Effective Multi-Agent Deep Reinforcement Learning Control With Relative Entropy Regularization

Chenyang Miao, Yunduan Cui, *Member, IEEE*, Huiyun Li, *Senior Member, IEEE*, and Xinyu Wu, *Senior Member, IEEE*

***Abstract*— This paper focused on developing an effective Multi-Agent Reinforcement Learning (MARL) approach that quickly explores optimal control policies of multiple agents through interactions with unknown environments. Multi-Agent Continuous Dynamic Policy Gradient (MACDPP) was proposed to tackle the issues of limited capability and sample efficiency in the current MARL approaches. It alleviates the inconsistency of multiple agents' policy updates by introducing the relative entropy regularization to the Centralized Training with Decentralized Execution (CTDE) framework with the Actor-Critic (AC) structure. Evaluated by multi-agent cooperation and competition tasks and traditional control tasks including OpenAI benchmarks and robot arm manipulation, MACDPP demonstrates its significant superiority in learning capability and sample efficiency compared with both related multi-agent and widely implemented signal-agent baselines. It converges to 62% higher average return and uses 38% fewer samples compared with the suboptimal baseline over all tasks, indicating the potential of MARL in challenging control scenarios, especially when the number of interactions is limited. The open source code of MACDPP is available at https://github.com/AdrienLin1/MACDPP.**

***Note to Practitioners*—Learning proper cooperation strategy over multiple agents in complicated systems has been a challenge in the domain of Reinforcement Learning. Our work extends the traditional MARL approach FKDPP that has been successfully implemented in the real-world chemical plant by Yokogawa to the CTDE framework and AC structure that supports continuous actions. This extension significantly expands its range of applications from cooperative/competitive tasks to the joint control of one complex system while maintaining its effectiveness.**

***Index Terms*— Reinforcement learning, Multi-Agent Reinforcement Learning (MARL), robot learning.**

Manuscript received 13 March 2024; accepted 5 May 2024. This article was recommended for publication by Associate Editor M. H. Anisi and Editor M. Dotoli upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 62103403, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2020B515130004, and in part by Shenzhen Research Project KCXST20221021111210023. *(Corresponding author: Yunduan Cui.)*

Chenyang Miao is with the University of Chinese Academy of Sciences, Beijing 101408, China, and also with Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518000, China.

Yunduan Cui, Huiyun Li, and Xinyu Wu are with Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518000, China (e-mail: cuiyunduan@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2024.3398712.

Digital Object Identifier 10.1109/TASE.2024.3398712

I. Introduction

GUIDED by task-related reward functions, Reinforcement Learning (RL) provides an effective solution to autonomously explore and gradually learn optimal or near-optimal control strategies by iteratively interacting with the environment in the absence of task-specific prior knowledge [1], [2]. Utilizing the power of deep neural networks [3] to adapt abstract features from high-dimensional input states, RL has demonstrated superior performances than humans in various complex scenarios including board games [4], video games [5], and robot control [6]. Based on the successful implementations of single-agent RL approaches, people naturally attempt to develop Multi-Agent Reinforcement Learning (MARL) to effectively explore optimal control policies of large-scale systems. MARL has achieved promising results in a wide range of tasks [7], [8], [9], [10], [11]. These results reveal the potential of MARL in multiple unmanned systems cooperative navigation and multi-target tracking tasks [12], [13], [14], [15]. On the other hand, transferring RL from single-agent environments to multi-agent environments raises a new challenge: the environments affected by the joint actions from multiple agents become non-stationary, and each agent faces a moving-target problem while its optimal strategy strongly depends on the frequently changing policies of other agents. This characteristic not only breaks the Markov property of the environment but also greatly compromises the learning capability and convergence speed of traditional RL approaches designed for single agent [16].

Compared with the approaches like Independent Q-Learning (IQL) [17] that directly implemented single-agent RL approaches in multi-agent scenarios to separately explore independent polices [18], [19], MARL methods based on the Centralized Training with Decentralized Execution (CTDE) framework provides an appealing prospect for addressing the issue of non-stationary [20]. It enables the multiple agents to learn decentralized policies in a centralized end-to-end fashion: all agents are accessible to the global information during the training stage while the decision-making of each agent is independent during the interaction with the environment. From the perspective of the value function, Value-Decomposition Networks (VDN) [21] decomposed the value function for multiple agents under a CTDE framework and achieved better cooperation behaviors in a

simulation maze environment. QMIX [22] further proposed a network-based mixture strategy for multiple agents' value function based on VDN and enjoyed astonishing results performances in StarCraft Multi-Agent Challenge (SMAC) [23]. Based on the policy-based RL approaches, Multi-Agent Proximal Policy Optimization (MAPPO) [24] demonstrated better performances than traditional MARL methods in both Multi-Agent Particle Environment (MPE) and SMAC. Employing an Actor-Critic (AC) structure, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [25] combines the strengths of both value function-based and policy-based approaches and has achieved good learning performances in both cooperative and competitive tasks. Based on this approach, Multi-Agent TD3 (MATD3) [26] tackled the issue of overestimated value function with additional critic networks. The minimax algorithm was further introduced by MiniMax Multi-Agent Deep Deterministic Policy Gradient (M3DDPG) [27] for enhanced learning capability in mixed cooperative and competitive scenarios. Although CTDE-based MARL approaches have improved the learning capability of agents in terms of structure, the issue of inherent inconsistency in policy updates among multiple agents and its negative impact on learning performance has not been adequately addressed from an algorithmic standpoint.

The relative entropy in RL was explored by [28] from the perspective of policy while the early version of Proximal Policy Optimization (PPO) [29] employed it as a constraint of smooth policy update. By incorporating the relative entropy between the current and previous policies as a regularization term into the value function, Dynamic Policy Programming (DPP) [30] effectively constrains excessive policy updates in single-agent environments. Theoretically, DPP significantly reduces the estimated error of the value function with superior error bounds [31], [32]. In engineering applications of single-agent scenarios, DPP enjoyed a thorough exploration in the local state-action space thanks to the policy update constrained by the relative entropy, contributing to superior sample efficiency and robustness in several robot control tasks [33], [34]. As one of the pioneering works applying relative entropy regularized RL to multi-agent scenarios, Factorial Kernel Dynamic Policy Programming (FKDPP) [35] was proposed to control large-scale chemical plants with multiple agents. By constraining multiple agents' policy updates by the relative entropy, the behavior of each agent in FKDPP will not drastically change in a short period, preserving the Markov property. FKDPP outperformed the control strategy designed by human experts in production rate, profit, and safety on a simulated Vinyl Acetate Monomer (VAM) plant [36] and has been successfully implemented in a real-world chemical plant for 35 days.[1] Although this work fully indicates the great potential of relative entropy regularized MARL in real-world systems, FKDPP was developed based on neither deep neural networks nor CTDE framework and was limited in discrete action space without supporting the AC structure.

[1]This implementation of FKDPP was conducted by Yokogawa Electric Corporation and JSR Corporation. For more details, please see: https://www.yokogawa.com/news/press-releases/2022/2022-03-22/.

TABLE I
COMPARISON OF MACDPP AND THE RELATED MARL BASELINES

Approach	Relative Entropy	AC	CTDE	Cooperative & Competitive
IQL [17]	×	×	×	○
QMIX [22]	×	×	×	×
MAPPO [24]	×	×	○	×
MADDPG [25]	×	○	○	○
M3DDPG [27]	×	○	○	○
MATD3 [26]	×	○	○	○
FKDPP [35], [36]	○	×	×	×
MACDPP (ours)	○	○	○	○

These characteristics restrict its application scope in more challenging and flexible control scenarios.

This paper focuses on integrating the relative entropy regularization to the modern MARL under the CTDE framework in order to alleviate the inconsistency of multiple agents' policy updates in various control scenarios. According to the characteristics compared with MARL baselines in Table I, our proposed approach, Multi-Agent Continuous Dynamic Policy Gradient (MACDPP) can be seen as the first extension of DPP to both AC structure and CTDE framework. It naturally extends the power of FKDPP from kernel-based value function approximation and discrete action space to the neural networks-based AC structure and CTDE framework with not only the support of continuous action space but also superior learning capability and sample efficiency. MACDPP reduces the intractable computational burden of FKDPP in the actor network with continuous actions by Monte Carlo sampling and naturally obtains a superior exploration strategy based on the Mellowmax softmax operator. Evaluated by cooperative/competitive tasks in MPE environment and traditional control tasks where multiple agents collaborate to control one high-dimensional system, the proposed method successfully demonstrated superiority in both learning capability and sample-efficiency compared with various multi-agent and signal-agent RL baselines, indicating the potential of MACDPP not only in competing or cooperating in multi-agent unmanned systems but also in jointly controlling large-scale systems including chemical and power process control. The contributions of this paper can be summarized as:

1) Our work first attempts to integrate relative entropy regularization into the CTDE framework-based MARL to address the inherent inconsistency of multiple agents' policy updates from an algorithmic perspective. We propose a novel MARL approach that is compatible with both cooperative and competitive tasks in a multi-agent scenario, as well as large-scale systems collaboratively controlled by multiple agents.
2) As one natural extension of previous work FKDPP [35], [36] to the deep neural networks and the AC structure with continuous action space, the proposed MACDPP can be seen as a comprehensive upgrade of FKDPP with enhanced learning capability and control flexibility in jointly controlling large-scale systems.
3) The proposed method was evaluated by several benchmarks from MPE to traditional control tasks in terms of the learning capability and sample efficiency compared to both related CTDE framework-based MARL and

widely-applied single-agent RL approaches. We further analyzed the impact of relative entropy regularization in the CTDE framework-based MARL on convergence and control behaviors.

The remainder of this paper is organized as follows. Section II introduces the preliminaries of Markov games, MARL, and CTDE framework. Section III details the proposed approach MACDPP. The experimental results are presented in Section IV. Finally, Section V concludes this paper.

II. Preliminaries

A. Markov Games

Markov games are widely utilized to model a multi-agent environment satisfying partially observable Markov processes (POMDP). It is generally defined by a sextuple $(N, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. N represents the number of agents in the target environment, $\mathcal{S}$ defines the general state space. The locally observed state of the i-th agent is denoted as $\boldsymbol{s}_i$ which is a subset of the global observed state $\boldsymbol{s} \in \mathcal{S}$. The joint action is made up of the local actions from all agents $\boldsymbol{a} = \boldsymbol{a}_1 \times \cdots \times \boldsymbol{a}_N \in \mathcal{A}$. The subspace of each agent's action is presented as $\mathcal{A}_i$. The state transition probability over all agents is presented as $\mathcal{P}$. $\mathcal{R}$ is a set of reward functions for specific tasks, and each agent has its own reward function $R_i(\boldsymbol{s}_i'|\boldsymbol{s}_i, \boldsymbol{a}_i) \in \mathcal{R}$ based on its local state, action and the next step state. The discount factor $\gamma \in (0, 1]$ is utilized to gradually ignore the accumulative rewards in a long-term horizon.

