An OCBA-Based Method for Efficient Sample Collection in Reinforcement Learning

Kuo Li, *Graduate Student Member, IEEE*, Xinze Jin, *Student Member, IEEE*, Qing-Shan Jia, *Senior Member, IEEE*, Dongchun Ren, and Huaxia Xia, *Member, IEEE*

***Abstract*— This work focuses on the sample collection in reinforcement learning (RL), where the interaction with the environment is typically time-consuming and extravagantly expensive. In order to collect samples in a more valuable way, we propose a confidence-based sampling strategy based on the optimal computing budget allocation algorithm (OCBA), which actively allocates the computing efforts to actions with different predictive uncertainties. We estimate the uncertainty with ensembles and generalize them from tabular representations to function approximations. The OCBA-based sampling strategy could be easily integrated into various off-policy RL algorithms, where we take Q-learning, DQN, and SAC as examples to show the incorporation. Besides, we provide the theoretical analysis towards convergence and evaluate the algorithms experimentally. According to the experiments, the incorporated algorithms obtain remarkable gains compared with modern ensemble-based RL algorithms.**

***Note to Practitioners*—Reinforcement learning is a powerful tool for handling sequential decision-making problems, e.g., autonomous driving and robotics control, where the behaviors typically have a long-term effect on future events. However, although RL achieves human-level control in some tasks, it severely suffers from low sample efficiency. Therefore, implementing RL in some practical areas, e.g., healthcare and rescue, is extremely hard due to the requirement of massive samples. This work aims to enhance the exploration of RL by incorporating OCBA, which provides an asymptotically optimal data-collection strategy for simulation-based optimization. Based on ensemble-based uncertainty estimation and OCBA-based action selection, the incorporated RL algorithms show competitive performance on many benchmarks and significantly reduce the sampling efforts during iterations.**

***Index Terms*— Reinforcement learning, OCBA, ensemble, uncertainty, exploration.**

Manuscript received 14 November 2022; revised 28 February 2023; accepted 27 May 2023. This article was recommended for publication by Associate Editor X. Zhong and Editor J. Li upon evaluation of the reviewers' comments. This work was supported in part by the NSFC under Grant 62125304, Grant 62192751, and Grant 62073182. *(Corresponding author: Qing-Shan Jia.)*

Kuo Li, Xinze Jin, and Qing-Shan Jia are with the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: li-k19@mails.tsinghua.edu.cn; jxz18@mails.tsinghua.edu.cn; jiaqs@tsinghua.edu.cn).

Dongchun Ren and Huaxia Xia are with the Meituan Group, Beijing 100102, China (e-mail: rendongchun@meituan.com; xiahuaxia@meituan.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2023.3282257.

Digital Object Identifier 10.1109/TASE.2023.3282257

I. Introduction

In THE past decade, reinforcement learning [1] has received widespread attention for its effective potential on a series of sequential decision-making problems. Furthermore, the combination with deep learning makes it possible to achieve human-level control on some complex tasks, e.g., playing Go [2], video games [3], [4], and robot control [5], [6], [7]. However, it usually comes with the tradeoff between exploration and exploitation. Namely, it might be extravagantly expensive to reach an expected performance. During the learning process, the data collection and policy evolution are strongly correlated, which implies that a well-designed strategy for action selection may benefit the following iterations.

This topic is intensively studied in multi-armed bandits (MAB) [1] and statistical ranking and selection (R&S) [8]. Both of them study the sampling strategy on finite actions, whose rewards have unknown distributions. Within limited sampling budget, MAB focuses on gathering more cumulative rewards (or equivalently fewer regret), while R&S aims to identify the best action with higher confidence. Classical MAB algorithms include optimistic initial values (OIV) [1], upper confidence bound (UCB) [9], and Thompson sampling (TS) [10], [11]. These algorithms perform competitively in minimizing regret but are more conservative in identifying the best action [12]. In contrast, R&S (or best arm identification, i.e., BAI, in computer science) performs better in identifying the best alternative with higher confidence. A representative approach is optimal computing budget allocation (OCBA) [13], [14], [15], which gives a closed-form budget-allocation strategy to maximize the probability of correct selection (PCS). The allocation problem could also be formulated as stochastic control, and an approximately optimal allocation strategy could be derived from the associated Bellman equation [16]. There are also approaches designed to maximize the expected value of information (EVI) [17]. For example, linear loss (LL) [17] and its variant LL_1 [18], [19] allocate sampling budget to minimize the expected opportunity cost (EOC) [17]. A more comprehensive review can be found in [20]. These algorithms provide effective sampling strategies based on the posterior performance distributions, but expanding them to Markov decision process (MDP) is not straightforward. On the one hand, the crucial interplay between states and actions is not under consideration. On the

other hand, the performance distributions in MAB and R&S are time-invariant, but in MDP, they frequently vary with the updating of behavior policies.

The most widely used sampling strategy for interacting with MDP is ϵ-greedy, which selects the empirically best action with probability $1-\epsilon$ or a random one with probability ϵ. It is widely applied in Q-learning [21] and Deep Q-networks (DQN) type algorithms [3], [22], [23], [24]. However, the ϵ-greedy strategy allocates sampling efforts based on the estimated discounted cumulative reward without considering the predictive uncertainty. The predictive uncertainty gives a measurement of the risk of false selection, without which a low training efficiency with exponentially many steps to learn might be caused [25]. Besides this, Bayesian Q-learning [26] selects the action with maximal posterior probability to be the best, where the probability is calculated by the approximated distribution of the remaining rewards. Similarly, the predictive uncertainty can also serve as a bonus during policy evaluation, which also improves efficiency [25]. However, these methods can only be applied to MDPs with discrete states and actions, where the action values (expected cumulative rewards starting from the given state-action pair) have tabular representations. These tabular-based algorithms are hard to expand to more complex tasks, especially those with continuous states or actions.

In practice, many real-world tasks can be characterized as MDPs with continuous states and actions, where typically RL algorithms with function approximations are adopted. These algorithms can be categorized into deterministic-policy-based algorithms and stochastic-policy-based algorithms. Similar to ϵ-greedy, the sampling strategies of deterministic-policy-based algorithms, e.g., deep deterministic policy gradient (DDPG) [27] and twin delayed deep deterministic policy gradient (TD3) [28], take the adjacent area of the evaluated best action into account, without considering the predictive uncertainty. For stochastic-policy-based algorithms, e.g., trust region policy optimization (TRPO) [29], proximal policy optimization (PPO) [30], and soft actor-critic (SAC) [31], [32], [33], the policies are trained with the estimated action values, without considering the predictive uncertainty neither. These may derive the accumulation of estimation errors along with the sequential states and lead to sub-optimal policies or even divergence [28].

A potential approach to address above issues is incorporating predictive uncertainties into the sampling process, as aforementioned MAB and R&S literature. However, for continuous MDPs, the efficient sampling strategies with counting-based uncertainty estimation are hard to be applied, since storing the statistics for infinite state-action pairs is impractical. Besides, the correlation between adjacent state-action pairs is not well utilized. Nevertheless, learning from this, it is feasible to incorporate confidence-based action selection with ensemble-based predictive-uncertainty estimation, which is realized by measuring the diversity of multiple function approximations. For example, Chen et al. [34] use Q-ensembles to estimate the predictive uncertainty and design an UCB-based sampling strategy for discrete actions. A similar idea arises in [35], where the UCB-based sampling strategy is further generalized to handle continuous actions by incorporating policy ensembles. As shown in [36], by incorporating some techniques to enforce diversity, the predictive uncertainty can be effectively estimated. This makes it possible to further improve the performance of modern off-policy RL algorithms by incorporating confidence-based sampling strategies, e.g., UCB and TS [35]. However, such MAB algorithms are a little conservative in identifying the best action, since exploitation is slightly overweighted [12].

Since RL algorithms are designed to accumulate more long-term rewards, a better sampling-effort allocation strategy has great potential to accelerate training. Namely, the tradeoff between exploration and exploitation should be adequately considered. On the one hand, if the action is selected with high confidence, we tend to exploit it and focus on the remaining trajectories. On the other hand, if the confidence is low, enhancing exploration has a more important long-term effect. A similar idea of efficiently accumulating evidence for decision-making arises in the aforementioned OCBA, which provides an asymptotically optimal strategy to maximize PCS for R&S problems. Motivated by this, we design an efficient sampling strategy by expanding the results of OCBA to the sampling process of RL. In the proposed algorithms, both the confidence evaluation and sampling-effort allocation are well considered. We firstly estimate the predictive uncertainty with ensembles. Then, under the tabular setting, we propose a confidence-based sampling strategy by relating the sampling process of RL to computing effort allocation in OCBA. We theoretically build the convergence property and further generalize it to continuous MDPs. The proposed OCBA-based sampling strategy can be incorporated into various off-policy algorithms. In this work, we take Q-learning, DQN, and SAC as instances to show the incorporation. Finally, we evaluate the proposed algorithms with experiments, where the OCBA-based RL algorithms show superior performance than the baselines. The standing of this work could be seen from three aspects. First, compared with prior arts in R&S [13], [17], [18], [19], we expand the sampling-effort allocation to more general situations of MDPs. Second, compared with existing ensemble-based RL algorithms [34], [35], we replace the widely used UCB with OCBA, which is more effective in identifying the best actions. Third, we expand the assumption of bounded reward in prior work [37] to a more general situation, where the rewards are only assumed to have bounded mean and variance.

