others, have attracted a great amount of attentions.

The hybridization of EAs has also achieved great success, such as DE/EDA [5], [6], SaDE [7], jSO [8] and others. The aim of hybrid EAs is to take advantages of the pros of different EAs and to compensate the cons of these EAs for favorable algorithmic performance. Very recently, the combination of univariate sampling and CMA-ES [4], [9], called HSES [10], has achieved the best performance for the CEC 2018 test functions.

In all the developed EAs, these always exist more or less hyper-parameters. Those hyper-parameters can be categorized in two dimensions as shown in Fig. 1. In one hand, the hyper-parameters can be time variant or invariant. For instances, the scaling factor F and crossover rate CR can be either fixed like in traditional DE, or adaptively changed such as in JADE [11]. Since the scaling factor and crossover rate are directly responsible for creating new solutions through arithmetic and/or logic operations, they are also categorized as "numerical hyper-parameters". On the other hand, for hyper-parameters such as the population size, the integer p in the current-to-best DE operator [8], the tournament size and others, are categorized as "structural hyper-parameters" since they do not directly involve in the solution creation procedure.

The performance of an EA depends not only on its core components including recombination and selection operations, but also on its hyper-parameters. Tuning hyper-parameter for optimal algorithmic performance can be cumbersome, time-consuming and tedious. What's worse, hybrid algorithms can bring extra structural hyper-parameters. For example, they will require not only the tuning of each composing algorithm's parameters, but also need to control the resource allocation for each of them.

Taking the winner of CEC 2018, HSES [10], as an example, two heuristics including a univariate sampling algorithm and CMA-ES are carried out sequentially. In HSES, except the numerical hyper-parameters used in univariate sampling and CMA-ES, the allocation of resources, or precisely the number of iterations K used by the first univariate sampling, is an important structural hyper-parameter. It could significantly influence the performance of HSES.

Algorithms such as Bayesian optimization (BOA) [12], sequential model-based optimization for algorithm configuration (SMAC) [13] and others, can be applied on tuning time-invariant numerical hyper-parameter. For examples, BOA has been successfully applied to tune numerical hyper-parameters in [14] [15]. Interested readers please see [16] for a review of the tuning techniques.

In these methods, tuning time-invariant numerical hyper-

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

parameters is modeled as optimizing black-box optimization problem. Once optimized, the resultant hyper-parameters are fixed during the optimization procedure. However, for time-variant hyper-parameters, we cannot use these methods simply because of the time-dependence.

It is usually beneficial to adaptively set the hyper-parameters during the search procedure in an EA. A great number of EAs with adaptive hyper-parameters have been studied (see [17] for detail). In most of these studies, it is the numerical hyper-parameters that are made time-variant, e.g. the scaling and crossover rate in DE are adaptively updated during the search by summarizing previous information in JADE [11]. Only recently, some EAs proposed to update structural hyper-parameter adaptively, e.g., the population size in jSO [8] is designed to be linearly decreasing.

However, there is no principle way to tune structural hyper-parameters. Their tuning is case-sensitive. For instance, controlling the selection of DE operators in SaDE [7] can be totally different from the switching between univariate sampling and CMA-ES in HSES.

Recall that to adaptively configure numerical hyper-parameters, new hyper-parameters are updated based on learning from previous search history as seen in JADE. The updating can be broadly considered as a learning problem. This perspective can be applied for adaptively tuning structural parameters. To implement this idea, we resort to reinforcement learning (RL) by modeling the tuning procedure as a finite-horizon Markov Decision Process [18].

In this paper, we propose to use RL, specifically the Q-learning algorithm, to tune the structural hyper-parameter of HSES, i.e. the number of iterations used by the first univariate sampling. In the rest of the paper, Section II briefly review HSES, and concepts and algorithms of RL. The proposed structural hyper-parameter tuning algorithm, called Q-HSES, is presented in Section III. Section IV summarizes experimental results on the CEC 2018 test functions in comparison with HSES. The conclusion is given in Section V.

II. Preliminaries

A. Reinforcement learning

RL has been playing an important role in the thriving of artificial intelligence. It aims to find a policy so that an agent is able to take optimal actions in an environment. Fig. 2 shows a typical representation of RL model. It can be modeled as a Markov decision process (MDP). Consider a finite-horizon MDP with discrete and finite state and action space defined by the tuple $(\mathcal{S}, \mathcal{A}, \mu_0, p, r, \pi, T)$ where $\mathcal{S} \in \mathbb{R}^D$ denotes the state space, $\mathcal{A} \in \mathbb{R}^d$ the action space, μ_0 the initial distribution of the state, $r : \mathcal{S} \to \mathbb{R}$ the reward, and T the time horizon, respectively. At each time t, there is an $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ and a transition probability $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, where $p(s_{t+1}|a_t, s_t)$ denotes the transition probability of s_{t+1} conditionally based on s_t and a_t. The policy $\pi : \mathcal{S} \times \mathcal{A} \times \{0, 1, \cdots T\} \to \mathbb{R}$, where $\pi(a_t|s_t)$ is the probability of choosing action a_t when observing current state s_t. The goal is to find a policy $\pi = p(a_t|s_t)$ so as to maximize the expectation of total

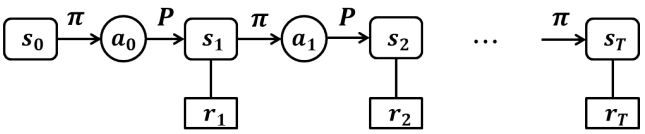

Fig. 2. The basic idea and elements involved in a RL model, where s, a, r represents state, action and reward, respectively, and π and P represents the policy and state transition probability, respectively.

rewards $\mathbb{E}\left(\sum_{t=1}^{T} \alpha_t r_t\right)$ where α_t denote time-step dependent weighting factors. In practice, weighting factors is always set to exponential power of constant i.e. $\alpha_t = \gamma^t$.

