Complex & Intelligent Systems
https://doi.org/10.1007/s40747-024-01421-3

ORIGINAL ARTICLE

Decentralized multi-agent cooperation via adaptive partner modeling

Chenhang Xu[1] · Jia Wang[2] · Xiaohui Zhu[1] · Yong Yue[1] · Weifeng Zhou[3] · Zhixuan Liang[4] · Dominik Wojtczak[5]

Received: 9 May 2023 / Accepted: 12 March 2024

Abstract
Multi-agent reinforcement learning encounters a non-stationary challenge, where agents concurrently update their policies, leading to changes in the environment. Existing approaches have tackled this challenge through communication among agents to obtain their partners' actions, but this introduces computational complexity known as partner sample complexity. An alternative approach is to develop partner models that generate samples instead of direct communication to mitigate this complexity. However, a discrepancy arises between the real policies distribution and the policy of partner models, termed as model bias, which can significantly impact performance when heavily relying on partner models. In order to achieve a trade-off between sample complexity and performance, a novel multi-agent model-based reinforcement learning algorithm called decentralized adaptive partner modeling (DAPM) is proposed, which utilizes fictitious self play (FSP) to construct partner models and update policies. Model bias is addressed by establishing an upper bound to restrict the usage of partner models. Coupled with that, an adaptive rollout approach is introduced, enabling real agents to dynamically communicate with partner models based on their quality, ensuring that agent performance can progressively improve with partner model samples. The effectiveness of DAPM is exhibited in two multi-agent tasks, showing that DAPM outperforms existing model-free algorithms in terms of partner sample complexity and training stability. Specifically, DAPM requires 28.5% fewer communications compared to the best baseline and exhibits reduced fluctuations in the learning curve, indicating superior performance.

Keywords Multi-agent reinforcement learning · Fictitious self play · Partner modeling · Partner sample complexity

Introduction

With the development of cutting-edge deep learning techniques, multi-agent reinforcement learning (MARL) has produced remarkable achievements and has been widely applied in diverse fields, such as gaming [1], sensor networks [2], autonomous driving [3], and multi-robot navigation [4]. Despite the advancements of MARL, it encounters challenges due to the non-stationarity of multi-agent environments, where policies are updated simultaneously by multiple agents, leading to dynamic changes. Existing solutions such as PBL [5], CommNet [6], and MAAC [7] have addressed the non-stationarity challenge by developing com-

✉ Xiaohui Zhu
Xiaohui.Zhu@xjtlu.edu.cn

Chenhang Xu
chenhang.xu21@student.xjtlu.edu.cn

Jia Wang
Jia.Wang02@xjtlu.edu.cn

Yong Yue
Yong.Yue@xjtlu.edu.cn

Weifeng Zhou
chandler.zhou@cpce-polyu.edu.hk

Zhixuan Liang
18094409g@connect.polyu.hk

Dominik Wojtczak
D.Wojtczak@liverpool.ac.uk

1 Department of Computing, Xi'an Jiaotong-Liverpool University, Ren'ai Road, Suzhou 215123, Jiangsu, China

2 Department of Intelligent Science, Xi'an Jiaotong-Liverpool University, Ren'ai Road, Suzhou 215123, Jiangsu, China

3 School of Professional Education and Executive Development, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

4 Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

5 Department of Computer Science, Liverpool University, Ashton Street, Liverpool, England L69 3BX, UK

Published online: 15 April 2024

munication protocols among agents and enabling access to partner actions. However, every communication causes computational complexity, known as partner sample complexity. These model-free approaches need an impractical number of such communications to be accessed for effective training. Consequently, partner sample complexity is a crucial factor that cannot be overlooked during training. On the other hand, model-based approaches have proven to be advantageous in addressing the issue of dynamic sample complexity resulting from agent-environment interactions in both single-agent reinforcement learning (SARL) and multi-agent reinforcement learning (MARL) [8, 9]. This is achieved by constructing the dynamic model of the real environment, which enables a reduction in the amount of required samples. Notably, previous approaches overlooked the importance of building models to represent others' actions as a means to minimize sample requirements. In a multi-agent system (MAS), the development of multiple partner models can be utilized to simulate the behavior of partners, thereby decreasing partner sample complexity. However, the accuracy of these models poses a significant challenge to policy quality, as model bias between real policies and partner policies may result in model-based approaches underperforming compared to model-free counterparts. The limitation of previous works bring up critical questions in MARL:

- How to effectively build partner models to accurately represent the actions of other agents?
- How can a suitable equilibrium be achieved between mitigating partner sample complexity in model-free methods and addressing the accuracy challenges of model-based techniques in MARL?

Introducing decentralized adaptive partner modeling (DAPM), a novel approach that integrates model-free and model-based learning techniques to effectively leverage partner models for policy optimization in decentralized MAS. To answer the first question, as illustrated in Fig. 1, DAPM leverages the FSP technique to maintain belief about the actions of partner agents and develop partner models that facilitate the modeling of their actions. In contrast to previous FSP-related works [10, 11], where each agent employs two policies: an average policy network trained through supervised learning and a best response policy network trained through reinforcement learning. The choice of which network to use was determined by a predetermined probability. This study proposes modifications to equip partners with both average policy networks and best response networks. However, the discrepancies in partner model accuracy can result in varying contributions to the compounded error when interacting with multiple partner models.

Addressing the second question requires achieving a delicate balance between partner sample complexity and partner model accuracy. In this regard, an adaptive rollout method is implemented to determine the necessity for communication with real partners (best response network) or partner models (average policy network). This method enables informed decisions on when communication is essential and optimizes communication efficiency in MAS. As the models are continuously optimized, more partner models can be included in the training process of MARL, reducing sample complexity associated with partnering agents. The main contributions of this paper can be summarized as follows:

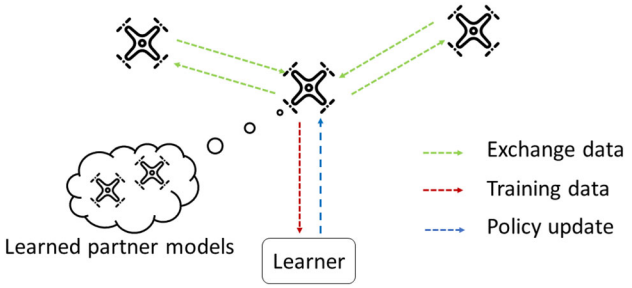


Fig. 1 Visualization of DAPM, with the DTDE framework. Agents communicate to share information. The partner models update their policy using the data from ego agent and generate training data under certain conditions to update the agent's policy

- Introduction of the pioneering approach with fictitious self play technique: DAPM is the first to comprehensively employ the FSP technique for modeling agents' actions, significantly advancing current methodologies in the field.
- Innovative adaptive rollout method: A dynamic approach to determining the need for communication with real partners, reducing partner sample complexity in MARL while maintaining algorithm performance, surpassing prior works that rely on predetermined probabilities.
- Superior performance in cooperative scenarios: Through experiments in MPE [12] and MAPDN [13] cooperative scenarios, DAPM outperforms SOTA model-free methods, achieving a remarkable 28.5% reduction in required communications. This underscores its potential to enhance sample efficiency in MARL.

Related work

This section discusses the related work and existing solutions for sample complexity. First of all, two different MARL paradigms are discussed in the first part. Following that, the model-based MARL will be introduced, with a special focus on reducing dynamic sample complexity. Lastly, the background of partner modeling will be introduced.