Based on the Markov games, MARL introduces the value function V_{i,π_i} and the state-action value function Q_{i,π_i} to measure the long-term accumulative rewards obtained by the i-th agent under its policy:

$$V_{i,\pi_i}(\boldsymbol{s}_i) = \mathbb{E}_{\substack{\boldsymbol{s}_{t+1}\sim\mathcal{P} \\ \boldsymbol{a}_{i,t}\sim\pi_i}}\left[\sum_{t=0}^{\infty}\gamma^t R_i(\boldsymbol{s}_{i,t+1}|\boldsymbol{s}_{i,t}, \boldsymbol{a}_{i,t}) \mid \boldsymbol{s}_{i,0} = \boldsymbol{s}_i\right], \quad (1)$$

$$Q_{i,\pi_i}(\boldsymbol{s}_i, \boldsymbol{a}_i) = \mathbb{E}_{\substack{\boldsymbol{s}_{t+1}\sim\mathcal{P} \\ \boldsymbol{a}_{i,t}\sim\pi_i}}\left[\sum_{t=0}^{\infty}\boldsymbol{\gamma}^t R_i(\boldsymbol{s}_{i,t+1}|\boldsymbol{s}_{i,t}, \boldsymbol{a}_{i,t}) \mid \substack{\boldsymbol{s}_{i,0}=\boldsymbol{s}_i; \\ \boldsymbol{a}_{i,0}=\boldsymbol{a}_i}\right] \quad (2)$$

where the global state in the next time step $\boldsymbol{s}_{t+1}$ is determined by the current global state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$ under $\mathcal{P}$. Define $\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}$ as the probability of translating from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ under action $\boldsymbol{a}$ in a global perspective, the goal of each agent in MARL is to learn an optimal control policy to maximize its optimal value function following a Bellman equation:

$$V_i^*(\boldsymbol{s}_i) = \max_{\pi_i}\sum_{\substack{\boldsymbol{a}\in\mathcal{A} \\ \boldsymbol{s}'\in\mathcal{S}}}\pi_i(\boldsymbol{a}_i|\boldsymbol{s}_i)\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s}\boldsymbol{s}'}\big(R_i(\boldsymbol{s}_i'|\boldsymbol{s}_i, \boldsymbol{a}_i) + \gamma V_i^*(\boldsymbol{s}_i')\big). \quad (3)$$

B. Multi-Agent Reinforcement Learning in CTDE Framework

In the CTDE-framework based MARL like MADDPG [25], the AC structure is implemented to separately estimate the state-action value function and model control policy by critic network $\hat{Q}_i(\cdot, \boldsymbol{\theta}_i)$ and actor network $\hat{\pi}_i(\cdot, \boldsymbol{\phi}_i)$ for each agent where $\boldsymbol{\theta}_i$ and $\boldsymbol{\phi}_i$ are the corresponding parameters. In the centralized training process, All critic networks are globally updated with the shared observation information. Define one global training sample from sample set $\mathcal{D}$ as $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}')$ where $\boldsymbol{r} = [R_1(\boldsymbol{s}_1'|\boldsymbol{s}_1, \boldsymbol{a}_1), \ldots, R_N(\boldsymbol{s}_N'|\boldsymbol{s}_N, \boldsymbol{a}_N)]$ is the vector of reward signal for all agents, the i-th agent's critic networks receive the states and actions from all agents $\hat{Q}_i(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_i)$ and measure its own long-term reward of $R_i(\boldsymbol{s}_i'|\boldsymbol{s}_i, \boldsymbol{a}_i)$. Determining the global action of the next step by all agents according to their local actions $\boldsymbol{a}_i' = \hat{\pi}_i(\boldsymbol{s}_i', \boldsymbol{\phi}_i), \forall i = 1, \ldots, N$, the corresponding Temporal-Difference (TD) errors that guide the update of critic networks in gradient descent optimization is calculated following:

$$\mathcal{L}(\boldsymbol{\theta}_i) = \big(\hat{Q}_i(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}_i) - R_i(\boldsymbol{s}_i'|\boldsymbol{s}_i, \boldsymbol{a}_i) - \gamma\,\hat{Q}_i(\boldsymbol{s}', \boldsymbol{a}', \boldsymbol{\theta}_i)\big)^2. \quad (4)$$

The actor networks are updated to maximize the returns of the current critic networks based on the local observation of each agent. The corresponding gradient of error is defined as:

$$\nabla_{\boldsymbol{\phi}_i} \leftarrow \nabla_{\boldsymbol{a}_i^*=\hat{\pi}_i(\boldsymbol{s}_i,\boldsymbol{\phi}_i)}\hat{Q}_i(\boldsymbol{s}, \boldsymbol{a}^*, \boldsymbol{\theta}_i)\nabla_{\boldsymbol{\phi}_i}\hat{\pi}_i(\boldsymbol{s}_i, \boldsymbol{\phi}_i) \quad (5)$$

where $\boldsymbol{a}^*$ is selected by all agents with shared information. In the decentralized execution process, on the other hand, the control actions of each agent are determined by the actor only without the consideration of other agents.

III. Approach

In this section, the proposed method MACDPP was detailed. It naturally extended the relative entropy regularization term from DPP [30] and FKDPP [35] to the modern MARL under the CTDE framework and AC structure. The multi-agent critic networks regularized by relative entropy were introduced in Section III-A, and the corresponding actor that supports continuous actions was introduced in Section III-B. The factorization strategy of MACDPP for cooperative and competitive tasks was discussed in Section III-C with a summary of MACDPP's learning procedure.

A. Relative Entropy Regularized Critics

Following the existing relative entropy regularized RL approaches [30], [31], the difference between the current policy π_i and previous policy $\bar{\pi}_i$ of the i-th agent on state $\boldsymbol{s}_i$ was defined as:

$$\mathbb{D}_{\mathrm{KL}}(\boldsymbol{s}_i) = \sum_{\boldsymbol{a}\in\mathcal{A}}\pi_i(\boldsymbol{a}_i|\boldsymbol{s}_i)\log\left(\frac{\pi_i(\boldsymbol{a}_i|\boldsymbol{s}_i)}{\bar{\pi}_i(\boldsymbol{a}_i|\boldsymbol{s}_i)}\right). \quad (6)$$

This term was then incorporated into the value function as a regularization term controlled by a parameter η:

$$V_i'(\boldsymbol{s}_i) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\left(R_i(\boldsymbol{s}_{i,t+1}|\boldsymbol{s}_{i,t}, \boldsymbol{a}_{i,t}) - \frac{1}{\eta}\mathbb{D}_{\mathrm{KL}}(\boldsymbol{s}_{i,t})\right)\right]. \quad (7)$$

Combining Eqs.(3) and (7), the resulted optimal value function was still a Bellman equation with an additional term $-\frac{1}{\eta}\mathbb{D}_{\mathrm{KL}}(\boldsymbol{s}_{i,t})$ in Eq. (3). Assume the action of each agent $\boldsymbol{a}_i \in \mathcal{A}_i$ is discrete, an iterative update form of both value function and policy can be found based on DPP [30]:

$$V_{i,t+1}'(\boldsymbol{s}_i) = \frac{1}{\eta}\log\sum_{\boldsymbol{a}_i\in\mathcal{A}_i}\exp\big(\eta\cdot\Psi_{i,t}(\boldsymbol{s}_i, \boldsymbol{a}_i)\big), \quad (8)$$

$$\bar{\pi}_{i,t+1}(\boldsymbol{a}_i|\boldsymbol{s}_i) = \frac{\exp\big(\eta\cdot\Psi_{i,t}(\boldsymbol{s}_i, \boldsymbol{a}_i)\big)}{\sum_{\boldsymbol{a}_i'\in\mathcal{A}_i}\exp\big(\eta\cdot\Psi_{i,t}(\boldsymbol{s}_i, \boldsymbol{a}_i')\big)} \quad (9)$$

where t is the iteration of update, $\Psi_{i,t}(\cdot)$ is the action preferences function [1] which can be treat as a regularized Q function:

$$\Psi_{i,t}(\boldsymbol{s}_i,\boldsymbol{a}_i)=\sum_{s'\in\mathcal{S}}\mathcal{P}^{\boldsymbol{a}}_{ss'}\big(R_i(\boldsymbol{s}'_i|\boldsymbol{s}_i,\boldsymbol{a}_i)+\gamma V'_{i,t}(\boldsymbol{s}'_i)\big)+\frac{1}{\eta}\log\bar{\pi}_{i,t}(\boldsymbol{a}_i|\boldsymbol{s}_i). \tag{10}$$

With a discrete action space, once the critic network accurately estimates the action preferences function $\Psi_i(\cdot)$, both the value function and policy can be directly calculated. In practice, the transition probability matrix $\mathcal{P}$ is usually too large and inaccessible. DPP proposed an update rule of $\Psi_i(\cdot)$ based on sampling by inserting Eqs (8) and (9) into Eq. (10). Given a sample $(\boldsymbol{s},\boldsymbol{a},\boldsymbol{r},\boldsymbol{s}')$, the action preferences function of the i-th agent was updated following:

$$\Psi_{i,t+1}(\boldsymbol{s}_i,\boldsymbol{a}_i)=\Psi_{i,t}(\boldsymbol{s}_i,\boldsymbol{a}_i)-\mathcal{B}_\eta\Psi_{i,t}(\boldsymbol{s}_i)+R_i(\boldsymbol{s}'_i|\boldsymbol{s}_i,\boldsymbol{a}_i)+\mathcal{B}_\eta\Psi_{i,t}(\boldsymbol{s}'_i), \tag{11}$$

$$\mathcal{B}_\eta\Psi_{i,t}(\boldsymbol{s}_i)=\sum_{\boldsymbol{a}_i\in\mathcal{A}_i}\frac{\exp\big(\eta\cdot\Psi_{i,t}(\boldsymbol{s}_i,\boldsymbol{a}_i)\big)\Psi_{i,t}(\boldsymbol{s}_i,\boldsymbol{a}_i)}{\sum_{\boldsymbol{a}'_i\in\mathcal{A}_i}\exp\big(\eta\cdot\Psi_{i,t}\big(\boldsymbol{s}_i,\boldsymbol{a}'_i\big)\big)}. \tag{12}$$

$\mathcal{B}_\eta(\cdot)$ is a Boltzmann softmax operator.

It is straightforward to estimate the action preferences function instead of the Q function by the critic neural networks under the CTDE framework when the action is discrete. Let the critic networks receive the global information of all agents, the loss function of the critic networks $\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i)$ is calculated by integrating Eq. (11) into Eq. (4):

$$\mathcal{L}(\boldsymbol{\theta}_i)=\big(\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i)-y(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}'_i,\boldsymbol{\theta}_i^-)\big)^2, \tag{13}$$

$$y(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}',\boldsymbol{\theta}_i^-)=R_i(\boldsymbol{s}'_i|\boldsymbol{s}_i,\boldsymbol{a}_i)+\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)-\mathcal{B}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)+\gamma\mathcal{B}_\eta\hat{\Psi}_i(\boldsymbol{s}',\boldsymbol{\theta}_i^-) \tag{14}$$

where $\boldsymbol{\theta}_i^-$ is the parameters of the corresponding target networks. The Boltzmann softmax operator is conducted over the global action of all agents:

$$\mathcal{B}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)=\sum_{\boldsymbol{a}\in\mathcal{A}}\frac{\exp\big(\eta\cdot\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)\big)\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)}{\sum_{\boldsymbol{a}'\in\mathcal{A}}\exp\big(\eta\cdot\hat{\Psi}_i\big(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}_i^-\big)\big)}. \tag{15}$$

However, the application of the relative entropy regularization in the AC structure remains limited due to the intractable calculation over the whole continuous action space in $\mathcal{B}_\eta(\cdot)$. We detailed our solution in the next subsection.