There are many RL algorithms, such as Q-learning, sarsa, deep Q network and policy gradient (interested readers please see details in [19]), which are developed to deal with different environments. Among them, Q-learning is developed for MDP with discrete state and action space, based on value iteration. Its core idea is to use the action-value function $Q(s, a)$ to estimate the reward in case $s_t = s$ and $a_t = a$, which is defined as

$$Q(s, a) = \mathbb{E}_\pi(r_{t+1} + \max_{a_{t+1}} \gamma Q(s_{t+1}, a_{t+1})|a_t = s, a_t = a)$$

As $\mathcal{A}$ is discrete and finite, policy can be regarded as

$$\pi(a|s) = \arg\max_{a \in \mathcal{A}} Q(s, a).$$

The Q-learning algorithm can be summarized in Alg. 1 (taken from [19]).

Algorithm 1 Q-learning

1: Initialize $Q(s, a) = 0$ for all $a \in \mathcal{A}, s \in \mathcal{S}$;
2: **for** $e = 1$: maxE **do**
3: **for** $t = 0 : T$ **do**
4: Choose a_t using policy derived from $Q(s, a)$ (ε-greedy) ;
5: Take a_t and observe s_{t+1} and r_{t+1};
6: Compute

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[\gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) + r_{t+1}]; \quad (1)$$

7: **end for**
8: **end for**

In Alg. 1, a maximum number (maxE) of epoch is used to learn the action-value function $Q(s, a)$ for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$. $Q(s, a)$ is initialized to be zero (line 1). Then at each epoch, a trajectory is obtained by applying the ε-greedy policy (line 3, which means in probability $1-\varepsilon$, action a_t is taken as $\arg\max Q(s_t, a)$ and in probability ϵ, action a_t is randomly chosen from $\mathcal{A}$. Based on the state and action at time t, the action-value function is updated according to the equation in line 6. The algorithm terminates at the maxE epoch.

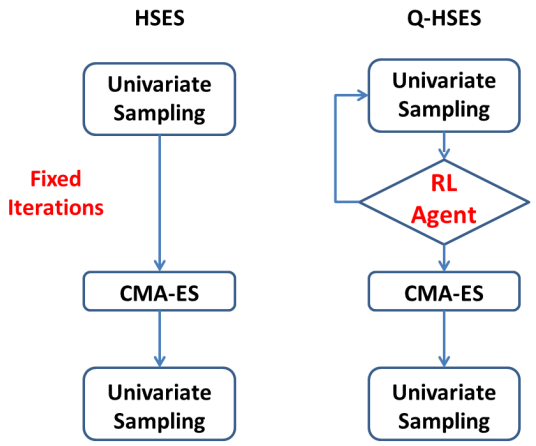

Fig. 3. The schematic diagram of HSES and Q-HSES.

B. HSES

HSES [10] is the winner algorithm in CEC 2018 competition. Its schematic diagram can be seen in the left plot of Fig. 3. The pseudo-code of HSES is shown in Alg. 2.

Algorithm 2 The pseudo-code of HSES

Require: an optimization function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n$, initial population $\mathbf{X}^0 = [\mathbf{x}_1^0, \cdots, \mathbf{x}_N^0] \in \mathbb{R}^{n \times N}$, the switch iteration $\mathbf{I}_1 \in \mathbb{N}_+$, maximum evaluations MaxNFE.

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

1: Set Idx $\leftarrow \emptyset$;
2: $[\mathbf{X}^{t_1}, \text{NFE}_1, \mathbf{x}_1^*] \leftarrow \text{UniSampling}(\mathbf{X}^0, f, \text{Idx}; \mathbf{I}_1)$;
3: $[\mathbf{X}^{t_2}, \text{NFE}_2, \mathbf{x}_2^*] \leftarrow \text{CMA-ES}(\mathbf{X}^{t_1}; \theta)$;
4: Idx $\leftarrow \text{Detect}(\mathbf{X}^{t_2})$;
5: $[\mathbf{X}^{t_3}, \text{NFE}_3, \mathbf{x}_3^*] \leftarrow \text{UniSampling}(\mathbf{X}^{t_2}, f, \text{Idx}; \mathbf{I}_2)$;
6: **return** $\mathbf{x}^* \leftarrow \mathbf{x}_3^*$.

In Alg. 2, the function UniSampling$(\cdot)$ performs univariate sampling for $\mathbf{I}_1$ iterations and returns a population of solutions $\mathbf{X}^{t_1}$ with relatively high qualities, the number of fitness evaluations used ($\text{NFE}_1 = N \times \mathbf{I}_1$), and the best solution $\mathbf{x}_1^*$ found by univariate sampling. $\mathbf{X}^{t_1}$ is then used as the initial population of CMA-ES with parameter θ. CMA-ES returns a population $\mathbf{X}^{t_2}$, the fitness evaluations used NFE_2 and the best solution it found $\mathbf{x}_2^*$. Once CMA-ES has terminated, the function Detect$(\cdot)$ is used to find which variables should be fixed for the next univariate sampling. In the new univariate sampling, a maximum of $\mathbf{I}_2$ iterations is carried out so that the whole number of evaluations is no more than the given constant MaxNFE. Interested readers please refer to [10] for algorithm details. The algorithm returns the best solution found $\mathbf{x}_3^*$ at termination.