Multi-agent reinforcement learning

The combination of metaheuristic and machine learning techniques has been successfully used in various areas. Key algorithms like the hybrid sine cosine algorithm and firefly algorithm have shown effectiveness in dealing with issues such as overfitting in computer vision [14, 15]. These algorithms are good at exploring large solution spaces and can handle changes within dynamic environments. In MARL, metaheuristic learning plays a significant role, especially in managing the balance between exploration and exploitation and adapting to changing situations. This impact is seen in policy optimization, coordination, and decision-making among autonomous agents. MARL methods align with successful metaheuristic principles. Additionally, centralized training with decentralized execution (CTDE) and decentralized training with decentralized execution (DTDE) are two primary training strategies in MARL. CTDE algorithms optimize local agent policies using a centralized reward and execute policies based on local histories, while DTDE allows agents to communicate and interact with the environment independently. VDN [16] is the first attempt known to us that combines centralized value function learning with decentralized execution, assuming equal contributions from agents and decomposing a central action-value function into individual Q-values. The QMIX [17] and weighted QMIX [18] algorithms represent a significant improvement upon the VDN method, as they incorporate a mixed network that considers the state information of agents. This approach allows for the determination of transformation weights that optimize the performance of the centralized training method. This enables the decomposition of the central return with non-linearity operations, leading to improved performance and more effective coordination among agents. MADDPG [12] is a CTDE approach that employs a critic for each agent to access the observations, actions, and policies of others. Local critics use joint observations and actions as input to compute Q-values, promoting coordination and decision-making in multi-agent systems. Although CTDE is a useful technique, it faces scalability challenges. These challenges arise from the exponential growth of state and action spaces, which increases the computational complexity. Additionally, the cost of centralizing information adds to the challenge. In order to address these challenges, various techniques have been proposed to augment the performance and scalability of centralized critic methods. ATT-MADDPG [19] utilized attention mechanism to explicitly capture the dynamic joint policy of other agents and promote cooperation among them. In the DTDE paradigm, agents are permitted to communicate with each other, and methods such as MAAC [7] collect all received messages to train the policy for the network. These approaches facilitate direct interaction and communication among agents, without considering sample complexity. In this study, DAPM adopts the DTDE training paradigm and trains the policy network independently with partner models. The partner model generates samples for training, thereby reducing the sample complexity associated with partners.

Model-based reinforcement learning

Recent model-based approaches, such as those proposed by Wang et al. [8] and Sun et al. [20], have demonstrated advantages in reducing sample complexity for SARL. However, in complex real-world environments, it may be challenging for these algorithms to learn accurate models. As a result, careful consideration is needed when choosing approaches for model learning and model usage. Early model-based methods employed simple models such as linear models [21] and Gaussian processes [22]. However, the limited expressiveness of these models makes them inadequate for handling non-linear and high-dimensional environments. More recent methods have utilized deep neural networks for improved performance, such as neural network ensembles [23] and Bayesian neural networks [24]. Previous research has employed various methods for model usage in SARL. Dyna-style approaches [25] have the ability to generate extra data using the learned models, effectively expanding the dataset for RL training. Monte Carlo (MC) value estimate [26] was one of the first models used to approximate state values. Many value-based RL algorithms rely on temporal-difference (TD) [27] prediction, which is another commonly used method for value approximation. MVE [28] showed that incorporating H-step TD value prediction can effectively mitigate estimation errors. However, MVE relies on a predetermined task-specific horizon H, which may need to be adjusted during training. To overcome this limitation, STEVE [29] proposes an approach that interpolates between different horizons H and leverages ensemble uncertainty to further enhance the performance of MVE. SLBO [30] employs the model to generate complete trajectories from the start state, but the rollout length is limited due to compounding model error. MBPO [31] uses branched rollouts in the model, starting from real environment states, and takes k steps based on the policy π and the learned model, allowing for diverse trajectory generation and policy refinement. AMLAPN [32] introduces a novel approach in model-based MARL by incorporating a centralized auxiliary prediction network. This network is designed to model the dynamics of the environment and actions of opponents, effectively addressing the issue of non-stationarity. Krupnik et al. [33] present a novel approach for trajectory planning by introducing a centralized multi-step generating model that utilizes a disentangled variational auto-encoder. MAMBPO [9] extends the MBPO approach to multi-agent settings, selectively using simulated data that is close to real data to

improve the policy. CPS [34] utilizes a dynamic model of the environment to predict the next state and uses a reward model to assess the quality of the predicted state. By combining these models, CPS can determine the priority of state-action updates. However, these methods primarily focus on leveraging the environment model to reduce dynamic sample complexity, without considering the utilization of partner models to mitigate the sample complexity arising from communication among agents. In this research, an approach has been proposed where each agent builds partner models to model actions for other agents, and learns its policy using samples generated from real agents and partner models simultaneously.

Partner modeling

Partner modeling is a viable approach for addressing non-stationarity in MARL. From an agent's perspective, considering partners as part of the environment can lead to instability and challenges in policy learning, as partners' policies may also change. However, when partners' information is explicitly incorporated through partner modeling, the environment becomes more stable, enabling the use of standard SARL algorithms for policy learning. While most previous works have focused on competitive scenarios and referred to the partner model as the opponent model in their literature, partner modeling has proven to be a useful solution for handling non-stationary problems in MARL.

Fictitious play [35] is an example of an agent estimating its opponent's strategy based on past experience to determine its next move. Recent studies have made progress in modeling opponents using neural networks, benefiting from their powerful representation capabilities. DRON [36] handles a multitasking Q-learning problem to learn an opponent policy representation and a play-against policy. VAE [37] has also been used to model opponents, by learning representations in MAS based on the opponent trajectories. From the perspective of an agent, self other modeling (SOM) [38] uses gradient ascent to learn the opponent's goal from its policy, opponent observation, and action. TomNet [39] learns embedding-based opponent representations for meta-learning. However, these methods without considering the model bias between the partner models and real agents, where agents exploit inaccuracies in a model when partner models generate too much useless samples, resulting in sub-optimal policies in the real-world scenario. In this research, an upper bound for the opponent model error has been utilized to limit the use of the opponent model, which guarantees the policy network will improve with partner models.

Preliminaries

Decentralized partially observable MDP

The cooperation MARL problem is formulated as a n agents decentralized partially observable Markov decision process (Dec-POMDP), which can be defined as a tuple $(\mathcal{N}, \mathcal{S}, \mathbf{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \gamma)$, where $\mathcal{N}$ is a set of n agents and $\mathcal{S}$ is the state space. The joint action space, denoted as $\mathcal{A} = \prod_i^n \mathcal{A}_i$, is formed by the individual action spaces of each agent, represented as $\mathcal{A}_i$. Similarly, the joint local observation space, denoted as $\mathcal{O} = \prod_i^n \mathcal{O}_i$, is formed by the individual observation spaces of each agent, represented as $\mathcal{O}_i$, and γ is the discount factor. From the preservative of agent i, its policy is denoted as π_i, which is the probability distribution over its action space. The joint policy of the other agents, denoted as $\pi_{-i}(a_{-i}|s) = \prod_{j\neq i} \pi_j(a_j|s)$, represents the probability distribution over the joint action space of all agents except agent i, where a_{-i} represents the joint action excluding agent i. At every time t, agent i takes action $a_i^t \in \mathcal{A}_i$ sampled from the individual policy $\pi^i(a_i^t|o_i^t)$, where o_i^t is the local observations of agent i's state s^t. After all the agents taking actions $\mathbf{u}^t = \{a_1^t, \ldots, a_n^t\}$, the environment moves to the next Markov state s^{t+1} based on the transition function $\mathcal{T}$. Then, each agent will receive a team reward $\mathcal{R}^t$. The objective of agent i is to maximize its expected rewards, denoted by η_i:

$$\max_{\pi_i} \eta_i\left[\pi_i, \pi_{-i}\right] = \mathbb{E}_{(s^t, a_i^t, a_{-i}^t)\sim\mathcal{T},\pi_i,\pi_{-i}} \times \left[\sum_{t=1}^{\infty} \gamma^t R^t(s^t, a_i^t, a_{-i}^t)\right]. \quad (1)$$

Followed by Zhang et al. [40], we define that each agent in the environment can observe the historical trajectories of all other agents but has no knowledge about their policies.