The main contributions are as follows:

1) We facilitate the sampling process of RL with computing effort allocation in OCBA, based on which an efficient sampling strategy is proposed. In addition, we give a criterion to measure confidence and provide a sampling strategy to efficiently identify the best action. To the best knowledge of authors, this is the first work to incorporate OCBA into the sampling process of infinite-horizon MDPs.
2) We adopt ensemble-based predictive-uncertainty estimation, which makes the OCBA-based sampling strategy effective for both tabular representations and nonlinear function approximations. Besides, by incorporating pol-

icy ensemble, the OCBA-based sampling strategy can be applied to almost all common RL settings.

3) We integrate the OCBA-based sampling strategy with three modern RL algorithms, e.g., Q-learning, DQN, and SAC, and validate their effectiveness through numerical experiments. The results show that the OCBA-based sampling strategy remarkably improves the performance compared with the baseline algorithms.

The rest of this paper is organized as follows. We provide the preliminary in Section II, introduce the proposed algorithms in Section III, give the convergence analysis in Section IV, present the experimental results in Section V, and briefly conclude in Section VI.

II. Preliminary

In this section, we first introduce some basic RL concepts and off-policy RL algorithms. Then, we present the main results of OCBA, which is used to develop the sampling strategy later.

A. Reinforcement Learning

We consider a sequential decision-making problem, which can be characterized as an MDP, $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $R(s,a): \mathcal{S}\times\mathcal{A}\to\mathbb{R}$[1] is the reward function that assigns each state-action pair a stochastic reward whose mean and variance are bounded, and $\gamma \in (0,1)$ is the discount factor for balancing instantaneous and future rewards. In the situation where $\mathcal{S}$ is discrete, $P(s'|s,a): \mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,1]$ defines the state transition probability of transiting from state s to s' by taking action a, while in the situation where $\mathcal{S}$ is continuous, $P(s'|s,a): \mathcal{S}\times\mathcal{A}\times\mathcal{S}\to[0,+\infty)$ defines corresponding state transition probability density. The agent intends to maximize the discounted cumulative reward (so-called "return"), $R_t = \sum_{k=0}^{\infty}\gamma^k r_{t+k}$, where r_{t+k} is the reward given by the reward function R at time $t+k$.

The action-value function $Q^\pi(s,a) = \mathbb{E}[R_t|s_t=s, a_t=a]$ is defined as the expected return of taking action a under state s and then following policy π, where in the situation where $\mathcal{A}$ is discrete, $\pi(a|s): \mathcal{S}\times\mathcal{A}\to[0,1]$ defines the probability of taking action a under state s, while in the situation where $\mathcal{A}$ is continuous, $\pi(a|s): \mathcal{S}\times\mathcal{A}\to[0,+\infty)$ defines corresponding probability density. The optimal action-value function is defined as

$$Q^*(s,a) = \max_\pi Q^\pi(s,a), \quad \forall (s,a)\in\mathcal{S}\times\mathcal{A}, \tag{1}$$

which follows the Bellman equation

$$Q^*(s,a) = \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}(R(s,a) + \gamma \max_{a'} Q^*(s',a')), \tag{2}$$

and could be obtained through value-iteration algorithms.

There are various algorithms to solve MDPs under different settings. We conclude the representative works in TABLE I. For example, when both $\mathcal{S}$ and $\mathcal{A}$ are finite, the action-value function could be described with a look-up table. Then, Q-learning [21] could be applied to estimate the optimal action-value function. It starts from random initialization and recursively updates estimation following

$$\begin{aligned} Q_{t+1}(s_t,a_t) = (1-\alpha_t(s_t,a_t))Q_t(s_t,a_t) \\ + \alpha_t(s_t,a_t)(r_t + \gamma \max_a Q_t(s_{t+1},a)), \end{aligned} \tag{3}$$

where $\alpha_t(s,a)\in[0,1]$ is the step size taking non-zero value only on $(s,a)=(s_t,a_t)$. If all state-action pairs are performed infinitely often, and

$$\sum_t \alpha_t(s,a) = \infty \sum_t \alpha_t^2(s,a) < \infty \tag{4}$$

holds for all $(s,a)\in\mathcal{S}\times\mathcal{A}$, the action values will converge with probability 1 (w.p.1) to Q^* [37].

Since tabular representations have limited capacity, they are typically replaced with function approximations if the states are continuous. For example, DQN approximates the action values with neural networks and realizes value iteration by minimizing the residual error

$$L(\theta) = \frac{1}{N}\sum_{e_t\in\mathcal{B}}(r_t + \gamma \max_a Q_{\theta^-}(s_{t+1},a) - Q_\theta(s_t,a_t))^2, \tag{5}$$

where θ, θ^- are the parameters of the current network Q_θ and target network Q_{θ^-}, respectively. The target network is adopted for stabilizing learning and is gradually updated towards θ during the training process. Besides, $e_t = (s_t,a_t,r_t,s_{t+1})\in\mathcal{B}$ is the experience collected at time t, and N is the size of mini-batch $\mathcal{B}$.

A more complex situation is that actions are continuous, which makes the $\max(\cdot)$ function in (3) and (5) hard to be calculated. In order to handle this situation, some actor-critic algorithms, e.g., DDPG [27], TD3 [28], and SAC [31], [32], [33], incorporate a separate actor to inference the best action. In this way, they separate the training process into two phases, i.e., policy evaluation and policy improvement. Taking SAC, a state-of-the-art off-policy actor-critic algorithm, as an example, during policy evaluation, the action-value function, i.e., critic, is updated by minimizing the residual error

$$L(\theta) = \frac{1}{N}\sum_{e_t\in\mathcal{B}}(y_t - Q_\theta(s_t,a_t))^2, \tag{6}$$

where

$$y_t = r_t + \gamma Q_{\theta^-}(s_{t+1},a'_{t+1}) - \nu \log\pi_\phi(a'_{t+1}|s_{t+1}) \tag{7}$$

TABLE I
Categorization of Representative RL Algorithms

		$\mathcal{S}$	
		discrete	continuous
$\mathcal{A}$	discrete	**Q-learning***[21], Sarsa°[38], Double Q-learning*[22]	**DQN***[3], TRPO°[29], PPO°[30], SAC*[39], A2C°[40], A3C°[40], Double DQN*[23], Dueling DQN*[24], Rainbow DQN*[41]
	continuous	A2C°[40], A3C°[40], TRPO°[29], PPO°[30], DDPG*[27], TD3*[28], **SAC***[32]	

* Off-policy algorithms.
° On-policy algorithms.

[1] $\mathbb{R}$ is the set of real numbers.

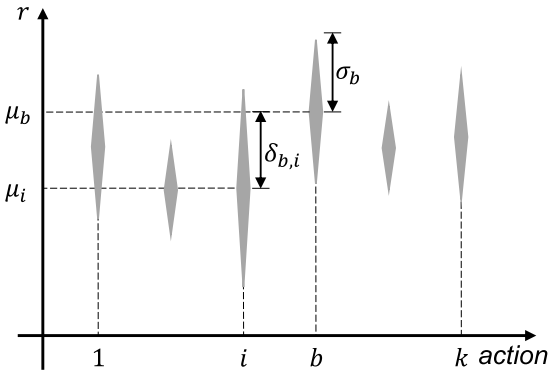


Fig. 1. Illustration of OCBA. The shaded regions denote the distributions of rewards.

is the soft target value, and the temperature ν determines the relative importance of the entropy against the reward. The action a'_{t+1} is freshly sampled from the actor $\pi_\phi(\cdot|s)$, which embeds a probability density over the continuous action space, and ϕ is the parameter. During policy improvement, the actor is updated by maximizing the entropy-regularized action value,

$$J(\phi) = \frac{1}{N}\sum_{e_t\in\mathcal{B}}(Q_\theta(s_t, a'_t) - \nu \log \pi_\phi(a'_t|s_t)), \tag{8}$$

where a'_t is freshly sampled from $\pi_\phi(\cdot|s_t)$. In order to address the overestimation error and stabilize learning, SAC also incorporates target networks and clipped double-Q estimation, whose details can be found in [32].

B. Optimal Computing Budget Allocation

OCBA [13] is an effective technology for resources allocation in the field of simulation-based optimization [42], [43]. As shown in Fig. 1, there are k alternatives $\{i\}_{i=1}^k$, whose performance follows Gaussian distributions $\mathcal{N}(\mu_1, \sigma_1^2)$, $\mathcal{N}(\mu_2, \sigma_2^2)$, $\cdots$, $\mathcal{N}(\mu_k, \sigma_k^2)$, respectively. The agent aims to identify the best alternative in the sense of mean performance, i.e., $b = \text{argmax}_i\, \mu_i$, with higher confidence.