In HSES, except the parameters like population size N in univariate sampling and the parameter θ of CMA-ES, the switch iteration $\mathbf{I}_1$ is vital to algorithmic performance. It determines how much resources are allocated for the first univariate sampling, which reflects the tradeoff between exploration and exploitation.

In our study, we care only on how to determine the structural hyper-parameter $\mathbf{I}_1$. The other hyper-parameters, such as N and θ, are fixed as used in the original paper of HSES.

III. Method

We model the tuning procedure of structural hyper-parameters as a finite-horizon MDP with discrete state and action space and horizon T. At each state, the RL agent chooses an action, i.e. switching to CMA-ES or not. If the action is not to switch, univariate sampling will be carried out. The next state will be observed. Otherwise, if the action is to switch to CMA-ES, the agent will stop and execute the rest of HSES (line 3-5 of Alg. 2).

In the following, we shall present the components used in RL, including state, action, transition probability and reward. We denote f_{best}^k the minimum function value obtained up to the k-th iteration.

State: In our RL, s_t includes two parts s_t^1 and s_t^2 which are defined as follows:

$$\begin{aligned} s_t^1 &\triangleq \frac{\log\left(f_{\text{best}}^{10(t-2)}\right) - \log\left(f_{\text{best}}^{10t}\right)}{\left|\log\left(f_{\text{best}}^{10(t-2)}\right)\right|}, t > 1 \\ s_t^2 &\triangleq \frac{\log\left(f_{\text{best}}^{0}\right) - \log\left(f_{\text{best}}^{10t}\right)}{\left|\log\left(f_{\text{best}}^{0}\right)\right|} \end{aligned} \tag{2}$$

and $s_1^1 \triangleq \frac{\log\left(f_{\text{best}}^{0}\right) - \log\left(f_{\text{best}}^{10}\right)}{\left|\log\left(f_{\text{best}}^{0}\right)\right|}$. In the definition, s^1 is used to measure the difference between the best function values in adjacent 20 steps; s^2 measures the descent rate from the first population. We use 10 times t because we don't want to judge switching or not every iteration but every 10 iterations.

It is seen that the range of s^1 and s^2 are all in $[0, +\infty)$. To make the state space discrete and finite, we divide the range to $[0, 0.005]$,$(0.005, 0.05]$,$(0.05, 0.09]$,$(0.09, 0.5]$,$(0.5, 1]$,$(1, +\infty)$ for s^1, and $[0, 0.2]$,$(0.2, 0.5]$,$(0.5, 0.8]$,$(0.8, 1.2]$,$(1.2, 3]$,$(3, +\infty)$ for s^2.

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

Transition probability: When $t < T$ and a_t is 0, the next state s_{t+1} is defined as mentioned above. When $t \geq T$ or a_t is 1, s_{t+1} will be the "terminal state". Here T is the horizon. It constrains the maximum iterations that can be used by univariate sampling. In this paper, we set $T = 20$.

Reward: In the terminal state, the reward will be the logarithm of the minimum function value found by the search. Otherwise, the reward is set to zero since the algorithm's performance is not known before the terminal state.

A. The proposed Q-learning based HSES

Given above definitions, the Q-learning algorithm, i.e. Alg. 1, can be used to train the agent. However, directly using Q-learning can result in the following three problems. First, to make Q-learning converge, the maximal number of epoch, MaxE, is generally large. This means HSES needs to be executed many times, which is not acceptable. Second,

in Q-learning, the action value function is updated at each trajectory based on current policy. Since EAs are stochastic, the performance of one trajectory is not stable, which means the learning of the action value function is not efficient. Third, the trajectory created based on current policy is unbalanced (as using ε-greedy policy). That is, some states may be rarely observed in the trajectories. This can make the learning converge slowly.

Algorithm 3 The training process for the RL agent.

Require: Training functions $f_1, \cdots, f_L$, the maximal epoch maxE, the horizon T and the learning rate α.
Ensure: an optimal policy $\pi(a|s)$.
1: Initialize $\text{meta}_Q(s,a) = 0$ for all $s \in \mathcal{S}, a \in \{0,1\}$;
2: **for** $l = 1 : L$ **do**
3: Initialize $Q(s,a) = 0$ for all $s \in \mathcal{S}, a \in \{0,1\}$;
4: Create T training trajectories $\text{Tr}_m,\ m = 1, \cdots, T$;
5: **for** $e = 1 : \text{maxE}$ **do**
6: **for** each trajectory m **do**
7: **for** $t = 1 : T$ **do**
8: $Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha\gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) + r_{t+1}$ where $\{s_t, a_t\} \in \text{Tr}_m$;
9: **end for**
10: **end for**
11: **end for**
12: **for** $s \in \mathcal{S}$ **do**
13: **if** $Q(s,1) > Q(s,0)$ **then**
14: $\text{meta}_Q(s,1) \leftarrow \text{meta}_Q(s,1) + 1$;
15: **end if**
16: **if** $Q(s,1) < Q(s,0)$ **then**
17: $\text{meta}_Q(s,0) \leftarrow \text{meta}_Q(s,0) + 1$;
18: **end if**
19: **end for**
20: **end for**
21: **return** $\pi(a|s) \leftarrow \arg\max_a \text{meta}_Q(s,a)$.

Our training process is summarized in Alg. 3. In line 4, we generate T trajectories $\text{Tr}_m,\ m = 1, 2, \cdots, T$. Each trajectory corresponds to the average of 51 runs of the HSES algorithm with switch iteration $\mathbf{I}_1$ to be $10, 20, \cdots, 200$ for a given training function. With these trajectories, the action value function $Q(s,a)$ is updated as seen in line 8. To eliminate the influence of different function scales, an auxiliary action-value function $\text{meta}_Q(s,a)$ is applied. It records the rank relationship between $Q(s,0)$ and $Q(s,1)$ from line 12 to line 19 for each training function. After training, the policy $\pi(a|s)$ is set to be $\arg\max_a \text{meta}_Q(s,a)$ (line 21). Note that for a state s, $\text{meta}_Q(s,0) = \text{meta}_Q(s,1)$ means that there is no evidence to tell which action (switch or not) is better. If this is the case, an action will be randomly chosen.