Fictitious selfplay

Fictitious self play [10] is an extensive form of fictitious play [35], which enables the fictitious player to update their policies in extensive form, resulting in linear time and space complexity. FSP agents update their best response network and average response network through reinforcement and supervised learning, respectively. In detail, FSP algorithms include two memories: M_{RL} stores each agent's history (s^t, a^t, r^t, s^{t+1}) for reinforcement learning. M_{SL} stores each agent's state action pair (s^t, a^t) for supervised learning.

In this method, each agent i hold belief $\mathcal{B}_{ij}^t$ on the strategy profile of other agents j, where $j \in \mathcal{N}_{-i} := \{1, \ldots, n\}_{n\neq i}$. The belief $\mathcal{B}_{ij}^t$ indicates the estimation of agent i on the synchronous action that the other agent j may taken at time t. These beliefs can be grouped to define the joint belief $\mathcal{B}_i^t$

that agent i has on the actions of other neighboring agents. By holding the belief $\mathcal{B}_i^t$, the agent i can locally estimate the team reward as follows:

$$\mathcal{R}_i^t(a_i, a_{-i}, s^t) = \int_{j \in \mathcal{N}_i} \mathcal{R}_i^t(a_i, \mathcal{B}_i^t, s_j^t) \mathrm{d}s_j^t. \tag{2}$$

These partner models are learned to infer others' actions by supervised learning, which are average policies. Subsequently, agents' best response policies are updated through reinforcement learning. It is important to note that in this process, the memory $\mathcal{D}$ is utilized, replacing M_{RL} and M_{SL}. All these networks are trained on this memory, but partner models are trained on the most recent data added to memory.

Methods

In this section, the introduction of the novel learning paradigm DAPM is presented to address the partner sample complexity problem in MARL and speed up the training process. Note that despite the method being proposed to be combined with MASAC, DAPM can be easily applied to other SOTA MARL algorithms.

Initially, an overview of the algorithm is presented, as illustrated in Fig. 2. First, agent i utilizes supervised learning to train partner models for each of the other agents using samples from the observation buffer, which consists of self-observation and partner action pairs $< o_i, a_{-i} >$. The partner models then generate all other agents' actions a_{-i}^t (purple) based on the self-observation o_i^t. Meanwhile, agent i continues to optimize its policy using the soft-actor-critic method, a reinforcement learning technique, by receiving rewards from the environment and updating its policy network parameters accordingly. As depicted in the figure, other agents' policy networks incorporate information about all agents in the environment and generate joint actions a_{-i}^t (black) based on their local observations. The adaptive rollout module determines whether to use samples from the partner models during this step. This method is further detailed in Fig. 3. In this figure, the rollout length is denoted as k. The process involves measuring the discrepancy between each partner model and its corresponding actual agent. Subsequently, the number of interactions required with partner models within k steps is computed.

Decentralized adaptive partner modeling

The main concept is that agent i can obtain the actions of partner models more readily compared to requesting actions from real partners. To simplify the complexity of sampling partners, the local agent can generate samples from partner models based on self-observation, instead of relying on communication to request actions from others. The DAPM approach involves acquiring and employing partner models to generate supplementary training data for the agent's policy. Afterward, the policy is updated using the multi-agent soft actor-critic (MASAC) scheme. Algorithm 1, which can be found in Sect. 5, provides a concise overview of the approach. As the agent interacts with both the environment and actual partners, the resulting transitions are collected and recorded in a centralized memory buffer denoted as $\mathcal{D}$. Periodically, all partner models are trained on this memory buffer. Subsequently, these partner models are employed to interact with agent i and generate additional samples, which are also stored in $\mathcal{D}$ for training the actor and critic networks, ensuring efficient utilization of partner models in the learning process.

Algorithm 1 DAPM algorithm

Require: Initialize replay buffer $\mathcal{D}$, actor π^ζ, critic Q^w for each agent i,
1: partner models π_j^ϕ for $j \in -i$
2: **for** N epochs **do**
3: Execute actions with real partners using π^ζ, store the resulting
4: transitions in $\mathcal{D}$
5: Update all partner models' parameters ϕ on $\mathcal{D}$
6: Calculate the errors for each partner model $\epsilon_j^{\hat{\pi}}$
7: For each partner, compute $n_j = \dfrac{\min_{j'} \epsilon_{j'}^{\hat{\pi}}}{\epsilon_j^{\hat{\pi}}}$
8: **for** M model rollouts **do**
9: Sample random state s from $\mathcal{D}$
10: Perform k-steps rollout start from s:
11: **for** $p = 1, 2, \ldots, k$ **do**
12: $a_i^{p-1} = \pi_\phi^i(s_{p-1})$
13: For each partner agent j:
14: **if** $p <= n_j$ **then**
15: $a_j^{p-1} = \pi_j^\phi(s_{p-1})$
16: **else**
17: $a_j^{p-1} = \pi_j^\zeta(s_{p-1})$
18: **end if**
19: Add the transitions to $\mathcal{D}$
20: **end for**
21: **end for**
22: Update parameters ζ and w using samples from $\mathcal{D}$
23: **end for**

Partner model learning

In line 5 of Algorithm 1, all partner models are trained using samples from the replay buffer $\mathcal{D}$. The partner model aims to approximate the nash equilibrium policies for the players by leveraging the advances in game theoretic FSP. Specifically, DAPM use a variation of FSP, in which agents form beliefs about other people's actions using an empirical distribution of other people's past actions. Typically, the method use the indicator function $\psi_i^t(a_i|o_i) = [\psi_1^t(a_i), \psi_2^t(a_i), \ldots, \psi_k^t(a_i)] :\rightarrow [0, 1]^k$, where $\psi_k^t(a_i) = 1$ if $a_i = k$ otherwise $\psi_k^t(a_i) = 0$, to count the number of the actions have been taken. Then, $f_i^t(o_i)$ are used to represent