Conduct a thought experiment that total T simulation replications are allocated to the alternatives, where each alternative i is allocated with N_i replications, and $\sum_{i=1}^k N_i = T$. Then, given the samples, the confidence is evaluated by PCS

$$\text{PCS} = \mathcal{P}(\cap_{i\neq b}(\mu_b > \mu_i)|X), \tag{9}$$

where $X = \{\{X_i^m\}_{m=1}^{N_i}\}_{i=1}^k$ is the collection of samples, X_i^m is the m-th sample from alternative i, and $\mathcal{P}(e)$ denotes the probability of event e occurring. If we use Gaussian distribution to approximate the posterior distribution for the unknown mean and adopt a non-informative prior distribution, the posterior distribution of μ_i is given in [44] as

$$\hat{\mu}_i \sim \mathcal{N}(\frac{1}{N_i}\sum_{m=1}^{N_i} X_i^m, \frac{\sigma_i^2}{N_i}). \tag{10}$$

Then, PCS could be rewritten as

$$\text{PCS} = \mathcal{P}(\cap_{i\neq b}(\hat{\mu}_b > \hat{\mu}_i)), \tag{11}$$

which is lower bounded by the approximate probability of correct selection (APCS)

$$\text{APCS} = 1 - \sum_{i\neq b}\mathcal{P}(\hat{\mu}_b \leq \hat{\mu}_i). \tag{12}$$

Since APCS provides an approximation of PCS and is easier to be calculated [13], it will be used as a criterion to control the sampling process of RL in the next section.

It can be seen from (10) that the computing budget allocation $\{N_i\}_{i=1}^k$ affects PCS and APCS by affecting the posterior distributions. Therefore, towards higher confidence, OCBA formulates the problem as

$$\underset{N_1,\cdots,N_k}{\text{argmax}}\ \text{APCS} \quad s.t. \sum_{i=1}^k N_i = T \tag{13}$$

and gives an asymptotically optimal solution

$$\begin{aligned} &\frac{N_i}{N_j} = (\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}})^2,\ i\neq b \text{ and } j\neq b \\ &N_b = \sigma_b(\sum_{i\neq b}\frac{N_i^2}{\sigma_i^2})^{\frac{1}{2}} \\ &\sum_i N_i = T \end{aligned} \tag{14}$$

where $\delta_{b,i} = \mu_b - \mu_i$ is the performance difference. Getting out of the thought experiment, when allocating budget as (14), both μ_i and σ_i are estimated by earlier samples. Although the estimated parameters are not accurate, the allocation results still make sense by alternating between re-estimation and re-allocation [8]. Other details of OCBA can be found in [13].

Based on the observation that OCBA performs effectively in identifying the best alternative in R&S problems, we attempt to expand it to MDP situations and develop an OCBA-based sampling strategy for RL. The details are introduced in the next section.

III. Methodology

In this section, we propose the main OCBA-based sampling strategy. We first put forward the ensemble-based predictive-uncertainty estimation, which is generalizable from tabular representations to function approximations, e.g., neural networks. Then, based on the distributions estimated with ensembles, the allocation strategy of OCBA is then mapped into a sampling distribution over the action space. In principle, this sampling strategy can be integrated into all off-policy algorithms in TABLE I. In this section, we take Q-learning, DQN, and SAC as instances to show the incorporation in the situation of 1) discrete states and actions, 2) continuous states and discrete actions, and 3) continuous actions, respectively. The incorporation with other off-policy algorithms can be realized in a similar way.

We firstly introduce the predictive-uncertainty estimation. Since the solution of (2) is hard to be precisely calculated in large-scale problems, the action values are typically estimated with sample trajectories. Therefore, the estimated action values might be different across multiple runs. Namely, for each state-action pair, the estimated action value at a given time step is

a random variable that follows certain distribution, and the estimated action value in each run is just a sample. Then, in order to estimate the distributions to make better decisions, we approximate the action values with an ensemble of M independent estimations $\{Q_{\theta^i}\}_{i=1}^{M}$ and calculate the mean and variance with

$$\mu(Q_\theta(s,a)) = \frac{1}{M}\sum_{i=1}^{M} Q_{\theta^i}(s,a) \tag{15}$$

$$\sigma^2(Q_\theta(s,a)) = \frac{1}{M}\sum_{i=1}^{M}(Q_{\theta^i}(s,a) - \mu(Q_\theta(s,a)))^2, \tag{16}$$

where by slightly abusing of notation, Q_θ represents the ensemble $\{Q_{\theta^i}\}_{i=1}^{M}$, and $\theta = \{\theta^i\}_{i=1}^{M}$ is the aggregation of parameters. As shown in [36], by setting a proper ensemble size M, $\sigma^2(Q_\theta)$ is effective in estimating the predictive uncertainty, and the estimation precision is even better than the benchmark Bayesian neural networks.

For each state-action pair, we empirically use Gaussian distributions to approximate the posterior distributions of estimated action values across multiple runs. As in OCBA, we adopt a non-informative prior distribution, and therefore the posterior distribution is estimated as $\mathcal{N}(\mu(Q_\theta(s,a)), \sigma^2(Q_\theta(s,a)))$. Then, maximizing the probability of identifying the best action with noisy action-value estimations falls into the scope of OCBA, where (14) provides an asymptotically optimal solution. Note that normalizing the results in (14) provides an allocation strategy that is independent of T, which implies that it actually provides a proportional relationship of the budget allocated to each action. Based on this observation, we normalize (14) to a sampling distribution $p_i = \frac{N_i}{T}$, with which the long-term allocation converges to the optimal solution. By replacing μ_i, σ_i in (14) with the ensemble-based estimations in (15) and (16), the probability of taking each action a under state s is obtained as

$$p_\theta(a|s) = \begin{cases} \dfrac{\sigma(Q_\theta(s,a^*))\tau_\theta(s)}{z_\theta(s)} & , \ a = a^* \\ \dfrac{\sigma^2(Q_\theta(s,a))}{\delta_\theta^2(s,a^*,a)z_\theta(s)} & , \ o.w. \end{cases}, \tag{17}$$

where

$$a^* = \underset{a}{\operatorname{argmax}}\, \mu(Q_\theta(s,a)), \tag{18}$$

$$\delta_\theta(s,a^*,a) = \mu(Q_\theta(s,a^*)) - \mu(Q_\theta(s,a)), \tag{19}$$

$$\tau_\theta(s) = \Big(\sum_{a\neq a^*} \frac{\sigma^2(Q_\theta(s,a))}{\delta_\theta^4(s,a^*,a)}\Big)^{\frac{1}{2}}, \tag{20}$$

and

$$z_\theta(s) = \sum_{a\neq a^*} \frac{\sigma^2(Q_\theta(s,a))}{\delta_\theta^2(s,a^*,a)} + \sigma(Q_\theta(s,a^*))\tau_\theta(s). \tag{21}$$

After that, we integrate the OCBA-based sampling strategy (17) into three off-policy algorithms to show the incorporation with RL.

A. OCBA-Based Q-Learning

Q-learning is an off-policy temporal-difference algorithm, which directly estimates the optimal action-value function Q^*. Different from the original ϵ-greedy policy, inspired by the concept of APCS, the proposed sampling strategy starts by estimating the decision confidence with

$$C_\theta(s) = 1 - \sum_{a\neq a^*} \int_{-\infty}^{\iota_\theta(s,a)} f(x)dx, \tag{22}$$

where $\iota_\theta(s,a) = \frac{-\delta_\theta(s,a^*,a)}{\sqrt{\sigma^2(Q_\theta(s,a))+\sigma^2(Q_\theta(s,a^*))}}$, a^* is the estimated best action defined in (18), and $f(\cdot)$ is the probability density function of standard Gaussian distribution. Since APCS provides a lower bound of PCS, $C_\theta(s)$ is used as a measurement of decision confidence. If $C_\theta(s)$ is larger than the threshold η, the agent will exploit a^*. Otherwise, the agent will explore actions to accumulate evidence for a reliable decision. In the latter situation, the OCBA-based sampling distribution in (17) will be adopted to accumulate evidence in a more efficient way.

Indeed, similar to ϵ-greedy, we perform an ϵ-OCBA policy

$$\tilde{p}_\theta(a|s) = \begin{cases} (1-\epsilon)\,p_\theta(a|s) + \dfrac{\epsilon}{|\mathcal{A}|} & , \ C_\theta(s) \leq \eta \\ (1-\epsilon)\,\mathrm{I}(a=a^*) + \dfrac{\epsilon}{|\mathcal{A}|} & , \ C_\theta(s) > \eta \end{cases} \tag{23}$$

for alleviating the model error, where $|\mathcal{A}|$ is the number of feasible actions, and $\mathrm{I}(e)$ is the indicator function taking value one if and only if event e occurs (otherwise taking value zero).

By recurrently updating the value estimations following

$$Q_{\theta^i}(s_t,a_t) = (1-\alpha_t(s_t,a_t))Q_{\theta^i}(s_t,a_t) + \alpha_t(s_t,a_t)y_t, \tag{24}$$

where $\alpha_t \in [0,1]$ satisfies (4), and

$$y_t = r_t + \gamma \max_{a'} \mu(Q_\theta(s_{t+1},a')), \tag{25}$$

all the action-value estimations converge to Q^*.

The pseudo-code is provided in Algorithm 1, and the analysis towards its convergence is given in Section IV. In this work, we finish the training process when the total sample budget $\mathcal{T}$ is used up.