The reason that the proposed training process can handle the mentioned learning problems is simply because the created trajectories Tr_m contains all possible situations for the switch iteration. That is, the performance of HSES with every possible switch iteration from 10 to 200 is observed, which is measured by the average of 51 HSES runs. This can make the training steady and the observed states balanced. Since we do not need to sample new trajectories during training, the training efficiency can be guaranteed.

Once the agent is learned (i.e. the optimal π is found), it is embedded within the HSES algorithm. The resultant algorithm is called Q-HSES. The right plot of Fig. 3 shows the digram of Q-HSES. Alg. 4 summarizes the algorithm. In the algorithm, after univariate sampling runs for M iterations (line 3), the current state s_t is computed (line 4) and an action is taken (line 5). If action is 0, it returns to the univariate sampling. The procedure repeats until action is 1, which means that the algorithm switches to CMA-ES. The rest operations are the same as in Alg. 2.

Algorithm 4 The pseudo-code of Q-HSES

Require: an optimization function $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n$, initial population $\mathbf{X}^0 = [\mathbf{x}_1^0, \cdots, \mathbf{x}_N^0] \in \mathbb{R}^{n \times N}$, maximum evaluations MaxNFE and the learned policy $\pi(a|s)$.
Ensure: an optimal solution $\mathbf{x}^*$.
1: Set Idx $\leftarrow \emptyset, t \leftarrow 0, \text{NFE}_1 \leftarrow 0$;
2: **repeat**
3: $[\mathbf{X}^{t+1}, \text{NFE}_u, \mathbf{x}_1^*] \leftarrow \text{UniSampling}(\mathbf{X}^t, f, \text{Idx}; M)$;
4: Compute the state s_t by (2);
5: Take $a_t \sim \pi(\cdot|s_t)$;
6: **if** $a_t = 1$ **then**
7: Exit;
8: **else**
9: $t \leftarrow t+1$, $\text{NFE}_1 \leftarrow \text{NFE}_1 + \text{NFE}_u$;
10: **end if**
11: **until** $t \geq T$
12: $[\mathbf{X}^{t_2}, \text{NFE}_2, \mathbf{x}_2^*] \leftarrow \text{CMA-ES}(\mathbf{X}^{t+1}; \theta)$;
13: Idx $= \text{Detect}(\mathbf{X}^{t_2})$;
14: $[\mathbf{X}^{t_3}, \text{NFE}_3, \mathbf{x}_3^*] \leftarrow \text{UniSampling}(\mathbf{X}^{t_2}, f, \text{Idx}; \mathbf{I}_2)$;
15: **return** $\mathbf{x}^* \leftarrow \mathbf{x}_3^*$.

IV. Experiments

In this paper, the 29 functions in the CEC 2018 competition ($f_1 - f_{30}$ except f_2) is used as benchmark. Metrics, including the rank and average function value, used in the competition are adopted for comparison. The proposed algorithm, Q-HSES, is compared with HSES.

A. Training

Training: The CEC 2018 test functions contain unimodal functions f_1 and f_3, multimodal functions $f_4 - f_{10}$, hybrid functions $f_{11} - f_{20}$ and composition functions $f_{21} - f_{30}$. Our training functions cover these 4 function types. 11 CEC 2018 functions $(f_1, f_6, f_7, f_8, f_{10}, f_{14}, f_{15}, f_{17}, f_{18}, f_{20}, f_{24})$ are used as the training functions. Parameters used in the Q-learning are set as $\gamma = 1$, $\alpha = 10^{-4}$, $M = 10$, $T = 20$ and $\text{MaxE} = 100,000$. As 50D and 100D problems are more complicated and the performance of algorithm is worse than 10D and 30D, the division to the range of s^2 is a little bit forward. For 50D and 100D, the interval node of s^2 is

multiplied by 0.05 and 0.025. Except the switch iteration, the other hyper-parameters are held the same as HSES (such as population size and the parameters of CMA-ES).

Testing: Q-HSES is used to optimize all the 29 functions of CEC 2018.

B. Comparison Results

In the original HSES, the switch iteration is fixed to be 100. The statistics of the obtained results of Q-HSES on test functions of CEC 2018 are summarized in Tables I,II,III and IV for 10D, 30D, 50D and 100D, respectively, in which the best, worst, mean and std. values are given.

Each of the statistics is obtained over 200, 100, 51, 51 runs on the error values (i.e. the difference between the obtained optimum and the known global optimum). When the error values are smaller or equal to 10^{-8}, they are treated as 0.