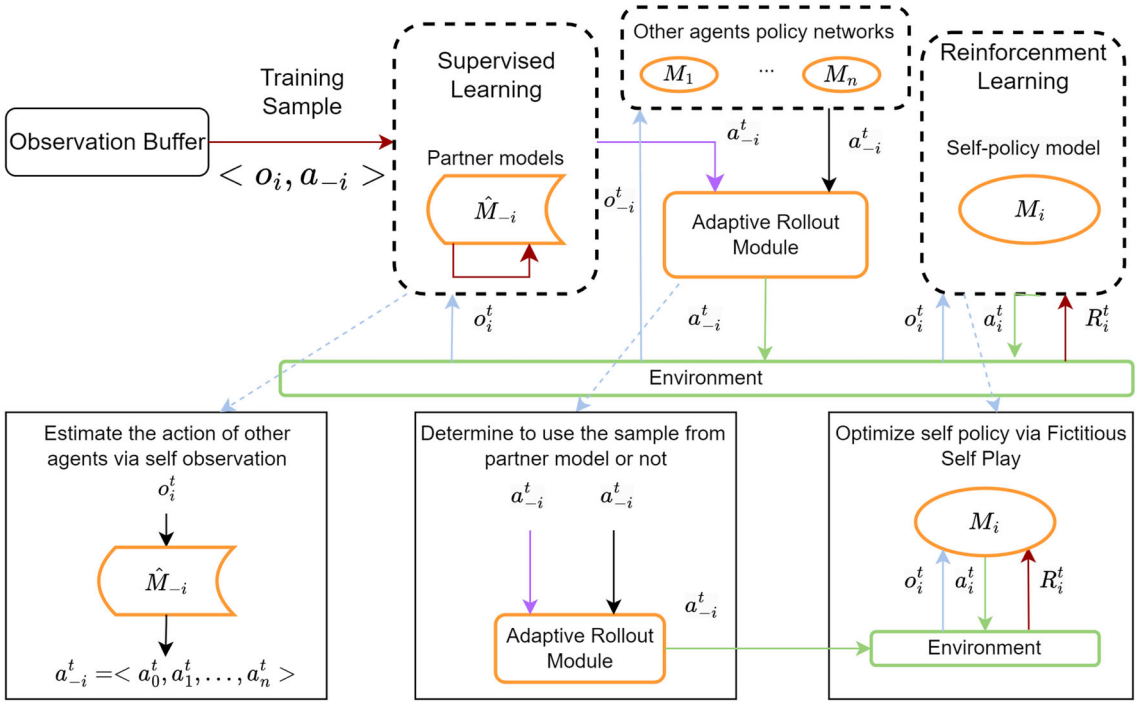


Fig. 2 General framework of the DAPM method from the agent i's perspective

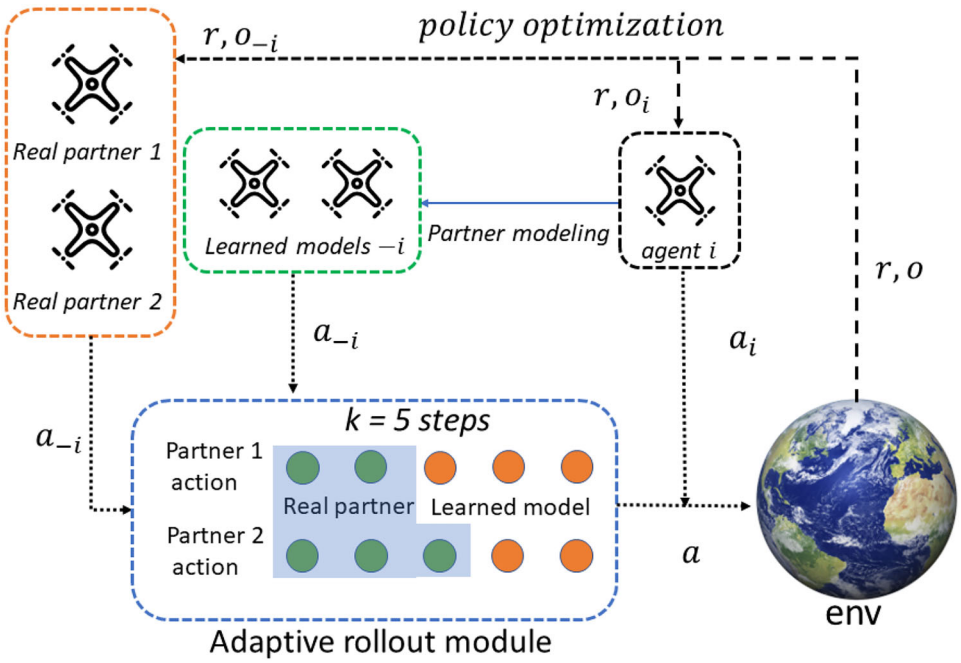


Fig. 3 Example of the adaptive rollout module from the agent i's perspective. Agent i will run k steps with other learned partner models. For each partner, agent i calculates the performance of its partner model and decides how many steps to interact with it within k steps

the empirical distribution of historical action that the agent i has taken at observation o_i until the time t, which can be computed as follow.

$$f_i^t = \frac{1}{1-t}\sum_{n=1}^{t}\psi_i^t(a_i). \tag{3}$$

Such that for each action interval $I \subset A$ the probability of the action a_i belonging to I is given by the integral of $f_i^t(o_i)$ over I. For example, if $I = [o_1, o_2]$, the probability can be expressed as:

$$\mathbb{P}(a_i) = \int_{o_1}^{o_2} f_i^t(\tau)\mathrm{d}t. \tag{4}$$

Adaptive partner rollout

Upon constructing partner models using the FSP technique, a theoretical analysis is conducted to establish an upper bound on the partner modeling error. This analysis allows us to identify the optimal number of steps that the real agent can communicate with the partner models, while still maintaining convergence of the DAPM algorithm under certain conditions. In model-based RL, the dynamic model error comes from two components, namely generalization error and distribution shift, which similarly affect partner model error. To ensure a consistent algorithmic improvement, the model can be enhanced by minimizing the discrepancy, represented as a constant value denoted as C, which quantifies the gap between the actual returns and the returns predicted by the model. In this research, the viewpoint of the agent i is adopted, and efforts are made to establish an upper bound C on the difference between the expected return of running policy with real partners $\eta_i[\pi_i, \pi_{-i}]$ and that of running policy with partner models $\hat{\eta}_i[\pi_i, \hat{\pi}_{-i}]$. This upper bound C can be formulated as follows:

$$\left|\eta_i[\pi_i, \pi_{-i}] - \hat{\eta}_i[\pi_i, \hat{\pi}_{-i}]\right| \leq C. \tag{5}$$

Since each partner model is trained using supervised learning based on historical trajectories taken by real partners, the error in each partner model can be quantified using standard Probably Approximately Correct (PAC) generalization bounds. In this work, the generalization error of partner agents for agent i is defined as $\epsilon_{-i}^{\hat{\pi}} = max_s D_{TV}(\pi_{-i}(\cdot|s)||\hat{\pi}_{-i}(\cdot|s))$, representing the validation loss of partner models on the time-dependent state distribution of the data-collecting policy π^D. The maximum total-variation distance $\epsilon_{-i}^{\pi} = \max_s D_{TV}(\pi_{-i}(\cdot|s)||\pi_{-i}^D(\cdot|s))$ denotes the distribution shift for partner models. By controlling the two errors in partner modeling, the research presents a bound:

$$\left|\eta_i[\pi_i, \pi_{-i}] - \hat{\eta}_i[\pi_i, \hat{\pi}_{-i}]\right| \leq 2r_{\max}\left[\frac{\gamma(\epsilon_{-i}^{\hat{\pi}} + 2\epsilon_{-i}^{\pi})}{1-\gamma} + 2\epsilon_{-i}^{\pi}\right] \tag{6}$$

Here, γ is the discount factor. The upper bound can be minimized by decreasing the discrepancy between real partner and learned partner models. To facilitate the dynamic adjustment of genuine agents and partner models, the model is trained in a two-step process. Initially, the model is trained with real agents over M steps to learn partner models for each agent. Subsequently, during the next k-steps, under certain conditions, the real partners are replaced with partner models to mitigate partner sample complexity. The method is like a length 1 branch k-steps rollout but performs in the real environment. Let η_i^{branch} denote the return obtained from this multi-agent k-steps rollout approach. By employing this approach, the returns can be bounded in the following manner:

$$\begin{aligned}&\left|\eta_i[\pi_i, \pi_{-i}] - \eta_i^{\text{branch}}[(\pi_0^D, \hat{\pi}_0), \ldots, (\pi_i^D, \hat{\pi}_i), \ldots, (\pi_n^D, \hat{\pi}_n)]\right| \\ &\leq 2r_{\max}\left[(k+1)\epsilon_{-i}^{\hat{\pi}} + \gamma^{k+1}\epsilon_{-i}^{\pi} + \frac{\gamma^{k+1}\epsilon_{-i}^{\pi}}{1-\gamma}\right] \\ &= C(\epsilon_{-i}^{\pi}, \epsilon_{-i}^{\hat{\pi}}, k)\end{aligned} \tag{7}$$

Here, the pair $(\pi_i^D, \hat{\pi}_i)$ means that the data collecting policy π_i^D and partner model policy $\hat{\pi}_i$ are used before and after the k-steps starts respectively for agent i. The upper bound denoted by C includes both the generalization errors and policy shifts for partner models.