B. Actors With Boltzmann Softmax Sampling

To effectively calculate Eq. (15) in continuous action space, we estimated it within a local range of the input action in MACDPP. The global action $\boldsymbol{a}$ was extended to a vector with $M+1$ Monte Carlo samples:

$$\mathcal{A}^{\mathrm{MC}}=[\boldsymbol{a},\boldsymbol{a}+\boldsymbol{e}^1,\boldsymbol{a}+\boldsymbol{e}^2,\ldots,\boldsymbol{a}+\boldsymbol{e}^M] \tag{16}$$

where $\boldsymbol{e}^j\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\zeta}^{\mathrm{MC}})$ for $j=1,\ldots,M$ is the Monte Carlo sampling noise controlled by $\boldsymbol{\zeta}^{\mathrm{MC}}$. The locally estimation of Eq. (12) therefore was calculated as:

$$\mathcal{B}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)\approx\sum_{\boldsymbol{a}\in\mathcal{A}^{\mathrm{MC}}}\frac{\exp\big(\eta\cdot\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)\big)\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)}{\sum_{\boldsymbol{a}'\in\mathcal{A}^{\mathrm{MC}}}\exp\big(\eta\cdot\hat{\Psi}_i\big(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}_i^-\big)\big)} \tag{17}$$

Algorithm 1 Estimation by Monte Carlo Sampling

Function `MC_Estimate`$(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i)$**:**
- # Build Monte Carlo sample set
- **for** $m=1$ *to* M **do**
  - $\boldsymbol{e}^m\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\zeta}^{\mathrm{MC}})$
- $\mathcal{A}^{\mathrm{MC}}=[\boldsymbol{a},\boldsymbol{a}+\boldsymbol{e}^1,\boldsymbol{a}+\boldsymbol{e}^2,\ldots,\boldsymbol{a}+\boldsymbol{e}^M]$
- # Locally estimate Mellowmax operation
- $\mathcal{M}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)\approx\frac{1}{\eta}\log\Big(\frac{\sum_{\boldsymbol{a}\in\mathcal{A}^{\mathrm{MC}}}\exp\left(\eta\cdot\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)-C\right)}{M+1}\Big)+C$
- Return $\mathcal{M}_\eta\hat{\Psi}_{\mathrm{target}}(\boldsymbol{s},\boldsymbol{\theta}^-)$

Algorithm 2 Exploration Using Boltzmann Softmax

Function `BS_Sampling`$(\boldsymbol{s},\boldsymbol{\theta}_i)$**:**
- **for** $j=1$ *to* N **do**
  - $\boldsymbol{a}_j^*=\hat{\pi}_j(\boldsymbol{s},\boldsymbol{\phi}_j)$
- $\boldsymbol{a}^*=[\boldsymbol{a}_1^*,\ldots,\boldsymbol{a}_N^*]$
- **for** $m=1$ *to* M **do**
  - $\boldsymbol{e}_i^m\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\zeta}^{\mathrm{Sampling}})$, $\boldsymbol{e}_{j\neq i}^m=\boldsymbol{0}$
  - $\boldsymbol{e}^m=[\boldsymbol{e}_1^m,\ldots,\boldsymbol{e}_N^m]$
- $\mathcal{A}^{\mathrm{explore}}=[\boldsymbol{a}^*,\boldsymbol{a}^*+\boldsymbol{e}^1,\boldsymbol{a}^*+\boldsymbol{e}^2,\ldots,\boldsymbol{a}^*+\boldsymbol{e}^M]$
- Select $\boldsymbol{a}^{\mathrm{explore}}\in\mathcal{A}^{\mathrm{explore}}$ following:
- $p(\boldsymbol{a}^{\mathrm{explore}})=\frac{\exp\left(\eta\cdot\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a}^{\mathrm{explore}},\boldsymbol{\theta}_i^-)\right)}{\sum_{\boldsymbol{a}'\in\mathcal{A}^{\mathrm{explore}}}\exp\left(\eta\cdot\hat{\Psi}_i\left(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}_i^-\right)\right)}$
- **return** $\boldsymbol{a}_i^{explore}$

According to [37], one critical issue of the Boltzmann softmax operator in RL is its multiple fixed points without non-expansion property which guarantees the convergence of " Q-learning-like" algorithms to a unique fixed point in theory. One effective solution is to replace it with the Mellowmax operator with a unique fix point and non-expansion property:

$$\mathcal{M}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)\approx\frac{1}{\eta}\log\left(\frac{\sum_{\boldsymbol{a}\in\mathcal{A}^{\mathrm{MC}}}\exp\big(\eta\cdot\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)\big)}{M+1}\right). \tag{18}$$

Algorithm 1 summarized the calculation of $\mathcal{M}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)$. In practice, numerical issues usually arise in Eq. (18) with a large η. We alternatively calculated it following:

$$\mathcal{M}_\eta\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{\theta}_i^-)\approx\frac{1}{\eta}\log\left(\frac{\sum_{\boldsymbol{a}\in\mathcal{A}^{\mathrm{MC}}}\exp\big(\eta\cdot\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_i^-)-C\big)}{M+1}\right)+C \tag{19}$$

where $C=\eta\cdot\max_{\boldsymbol{a}'\in\mathcal{A}^{\mathrm{MC}}}[\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a}',\boldsymbol{\theta}_i^-)]$.

Employing the Monte Carlo sampling to estimate $\mathcal{M}_\eta\hat{\Psi}_i(\cdot)$, any policy network maps the local states to the local actions can be used as an actor. In the decentralized execution process, the trained agent made decisions based on its own actor with local observation $\boldsymbol{a}_i^*=\hat{\pi}_i(\boldsymbol{s}_i,\boldsymbol{\phi}_i)$ The actor was updated in the centralized training following the gradient below:

$$\nabla_{\boldsymbol{\phi}_i}\leftarrow\nabla_{\boldsymbol{a}_i^*=\hat{\pi}_i(\boldsymbol{s}_i,\boldsymbol{\phi}_i)}\hat{\Psi}_i(\boldsymbol{s},\boldsymbol{a}^*,\boldsymbol{\theta}_i)\nabla_{\boldsymbol{\phi}_i}\hat{\pi}_i(\boldsymbol{s}_i,\boldsymbol{\phi}_i) \tag{20}$$

where the $\boldsymbol{a}^*$ was jointly calculated by all agents' policies.

Unlike MADDPG which explores by directly adding noises to its actions, MACDPP proposed an effective exploitation that naturally related to the relative entropy regularized value function based on the shared information in the centralized training process. Define the global action as $\boldsymbol{a}^* = [\boldsymbol{a}_1^*, \ldots, \boldsymbol{a}_N^*]$ where $\boldsymbol{a}_i^* = \hat{\pi}_i(\boldsymbol{s}_i, \boldsymbol{\phi}_i), i = 1, \ldots N$. An exploration set for the i-th agent with $M+1$ candidates was built following Algorithm 2: $\mathcal{A}^{\text{explore}} = [\boldsymbol{a}^*, \boldsymbol{a}^* + \boldsymbol{e}^1, \boldsymbol{a}^* + \boldsymbol{e}^2, \ldots, \boldsymbol{a}^* + \boldsymbol{e}^M]$ where exploration noises $\boldsymbol{e}^m$ added Gaussian noises only to the local actions related to the i-th agent. Please note that the variance of sampling $\boldsymbol{\zeta}^{\text{Sampling}}$ which affected the decision-making of the agent was independent of $\boldsymbol{\zeta}^{\text{MC}}$ in Algorithm. 1 which locally estimated the Mellowmax operation. An effective exploration action was randomly selected following the probability below:

$$p(\boldsymbol{a}^{\text{explore}}) = \frac{\exp\left(\eta \cdot \hat{\Psi}_i(\boldsymbol{s}, \boldsymbol{a}^{\text{explore}}, \boldsymbol{\theta}_i^-)\right)}{\sum_{\boldsymbol{a}' \in \mathcal{A}^{\text{explore}}} \exp\left(\eta \cdot \hat{\Psi}_i\left(\boldsymbol{s}, \boldsymbol{a}', \boldsymbol{\theta}_i^-\right)\right)}. \tag{21}$$

The i-th agent utilized the corresponding local action $\boldsymbol{a}_i^{\text{explore}}$ in $\boldsymbol{a}^{\text{explore}}$ to interact with the environment. Although Eq. (21) required the global actions of all agents for an effective exploration, the execution of all agents can be decentralized in evaluation since the deterministic local action $\boldsymbol{a}_i^* = \hat{\pi}_i(\boldsymbol{s}_i, \boldsymbol{\phi}_i)$ was related to only the local observation $\boldsymbol{s}_i$.

C. Factorization of Multi-Agents in Different Tasks

In this subsection, we detailed the training procedure of MACDPP in both multi-agent cooperative/competitive environments and single systems that are collaboratively controlled by multiple agents. The learning process of the proposed method in a multi-agent cooperative/competitive scenario was summarized in Algorithm 3. Given the length of episode E and the length of one rollout T, at the beginning, the parameters of both critic and actor networks were randomly initialized as $\boldsymbol{\theta}_i, \boldsymbol{\phi}_i, i = 1, \ldots, N$. Those weights were copied to the target networks as $\boldsymbol{\theta}_i^-, \boldsymbol{\phi}_i^-$. At each step, the global state $\boldsymbol{s}$ was first observed. The control action of each agent $\boldsymbol{a}_i$ was determined by BS_Sampling$(\cdot)$ following Algorithm 2. Conducted global action $\boldsymbol{a} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$ by all agents, the global state in the next step and the vector of N reward functions $\boldsymbol{r} = [R_1(\boldsymbol{s}_1'|\boldsymbol{s}_1, \boldsymbol{a}_1), \ldots, R_N(\boldsymbol{s}_N'|\boldsymbol{s}_N, \boldsymbol{a}_N)]$ were then observed and stored to the separate replay buffer. During the centralized training buffer, the update of each agent was separately conducted with its own J mini-batch samples while the samples from other agents' buffers were used to restore the global information. The TD error was calculated following Eq. (13) and Algorithm 1 to updated actor and critic networks. The target networks were then smoothly updated with a smooth parameter τ according to $\boldsymbol{\theta}_i, \boldsymbol{\phi}_i, i = 1, \ldots, N$.

When implementing the proposed MACDPP to jointly control one complex system by multiple agents following our previous work [35], [36], only one global replay buffer $\mathcal{D}$ was built. At each step, the global observed state $\boldsymbol{s}$ was sent in parallel to all agents. The control actions of all actors were then integrated as $\boldsymbol{a}$ and conducted to the target system. The resulting next step state $\boldsymbol{s}'$ and the corresponding reward were received and stored in $\mathcal{D}$. Please note that all agents shared one reward function $\boldsymbol{r} = [R(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})]$ designed for the whole system. Unlike the case in charge of multi-agent cooperative/competitive environments, MACDPP did not separately conduct J mini-batch sampling for each agent when jointly controlling one system but rather shared J samples during the update. In addition, the actor network of the i-th agent $\hat{\pi}_i(\boldsymbol{s}, \boldsymbol{\phi}_i)$ directly received the global observed state in the decision-making process. The difference between the learning process of MACDPP in cooperative/competitive and joint control scenarios was illustrated in Fig. 1.