TABLE I
RESULTS IN 10D FOR 200 RUNS OBTAINED BY Q-HSES.

	best	worst	median	mean	std
f_1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_5	0.00e+00	3.98e+00	9.95e-01	8.40e-01	8.05e-01
f_6	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_7	1.04e+01	1.28e+01	1.10e+01	1.11e+01	4.96e-01
f_8	0.00e+00	2.98e+00	0.00e+00	5.87e-01	7.54e-01
f_9	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_{10}	6.25e-02	6.12e+02	3.73e+00	1.07e+02	1.42e+02
f_{11}	0.00e+00	1.99e+00	0.00e+00	1.29e-01	3.77e-01
f_{12}	0.00e+00	4.14e+02	2.09e-01	2.43e+01	6.42e+01
f_{13}	0.00e+00	8.51e+00	5.20e+00	3.39e+00	2.55e+00
f_{14}	0.00e+00	3.11e+02	9.85e-04	7.28e+00	3.25e+01
f_{15}	6.71e-05	5.10e+00	4.35e-01	5.58e-01	7.08e-01
f_{16}	1.17e-01	1.19e+02	7.28e-01	1.94e+00	1.18e+01
f_{17}	1.97e-02	3.87e+01	1.73e+01	1.27e+01	1.07e+01
f_{18}	2.24e-05	2.15e+01	9.39e-01	2.15e+00	3.25e+00
f_{19}	2.44e-02	8.99e+00	1.74e-01	5.41e-01	1.34e+00
f_{20}	0.00e+00	1.20e+02	1.31e+00	8.82e+00	1.81e+01
f_{21}	1.00e+02	2.06e+02	2.02e+02	1.97e+02	2.13e+01
f_{22}	1.00e+02	1.00e+02	1.00e+02	1.00e+02	2.85e-14
f_{23}	3.00e+02	3.06e+02	3.00e+02	3.01e+02	1.51e+00
f_{24}	1.00e+02	3.32e+02	3.29e+02	3.27e+02	1.62e+01
f_{25}	3.98e+02	4.50e+02	4.46e+02	4.46e+02	3.62e+00
f_{26}	2.00e+02	3.00e+02	3.00e+02	3.00e+02	7.07e+00
f_{27}	3.91e+02	4.01e+02	3.98e+02	3.97e+02	1.95e+00
f_{28}	3.00e+02	6.46e+02	5.84e+02	5.92e+02	4.77e+01
f_{29}	2.41e+02	2.97e+02	2.63e+02	2.64e+02	1.00e+01
f_{30}	3.95e+02	4.84e+02	3.95e+02	4.11e+02	2.21e+01

The rank sum hypothesis test is carried out at 5% significance level between the results obtained by Q-HSES and HSES. The results are summarized in Table V. From Table V, we see that Q-HSES performs significantly better than HSES in general for all dimensions.

Table VI shows the average function values obtained for the test functions in four different dimensions. It is clear that the average function value obtained by Q-HSES is smaller than HSES in 10D, 30D, 50D, but a little bit greater in 100D.

Overall, we may conclude that the proposed control algorithm based on Q-learning can indeed find a better structural hyper-parameter setting for HSES.

TABLE II
RESULTS IN 30D FOR 100 RUNS OBTAINED BY Q-HSES.

	best	worst	median	mean	std
f_1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_4	0.00e+00	3.99e+00	3.99e+00	2.91e+00	1.78e+00
f_5	2.98e+00	1.19e+01	6.96e+00	6.77e+00	2.04e+00
f_6	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_7	3.29e+01	4.04e+01	3.50e+01	3.53e+01	1.43e+00
f_8	2.98e+00	1.09e+01	6.96e+00	6.64e+00	1.92e+00
f_9	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_{10}	1.27e+02	1.81e+03	8.53e+02	8.69e+02	3.44e+02
f_{11}	0.00e+00	7.29e+01	4.97e+00	1.22e+01	2.01e+01
f_{12}	6.96e-02	3.59e+02	3.93e+00	2.89e+01	7.39e+01
f_{13}	3.81e+00	8.59e+01	2.65e+01	3.08e+01	1.57e+01
f_{14}	1.47e-04	3.58e+01	1.40e+01	1.18e+01	9.89e+00
f_{15}	3.92e-01	3.45e+01	3.49e+00	4.45e+00	4.59e+00
f_{16}	1.26e+00	8.72e+02	2.43e+02	2.64e+02	2.00e+02
f_{17}	2.37e+00	6.03e+02	2.48e+01	7.15e+01	1.12e+02
f_{18}	4.98e-01	2.74e+01	2.07e+01	1.89e+01	6.15e+00
f_{19}	1.72e+00	3.10e+01	3.49e+00	4.25e+00	4.20e+00
f_{20}	1.20e+02	4.21e+02	1.43e+02	1.62e+02	6.01e+01
f_{21}	2.02e+02	2.21e+02	2.07e+02	2.08e+02	3.41e+00
f_{22}	1.00e+02	1.00e+02	1.00e+02	1.00e+02	3.03e-13
f_{23}	3.37e+02	3.69e+02	3.50e+02	3.51e+02	8.18e+00
f_{24}	4.08e+02	4.32e+02	4.20e+02	4.20e+02	4.70e+00
f_{25}	3.87e+02	3.87e+02	3.87e+02	3.87e+02	2.38e-02
f_{26}	2.00e+02	1.47e+03	9.17e+02	8.84e+02	2.04e+02
f_{27}	5.07e+02	5.50e+02	5.24e+02	5.25e+02	1.02e+01
f_{28}	3.00e+02	4.03e+02	3.00e+02	3.19e+02	3.99e+01
f_{29}	4.09e+02	8.11e+02	4.39e+02	4.68e+02	7.66e+01
f_{30}	1.97e+03	2.20e+03	2.06e+03	2.06e+03	4.45e+01

TABLE III
RESULTS IN 50D FOR 51 RUNS OBTAINED BY Q-HSES.