B. OCBA-Based DQN

As an extension of Q-learning, DQN replaces the look-up tables with deep neural networks. Namely, each item in the

Algorithm 1 OCBA-Based Q-Learning

Randomly initialize $\{Q_{\theta^i}\}_{i=1}^{M}$; set the step size α_t, step counter $t=0$, parameter for exploration ϵ, and total sample budget $\mathcal{T}$; observe the initial state s_0.
repeat
  Sample action a_t with the OCBA-based sampling strategy $a_t \sim \tilde{p}_\theta(\cdot|s_t)$.
  Execute a_t and observe s_{t+1}, r_t.
  Update the action-value estimations with (24).
  Set the iteration counter $t \leftarrow t+1$.
until $t = \mathcal{T}$.

ensemble $\{Q_{\theta^i}\}_{i=1}^M$ is a separate neural network. Besides, OCBA-based DQN adopts the same sampling strategy as OCBA-based Q-learning, which firstly estimates the decision confidence with $C_\theta(s)$ and then decides to exploit a^* or explore actions with the OCBA-based sampling strategy. The overall sampling strategy is given in (23). In order to estimate the optimal action values, the agent alternates between interacting with the environment and updating the estimations. After each iteration, an experience $e_t = (s_t, a_t, r_t, s_{t+1})$ will be stored in the replay buffer $\mathcal{D}$. Then, a mini-batch $\mathcal{B}$ will be randomly sampled from $\mathcal{D}$ to update the action-value estimations. The loss for each neural network is given by the residual error

$$L(\theta^i) = \frac{1}{N}\sum_{e_t\in\mathcal{B}}(y_t - Q_{\theta^i}(s_t, a_t))^2, \tag{26}$$

where $y_t = r_t + \gamma \max_a \mu(Q_{\theta^-}(s_{t+1}, a))$, Q_{θ^-} is the ensemble of target networks $\{Q_{\theta^{i-}}\}_{i=1}^M$, and $\theta^- = \{\theta^{i-}\}_{i=1}^M$ is the aggregation of parameters. Finally, the neural networks are updated by performing stochastic gradient descent on $L(\theta^i)$ recurrently.

The pseudo-code is provided in Algorithm 2, where we incorporate experience replay and target networks to improve the performance. In order to stabilize training, in each time step, the target networks are slightly modified towards the current networks with a small step size κ.

Algorithm 2 OCBA-Based DQN

Randomly initialize $\{Q_{\theta^i}\}_{i=1}^M$, and set the target parameters $\theta^- \leftarrow \theta$; set the step size for target networks κ, replay buffer $\mathcal{D} = \varnothing$, step counter $t = 0$, parameter for exploration ϵ, and total sample budget $\mathcal{T}$; observe the initial state s_0.
repeat
Sample action a_t with the OCBA-based sampling strategy $a_t \sim \tilde{p}_\theta(\cdot|s_t)$.
Execute a_t and observe s_{t+1}, r_t.
Store $e_t = (s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$.
Update the action-value estimations by performing gradient descent on $L(\theta^i)$, which is defined in (26).
Update the target networks with

$$\theta^- \leftarrow (1-\kappa)\theta^- + \kappa\theta \tag{27}$$

Set the iteration counter $t \leftarrow t + 1$.
until $t = \mathcal{T}$.

C. OCBA-Based SAC

As a state-of-the-art actor-critic algorithm for handling MDPs with continuous actions, SAC maintains a separate actor to generate actions. Similar to that, OCBA-based SAC maintains an action-value ensemble $\{Q_{\theta^i}\}_{i=1}^M$ and a policy ensemble $\{\pi_{\phi^i}\}_{i=1}^M$, where $\pi_{\phi^i}(a|s)$ is the probability density of taking action a under state s. Correspondingly, the optimization process is divided into two phases, i.e., policy evaluation and policy improvement.

In policy-evaluation phase, each action-value estimation is updated towards minimizing the residual error

$$L(\theta^i) = \frac{1}{N}\sum_{e_t\in\mathcal{B}}\left(y_t^i - Q_{\theta^i}(s_t, a_t)\right)^2, \tag{28}$$

where

$$y_t^i = r_t + \gamma(\mu(Q_{\theta^-}(s_{t+1}, a_{t+1}^{i\prime})) - \nu \log \pi_{\phi^i}(a_{t+1}^{i\prime}|s_{t+1})), \tag{29}$$

and $a_{t+1}^{i\prime}$ is freshly sampled from $\pi_{\phi^i}(\cdot|s_{t+1})$. The policy ensemble makes it possible to apply OCBA-based sampling strategy (17) on the proposed action set. Namely, in each time step, we firstly collect a set of actions $\{a_t^i \sim \pi_{\phi^i}(\cdot|s_t)\}_{i=1}^M$ and then select one based on the OCBA-based sampling strategy.

In policy-improvement phase, each actor is updated by performing gradient ascent on the entropy-regularized action values

$$J(\phi^i) = \frac{1}{N}\sum_{e_t\in\mathcal{B}}(Q_{\theta^i}(s_t, a_t^{i\prime}) - \nu \log \pi_{\phi^i}(a_t^{i\prime}|s_t)), \tag{30}$$

where $a_t^{i\prime}$ is freshly sampled from $\pi_{\phi^i}(\cdot|s_t)$. By training each actor with a separate critic as in (30), we further enhance the diversity of policies to learn multi-modal behaviors.

The pseudo-code is provided in Algorithm 3, where the ϵ-OCBA policy is defined as

$$\tilde{p}_\theta(a_t^i|s_t) = \begin{cases} (1-\epsilon)\,p_\theta(a_t^i|s_t) + \dfrac{\epsilon}{M} & , \quad C_\theta(s_t) \le \eta \\ (1-\epsilon)\,\mathrm{I}(a_t^i = a_t^*) + \dfrac{\epsilon}{M} & , \quad C_\theta(s_t) > \eta \end{cases}, \tag{31}$$

where $a_t^* = \mathrm{argmax}_{a_t^i\in\mathcal{A}_t}\,\mu(Q_\theta(s_t, a_t^i))$ in this situation.

Algorithm 3 OCBA-Based SAC

Randomly initialize $\{Q_{\theta^i}\}_{i=1}^M$ and $\{\pi_{\phi^i}\}_{i=1}^M$; set the target parameters $\theta^- \leftarrow \theta$, step size for target networks κ, replay buffer $\mathcal{D} = \varnothing$, step counter $t = 0$, and total sample budget $\mathcal{T}$; observe the initial state s_0.
repeat
Collect actions $\mathcal{A}_t = \{a_t^i \sim \pi_{\phi^i}(\cdot|s_t)\}_{i=1}^M$.
Sample an action $a_t \sim \tilde{p}_\theta(\cdot|s_t)$ from $\mathcal{A}_t$.
Execute a_t and observe s_{t+1}, r_t.
Store $e_t = (s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$.
Update the critics by performing gradient descent on $L(\theta^i)$, which is defined in (28).
Update the actors by performing gradient ascent on $J(\phi^i)$, which is defined in (30).
Update the target networks with

$$\theta^- \leftarrow (1-\kappa)\theta^- + \kappa\theta. \tag{32}$$

Set the iteration counter $t \leftarrow t + 1$.
until $t = \mathcal{T}$.

IV. Theoretical Results

In this section, we discuss the convergence property of the proposed OCBA-based Q-learning algorithm. Compared with prior works [37], [45], which consider a single action-value estimation and bounded rewards, we study a more general case, where multiple action-value estimations are updated dependently, and the rewards are assumed to have bounded mean and variance. For ease of presentation, the notations are slightly different from prior sections. We replace the prior notation $Q_\theta = \{Q_{\theta^i}\}$ with $Q_t = \{Q_t^i\}_{i=1}^M$ to represent the estimated action values at time step t. Besides, for consistency of notations, we use p_t and $\tilde{p}_t$ to represent the same quantities defined in (17) and (23), respectively.

The main theorem is developed on the following lemmas.

Lemma 1: [46] The random process $\{\Delta_t\}$ taking values in $\mathbb{R}^n$ is defined as

$$\Delta_{t+1}(x) = (1-\alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x), \tag{33}$$

where $\alpha_t(x)$ is the step size for x at time step t, and F_t is a random process. Let $\mathcal{F}_t = \{F_i | \forall i < t\}$, then Δ_t converges to 0 w.p.1 if the following conditions are satisfied for all x.

1) $0 \le \alpha_t(x) \le 1$, $\sum_{t=0}^{\infty}\alpha_t(x) = \infty$, $\sum_{t=0}^{\infty}\alpha_t^2(x) < \infty$;
2) $\exists \vartheta < 1$, *so that* $|\mathbb{E}[F_t(x)|\mathcal{F}_t]| \le \vartheta ||\Delta_t||_\infty$;
3) $\exists \mathcal{C} > 0$, *so that* $var[F_t(x)|\mathcal{F}_t] \le \mathcal{C}(1 + ||\Delta_t||_\infty^2)$.