Not like the classic k-steps rollout choosing the optimal $k = \text{argmin}_{k>0} C(\epsilon_{\pi}^{-i}, \epsilon_{\hat{\pi}}^{-i}, k)$, with low generalization error, the approach aims to maintain the rollout length while making adjustments to the interaction steps between real agents and partner models. As a result, the adaptive rollout approach shortens the rollout length for inaccurate partner models, while retaining a longer rollout length for accurate models, as depicted in Fig. 3.

Throughout the training process, both real agents and partner models exhibit performance enhancement and error reduction, leading to decreasing discrepancies. In the event of favorable model quality, an increased utilization of partner models becomes viable. Consequently, the proposed implementation involves an adaptive approach to dynamically use partner models. Technically, using partner models in too few steps with relatively accurate models, results in low partner sample efficiency, especially at the end of training. Long steps may deviate from the real trajectory distribution due to partner models' inaccuracies. Thus, the adaptive method reduces interaction steps with incorrect partner models but keeps more accurate ones.

From the previous discussion and referencing line 6 of Algorithm 1, the generalization error for each partner model, denoted as $\epsilon_j^{\hat{\pi}}$ is initially computed. Subsequently, the total generalization error for all partner models over k steps is calculated as: $(k+1)\epsilon_{-i}^{\hat{\pi}} = (k+1)\sum_{j\subset -i}\epsilon_j^{\hat{\pi}}$. Considering agent i's perspective, it can interact either with the real partner j following policy π_j, or with the partner model employing policy $\hat{\pi}_j$. Hence, in line 7, $\hat{\pi}_j$ is utilized for the first $n_j = \dfrac{\min_{j'}\epsilon_{j'}^{\hat{\pi}}}{\epsilon_j^{\hat{\pi}}}$ steps, then the real partner is interacted with by agent i for the remaining $k - n_j$ steps, as illustrated from lines 10 to 18. In this case, the generalization for each partner model is bounded by $k\min_{j'}\epsilon_{j'}^{\hat{\pi}}$. With such n_j defined, the generalization error for all partner models becomes $\sum_{j\subset -i}(n^j+1)\epsilon_j^{\hat{\pi}}$, which improves the η_i^{branch} by make the discrepancy bound tighter.

Policy learning

Finally, the MASAC algorithm is employed for policy learning, which utilizes an actor-critic scheme involving an actor that selects actions based on local observations, and a critic that estimates rewards considering all agents' observations and actions. In line 1 of Algorithm 1, the parameter ζ_i is assigned to the actor network, and the parameter w_i is assigned to the critic network.

After every epoch, the actor and critic networks are trained using a batch of samples that are randomly drawn from the replay buffer $\mathcal{D}$, employing a specified number of gradient descent steps. During training, the policy entropy is maximized to foster exploration, facilitate continuous learning, and minimize sample complexity. The critic network leverages TD learning method to enhance its policy, and the target for the critic is determined as follows:

$$y = r^{t+1} + \gamma\left(Q^{w^i}(s^{t+1}, a_i^{t+1}, a_{-i}^{t+1}) - \alpha \log \pi_{\zeta_i}(a_i^{t+1}|s^{t+1})\right) \tag{8}$$

The MASAC algorithm employs a log term to reward high-entropy policies, with the weight α used to control the level of entropy. The critic network's loss function is formulated as the mean squared error between the expected return from the target and the expected return predicted by the current critic network:

$$\mathcal{L}_{Q_i}(w_i) = \left(Q^{w_i}(s^t, a_i^t, a_{-i}^t) - y\right)^2 \tag{9}$$

The aim of the actor network is to optimize the agent's policy by maximizing both the expected return, which represents the cumulative reward, and the entropy, which promotes exploration and prevents determinism. The loss function used to update the policy gradient is as follows:

$$\mathcal{L}_{\pi_i}(\zeta_i) = -\left(Q^{w_i}(s^t, a_i^t, a_{-i}^t) - \alpha \log \pi^{\zeta_i}(a_i^t|s_i^t)\right) \tag{10}$$

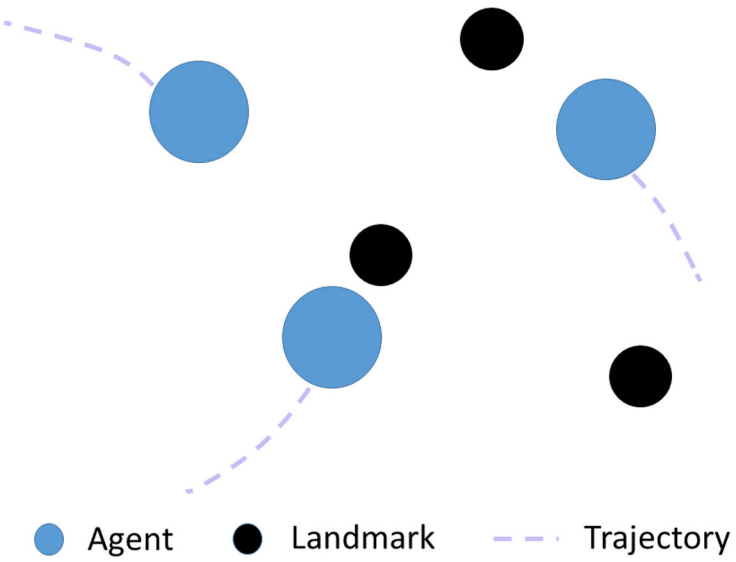
Experiments

In this section, the effectiveness of DAPM is assessed in comparison to various model-free algorithms such as MADDPG, MAPPO, and MATD3. Through the experiments, it is illustrate that DAPM exhibits higher sample-efficiency compared to these baselines in two distinct cooperative tasks.

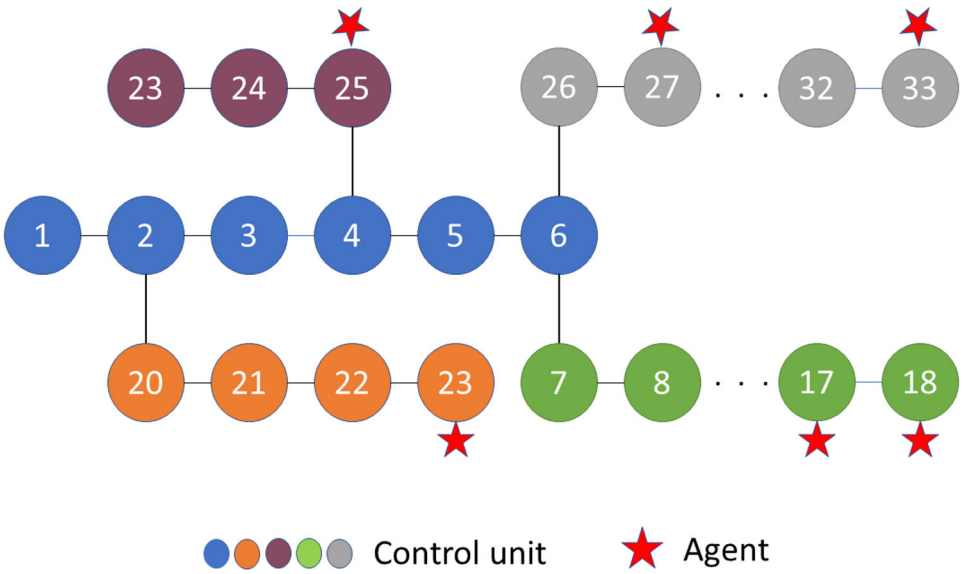
Environment descriptions

The algorithm is tested with other baselines in two different cooperative benchmarks, which are the multi-agent particle (MPE) [12] environment benchmark and multi-agent active voltage control on power distribution networks (MAPDN) [13].