	best	worst	median	mean	std
f_1	0.00e+00	1.70e-08	0.00e+00	0.00e+00	0.00e+00
f_3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_4	0.00e+00	1.14e+02	2.85e+01	4.56e+01	4.72e+01
f_5	0.00e+00	3.98e+00	9.95e-01	1.11e+00	1.08e+00
f_6	3.59e-08	2.95e-05	1.05e-05	1.27e-05	1.08e-05
f_7	5.40e+01	5.81e+01	5.53e+01	5.54e+01	8.05e-01
f_8	0.00e+00	3.98e+00	9.95e-01	1.44e+00	9.81e-01
f_9	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_{10}	1.24e+02	7.39e+02	1.33e+02	2.60e+02	1.69e+02
f_{11}	1.83e+01	2.62e+01	2.33e+01	2.30e+01	1.90e+00
f_{12}	2.40e+00	4.08e+02	1.33e+02	1.53e+02	1.23e+02
f_{13}	2.54e-06	7.85e+01	4.54e+01	3.93e+01	2.54e+01
f_{14}	1.24e-04	2.24e+01	2.03e+01	1.41e+01	9.51e+00
f_{15}	3.14e+00	1.85e+01	1.74e+01	1.73e+01	2.05e+00
f_{16}	1.28e+02	1.24e+03	7.62e+02	6.92e+02	2.73e+02
f_{17}	2.96e+01	9.75e+02	1.77e+02	2.74e+02	1.70e+02
f_{18}	5.72e-01	2.11e+01	2.09e+01	2.05e+01	2.85e+00
f_{19}	3.57e+00	2.61e+01	5.55e+00	7.13e+00	4.79e+00
f_{20}	2.04e+01	2.44e+02	2.50e+01	3.45e+01	4.25e+01
f_{21}	2.01e+02	2.10e+02	2.05e+02	2.05e+02	1.21e+00
f_{22}	1.00e+02	1.00e+02	1.00e+02	1.00e+02	1.16e-06
f_{23}	4.04e+02	4.41e+02	4.24e+02	4.25e+02	9.29e+00
f_{24}	4.84e+02	4.94e+02	4.90e+02	4.89e+02	2.51e+00
f_{25}	4.71e+02	5.66e+02	5.63e+02	5.49e+02	2.38e+01
f_{26}	4.00e+02	8.75e+02	6.58e+02	6.16e+02	1.45e+02
f_{27}	5.24e+02	6.07e+02	5.52e+02	5.56e+02	2.18e+01
f_{28}	4.70e+02	5.08e+02	5.08e+02	5.00e+02	1.20e+01
f_{29}	3.03e+02	8.19e+02	3.47e+02	4.50e+02	1.50e+02
f_{30}	5.80e+05	6.71e+05	5.96e+05	5.99e+05	1.56e+04

TABLE IV
RESULTS IN 100D FOR 51 RUNS OBTAINED BY Q-HSES.

	best	worst	median	mean	std
f_1	0.00e+00	2.45e-08	0.00e+00	0.00e+00	0.00e+00
f_3	0.00e+00	3.88e-08	0.00e+00	0.00e+00	0.00e+00
f_4	0.00e+00	6.88e+01	0.00e+00	7.74e+00	1.92e+01
f_5	9.95e-01	6.96e+00	2.98e+00	3.58e+00	1.49e+00
f_6	6.51e-08	1.19e-07	8.95e-08	9.23e-08	1.76e-08
f_7	1.08e+02	1.13e+02	1.10e+02	1.10e+02	1.19e+00
f_8	9.95e-01	7.96e+00	3.98e+00	3.98e+00	2.14e+00
f_9	0.00e+00	6.22e+00	0.00e+00	7.88e-01	1.61e+00
f_{10}	7.63e+02	2.04e+03	1.23e+03	1.31e+03	3.07e+02
f_{11}	1.69e-06	8.91e+01	1.27e+01	3.55e+01	3.80e+01
f_{12}	3.43e+02	1.97e+03	8.70e+02	9.13e+02	4.13e+02
f_{13}	3.77e+01	5.96e+01	4.15e+01	4.32e+01	5.65e+00
f_{14}	1.99e+00	2.38e+01	2.18e+01	2.11e+01	4.54e+00
f_{15}	6.89e+01	1.33e+02	9.06e+01	9.87e+01	1.96e+01
f_{16}	4.31e+02	1.72e+03	1.10e+03	1.04e+03	4.04e+02
f_{17}	5.80e+01	8.72e+02	4.84e+02	4.96e+02	2.37e+02
f_{18}	5.19e-01	2.26e+01	1.53e+00	9.41e+00	9.87e+00
f_{19}	1.10e+01	3.16e+01	1.40e+01	1.50e+01	5.00e+00
f_{20}	2.77e+02	1.33e+03	5.09e+02	5.52e+02	2.78e+02
f_{21}	2.17e+02	2.32e+02	2.27e+02	2.26e+02	4.01e+00
f_{22}	1.00e+02	1.00e+02	1.00e+02	1.00e+02	2.40e-09
f_{23}	5.28e+02	5.58e+02	5.43e+02	5.45e+02	7.51e+00
f_{24}	8.32e+02	8.52e+02	8.45e+02	8.43e+02	5.97e+00
f_{25}	6.58e+02	7.87e+02	7.41e+02	7.34e+02	3.36e+01
f_{26}	2.15e+03	2.55e+03	2.37e+03	2.37e+03	1.23e+02
f_{27}	6.27e+02	6.47e+02	6.37e+02	6.37e+02	6.01e+00
f_{28}	3.00e+02	6.33e+02	5.35e+02	5.03e+02	1.10e+02
f_{29}	8.36e+02	2.14e+03	1.32e+03	1.31e+03	3.71e+02
f_{30}	2.51e+03	2.91e+03	2.71e+03	2.72e+03	1.16e+02

TABLE V
RANK COMPARISON OF Q-HSES AND HSES

	better	$\approx$	worse
10D	8	20	1
30D	7	20	2
50D	3	24	2
100D	3	25	1

C. Validation

The switch iteration is fixed in HSES as 100. To validate the performance of the learned agent, Q-HSES is compared with HSES on different switch iterations in 10D. Table VII summarizes the results, in which the hypothesis test results at 5% significant level are listed. The indices of the functions that Q-HSES performs better are also shown. The indices of the functions that belong to the training functions are typeset in bold.