Lemma 2: [37] Given a finite MDP, $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, the Q-learning algorithm given by the update rule

$$\mathcal{Q}_{t+1}(s_t, a_t) = \mathcal{Q}_t(s_t, a_t) + \alpha_t(s_t, a_t)[y_t - \mathcal{Q}_t(s_t, a_t)], \tag{34}$$

where $y_t = r_t + \gamma \max_{a'} \mathcal{Q}_t(s_{t+1}, a')$, converges w.p.1 to the optimal action-value function as long as each state-action pair is performed infinitely often, and

$$\sum_t \alpha_t(s,a) = \infty \sum_t \alpha_t^2(s,a) < \infty \tag{35}$$

holds for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Lemma 1 gives a general criterion for the convergence of random process, based on which Lemma 2 establishes the convergence property of Q-learning. However, in the original proof of Lemma 2 [37], they assume the rewards to be bounded. In order to relax it to rewards with bounded mean and variance, e.g., Gaussian rewards, we give a modified proof in Appendix A.

Then, based on the lemmas, we provide the main theorem as follows.

Theorem 1: Give a finite MDP $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, whose rewards have bounded mean and variance. The OCBA-based Q-learning agent maintains an ensemble of action-value estimations $\{Q_t^i\}_{i=1}^M$ and selects actions following the ϵ-OCBA policy

$$\tilde{p}_t(a|s_t) = \begin{cases} (1-\epsilon)p_t(a|s_t) + \dfrac{\epsilon}{|\mathcal{A}|} & , \quad C_t(s_t) \le \eta \\ (1-\epsilon)\,\mathrm{I}(a = a^*) + \dfrac{\epsilon}{|\mathcal{A}|} & , \quad C_t(s_t) > \eta \end{cases} \tag{36}$$

where p_t is defined in (17). The estimations are updated with

$$Q_{t+1}^i(s_t, a_t) = (1-\alpha_t(s_t,a_t))Q_t^i(s_t,a_t) + \alpha_t(s_t,a_t)y_t, \tag{37}$$

where $y_t = r_t + \gamma \max_{a'} \mu(Q_t(s_{t+1}, a'))$, $\alpha_t(s,a) \in [0,1]$ takes non-zero values only on $(s,a) = (s_t, a_t)$, and (35) holds for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Then the estimations converge w.p.1 to the optimal action-value function, i.e.,

$$\lim_{t\to\infty} Q_t^i(s,a) = Q^*(s,a), \ \forall (s,a) \in \mathcal{S}\times\mathcal{A}, \ \forall i \in 1, \cdots, M. \tag{38}$$

Proof: Let us start by proving that $\mu(Q_t)$ converges to the optimal action-value function Q^*, which is defined in (1). By averaging the two hands of (37), we obtain the update rule for $\mu(Q_t)$, i.e.,

$$\begin{aligned} &\mu(Q_{t+1}(s_t,a_t)) \\ &\quad = (1-\alpha_t(s_t,a_t))\mu(Q_t(s_t,a_t)) + \alpha_t(s_t,a_t)y_t. \end{aligned} \tag{39}$$

Since $\tilde{p}_t(a|s) \ge \frac{\epsilon}{|\mathcal{A}|} > 0$ holds for all $(s,a) \in \mathcal{S}\times\mathcal{A}$, which implies that each reachable state-action pair will be performed infinitely often, it is easy to validate that $\mu(Q_t)$ converges to the optimal action-value estimation, i.e.,

$$\lim_{t\to\infty} \mu(Q_t(s,a)) = Q^*(s,a), \ \forall (s,a) \in \mathcal{S}\times\mathcal{A}, \tag{40}$$

by replacing $\mathcal{Q}_t$ in Lemma 2 with $\mu(Q_t)$.

Then, we prove that each action-value estimation converges to $\mu(Q_t)$. We first derive the update rule of $\sigma(Q_t)$ by

$$\begin{aligned} &\sigma^2(Q_{t+1}(s_t,a_t)) \\ &= \frac{1}{M}\sum_{i=1}^{M}(Q_{t+1}^i(s_t,a_t) - \mu(Q_{t+1}(s_t,a_t)))^2 \\ &= \frac{1}{M}\sum_{i=1}^{M}(1-\alpha_t(s_t,a_t))^2(Q_t^i(s_t,a_t) - \mu(Q_t(s_t,a_t)))^2 \\ &= (1-\alpha_t(s_t,a_t))^2\frac{1}{M}\sum_{i=1}^{M}(Q_t^i(s_t,a_t) - \mu(Q_t(s_t,a_t)))^2 \\ &= (1-\alpha_t(s_t,a_t))^2\sigma^2(Q_t(s_t,a_t)). \end{aligned} \tag{41}$$

Since $\alpha_t(s_t,a_t) \in [0,1]$, the update rule of $\sigma(Q_t)$ could be obtained as,

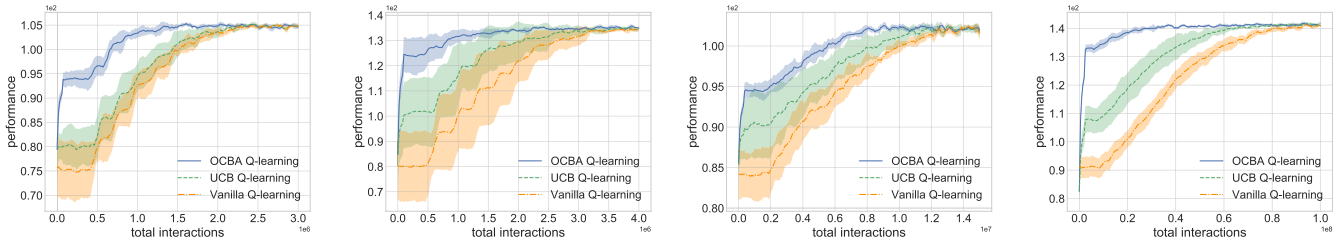
$$\sigma(Q_{t+1}(s_t,a_t)) = (1-\alpha_t(s_t,a_t))\sigma(Q_t(s_t,a_t)). \tag{42}$$

Further, by replacing Δ_t with $\sigma(Q_t)$ and setting $F_t(x) = 0$, Lemma 1 establishes the convergence property, i.e.,

$$\lim_{t\to\infty} \sigma(Q_t(s,a)) = 0, \ \forall (s,a) \in \mathcal{S}\times\mathcal{A}. \tag{43}$$

Based on (40) and (43), all of the action-value estimations converge to Q^*, which concludes the proof. □

To sum up, Theorem 1 establishes the convergence property of OCBA-based Q-learning. We remark that like in related works [37], [45], [47], [48], the convergence property is only established under the finite settings. However, as shown in most temporal-difference learning-based algorithms, e.g., [3], [27], [28], replacing the look-up tables with function approximations, e.g., neural networks, also show satisfying results.



(a) $|\mathcal{S}| = 10, |\mathcal{A}| = 5$ (b) $|\mathcal{S}| = 10, |\mathcal{A}| = 10$ (c) $|\mathcal{S}| = 30, |\mathcal{A}| = 10$ (d) $|\mathcal{S}| = 50, |\mathcal{A}| = 50$

Fig. 2. Testing results of vanilla Q-learning, UCB-based Q-learning (denoted as UCB Q-learning), and OCBA-based Q-learning (denoted as OCBA Q-learning). The lines and shaded regions represent the mean and standard deviation across five runs.

V. Experimental Results

In this section, we conduct several experiments to evaluate the effectiveness of the proposed OCBA-based sampling strategy. For the selection of ensemble size M, we give a qualitative analysis in Appendix B, based on which we set $M = 5$ for all the experiments. Besides, detailed hyper-parameters are provided in Appendix C.

Firstly, we compare the vanilla Q-learning [21], UCB-based Q-learning [34], and OCBA-based Q-learning on some MDPs with finite states and actions. Taking the MDP with state space $\mathcal{S} = \{s\}_{s=1}^{S}$ and action space $\mathcal{A} = \{a\}_{a=1}^{A}$ as example, the state transition probability is designed as

$$P(s'|s,a) = \frac{\exp(-20(g(s',s,a)/(S-1))^2)}{\sum_{\tilde{s}=1}^{S}\exp(-20(g(\tilde{s},s,a)/(S-1))^2)}, \quad (44)$$

where $g(\tilde{s},s,a) = \min_{j\in\mathbb{Z}} |\tilde{s} + jS - a + (A+1)/2|$, and $\mathbb{Z}$ is the set of integers. Besides, the reward is given by

$$R(s,a) = e^{-5((2s-S-1)/(S-1))^2} + 0.1e^{-5((2a-A-1)/(A-1))^2} + \varsigma, \quad (45)$$

where $\varsigma \sim \mathcal{N}(0, 0.1^2)$ is a Gaussian noise. We remark that UCB-based Q-learning adopts a similar ensemble-based uncertainty estimation but selects actions following $p_\theta(a|s_t) = (1-\epsilon)\,\mathrm{I}(a = a_t^*) + \frac{\epsilon}{|\mathcal{A}|}$, where $a_t^* = \mathrm{argmax}_a(\mu(Q_\theta(s_t,a)) + \alpha\sigma(Q_\theta(s_t,a)))$, and $\alpha > 0$ is a temperature coefficient. In the experiments, the sizes of decision spaces, i.e., $S \times A$, are set as 10×5, 10×10, 30×10, and 50×50, respectively. The learned policies are tested at every fixed interval, and the performance is evaluated by cumulative rewards. The results are shown in Fig. 2, where OCBA-based Q-learning shows the best performance in all tasks. Besides, OCBA-based Q-learning has the smallest variance, which implies that the OCBA-based sampling strategy makes the training process stable. Compared with UCB-based Q-learning, OCBA-based Q-learning converges faster, which is reasonable since the situation of identifying the best action for each state in RL is similar to R&S, where typically OCBA is more efficient. This also reveals that the OCBA-based sampling strategy makes a better tradeoff between exploration and exploitation. If the decision confidence is low, the OCBA-based sampling strategy provides an effective approach to accumulate more evidence for a reliable decision.