1. *Cooperative navigation* The cooperative navigation task is a challenging scenario in MPE where multiple agents are trained to navigate in a shared environment with the goal of covering multiple landmarks without colliding with each other. This task requires agents to coordinate their actions and make decisions that are mutually beneficial in order to achieve the collective objective of covering all landmarks. In this task, each agent has access to several types of information, including its own speed and location, as well as the relative location of other agents and landmarks. The locations for all landmarks and agents are randomly generated at the beginning of each episode, which introduces variability and requires agents to adapt to changing environments. The agents are rewarded based on their performance. Specifically, the agents are rewarded based on a cumulative sum of negative minimum distances calculated between each landmark and the corresponding agent's position. This encourages agents to get as close as possible to the landmarks in order to cover them effectively. However, agents are also punished for collisions, which motivates them to avoid colliding with each other during navigation. Figure 4a provides a visual representation of cooperative navigation, showcasing the agents moving in the environment and covering the landmarks without colliding. This task poses several challenges, including the need for agents to learn effective navigation strategies, coordinate their actions to avoid collisions and adapt to changing environments with randomly generated agents and landmarks.
2. *MAPDN* The power distribution network employs distributed active voltage regulation, where multiple agents work collaboratively to control PV inverters that generate reactive power to maintain the voltage at each bus within a safe range of 0.95 p.u. $\leq v_k \leq$ 1.05 p.u., for all buses k in the system. The term "p.u." refers to the voltage unit used in the system. In this task, each agent is responsible for controlling a PV inverter to regulate the voltage at a specific bus in the distribution network. The agents need to cooperate with each other to manage the voltage of all the buses in the network effectively, as the decisions of one agent can impact the voltage levels observed by other agents due to the interconnected nature of power networks. It is important to note that not all buses in the network have a PV installed, making cooperation among agents crucial to ensure voltage stability across the entire network. Figure 4b provides a visual representation of the power distribution network, with different zones denoted by different colors. Each agent has limited information and can only observe par-



(a) Cooperative navigation



(b) MAPDN 33-bus network

Fig. 4 Illustrations of the two benchmark domains

tial information about the system. For example, the agent responsible for controlling the PV inverter at bus 23 can only access information from zone 2 of the network. To maintain the voltage within the desired range, agents need to make coordinated decisions on reactive power generation by adjusting the output of their respective PV inverters. The agents rely on communication and coordination to exchange information and take actions based on the observed voltage levels and other relevant data. The objective is to collectively regulate the voltage levels at all the buses in the network while considering the limitations of each agent's observation capabilities and the interconnected nature of the power distribution network.

Table 1 Hyperparameters used in different benchmarks

Hyperparameter	Environment	
	MPE	MAPDN
Number of steps in one epoch	25	240
Learning rates [actor, critic, partner model]	[0.0005, 0.0005, 0.0005]	[0.0001, 0.0001, 0.0001]
Hidden layer sizes	64	64
Batch size	1024	32
Rollout length	$1 \rightarrow 20$	$1 \rightarrow 100$
Discount factor	0.95	0.99
Target networks update rate	0.01	0.01

Simulation settings

To ensure a fair comparison between DAPM and other baselines, the hyperparameters of both algorithms are kept consistent across different environments. However, the only distinction is a binary switch that determines the choice between communicating with the partner models π_j^{ϕ} and real partners π_j^{ζ}. In the MPE experiment, the action space is discrete, and the actor, critic, and partner model networks in DAPM are parameterized with a three-layer MLP using ReLU activation functions. The optimizer used for all networks is Adam optimizer. The length of each episode is fixed at 25 timesteps, and the performance of the partner model is evaluated using accuracy. The MAPDN experiment involves a 33-bus network with 5 distinct regions and 6 agents, as shown in Fig. 4b. Each zone is visually differentiated by a unique color scheme, and each agent is represented by a star symbol. Similar to the MPE experiment, all networks in this environment are parameterized by a three-layer MLP with ReLU activation functions, but DAPM use RMSprop optimizer for parameter updates. The action range is set within $[-0.8, 0.8]$ to ensure the safety of distribution networks following the MAPDN experiment. The episode length is set to be 240 timesteps, which exceeds the episode duration typically employed in the MPE. During training, test results are reported using the median and 25–75% quartile shading with 5 random seeds. The evaluation metric for this experiment is the controllable rate (CR), which measures the ratio of time steps where all buses' voltages are under control. A comprehensive list of hyperparameter settings can be found in Table 1.

Results

The convergence speed of the average reward value is first analyzed with a different number of agents in various tasks to demonstrate the sample efficiency of DAPM. Second, win rate and collision rate are utilized to demonstrate DAPM is superior to other algorithms. Finally, the adaptive rollout method is demonstrated to avoid the performance drop of DAPM with four different rollout lengths k.

Sample efficiency analysis

To establish the superior sample efficiency of the method over other baseline approaches across various tasks and agent quantities, the illustration begins with MPE. Experiments are conducted with fixed agent quantities of 3 and 5, as depicted in Fig. 5. In Fig. 5a, the training involves 3 agents with 3 landmarks in the environment. The training results reveal that the DAPM converges about 0.75 M interactions while other baselines are still divergent. Here, MATD3 is the best performance baseline. It converges at 1.1 M interactions, resulting in a reduction of approximately 31.82% in interaction. In Fig. 5b, the environment contains 5 agents, which is more complex than the previous task. The figure shows that MADDPG and MATD3 have not converged at the end of training steps and resulted in a significantly lower average reward. DAPM starts to converge at 0.75 M interactions, which takes 28.57% fewer interactions than MAPPO. In Fig. 5c, the policy is trained in MAPDN with 6 agents. The figure shows that DAPM starts to converge at 140 k interactions, while MADDPG converge at 187 k interactions, resulting in a reduction of approximately 25.13% in interaction. MAPPO can converge around 140 k interactions like DAPM, however, it has 0.21 less controllable rate, and a more detailed comparison can be seen in Table 2. Therefore, partner models indeed generate useful extra samples when running rollouts, which help agents learn fast. Based on the analysis, it can be concluded that the DAPM can reach asymptotic performance and final performance as others with fewer interactions. This indicates a reduction in partner sample complexity and an accelerated learning process for agent policies through the utilization of partner models with the adaptive rollout method.