Algorithm 3 Learning Process of MACDPP

for $i = 1$ *to* N **do**
  Initialize buffer D_i, networks weights $\boldsymbol{\theta}_i$, $\boldsymbol{\phi}_i$.
  Copy the target networks with parameters $\boldsymbol{\theta}_i^-$, $\boldsymbol{\phi}_i^-$.
for $e = 1$ *to* E **do**
  for $t = 1$ *to* T **do**
    # Interaction phase
    Observe state $\boldsymbol{s}$
    for $i = 1$ *to* N **do**
      $\boldsymbol{a}_i = \text{BS\_Sampling}(\boldsymbol{s}, \boldsymbol{\theta}_i^-)$
    Execute $\boldsymbol{a} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$
    Observe next state $\boldsymbol{s}'$ and reward $\boldsymbol{r}$
    Separately store sample to $\mathcal{D}_i, i = 1, \ldots, N$
    # Centralized training phase
    for $i = 1$ *to* N **do**
      Sample mini-batch of J samples from $\mathcal{D}_i$
      for $j = 1$ *to* J **do**
        # Restore information from $\mathcal{D}_k, k \neq i$
        $\boldsymbol{s}_j = [\boldsymbol{s}_{1,j}, \ldots, \boldsymbol{s}_{N,j}]$
        $\boldsymbol{a}_j = [\boldsymbol{a}_{1,j}, \ldots, \boldsymbol{a}_{N,j}]$
        $\boldsymbol{s}_j' = [\boldsymbol{s}_{1,j}', \ldots, \boldsymbol{s}_{N,j}']$
        # Calculate the next action of all agents
        for $k = 1$ *to* N **do**
          $\boldsymbol{a}_{k,j}' = \hat{\pi}_k(\boldsymbol{s}_k', \boldsymbol{\phi}_k^-)$
        $\boldsymbol{a}_j' = [\boldsymbol{a}_{1,j}', \ldots, \boldsymbol{a}_{N,j}']$
        # Monte Carlo Estimation
        $\mathcal{M}_\eta \hat{\Psi}_i(\boldsymbol{s}_j) = \text{MC\_Estimate}(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}_i^-)$
        $\mathcal{M}_\eta \hat{\Psi}_i(\boldsymbol{s}_j') = \text{MC\_Estimate}(\boldsymbol{s}_j', \boldsymbol{a}_j', \boldsymbol{\theta}_i^-)$
        # Calculate TD error following Eq. (14)
        $y_j = R_i(\boldsymbol{s}_{i,j}'|\boldsymbol{s}_{i,j}, \boldsymbol{a}_{i,j}) + \gamma \mathcal{M}_\eta \hat{\Psi}_i(\boldsymbol{s}_j') + \hat{\Psi}_i(\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{\theta}_i^-) - \mathcal{M}_\eta \hat{\Psi}_i(\boldsymbol{s}_j)$
      # Update critic $\hat{\Psi}_i(\cdot, \boldsymbol{\theta}_i)$ and actor $\hat{\pi}_i(\cdot, \boldsymbol{\phi}_i)$
      $\boldsymbol{\theta}_i \leftarrow \arg\min_{\boldsymbol{\theta}_i} \frac{1}{J} \sum_{j=1}^{J} (y_j - \hat{\Psi}(\boldsymbol{s}_j, \boldsymbol{a}_{i,j}, \boldsymbol{\theta}_i))^2$
      # Update actor $\hat{\pi}_i(\cdot, \boldsymbol{\phi}_i)$
      $\nabla_{\boldsymbol{\phi}_i} \leftarrow \frac{\sum_{j=1}^{J} \nabla_{\boldsymbol{a}_{i,j}'} \hat{\Psi}(\boldsymbol{s}_j, \boldsymbol{a}_j', \boldsymbol{\theta}) \nabla_{\boldsymbol{\phi}_i} \hat{\pi}_i(\boldsymbol{s}_{i,j}, \boldsymbol{\phi}_i)}{J}$
      # Update target networks
      $\boldsymbol{\theta}_i^- \leftarrow \tau \boldsymbol{\theta}_i + (1-\tau)\boldsymbol{\theta}_i^-$
      $\boldsymbol{\phi}_i^- \leftarrow \tau \boldsymbol{\phi}_i + (1-\tau)\boldsymbol{\phi}_i^-$
return $\hat{\Psi}_i, \hat{\pi}_i, i = 1, \ldots, N$

IV. Experimental Results

A. Experimental Settings

In this section, we evaluated MACDPP in multi-agent and traditional control tasks in terms of learning capability and

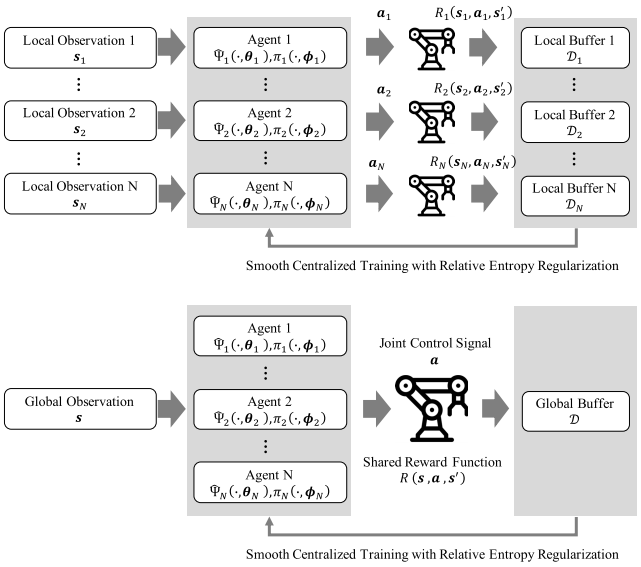


Fig. 1. The differences between the learning process of MACDPP in cooperative/competitive (top) and joint control (bottom) scenarios.

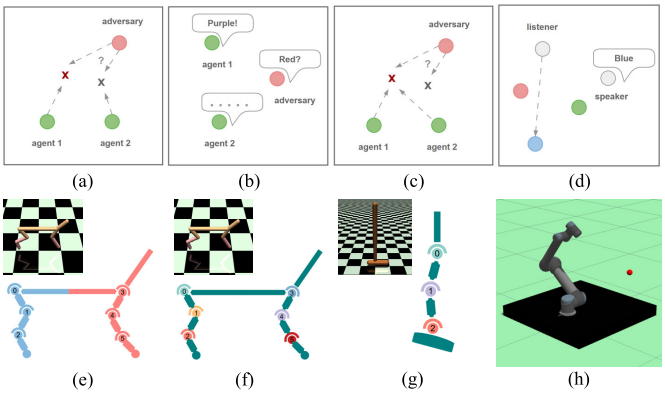


Fig. 2. Eight Benchmark tasks for evaluation: (a) Physical Deception from MPE (mixed cooperative competitive); (b) Covert Communication from MPE (mixed cooperative competitive); (c) Keep Away from MPE (mixed cooperative competitive); (d) Cooperative Communication from MPE (pure cooperative); (e) HalfCheetah from Mujoco (joint control by two agents); (f) HalfCheetah from Mujoco (joint control by six agents); (g) Hopper from Mujoco (joint control by three agents); (h) UR5 End Effector Positioning from robo-gym (joint control by five agents).

sample efficiency. For the multi-agent scenario, we selected the physical deception, Covert Communication, keep away and cooperative communication tasks from the MPE [25].[2] The first three are mixed cooperative-competitive tasks and the last one is a pure cooperative task. Following the top of Fig. 2, the Physical Deception task trains two agents (green) to cooperate. One agent aims to reach the actual target while another one attempts to mislead the adversary (red) into reaching an incorrect target. In the Covert Communication task, the two agents are trained to communicate through a private key generated randomly at the beginning of each episode, a negative reward is obtained when the adversary reconstructs the message. In the Keep Away task, two agents are trained to push the adversary as far as possible from the target. The Cooperative Communication task trains the listener to reach the target following the speaker's instructions. MADDPG [25], MATD3 [26] and M3DDPG [27] were selected as the compared MARL baselines.

We further test the proposed method by the HalfCheetah, the Hopper tasks from Mujoco simulation [38] and the UR5 robot arm simulation task ur_ee_position developed in robo-gym [39] to evaluate its performances in more traditional control scenarios. For the compared MARL approaches including MACDPP and MADDPG, the HalfCheetah was divided into two scenarios: two agents separately controlled the front and back body, and six agents controlled six joints. The Hopper and UR5 were jointly controlled by three and five agents for each controllable joint. All benchmark control tasks were illustrated at the bottom of Fig. 2. This experiment verifies the potential of MACDPP in the application of real-world robots. The results of the HalfCheetah and Hopper tasks reflect the MACDPP's effectiveness on quadruped robots while the simulation results on robo-gym simulation can be quickly applied to the real-world UR5 robot with less engineering cost.

We not only compared the proposed method with MARL approaches MADPPG [25] but also the widely implemented single-agent RL approaches including Deep Deterministic Policy Gradient (DDPG) [40], Twin Delayed Deep Deterministic Policy Gradient (TD3) [41] and Soft Actor-Critic (SAC) [42]. The hyperparameters of all compared methods for each task were summarized in Table II. All actors and critics shared the same network structures. The tunable hyperparameters of the proposed MACDPP including η, the Monte Carlo sampling numbers in Algorithms 1 and 2 M_1, M_2 and the sampling noise $\zeta^{\mathrm{MC}}, \zeta^{\mathrm{Sampling}}$ were listed in Table III. The proposed MACDPP was developed by PaddlePaddle [43] under its RL toolkit PARL.[3] All experiments were conducted on a workstation with Intel Xeon W2265 CPU, NVIDIA GeForce RTX 3080 GPU, 64GB memory and Ubuntu 20.04 OS. The experimental results were summarized over five independent trials with different random seeds for statistical evidence. The open source code of MACDPP is available at https://github.com/AdrienLin1/MACDPP.

B. Cooperation and Competition in MPE Benchmarks

1) Evaluation of the Learning Capability: We first compared the proposed methods with the related MARL approaches in four benchmark tasks from the MPE environment. The learning curves of all compared approaches were shown in Fig. 3 while the maximum average returns of each method in the evaluation phase during the learning were listed in Table IV (the number in red indicates the best result in the corresponding term). The trials of all four tasks were conducted by 25k episodes, each episode had 25 steps.

In the Physical Deception task, MADDPG, MATD3 and M3DDPG converged to close performances near 0 average return after 25k episodes. As a comparison, our method quickly suppressed other baselines in the first 2000 episodes and converged to 40 average return after 25k episodes.