Then, we conduct experiments on four classic-control tasks, i.e., CartPole, MountainCar, Acrobot, and Spread, to evaluate OCBA-based DQN. The first three tasks are provided by OpenAI Gym [49], and the last is provided in [50]. Since these tasks have continuous states and discrete actions, we provide their dimension of states $d(\mathcal{S})$ and number of feasible actions $|\mathcal{A}|$ in TABLE II for a comparison. We set two baseline algorithms, SUNRISE (DQN version) [35] and vanilla DQN [3]. SUNRISE incorporates ensemble-based uncertainty estimation and UCB-based sampling strategy, and vanilla DQN [3] adopts ϵ-greedy sampling strategy. The results are shown in Fig. 3, where OCBA-based DQN outperforms the baseline algorithms in all the environments. Similar to the tabular situations in Fig. 2, OCBA-based DQN converges faster and performs more stable, which implies that both the ensemble-based predictive-uncertainty estimation and OCBA-based sampling strategy can be effectively expanded to the situations of nonlinear function approximations. Besides, the OCBA-based sampling strategy performs better in complex situations, e.g., Fig. 2d and Fig. 3d, which implies its potential to handle large-scale problems. Moreover, from these experiments, it can be found that the OCBA-based algorithms significantly accelerate training in the initial phase, which further benefits later iteration due to the sequential relationship among states. As a consequence, the OCBA-based sampling strategy shows advantages in reducing sampling efforts.

TABLE II
Descriptions of Decision Spaces

	CartPole	MountainCar	Acrobot	Spread
$d(\mathcal{S})$	4	2	6	12
$\lvert\mathcal{A}\rvert$	2	3	3	15

TABLE III
Descriptions of Decision Spaces

	HalfCheetah	Walker	Hopper	Ant
$d(\mathcal{S})$	17	17	11	27
$d(\mathcal{A})$	6	6	3	8

Finally, we evaluate OCBA-based SAC on some continuous-control tasks, i.e., HalfCheetah, Walker, Hopper, and Ant, which are provided by OpenAI Gym and MuJoCo simulator. Since these tasks have continuous states and actions, we provide their dimension of states $d(\mathcal{S})$ and dimension of actions $d(\mathcal{A})$ in TABLE III. We take eight state-of-the-art algorithms as baselines, including three model-based algorithms (PETS [53], POPLIN [52], and ME-TRPO [54]), two on-policy model-free algorithms (TRPO [29] and PPO [30]), two off-policy model-free algorithms (TD3 [28] and SAC [32]), and an ensemble-based algorithm (SAC-version SUNRISE [35]) which incorporates ensemble-based predictive-uncertainty estimation and UCB-based sampling strategy. The results are reported in TABLE IV. In the first three environments, OCBA-based SAC obtains the best scores, especially in HalfCheetah, where it surpasses other algorithms

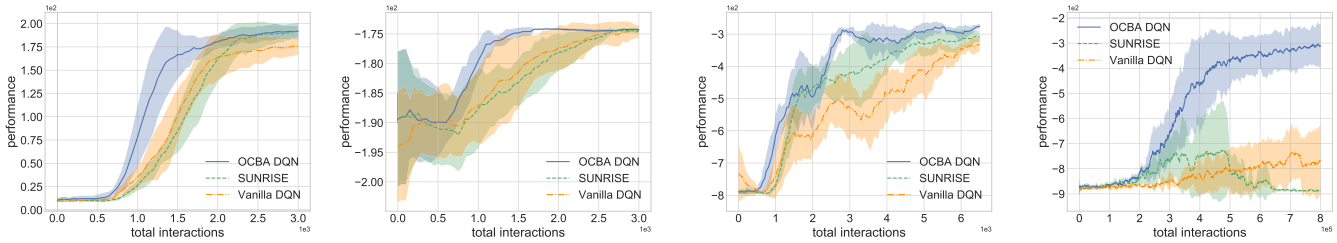


Fig. 3. Testing results of vanilla DQN, SUNRISE (DQN version), and OCBA-based DQN (denoted as OCBA DQN). The lines and shaded regions represent the mean and standard deviation across five runs.

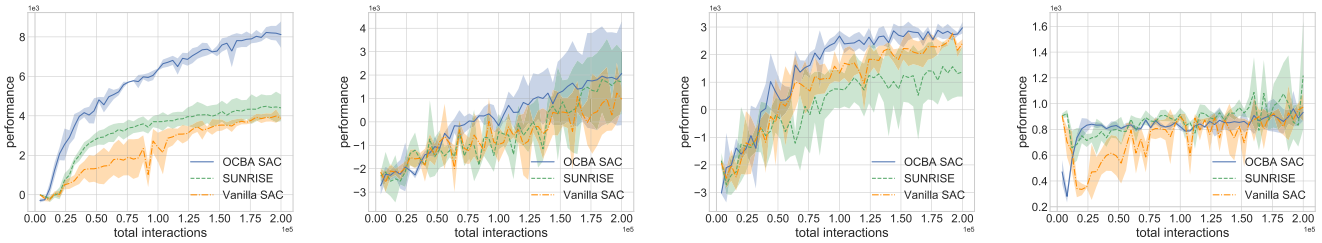


Fig. 4. Testing results of vanilla SAC, SUNRISE (SAC version), and OCBA-based SAC (denoted as OCBA SAC). The lines and shaded regions represent the mean and standard deviation across three runs.

TABLE IV
PERFORMANCE COMPARISON ON CONTINUOUS TASKS

	HalfCheetah	Walker2d	Hopper	Ant
Random	-288.3 ± 65.8	-2456.9 ± 345.3	-2572.7 ± 631.3	473.8 ± 40.8
PETS	2795.3 ± 879.9	312.5 ± 493.4	1125.0 ± 679.6	1852.1 ± 141.0
POPLIN	4235.0 ± 1133.0	597.0 ± 478.8	2055.2 ± 613.8	$\mathbf{2330.1} \pm 320.9$
ME-TRPO	2283.7 ± 900.4	-1609.3 ± 657.5	1272.5 ± 500.9	282.2 ± 18.0
TRPO	-12.0 ± 85.5	-2286.3 ± 373.3	-2100.1 ± 640.6	323.3 ± 24.9
PPO	17.2 ± 84.4	-1893.6 ± 234.1	-103.8 ± 1028.0	321.0 ± 51.2
TD3	3614.3 ± 82.1	-73.8 ± 769.0	2245.3 ± 232.4	956.1 ± 66.9
Vanilla SAC*	3875.4 ± 75.9	966.4 ± 427.7	2406.6 ± 148.0	972.6 ± 105.4
SUNRISE*	4403.7 ± 675.6	1699.5 ± 1361.3	1367.8 ± 889.4	1217.9 ± 418.5
OCBA SAC	$\mathbf{8115.8} \pm 657.8$	$\mathbf{2071.9} \pm 2210.3$	$\mathbf{2973.2} \pm 206.9$	932.7 ± 67.0

We report the score at 200K iterations averaged over three runs. For the baseline algorithms, we report the best results in prior works [35], [51], [52].
* Our implementations based on the code provided in [35].

by a remarkable margin. In Ant task, OCBA-based SAC does not obtain the best score but still performs similarly to TD3 and SAC, which are state-of-the-art off-policy model-free algorithms. In TABLE IV, compared with model-based algorithms, i.e., PETS, POPLIN, and ME-TRPO, OCBA-based SAC performs competitively from the aspect of sample efficiency. Empirically, model-based algorithms have higher sample efficiency but suffer from relatively worse asymptotic performance due to the bias of models. Therefore, OCBA-based SAC provides an effective approach that not only has competitive sample efficiency but executes in a model-free manner. Compared with the on-policy model-free algorithms, i.e., TRPO and PPO, OCBA-based SAC performs better in all the tasks, which validates the positive impact of the OCBA-based sampling strategy. In order to intuitively show the superior performance of OCBA-based SAC, we provide the learning curves in Fig. 4, where OCBA-based SAC shows remarkable gains towards the baseline algorithms in the first three tasks. In the last task, OCBA-based SAC and UCB-based SAC perform similarly, but compared with vanilla SAC, the proposed algorithm also shows a remarkable improvement in the initial training phases. These results verify the effectiveness of the OCBA-based sampling strategy in continuous control tasks.