Stability and win rate analysis

As observed in Fig. 5, the fluctuation of DAPM curve will be smaller than other algorithms, which means DAPM is more

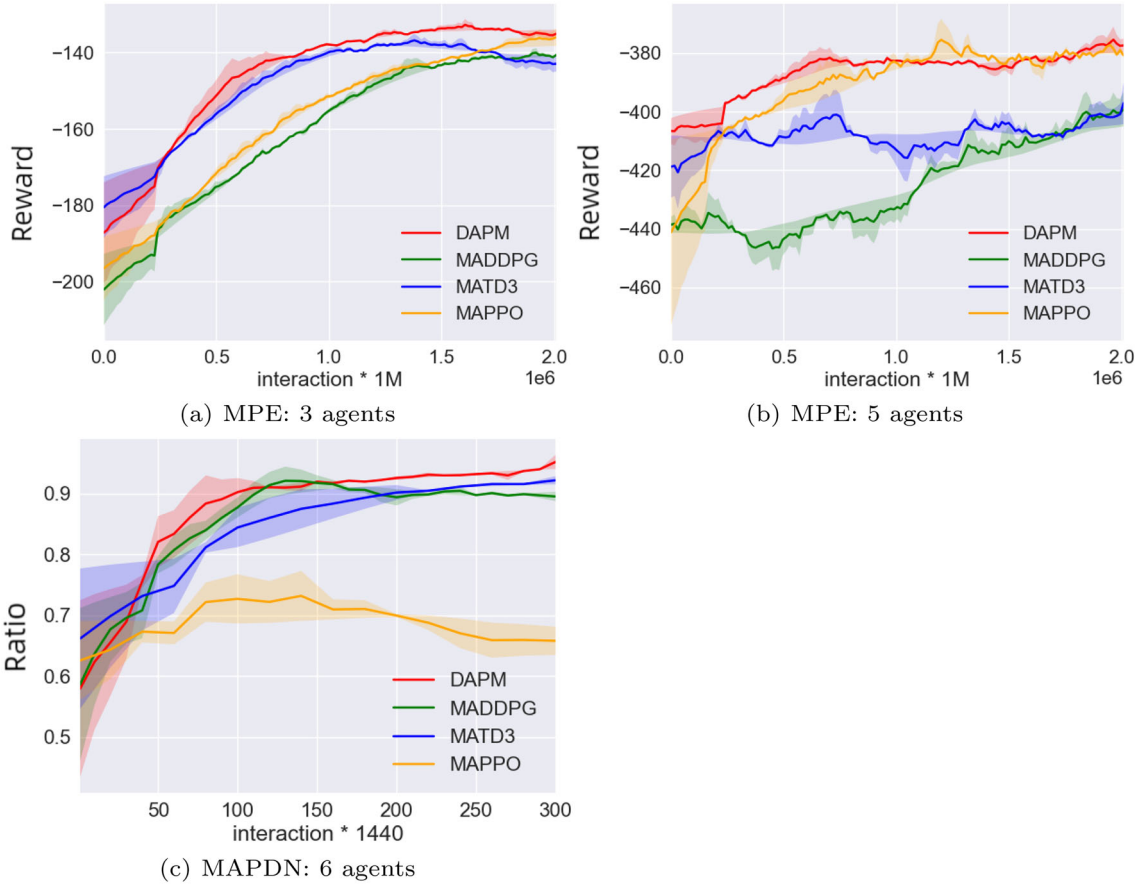


Fig. 5 Comparison between the performance of DAPM in MPE and MAPDN. Interaction means the communication times between the ego agent with other agents. The mean and standard error are depicted using bold lines and shaded areas, respectively. **a** MPE environment and the number of agents in is 3. **b** MPE environment and the number of agents is 5. **c** MAPDN environment and the number of agents is 6

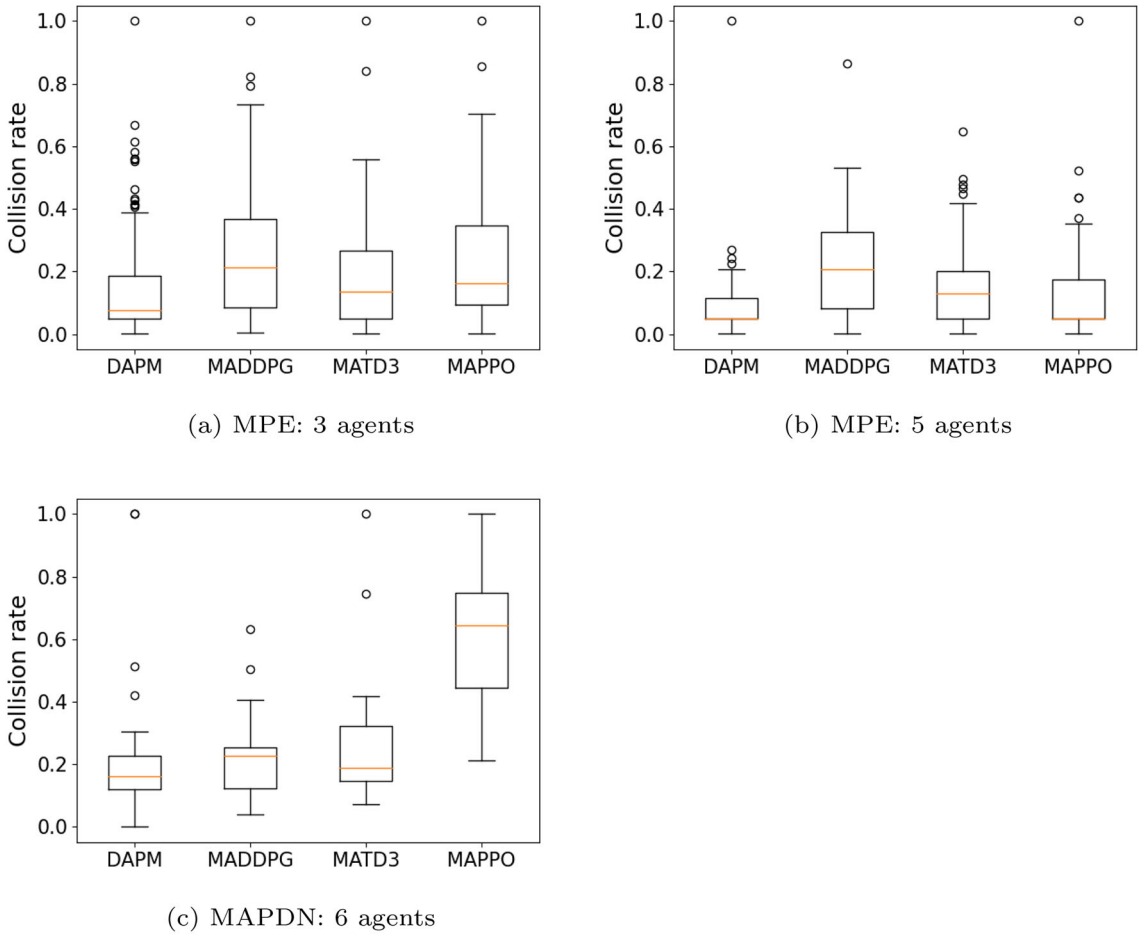
Table 2 Performance analysis on converge speed ($\star$) and average reward ($*$) or controllable rate ($\circ$)

Task	MPE: 3 agents		MPE: 5 agents		MAPDN: 6 agents	
	$\star$	$*$	$\star$	$*$	$\star$	$\circ$
Algorithm						
MADDPG	1.8 M	−141	None	None	187 k	0.9
MATD3	1.1 M	−140	None	None	244 k	0.9
MAPPO	1.7 M	**−139**	1.05 M	**−380**	**140 k**	0.7
DAPM	**0.75 M**	**−139**	**0.75 M**	**−380**	**140 k**	**0.91**

The bolded part under 'converage speed' shows how often the agent communicates with the environment, with 'M' representing million times and 'k' representing thousand times. Less communication means better sample efficiency. Also, bolded parts related to reward and controllable rate indicate that DAPM achieves the same reward as the best baseline. This means DAPM improves sample efficiency without sacrificing final performance

stable than others during the training process. In the cooperative MPE environment, agents are penalized for collisions, contributing to curve fluctuations. To compare the stability of each method more clearly, boxplots are utilized to depict the collision rate for each algorithm in different tasks. The collision rate indicates the frequency of agents crashing into each other in MPE and the occurrence of voltage going out of control in MAPDN.