[2]https://github.com/openai/multiagent-particle-envs

[3]https://github.com/PaddlePaddle/PARL

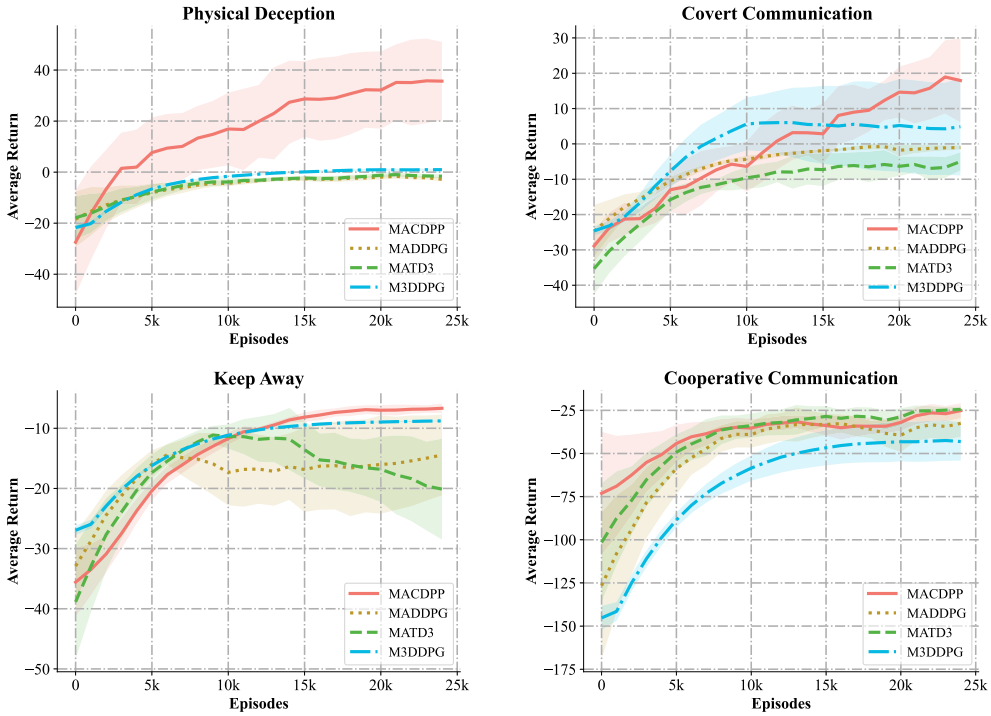


Fig. 3. Learning curves of MACDPP and other MARL baselines in MPE benchmark tasks. The shaded region represents the corresponding standard deviation over five trials.

TABLE II
COMMON HYPERPARAMETER SETTINGS OF COMPARED APPROACHES

	Physical Deception	Covert Communication	Keep Away	Cooperative Communication	HalfCheetah	Hopper	UR5
Critic Learning Rate	0.01	0.01	0.1	0.1	10^{-3}	5×10^{-4}	5×10^{-4}
Actor Learning Rate	0.01	0.01	0.01	0.01	10^{-3}	5×10^{-5}	5×10^{-5}
Actor and Critic Structure	$(64, 64)$	$(64, 64)$	$(64, 64)$	$(64, 64)$	$(400, 300)$	$(400, 300)$	$(400, 300)$
Target Update Rate (τ)	1×10^{-3}	1×10^{-4}	1×10^{-4}	1×10^{-4}	5×10^{-3}	5×10^{-3}	5×10^{-3}
Batch Size (J)	1024	1024	1024	1024	100	100	100
Discount Factor (γ)	0.95	0.95	0.95	0.95	0.99	0.99	0.99
Memory Size	10^5	10^5	10^5	10^5	10^6	10^6	10^6
Warmup Steps	0	0	0	0	10^4	10^4	10^4
Steps per Update	100	50	25	25	1	1	1

TABLE III
SPECIFIC HYPERPARAMETERS OF MACDPP

	η	M_1	M_2	ς^{MC}	$\varsigma^{\text{Sampling}}$
Physical Deception	20	30	50	**0.1**	**0.2**
Covert Communication	20	30	50	**0.1**	**0.2**
Keep Away	0.1	30	50	**0.1**	**0.1**
Cooperative Communication	0.1	30	50	**0.1**	**0.1**
HalfCheetah	20	30	50	**0.1**	**0.1**
Hopper	5	30	50	**0.1**	**0.1**
UR5	0.05	30	50	**0.1**	**0.1**

In the evaluation task, MACDPP outperformed MATD3 and M3DDPG with over 7000% and 1200% maximum average return while MADDPG achieved a negative average return. In the Covert Communication task, M3DDPG outperformed MADDPG and MATD3 in both average return and convergence speed thanks to its Minimax operator. On the other hand, MACDPP converged to over 100% more average return during learning and achieved 135% maximum average in evaluation compared with the suboptimal method M3DDPG. In the Keep Away task, MADDPG converged to a relatively low average return. MATD3 converged to near -10 average return within 10k episodes but could not maintain its performance. Although M3DDPG converged quickly in the first 10k episodes, MACDPP outperformed it with 5% more maximum average return and the lowest standard deviation in the learning curve which indicated a more stable training procedure. In the Cooperative Communication task which requires pure cooperation over all agents, it is observed that M3DDPG which is good at competition failed to learn good cooperation policies. Although MACDPP, MADDPG and MATD3 all learned to a close performance, the proposed method enjoyed the fastest convergence speed. Overall, the proposed method demonstrated significantly superior learning capability than related MARL baselines in various MPE benchmark tasks.

2) Evaluation of the Sample Efficiency: The sample efficiency which is important to the implementation of RL in real-world systems was evaluated in Fig. 4. We define the

TABLE IV
MAXIMUM AVERAGE RETURNS OVER FOUR MPE BENCHMARK TASKS

	MACDPP	MADDPG	MATD3	M3DDPG
Physical Deception	$\mathbf{47.02 \pm 14.28}$	-0.36 ± 2.20	0.67 ± 2.63	3.77 ± 0.23
Covert Communication	$\mathbf{31.68 \pm 19.07}$	0.72 ± 2.71	1.76 ± 8.78	13.48 ± 10.52
Keep Away	$\mathbf{-5.27 \pm 1.27}$	-6.27 ± 1.01	-5.55 ± 1.33	-8.53 ± 0.10
Cooperative Communication	-13.10 ± 6.79	-15.88 ± 10.78	$\mathbf{-11.90 \pm 6.95}$	-40.07 ± 11.90

TABLE V
AVERAGE CALCULATION TIME OF 1000 EPISODES OVER FOUR MPE BENCHMARK TASKS

	MACDPP	MADDPG	MATD3	M3DDPG
Physical Deception	124.77 ± 1.77 s	32.84 ± 0.20 s	33.31 ± 0.20 s	105.86 ± 2.41 s
Covert Communication	161.73 ± 1.40 s	41.23 ± 0.35 s	41.27 ± 0.45 s	125.83 ± 2.38 s
Keep Away	149.58 ± 1.33 s	39.30 ± 0.32 s	42.94 ± 0.39 s	115.28 ± 1.86 s
Cooperative Communication	144.11 ± 0.55 s	37.10 ± 0.23 s	39.19 ± 0.41 s	108.56 ± 2.44 s

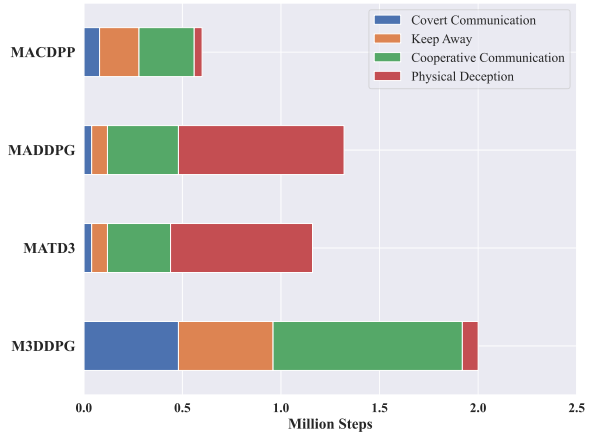


Fig. 4. Number of interactions utilized by all compared approaches in MPE benchmark tasks to reach the lower boundary of maximum average return.

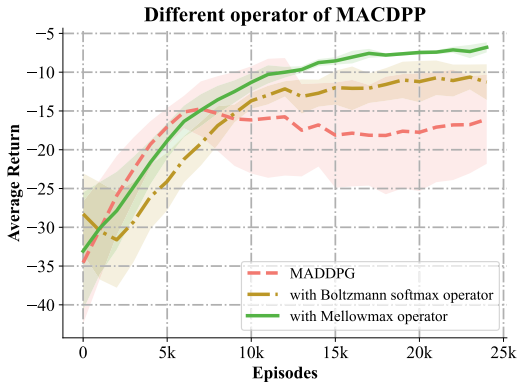


Fig. 5. Learning curves of MACDPP with different operators in the Keep Away task. The shaded region represents the standard deviation of the average evaluation over five trials.

measure of sample efficiency in this subsection as the number of interactions used by each approach to reach the lower boundary of the maximum average returns in Table IV over the four benchmark tasks from the MPE environment. It is clearly observed that the proposed method achieved the overall superior sample efficiency among all compared MARL approaches, it reduced 70%, 54.6% and 48.3% usage of samples than M3DDPG, MADDPG and MATD3 to reach a certain level of control performances. At the same time, we found that MACDPP has overall effectiveness in improving sample efficiency, whether in cooperative or competitive tasks. This result demonstrated the great potential of the proposed method in quickly learning proper multi-agent control policies in complex scenarios with few sampling costs.

3) Evaluation of the Computational Efficiency: In this subsection, we investigated the impact of additional Monte Carlo sampling and Boltzmann softmax operator in MACDPP on computational efficiency. We measured the average calculation time of 1000 episodes including training and decision-making over all compared MARL baselines in Table V. It is observed that MACDPP brought additional computational burdens in all four tasks. It required 285.54%, 270.19% and 27.36% more calculation time compared with MADDPG, MATD3 and M3DDPG. On the other hand, considering the obvious advantages of our approach in learning ability, convergence speed, and sample efficiency, we believed that these increased computational complexities in training and decision-making were acceptable.

4) Impact of the Relative Entropy Regularization: In this subsection, we explored the impact of relative entropy regularization. We first evaluated the impact of different operators regularized by the relative entropy including Boltzmann softmax and Mellowmax on MADDPG in the Keep Away task in Fig. 5. It is clearly observable that employing Mellowmax instead of Boltzmann softmax could improve the MACDPP's performances in both convergence speed and the maximum average return. As a comparison, MACDPP using the Boltzmann softmax operator finally reached a higher average return, its learning process significantly fell behind MADDPG at the first 10k episodes due to its low efficiency in converging to a unique fixed point. This result was consistent with the advantages of Mellowmax described in [37].

We evaluated the special parameter η that controls the strength of the relative entropy term in MACDPP. The learning curves MACDPP in the Keep Away task with different values

TABLE VI
MAXIMUM AVERAGE RETURNS OVER FOUR BENCHMARK CONTROL TASKS

	MACDPP	MADDPG	DDPG	TD3	SAC
HalfCheetah (2 agents)	**10590.80 ± 443.92**	9404.49 ± 1252.08	9991.71 ± 1533.64	9754.40 ± 955.57	10426.10 ± 1577.89
HalfCheetah (6 agents)	9767.99 ± 419.40	6482.54 ± 1296.47	9821.45 ± 1393.96	9730.61 ± 962.49	**9828.28 ± 803.50**
Hopper (3 agents)	**3452.87 ± 60.85**	2741.40 ± 935.69	2212.34 ± 1139.88	3384.24 ± 230.43	3325.33 ± 270.24
UR5 (5 agents)	**1.86 ± 0.03**	1.66 ± 0.73	0.60 ± 1.04	1.60 ± 0.30	−1.13 ± 0.19

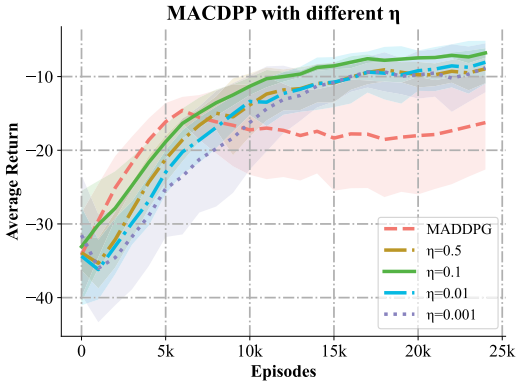


Fig. 6. Learning curves of MACDPP with different values of parameter η in the Keep Away task. The shaded region represents the standard deviation of the average evaluation over five trials.

of η are compared in Fig. 6. With a wide range of η from 0.001 to 0.5, MACDPP consistently outperformed the baseline method MADDPG in both the mean and standard deviation of the average returns. It is also observed that a proper selection of η could significantly improve MACDPP's learning performance. The most superior learning curve was obtained when $\eta = 0.1$. An over-small parameter $\eta = 0.001$ resulted in extremely slow convergence and a very large standard deviation of return at the beginning. As a comparison, the large one has less effect on smooth policy updates and may fail in learning more optimal control policies within a limited number of interactions.