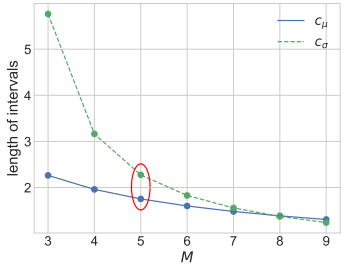


Fig. 5. Length of confidence intervals with respect to M.

To sum up, we implement the proposed algorithms on some benchmarks and compare them with baseline algorithms. It is shown that the OCBA-based sampling strategy significantly reduces the sampling efforts and meanwhile stabilizes training. Moreover, as an orthogonal technique, OCBA-based sampling

strategy can be easily superimposed with other techniques, e.g., dueling network [24], prioritized experience replay [55], and noisy net [56], to further improve the performance.

VI. Conclusion

In this work, we focus on the sample collection in RL and develop an OCBA-based sampling strategy. Firstly, we estimate the action values with ensembles, with which the predictive uncertainties can be estimated. Based on this, we develop an OCBA-based sampling strategy and integrate it with three modern off-policy algorithms, i.e., Q-learning, DQN, and SAC. Then, we establish the convergence property and evaluate its performance with several experiments. It is shown that the OCBA-based sampling strategy effectively reduces the sampling efforts and surpasses other ensemble-based algorithms by a remarkable margin.

To the best knowledge of authors, this work is the first one to incorporate OCBA-based sampling-effort allocation into ensemble-based RL algorithms. In current work, we only focus on off-policy RL algorithms. As for future works, we will take on-policy RL algorithms, e.g., TRPO and PPO, into consideration. Besides, it is also interesting to incorporate the OCBA-based sampling strategy into curiosity-driven algorithms, e.g., [57], and decentralized networked systems, e.g., [48], [58], [59], [60], where the predictive-uncertainty estimation is more complicated. We hope this work will shed light on related directions.

Appendix A
Modified Proof of Lemma 2

In this section, we prove that Lemma 2 holds for rewards with bounded mean and variance. Compared with the original work [37], which establishes the convergence property for bounded rewards, we consider a more general situation, where the variance term in Condition 3 of Lemma 1 cannot be easily bounded by a given constant. For the strictness of the theoretical analysis, we give a modified proof below.

Firstly, by defining

$$\Delta_t(s,a) = \mathcal{Q}_t(s,a) - Q^*(s,a) \tag{46}$$

and subtracting $Q^*(s_t,a_t)$ from both hands of (34), we have

$$\Delta_{t+1}(s_t,a_t) = (1-\alpha_t(s_t,a_t))\Delta_t(s_t,a_t) + \alpha_t(s_t,a_t)F_t(s_t,a_t), \tag{47}$$

where

$$F_t(s,a) = (r_t + \gamma \max_{a'} \mathcal{Q}_t(s_{t+1},a') - Q^*(s_t,a_t))\,\mathrm{I}(s=s_t, a=a_t). \tag{48}$$

Then, for all $(s,a) \in \mathcal{S}\times\mathcal{A}$,

$$\begin{aligned}
&|\mathbb{E}[F_t(s,a)|\mathcal{F}_t]| \le |\mathbb{E}[F_t(s_t,a_t)|\mathcal{F}_t]| \\
&\overset{①}{=}\gamma|\mathop{\mathbb{E}}_{s_{t+1}\sim P(\cdot|s_t,a_t)}(\max_{a'}\mathcal{Q}_t(s_{t+1},a) - \max_{a'}Q^*(s_{t+1},a'))| \\
&\le \gamma \mathop{\mathbb{E}}_{s_{t+1}\sim P(\cdot|s_t,a_t)}(|\max_{a'}\mathcal{Q}_t(s_{t+1},a') - \max_{a'}Q^*(s_{t+1},a')|) \\
&\le \gamma \mathop{\mathbb{E}}_{s_{t+1}\sim P(\cdot|s_t,a_t)}(\max_{a'}|\mathcal{Q}_t(s_{t+1},a') - Q^*(s_{t+1},a')|) \\
&\le \gamma \max_{s_{t+1},a'}|\mathcal{Q}_t(s_{t+1},a') - Q^*(s_{t+1},a')| = \gamma||\Delta_t||_\infty,
\end{aligned} \tag{49}$$

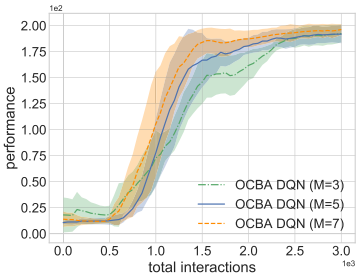


Fig. 6. The impact of M on OCBA-based DQN (denoted as OCBA DQN). The lines and shaded regions represent the mean and standard deviation across five runs.

where ① is obtained by substituting (2) in. Therefore, condition 2 in Lemma 1 is satisfied. Besides,

$$\begin{aligned}
&\mathrm{var}[F_t(s,a)|\mathcal{F}_t] \le \mathrm{var}[F_t(s_t,a_t)|\mathcal{F}_t] \\
&\overset{②}{=}\mathrm{var}(r_t) + \gamma^2\mathrm{var}[\max_{a'}\mathcal{Q}_t(s_{t+1},a')|\mathcal{F}_t] \\
&\quad = \mathrm{var}(r_t) + \gamma^2\mathrm{var}[\max_{a'}(\Delta_t(s_{t+1},a') + Q^*(s_{t+1},a'))|\mathcal{F}_t] \\
&\overset{③}{\le}\mathrm{var}(r_t) + \gamma^2\,\mathbb{E}[(\max_{a'}(\Delta_t(s_{t+1},a') + Q^*(s_{t+1},a')))^2|\mathcal{F}_t] \\
&\quad \le \mathrm{var}(r_t) + \gamma^2\,\mathbb{E}[(||\Delta_t||_\infty + ||Q^*||_\infty)^2|\mathcal{F}_t] \\
&\quad \le \mathrm{var}(r_t) + 2\gamma^2(||\Delta_t||_\infty^2 + ||Q^*||_\infty^2),
\end{aligned} \tag{50}$$

where ② holds since that r_t is independent of $\mathcal{F}_t$ and s_{t+1} given (s_t,a_t), and ③ holds since that for any random variable X, $\mathrm{var}(X) = \mathbb{E}X^2 - (\mathbb{E}X)^2 \le \mathbb{E}X^2$. Because the rewards have bounded mean and variance, both $\mathrm{var}(r_t)$ and $||Q^*||_\infty$ are bounded, which verifies condition 3 in Lemma 1. Finally, since condition 1 is naturally satisfied, Lemma 1 establishes the convergence property of Lemma 2.

Appendix B
Impact of Ensemble Size

In this section, we give qualitative analyses about the impact of ensemble size M, which can be seen from the following two aspects.

Firstly, we show the impact of M from the perspective of statistical inference. In the algorithms, we use M estimated action values to approximate the posterior distributions, which can be related to the situation of estimating the parameters of a Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$ with M samples. From the results in [61], the expected length of confidence intervals for estimated μ and σ are $c_\mu = \frac{2\sigma\mathcal{N}_{\upsilon/2}}{\sqrt{M}}$ and $c_\sigma = \sigma\sqrt{\frac{M-1}{\mathcal{X}^2_{M-1,1-\upsilon/2}}} - \sigma\sqrt{\frac{M-1}{\mathcal{X}^2_{M-1,\upsilon/2}}}$, respectively, where $\mathcal{N}_{\upsilon/2}$, $\mathcal{X}^2_{M-1,\upsilon/2}$, and $\mathcal{X}^2_{M-1,1-\upsilon/2}$ are quantiles of Gaussian and chi-square distributions. The lengths of confidence intervals with respect to M are shown in Fig. 5, where the improvement of increasing M gradually gets smaller. However, since the required computational resources increase linearly, we must select an appropriate ensemble size to make a tradeoff.

In practice, due to the noise of function approximations, e.g., neural networks, increasing the ensemble size over a threshold will not lead to a remarkable improvement. In order to intuitively show this, we take the CartPole task [49] and OCBA-based DQN algorithm as an example to show the practical impact of M. We set the ensemble size as $M = 3$,

$M = 5$, and $M = 7$, respectively, and the results are shown in Fig. 6. It can be found that $M = 5$ has shown a satisfying performance, and increasing it from $M = 5$ to $M = 7$ only shows a slight improvement. This is also pointed out in [36], where they show that $M = 5$ has provided effective estimations, especially when neural networks are adopted.

Based on above observations, to make a tradeoff between the performance and required computational resources, we choose $M = 5$ for all the following experiments.

APPENDIX C
DETAILS OF EXPERIMENTS

In this section, we provide detailed hyper-parameters for all the experiments.

Firstly, the following hyper-parameters are set as the same for all the experiments, including $M = 5$, $\epsilon = 0.1$, $\eta = 0.99$, and $\gamma = 0.99$. Then, for the hyper-parameters that arise in each group, we describe them as follows.

In the first group (Fig. 2), we set $\alpha_t = 0.01$. We remark that slightly different from the original step size that satisfies (35), we adopt a small constant step size, which is widely adopted in practice [3], [27], [28], [32].