From Table VII, it is seen that with different switch iterations, Q-HSES performs better than HSES in general: Q-HSES performs better on more functions than HSES. This indicates that the learned agent works well. It is seen that among the functions that Q-HSES performs better, half of them do not belong to the training functions. This indicates the learned agent generalizes well.

Table VIII shows the average $\mathbf{I}_1$ values learned for each function. From the table, it is seen that the $\mathbf{I}_1$ values are different for different functions.

TABLE VI
FUNCTION VALUE COMPARISON OF Q-HSES AND HSES

	Q-HSES	HSES
10D	3504.1	3511.1
30D	7246.6	7364.5
50D	604390.5	611750.5
100D	14648.3	14602.3

TABLE VII
VALIDATION OF Q-HSES

fixed iterations	better / $\approx$ / worse	index
30	12/ 14 / 3	5 **7 8 10** 11 **17 18 20** 21 22 23 26
50	10/ 17 / 2	5 **7 8 10** 11 16 **17 20** 23 26
120	7/ 20 / 2	**7** 16 **17** 19 **20** 23 26
160	7/ 20 / 2	13 **14 18** 19 22 23 26
200	5/ 24 / 1	13 **14 15 18** 19

D. Training Process

To verify the training process is convergent, we define a criterion named "the rate of action value change". As the state space is divided into 36 intervals (6 intervals for s^1 and 6 intervals for s^2) and action is $\{0, 1\}$, then the action-value function $Q(s, a)$ is a 36×2 matrix. Regarding it as a 72D vector, the rate of action value change is defined as the Frobenius norm between two adjacent epochs:

$$\|Q^e(s, a) - Q^{e+1}(s, a)\|_F^2$$

where $Q^e(s, a)$ and $Q^{e+1}(s, a)$ is the action value function at the eth and $(e + 1)$th epoch, respectively. This definition can be used to show how the training goes. Fig. 4 shows the rates of change for the training functions. From the figure, it is clear that the training process of action value function is convergent. It also shows that the range of the changes is very much different among different functions which roots from different scales of the function values.

V. CONCLUSION

In this paper, we first categorized the hyper-parameters of EAs from two perspectives. We then proposed to model the adaptive control of the structural hyper-parameters as a Markov decision process. Based on the formalization, Q-learning was applied to learn an agent for time-variant hyper-parameter tuning for the winner algorithm of CEC 2018, called HSES. We proposed the fundamental elements of the Q-learning for the agent, including states, action and reward. In the experiments, we trained the agent on a selection of functions from the CEC 2018 competition. By embedding the learn agent, Q-HSES was developed. The comparison between Q-HSES and HSES showed that the structural hyper-parameter in HSES controlled by the learned agent performs generally better than HSES.

As a first attempt to use Q-learning for hyper-parameter controlling, the experiments showed that the proposed method is promising. In the future, we intend to combine RL to advancing the development of evolutionary algorithms.

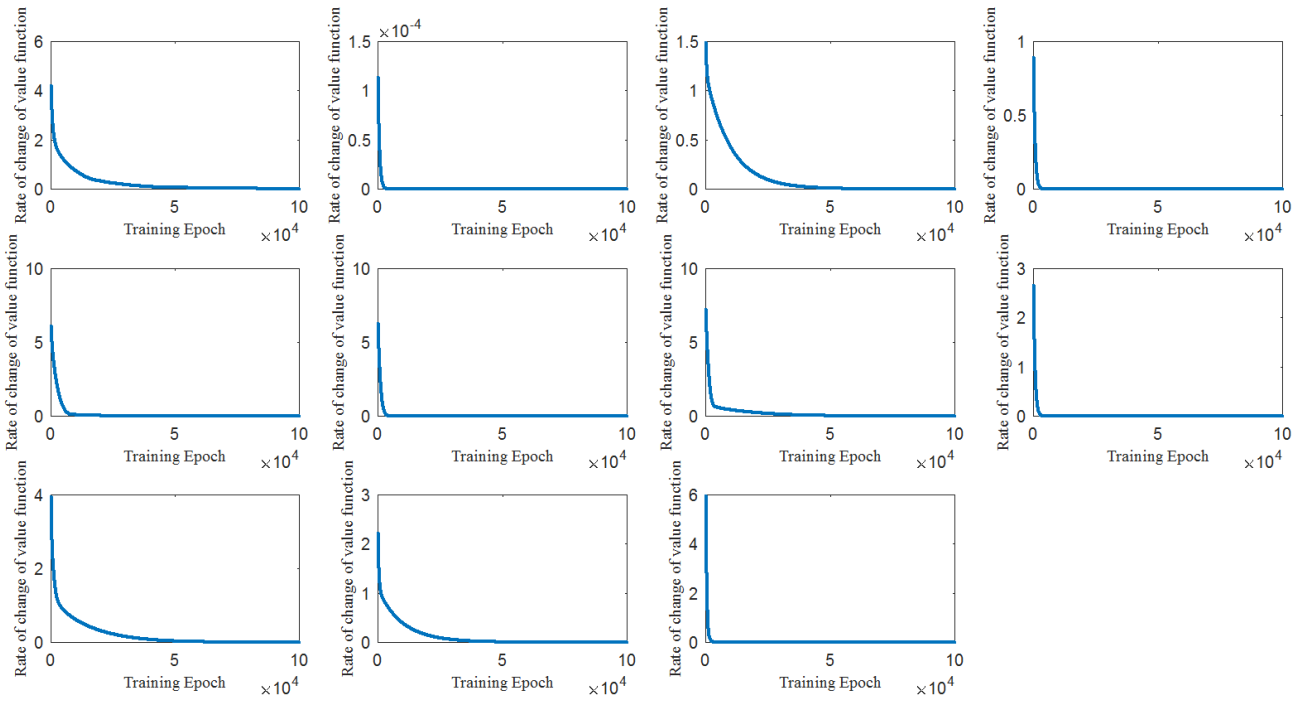

Fig. 4. The rate of function value change against epoch for the 11 training functions.