C. Cooperation in Traditional Control Task

1) Evaluation of the Learning Capability: In this section, we moved to the traditional control scenarios where the MARL approaches were employed to jointly control one system. In this section, we compared two MARL approaches MACDPP and MADDPG which were treated as the proposed method without using relative entropy regularization. Three widely implemented single-agent RL approaches DDPG, TD3 and SAC were also compared. The learning curves of all compared approaches in 2-agent HalfCheetah (the system was jointly controlled by MARL with two agents), 6-agent HalfCheetah, 3-agent Hopper and 5-agent UR5 robot arm were illustrated in Fig. 7. The average maximum returns in the evaluation phase are listed in Table. VI. Please note that the results of single-agent methods in two HalfCheetah were slightly different since we used difficult random seeds for each task.

In the 2-agent HalfCheetah task, MADPPG learned the worst policy with the lowest average return. In this case, the joint control strategy failed to suppress the single-agent approach while introducing an additional computational burden. As a comparison, our method converged to the best average return overall compared approaches with a significant advantage in convergence speed. Regarding the maximum average return in evaluation, MACDPP outperformed 12.61%, 8.57% and 6.0% compared with MADDPG, TD3 and DDPG while achieving slightly better results than SAC with significantly superior convergence speed. In the 6-agent HalfCheetah task, the learning capability of MADPPG hugely deteriorated so that the six joints could not effectively cooperate. As a comparison, the proposed method successfully learned the task as well as other single-agent baselines with not only a superior convergence speed but also less standard deviation in the average returns. It enjoyed over 50% more maximum average returns than MADDPG which did not employ the relative entropy regularization. In the 3-agent Hopper task, MADDPG outperformed DDPG in average return and convergence speed while the proposed MACDPP achieved overall superior performances than all compared baselines. It quickly converged to the best maximum average return which was 56.07%, 25.95%, 3.84% and 2.03% than DDPG, MADDPG, SAC and TD3, respectively. In the more practical UR5 control scenarios where five independent joints were jointly controlled by five agents in MARL methods, both MACDPP and MADDPG outperformed single-agent RL approaches. Within 200k steps, SAC and DDPG could not learn the task (SAC could not converge at all, and DDPG learned extremely slowly at the first 70k steps) while only TD3 achieved a close average return to MADDPG. As a comparison, MACDPP consistently demonstrated superiority in both learning capability and convergence speed, it quickly reached an average return over 0 within 70k steps and finally obtained 210%, 12.05%, 264.6% and 16.3% higher maximum average return than other baselines.

2) Evaluation of the Sample Efficiency: The sample efficiency of MACDPP was evaluated in Fig. 8. Compared with MADDPG which converged slower to a certain level of control performances than the signal-agent baselines, the proposed method demonstrated great advantage in sample efficiency with the regularization of relative entropy. It only spent 36.6% samples to reach the same performance. This result indicated the importance of properly restricting large policy updates in MARL for superior effectiveness. MCDPP successfully reduced 44.91%, 37.94% and 27.98% usage of interactions than DDPG, SAC and TD3.

3) Evaluation of the Computational Efficiency: The computational burden was evaluated in Table VII by summarizing the average calculation time including agent training and simulation operating of the first 1000 steps in four control

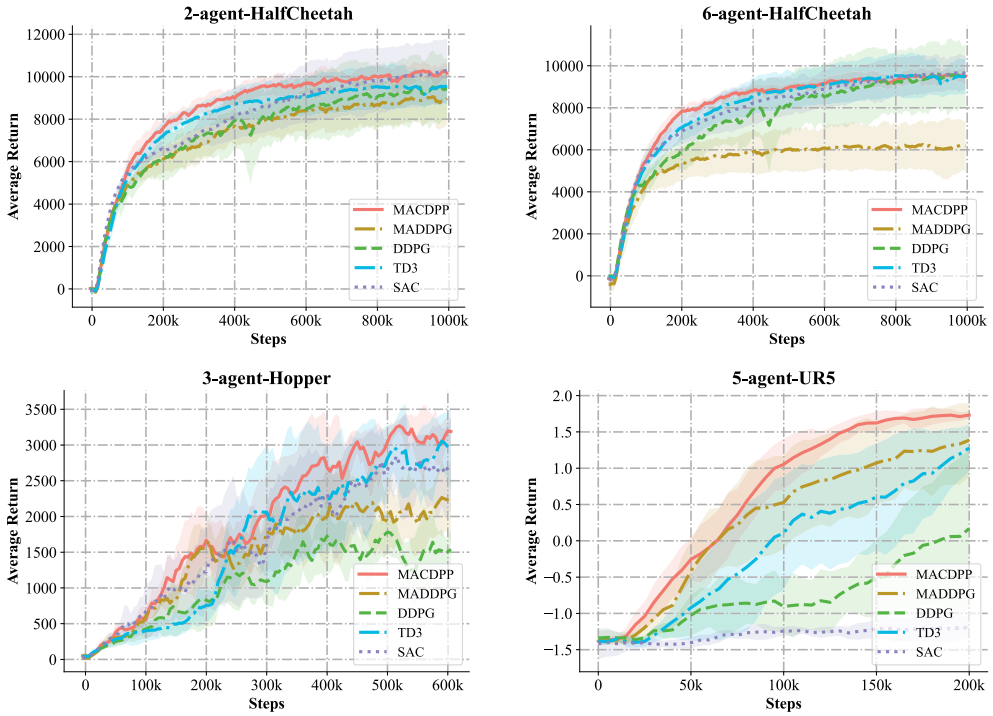


Fig. 7. Learning curves of MACDPP and all baselines in Mujoco and robo-gym benchmark tasks. The shaded region represents the corresponding standard deviation over five trials.

TABLE VII
AVERAGE CALCULATION TIME OF 1000 STEPS OVER FOUR BENCHMARK CONTROL TASKS

	MACDPP	MADDPG	DDPG	TD3	SAC
HalfCheetah (2 agents)	13.84 ± 0.29 s	9.44 ± 0.30 s	4.51 ± 0.20 s	4.18 ± 0.07 s	8.36 ± 0.10 s
HalfCheetah (6 agents)	54.84 ± 0.15 s	38.35 ± 0.29 s	4.66 ± 0.24 s	4.34 ± 0.29 s	8.63 ± 0.41 s
Hopper (3 agents)	22.61 ± 0.89 s	15.83 ± 0.72 s	4.86 ± 0.36 s	4.69 ± 0.32 s	8.84 ± 0.64 s
UR5 (5 agents)	284.58 ± 28.80 s	219.32 ± 29.31 s	162.03 ± 16.44 s	177.65 ± 22.73 s	144.68 ± 0.64 s

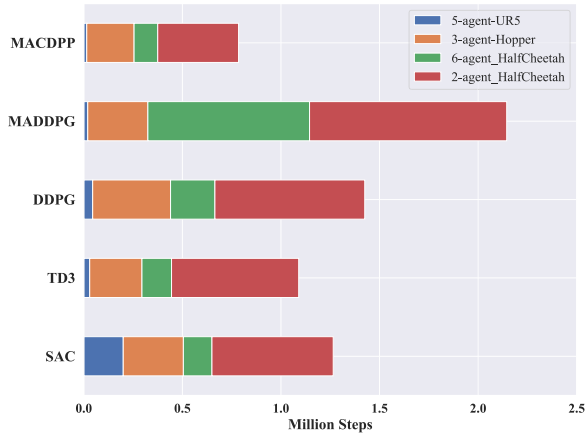


Fig. 8. Number of interactions utilized by all compared approaches in Mujoco and robo-gym benchmark tasks to reach the lower boundary of maximum average return.

scenarios. Although the proposed method required 132.85%, 213.5%, 196.92% and 220.43% more computational times compared with MADDPG, DDPG, TD3 and SAC, it did not have as much disadvantage in terms of computational burden as the results of MPE benchmark tasks in Table V compared with MADDPG since the most time was spent on operating the simulator. With the increasing system operation and communication times (i.e., from Mujoco to robo-gym based on ROS toolkit), the phenomenon above became more and more noticeable. It demonstrated the potential and effectiveness of MACDPP in jointly controlling large-scale systems.

4) Impact of the Relative Entropy Regularization: In this subsection, we explored the impact of relative entropy regularization in the traditional control tasks. The impact of different operators regularized by the relative entropy including Boltzmann softmax and Mellowmax on MACDPP in the HalfCheetah task was demonstrated in Fig. 9. The Mellowmax operator contributed to higher average returns and faster convergence. Despite a slightly faster convergence, MACDPP using the Boltzmann softmax operator did not suppress MADDPG in terms of the average return after 1000k steps of learning.

The impact of different values of η in MACDPP with Mellowmax was further evaluated by the HalfCheetah task, as shown in Fig. 10. We observed that the policy update of MACDPP was overly constrained when $\eta = 0.1$, resulting in a

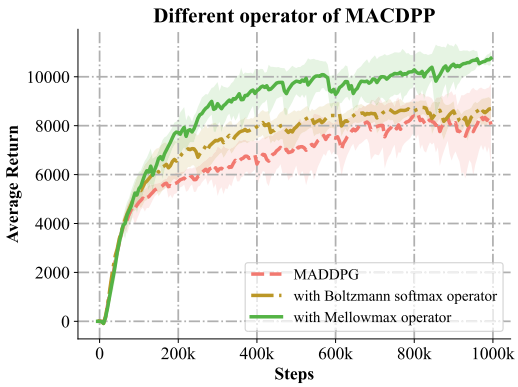


Fig. 9. Learning curves of MACDPP with different operators in the HalfCheetah task. The shaded region represents the standard deviation of the average evaluation over five trials.

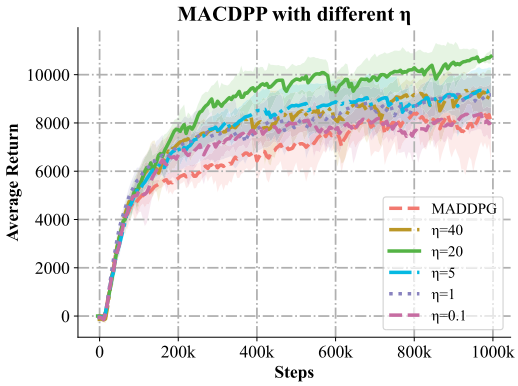


Fig. 10. Learning curves of MACDPP with different values of parameter η in the HalfCheetah task. The shaded region represents the standard deviation of the average evaluation over five trials.