In the second group (Fig. 3), we set $N = 64$, $\kappa = 0.01$, and learning rate $l = 0.001$. The neural networks have two fully connected hidden layers with Relu activation function, where each layer has $h = 64$ hidden nodes. The output layer is a fully connected layer with Identity activation function. Relu function is defined as $\mathcal{R}(x) = \max(x, 0)$, and Identity function is defined as $\mathcal{I}(x) = x$.

In the third group (Fig. 4), we set $N = 100$, $\nu = 0.2$, $\kappa = 0.005$, and $l = 0.001$. Both the critic and actor networks have two fully connected hidden layers with Relu activation function, where each layer has $h = 256$ hidden nodes. Besides, each critic network has a fully connected output layer with Identity activation function, and each actor network has two fully connected output layers with Identity activation function.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[2] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[3] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[4] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. AAAI*, Nov. 2015, pp. 1–9.

[5] Z. Liu et al., "Visuomotor reinforcement learning for multirobot cooperative navigation," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3234–3245, Oct. 2022.

[6] W. Zhang, K. Song, X. Rong, and Y. Li, "Coarse-to-fine UAV target tracking with deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1522–1530, Oct. 2019.

[7] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 29, no. 19, pp. 70–76, Jan. 2017.

[8] C.-H. Chen, S. E. Chick, L. H. Lee, and N. A. Pujowidianto, "Ranking and selection: Efficient simulation budget allocation," in *Handbook of Simulation Optimization*. New York, NY, USA: Springer, 2015, pp. 45–80.

[9] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Proc. Int. Conf. Algorithmic Learn. Theory*, Lyon, France, Oct. 2011, pp. 174–188.

[10] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, nos. 3–4, pp. 285–294, Dec. 1933.

[11] W. R. Thompson, "On the theory of apportionment," *Amer. J. Math.*, vol. 57, no. 2, pp. 450–456, 1935.

[12] Y. Li, M. C. Fu, and J. Xu, "An optimal computing budget allocation tree policy for Monte Carlo tree search," *IEEE Trans. Autom. Control*, vol. 67, no. 6, pp. 2685–2699, Jun. 2022.

[13] H.-C. Chen, C.-H. Chen, and E. Yucesan, "Computing efforts allocation for ordinal optimization and discrete event simulation," *IEEE Trans. Autom. Control*, vol. 45, no. 5, pp. 960–964, May 2000.

[14] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dyn. Syst.*, vol. 10, no. 3, pp. 251–270, Jul. 2000.

[15] J. Zhang, C. Wang, D. Zang, and M. Zhou, "Incorporation of optimal computing budget allocation for ordinal optimization into learning automata," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 1008–1017, Apr. 2016.

[16] Y. Peng, E. K. P. Chong, C.-H. Chen, and M. C. Fu, "Ranking and selection as stochastic control," *IEEE Trans. Autom. Control*, vol. 63, no. 8, pp. 2359–2373, Aug. 2018.

[17] S. E. Chick and K. Inoue, "New two-stage and sequential procedures for selecting the best simulated system," *Operations Res.*, vol. 49, no. 5, pp. 732–743, Oct. 2001.

[18] P. I. Frazier, W. B. Powell, and S. Dayanik, "A knowledge-gradient policy for sequential information collection," *SIAM J. Control Optim.*, vol. 47, no. 5, pp. 2410–2439, Jan. 2008.

[19] S. E. Chick, J. Branke, and C. Schmidt, "Sequential sampling to myopically maximize the expected value of information," *INFORMS J. Comput.*, vol. 22, no. 1, pp. 71–80, Feb. 2010.

[20] L. J. Hong, W. Fan, and J. Luo, "Review on ranking and selection: A new perspective," *Frontiers Eng. Manag.*, vol. 8, no. 3, pp. 321–343, Sep. 2021.

[21] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[22] H. V. Hasselt, "Double Q-learning," in *Proc. Neural Inf. Process. Syst. (NIPS)*, Vancouver, BC, Canada, vol. 23, Dec. 2010, pp. 2613–2621.

[23] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[24] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. ICML*, New York, NY, USA, Jun. 2016, pp. 1995–2003.

[25] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?" in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Montréal, QC, Canada, Dec. 2018, pp. 4868–4878.

[26] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proc. 15th Nat. Conf. Artif. Intell.*, Jul. 1998, pp. 761–768.

[27] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, May 2016, pp. 1–14.

[28] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jun. 2018, pp. 1587–1596.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, Lille, France, Jun. 2015, pp. 1889–1897.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[31] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2017, pp. 1352–1361.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 1861–1870.

[33] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[34] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "UCB exploration via Q-ensembles," 2017, *arXiv:1706.01502*.

[35] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 6131–6141.

[36] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Los Angeles, CA, USA, Dec. 2017, pp. 6405–6416.

[37] F. S. Melo, "Convergence of Q-learning: A simple proof," Inst. Syst. Robot., Zürich, Switzerland, Tech. Rep., pp. 1–4, 2001.

[38] G. A. Rummery and M. Niranjan, *On-Line Q-Learning Using Connectionist Systems*, vol. 37. Princeton, NJ, USA: Citeseer, 1994.

[39] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv:1910.07207*.

[40] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 1928–1937.

[41] M. Hessel et al., "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2018, pp. 3215–3222.

[42] A. Gosavi, *Simulation-Based Optimization*. Berlin, Germany: Springer, 2015.

[43] F. Gao, S. Gao, H. Xiao, and Z. Shi, "Advancing constrained ranking and selection with regression in partitioned domains," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 382–391, Jan. 2019.

[44] S. E. Chick, "Bayesian analysis for simulation input and output," in *Proc. 29th Conf. Winter Simul.*, 1997, pp. 253–260.

[45] K. Li and Q.-S. Jia, "Decentralized multi-agent reinforcement learning: An off-policy method," 2021, *arXiv:2111.00438*.

[46] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Comput.*, vol. 6, no. 6, pp. 1185–1201, Nov. 1994.

[47] S. Bhatnagar, "An actor–critic algorithm with function approximation for discounted cost constrained Markov decision processes," *Syst. Control Lett.*, vol. 59, no. 12, pp. 760–766, Dec. 2010.

[48] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 5872–5881.

[49] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

[50] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Los Angeles, CA, USA, Dec. 2017, pp. 1–12.

[51] T. Wang et al., "Benchmarking model-based reinforcement learning," 2019, *arXiv:1907.02057*.

[52] T. Wang and J. Ba, "Exploring model-based planning with policy networks," in *Proc. Int. Conf. Learn. Represent.*, May 2019, pp. 1–20.

[53] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Montréal, QC, Canada, Dec. 2018, pp. 4759–4770.

[54] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr. 2018, pp. 1–15.

[55] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent.*, May 2016, pp. 1–21.

[56] M. Fortunato et al., "Noisy networks for exploration," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr. 2018, pp. 1–21.

[57] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 488–489.

[58] J. Yan, C. Deng, and C. Wen, "Resilient output regulation in heterogeneous networked systems under Byzantine agents," *Automatica*, vol. 133, Nov. 2021, Art. no. 109872.

[59] J. Yan, F. Guo, and C. Wen, "False data injection against state estimation in power systems with multiple cooperative attackers," *ISA Trans.*, vol. 101, pp. 225–233, Jun. 2020.

[60] M. Brittain and P. Wei, "Scalable autonomous separation assurance with heterogeneous multi-agent reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 2837–2848, Oct. 2022.

[61] G. Casella and R. L. Berger, *Statistical Inference*. Bosto, MA, USA: Cengage, 2021.

Kuo Li (Graduate Student Member, IEEE) received the B.E. degree in automation from Tsinghua University, Beijing, China, in 2019, where he is currently pursuing the Ph.D. degree with the Center for Intelligent and Networked Systems (CFINS), Department of Automation.

His research interests include theories and applications of reinforcement learning and transfer learning in cyber-physical systems.

Xinze Jin (Student Member, IEEE) received the B.E. degree (Hons.) from Beihang University, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree with the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Tsinghua University, Beijing.

His research interests include safe reinforcement learning, event-based optimization, and their applications in cyber-physical systems.

Qing-Shan Jia (Senior Member, IEEE) received the B.E. degree in automation and the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 2002 and 2006, respectively.

He was a Visiting Scholar with Harvard University, Cambridge, MA, USA; The Hong Kong University of Science and Technology, Hong Kong; and the Massachusetts Institute of Technology, Cambridge, in 2006, 2010, and 2013, respectively. He is currently a Professor with the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University. His research interests include theories and applications of discrete-event dynamic systems (DEDSs) and simulation-based optimization of cyber-physical systems.

Dongchun Ren received the B.E. degree in automation from Nankai University, Tianjin, China, in 2008, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014.

He is currently a Researcher with the Department of Autonomous Driving, Meituan Group, Beijing. His research interests include the areas of robotics, with a focus on motion planning, behavior planning, and trajectory prediction.

Huaxia Xia (Member, IEEE) received the Ph.D. degree from the University of California, San Diego, in 2006.

He is currently a Chief Scientist with the Meituan Group and the Head of the Meituan Autonomous Driving Department. Before joining the Meituan Group, he worked as a Senior Software Engineer at Google and a Principal Architect at Baidu. His research interests include software infrastructure, artificial intelligence, and autonomous driving.