TABLE VIII
THE $\mathbf{I}_1$ VALUES LEARNED BY THE RL AGENT USED IN Q-HSES.

	10D	30D	50D	100D		10D	30D	50D	100D
f_1	90	200	200	200	f_3	60	200	200	200
f_4	30	60	200	200	f_5	200	200	40	200
f_6	200	200	30	200	f_7	200	200	200	200
f_8	200	200	40	200	f_9	30	200	200	200
f_{10}	120	120	30	40	f_{11}	30	200	200	200
f_{12}	30	60	200	200	f_{13}	50	200	200	200
f_{14}	30	200	200	200	f_{15}	60	200	200	200
f_{16}	200	200	30	200	f_{17}	200	200	200	200
f_{18}	140	200	200	200	f_{19}	50	200	200	200
f_{20}	200	200	30	200	f_{21}	200	200	30	200
f_{22}	200	200	200	200	f_{23}	200	200	200	40
f_{24}	200	200	110	200	f_{25}	200	200	200	200
f_{26}	200	200	50	200	f_{27}	200	200	30	200
f_{28}	200	200	200	200	f_{29}	200	200	200	200
f_{30}	30	90	200	200					

VI. APPENDIX

In this section, we briefly introduce the concepts used in reinforcement learning. Basically, RL aims to maximize the expected cumulative reward, i.e. $\mathbb{E}\left(\sum_{t=1}^{T}\gamma^t r_t\right)$. First define

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T$$

The expectation of G_t measures the benefit on time t. Further, we define the state-value function $v(s)$ and action-value function $Q(s,a)$ as follows:

$$v(s) \triangleq \mathbb{E}[G_t | s_t = s] \tag{3}$$

$$Q(s,a) \triangleq \mathbb{E}[G_t | a_t = a, s_t = s] \tag{4}$$

Without loss of generality, set $\gamma = 1$, we have the following Bellman's equality:

$$\begin{aligned} v(s) &= \mathbb{E}[r_{t+1}|s_t = s] \\ &\quad + \mathbb{E}[\mathbb{E}[r_{t+2} + \cdots + r_T | s_{t+1} = s']|s_t = s] \\ &= \mathbb{E}[r_{t+1} + v(s')|s_t = s] \end{aligned}$$

For action value function, we also have the Bellman's formula:

$$Q(s,a) = \mathbb{E}[r_{t+1} + v(s')|s_t = s, a_t = a]$$

For optimal policy $\pi(a|s)$, we have:

$$v(s) = \max_a Q(a,s)$$

which induces the optimal Bellman equation:

$$Q(s,a) = \mathbb{E}[r_{t+1} + \max_{a'} Q(a', s')|s_t = s, a_t = a] \tag{5}$$

This resembles line 6 of Alg. 1.

ACKNOWLEDGEMENT

This work was partly supported by the National Natural Science Foundation of China (grant no. 11991023), the Major Project of National Science Foundation of China (grant no. U1811461), Key Project of National Science Foundation of China (grant no. 11690011), and the Project of National Science Foundation of China (grant no. 61721002).

REFERENCES

[1] K. V. Price, *An Introduction to Differential Evolution*. GBR: McGraw-Hill Ltd., UK, 1999, pp. 79–108.

[2] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

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

[4] N. Hansen and A. Ostermeier, "Completely derandomized self adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[5] J. Sun, Q. Zhang, and E. Tsang, "DE/EDA: a new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3, pp. 249–262, 2005.

[6] J. Sun, "Two-stage EDA-based approach for all optical wdm mesh network survivability under srlg constraints," *Applied Soft Computing*, vol. 11, pp. 916–926, 2011.

[7] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions On Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[8] J. Brest, M. S. Maucec, and B. BOskovic, "Single objective realparameter optimization: Algorithm jSO," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2017, pp. 1311–1318.

[9] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[10] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2018, pp. 1–7.

[11] J. Zhang and A. C.Sanderson, "JADE: Adaptive differential evolution with optimal external archive," *IEEE Transactions On Evolutionary Computation*, vol. 13, no. 5, pp. 945–458, 2009.

[12] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[13] F. Hutter, H. H. Hoos, and K. Leytonbrown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of International Conference on Learning and Intelligent Optimization*, 2011, pp. 507–523.

[14] I. Roman, J. Ceberio, A. Mendiburu, and J. A. Lozano, "Bayesian optimization for parameter tuning in evolutionary algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2016, pp. 4839–4845.

[15] C. Huang, B. Yuan, Y. Li, and X. Yao, "Automatic parameter tuning using bayesian optimization method," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2019, pp. 2090–2097.

[16] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Transactions on Evolutionary Computation*, pp. 1–16, 2019. [Online]. Available: 10.1109/ TEVC.2019.2921598

[17] A. Aleti and I. Moser, "A systematic literature review of adaptive parameter control methods for evolutionary algorithms," *ACM Computing Surveys*, vol. 49, no. 3, Oct. 2016.

[18] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley and Sons, Inc., 1994.

[19] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.