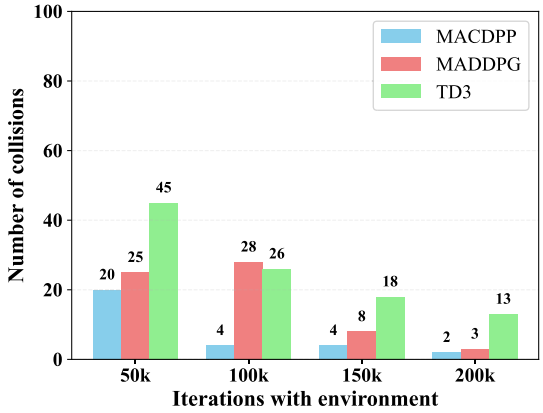


Fig. 11. Average number of collisions of MACDPP in the UR5 control task during the first 200k steps compared with MADDPG and TD3.

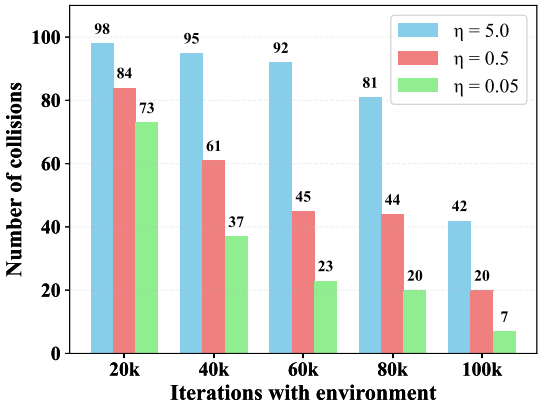


Fig. 12. Average number of collisions of MACDPP with different values of parameter η in the UR5 control task during the first 100k steps.

close performance to MADDPG. Increased η to 1, the average return of MACDPP reached about 9000. With a proper setting $\eta = 20$, the proposed method converged to an average return of over 10000 and demonstrated a significant advantage in terms of convergence speed. However, recklessly increasing η will not lead to further improvement in MACDPP's learning capability. When η was increased to 40, there was a rapid decline in both average return and convergence speed, which aligns with the experimental results of KDPP [33].

The effect of MACDPP's relative entropy regularization on the robot's behavior was evaluated using the UR5 control task. We tested the agents of MACDPP, MADDPG and TD3 saved at the 50k, 100k, 150k and 200k steps by five episodes with total 5×100 steps and compared their average number of collisions in Fig. 11. Compared to MADDPG and TD3 which could not quickly reduce the robot's collision during learning, the proposed method with the Mellowmax operator ($\eta = 0.05$) significantly reduced the number of collisions at the early stage of the learning process. Figure 12 further demonstrated the impact of η on the average number of collisions during the early stage (20k to 100k steps) of the UR5 control task. The effect of η in suppressing high-risk actions can be clearly observed. As η decreased, the relative entropy constraint on MACDPP's policy update became increasingly strong, leading to a significant reduction in the number of collisions. The results above show the advantage of the proposed method in engineering applications. The policy update constrained by the relative entropy could drastically reduce high-risk and harmful exploration actions while enhancing learning capability.

All the experimental results above indicated the good generalization capability of MACDPP in various tasks including MPE and traditional control tasks. However, it is also observed that the selection of η affected the performances of MACDPP and became the core of its successful implementation in different control scenarios. Compared to the relatively simple Hopper task, the HalfCheetah task requires a larger η to achieve an efficient exploration. For more practical implementation on the UR5 robot, a smaller η contributed to more stable and safe exploration actions that prevent frequent robot resetting due to collisions, thereby significantly improving the learning performance. In the implementation of MACDPP, we suggested initializing the value of η within the range from 0.01 to 20, e.g., trying 0.01, 0.1, ; 1.0, 10.0, and 20.0. After evaluating the learning performances of these preliminary selections, we selected the best one and further fine-tuned it according to the requirements of the specific task. Lowering η contributed to smooth exploration and policy updates while increasing η sped up the learning process in the early stages

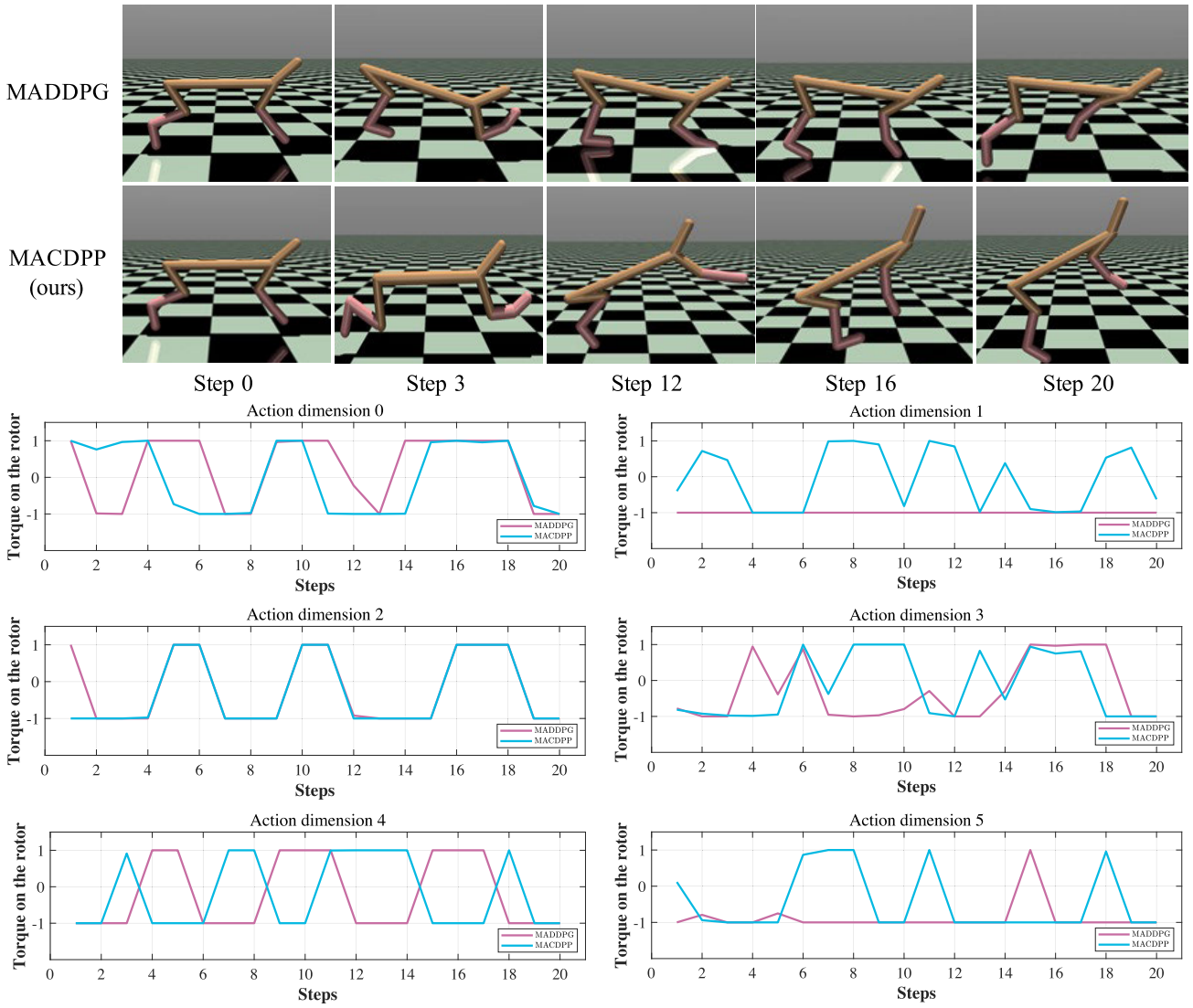


Fig. 13. Trajectories of control actions using MADDPG and MACDPP in one test rollout of 6-agent HalfCheetah task. The actions of MADDPG and MACDPP were drawn in purple and blue respectively.

of training at the expense of stability. Please also note that an overly large η could cause numerical issues in the action preferences function due to the use of exponential functions, resulting in divergence of learning.

5) Case Study: In this subsection, we investigated the superior control behaviors of MACDPP through the rollouts of learned policies in both 6-agent HalfCheetah and 5-agent UR5 control scenarios. In the first case study, we explored the learned policies of MADDPG and MACDPP under the same setting of parameter and random seed. The test rollouts with 20 steps and the corresponding trajectories of each joint were analyzed in Fig. 13. It is observed that the learned control behavior of MACDPP is more effective than the one of MADDPG. Effectively coordinating six joints by six agents, the proposed method learned a superior control strategy. Each joint timely conducted proper torque according to the current system states, resulting in a faster movement. In comparison, MADDPG had significant disadvantages in terms of coordinating six joints by separate agents. Although the agent in charge of dimension 2 successfully learned a similar policy to the one of MACDPP, the whole multi-agent system struggled to generate proper torques from other agents: the agent of dimension 5 only produced effective torque near the 15-th step while the agent of dimension 1 was fixed with -1 torque during the how rollout. Due to the lack of relative entropy regularization from the algorithmic perspective, the multiple agents in MADDPG were unable to learn effective and cooperative control strategies.

In the next case study, we studied the test rollouts using MADDPG and MACDPP in the ur_ee_position task which aims to control the end-effector of the UR5 robot arm to reach the randomly generated targets. It is observed that MACDPP quickly drove the robot to finish the task within 60 steps. The action trajectories showed proper cooperation between each joint. The base and elbow joints continuously output $-180°$ and $60°$ throughout the task. At step 25, the shoulder joint and wrist 1 joint were coordinated to guide the end-effector to move forward to the target position. Around step 60, the

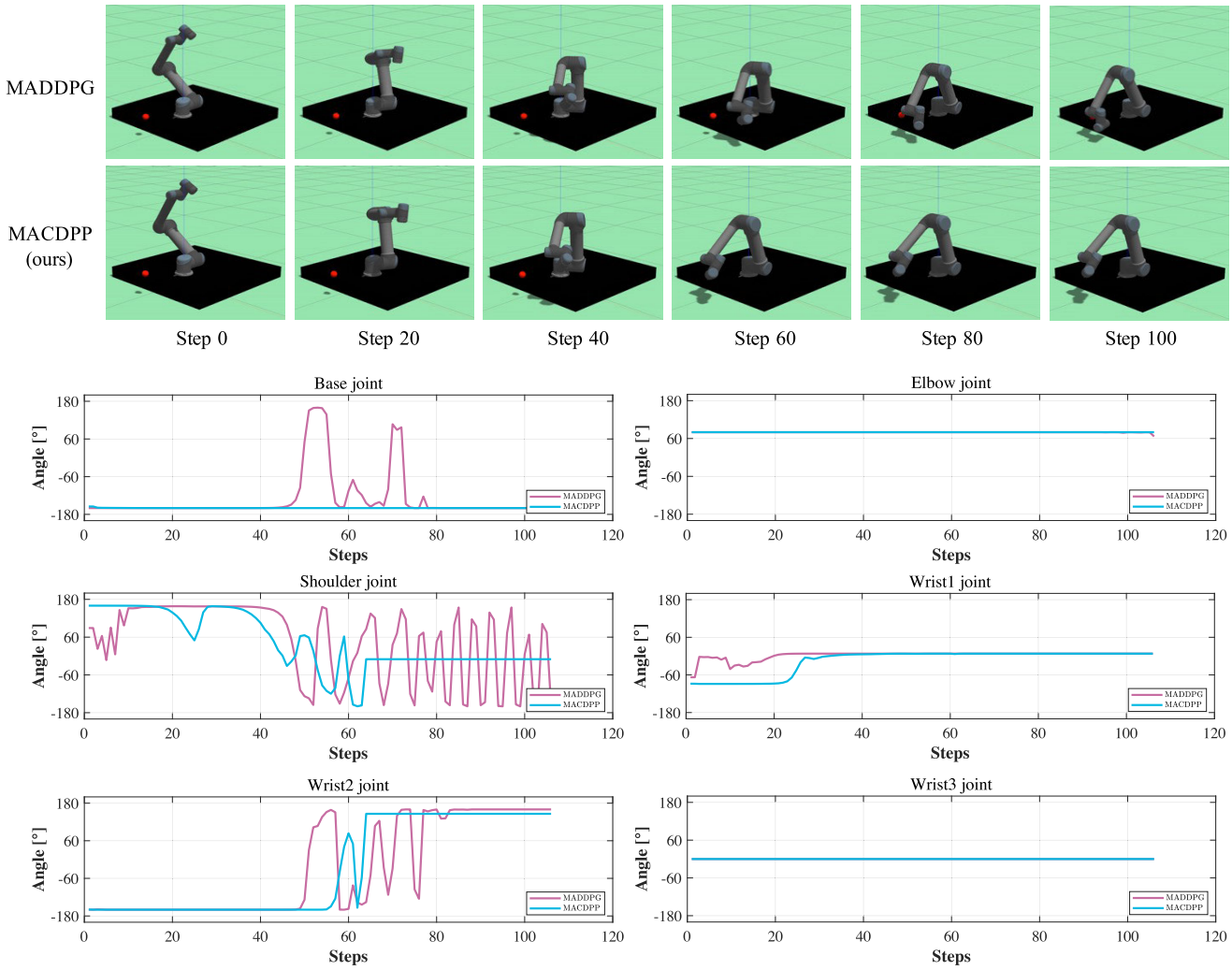


Fig. 14. Trajectories of control actions using MADDPG and MACDPP in one test rollout of 5-agent UR5 control task ur_ee_position. The actions of MADDPG and MACDPP were drawn in purple and blue respectively. The last dimension Wrist3 joint was not included in the controllable action as it was fixed to 0 in the ur_ee_position task.

shoulder and wrist 2 joints worked together to quickly reach the target. The action Trajectories of all dimensions were effective with minimal jitter. Compared with our method, MADDPG could not sufficiently learn the cooperative strategy over five joints. The base joint could not continuously output a certain degree, it had strong trembling between steps 40 to 80. The shoulder and wrist 1 joints failed to cooperate effectively at the beginning, resulting in redundant movements of the end-effector. After step 40, the shoulder and wrist 2 joints could not achieve seamless coordination. Both of them experienced sustained tremors which ultimately resulted in highly degraded control performance.

The experimental results above revealed the advantages of MACDPP in the joint control of robot systems. The multiple agents reduced the exploration complexity in one system and resulted in faster policy convergence compared to single-agent approaches with the same amount of interactions. At the same time, the relative entropy regularization significantly avoided the mismatch between the update of multi-agent policies during the learning, promoting the effectiveness in the learning of cooperative control strategies.

V. Conclusion and Discussions

This article proposed a novel MARL approach MACDPP to improve the learning capability and sample effectively in a wide range of control scenarios including multiple agents cooperative/competitive tasks and joint control of a single complicated system. It naturally alleviated the inherent inconsistency over multiple agents policy updates by integrating the relative entropy regularization to the AC structure and CTDE framework. MACDPP successfully extended FKDPP which has been successfully implemented in the real-world chemical plant by Yokogawa [35], [36] towards a modern approach that supports deep neural networks, AC structure and CTDE framework to fit a wider range of control scenarios. Through evaluation of different benchmark tasks, ranging from multi-agent cooperation/competition to Mujoco simulator and robot arm manipulation, our proposed method consistently demonstrated significant superiority in both learning capability and sample efficiency compared with related multi-agent and single-agent RL baselines. All these results indicated the potential of relative entropy regularized MARL in effectively learning complex systems divided into

multiple agents with lower sampling costs and better control performance.

For future work, the proposed method can be improved in the cooperative tasks by independently and dynamically setting the parameters of relative entropy regularization for each agent according to its current performances [44]. Reducing the computational burden of MACDPP by efficiently sharing knowledge over multiple agents could contribute to better feasibility for complex and large-scale control scenarios. The combination of MACDPP with state-of-the-art single-agent RL methods like TD7 [45] for more effective learning capability also remains unexplored.

REFERENCES

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

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

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

[6] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Robot. Res.*, vol. 40, nos. 4–5, pp. 698–721, Apr. 2021.

[7] C. Sun, W. Liu, and L. Dong, "Reinforcement learning with task decomposition for cooperative multiagent systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2054–2065, May 2021.

[8] A. Wong, T. Bäck, A. V. Kononova, and A. Plaat, "Deep multiagent reinforcement learning: Challenges and directions," *Artif. Intell. Rev.*, vol. 56, no. 6, pp. 5023–5056, Jun. 2023.

[9] X. Yao, C. Wen, Y. Wang, and X. Tan, "SMIX(λ): Enhancing centralized value functions for cooperative multiagent reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 52–63, Jan. 2023.

[10] J. Chai et al., "UNMAS: Multiagent reinforcement learning for unshaped cooperative scenarios," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2093–2104, Apr. 2023.

[11] 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.

[12] N. Patrizi, G. Fragkos, K. Ortiz, M. Oishi, and E. E. Tsiropoulou, "A UAV-enabled dynamic multi-target tracking and sensing framework," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[13] Y. Jin, S. Wei, J. Yuan, and X. Zhang, "Hierarchical and stable multiagent reinforcement learning for cooperative navigation control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 90–103, Jan. 2023.

[14] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Appl. Intell.*, vol. 53, no. 11, pp. 13677–13722, 2023.

[15] Z. Yan, A. R. Kreidieh, E. Vinitsky, A. M. Bayen, and C. Wu, "Unified automatic control of vehicular systems with reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 2, pp. 789–804, Apr. 2023.

[16] L. Busoniu, R. Babuska, and S. B. De, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Feb. 2008.

[17] A. Tampuu et al., "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, Apr. 2017, Art. no. e0172395.

[18] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, no. 1, pp. 1–31, Feb. 2012.

[19] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems (AAMS)*. São Paulo, Brazil: Springer, 2017, pp. 66–83.

[20] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 29, 2016, pp. 1–9.

[21] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.* 2018, p. 2085–2087.

[22] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 7234–7284, 2020.

[23] M. Samvelyan et al., "The StarCraft multi-agent challenge," in *Proc. Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2019, pp. 2186–2188.

[24] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 35, 2022, pp. 24611–24624.

[25] R. Lowe et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 30, 2017, pp. 1–12.

[26] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," 2019, *arXiv:1910.01465*.

[27] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 4213–4220, 2019.

[28] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *Proc. Assoc. Advancement Artif. Intell. (AAAI)*, 2010, pp. 1607–1612.

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

[30] M. G. Azar, V. Gómez, and H. J. Kappen, "Dynamic policy programming," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 3207–3245, 2012.

[31] T. Kozuno, E. Uchibe, and K. Doya, "Theoretical analysis of efficiency and robustness of softmax and gap-increasing operators in reinforcement learning," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2019, pp. 2995–3003.

[32] N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist, "Leverage the average: An analysis of KL regularization in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 12163–12174.

[33] Y. Cui, T. Matsubara, and K. Sugimoto, "Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states," *Neural Netw.*, vol. 94, pp. 13–23, Oct. 2017.

[34] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robot. Auto. Syst.*, vol. 112, pp. 72–83, Feb. 2019.

[35] Y. Cui, L. Zhu, M. Fujisaki, H. Kanokogi, and T. Matsubara, "Factorial kernel dynamic policy programming for vinyl acetate monomer plant model control," in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2018, pp. 304–309.

[36] L. Zhu, Y. Cui, G. Takami, H. Kanokogi, and T. Matsubara, "Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process," *Control Eng. Pract.*, vol. 97, Apr. 2020, Art. no. 104331.

[37] K. Asadi and M. L. Littman, "An alternative softmax operator for reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 243–252.

[38] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.

[39] M. Lucchi, F. Zindler, S. Mühlbacher-Karrer, and H. Pichler, "Robo-gym—An open source toolkit for distributed deep reinforcement learning on real and simulated robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5364–5371.

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

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

[42] 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.*, 2018, pp. 1861–1870.

[43] Y. Ma, D. Yu, T. Wu, and H. Wang, "PaddlePaddle: An open-source deep learning platform from industrial practice," *Frontiers Data Domputing*, vol. 1, no. 1, pp. 105–115, 2019.

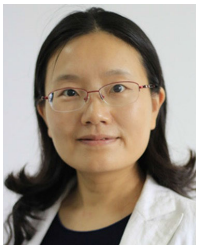
[44] Z. Shang, H. Li, and Y. Cui, "Shiftable dynamic policy programming for efficient and robust reinforcement learning control," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2021, pp. 1688–1693.

[45] S. Fujimoto et al., "For SALE: State-action representation learning for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–52.

Chenyang Miao received the B.S. degree in network engineering from Anhui University, Hefei, China, in 2021. He is currently pursuing the M.Sc. degree in computer technology with Shenzhen Institute of Advanced Technology, University of Chinese Academy of Science, Shenzhen, China. His current research interests include deep reinforcement learning, multi-agent reinforcement learning, and robot control.

Yunduan Cui (Member, IEEE) received the B.E. degree in electronic engineering from Xidian University, China, in 2012, the M.E. degree in computer science from Doshisha University, Japan, in 2014, and the Ph.D. degree in computer science from Nara Institute of Science and Technology, Japan, in 2017.

From 2017 to 2020, he was a Post-Doctoral Researcher with Nara Institute of Science and Technology. He is currently an Associate Professor with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China. His research focuses on machine learning, especially reinforcement learning-driven unmanned systems. He received the Best Oral Paper Award of IEEE-RAS 16th International Conference on Humanoid Robots in 2016, the Best Student Award of Nara Institute of Science and Technology in 2018, the Best Paper Award of Japanese Neural Network Society (JNNS) in 2018, and the SICE International Young Authors Award for IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in 2019.



Huiyun Li (Senior Member, IEEE) received the M.Eng. degree in electronic engineering from Nanyang Technological University in 2001 and the Ph.D. degree from the University of Cambridge, U.K., in 2006. She is currently a Professor with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, and The Chinese University of Hong Kong. Her research interests automobile electronics and autonomous vehicles.

Xinyu Wu (Senior Member, IEEE) received the B.E. and M.E. degrees from the Department of Automation, University of Science and Technology of China, Hefei, China, in 2001 and 2004, respectively. He is currently a Professor with Shenzhen Institutes of Advanced Technology, Shenzhen, China, and the Director of the Center for Intelligent Bionic. He has authored or coauthored more than 180 articles and two monographs. His research interests include computer vision, robotics, and intelligent systems. His Ph.D. degree was awarded from The Chinese University of Hong Kong in 2008. He is an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS MAN CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and IEEE ROBOTICS AND AUTOMATION LETTERS.