Figure 6 clearly shows that DAPM exhibits the lowest collision rate in every subfigure, signifying its superior stability. As partner models continue to refine their policies during the training process, the expectation is to utilize more refined average policies towards the conclusion of training. This keeps the model from exploiting the environment, which can lead to a high collision rate, especially if there are many agents. From Fig. 5, DAPM's average reward is slightly higher than the best baseline for each task. As shown



(a) MPE: 3 agents

(b) MPE: 5 agents

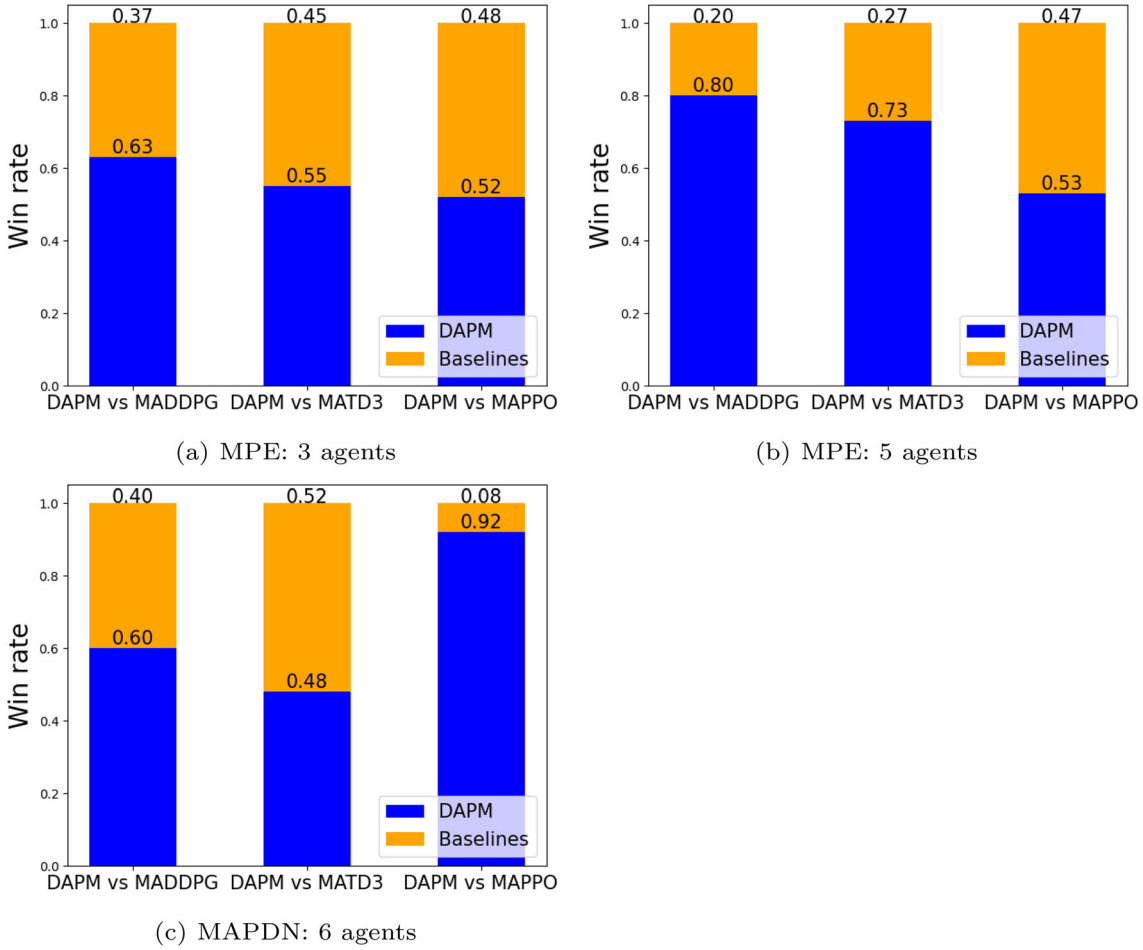
(c) MAPDN: 6 agents

Fig. 6 Stability comparison in different tasks

in Fig. 7, the rewards obtained by different algorithms within 1000 episodes are measured to demonstrate the superiority of DAPM in cooperative tasks. From Fig. 7a, b, the win rate of DAPM is slightly superior to the best performance baseline and 0.04 less than MATD3 in Fig. 7c. Based on the results, DAPM has the same performance as the best baseline. This is because the policy network still trains using the soft-actor-critic framework, and the only thing that has changed is where the samples in the buffer come from. This method does not impact the performance but achieves it with fewer interactions. Overall, the approach makes the training process more stable and faster.

Performance and statistical analysis with different rollout length

The performance of agents in DAPM is strongly related to the quality and usage of partner models. It is acknowledged that partner models have a model bias with real partners, and agents learn to exploit model inaccuracies after interacting with them for too many steps will lead to a lower performance. To prove the adaptive rollout method can avoid performance dropping by increasing the rollout length, the performance of DAPM with four different rollout lengths is compared in Fig. 8. Additionally, the Analysis of Variance method (ANOVA) [41, 42] was used to test whether changing the rollout length affects the final average reward. In this statistical analysis, each ANOVA is summarized with an F-statistic and a p value. In this context, where ANOVAs are repeated for each task, a Holm–Bonferroni correction is applied to control the probability of false positives. From Fig. 5a and b, it is evident that the average reward converges, regardless of the number of involved agents. For simplicity, the analysis is conducted with 3 agents in the MPE task, and two analyses are performed for the two tasks. As shown in Fig. 8a, length 10 got the best performance compared with other settings, which can achieve a higher reward and use fewer steps to converge while others can still converge to a reasonable average reward with extra steps. Surprisingly, length 5 got the worst performance. One possible explanation is that using a small number of partner models may create samples that are far from the normal distribution. When mod-



(a) MPE: 3 agents

(b) MPE: 5 agents

(c) MAPDN: 6 agents

Fig. 7 Win rate comparison in different tasks

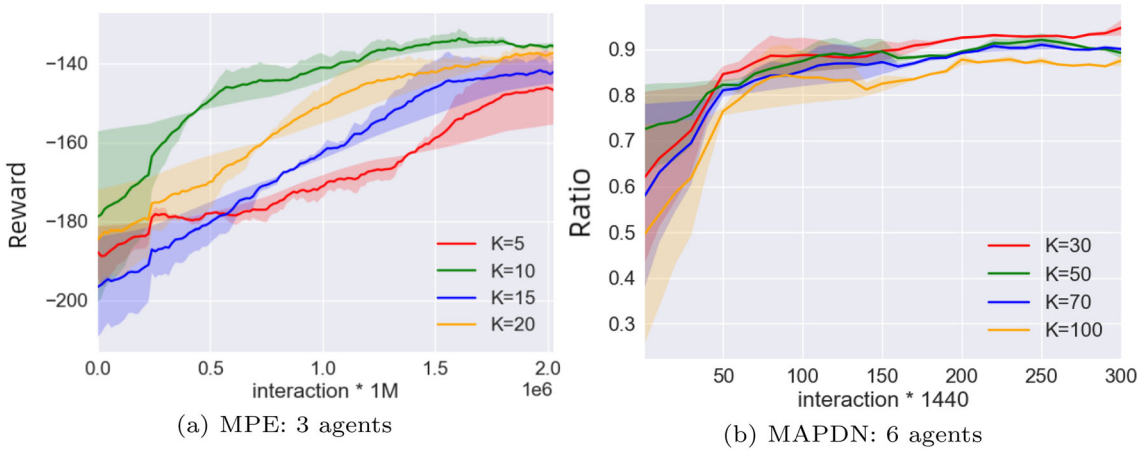
els are trained on these polluted batch samples, they tend to explore in the wrong direction. In Fig. 8b, length 30 performed the best compared to the other settings. The results show that as the partner model is used more, the algorithm will get a lower CR and converge later. This shows that the adaptive rollout method reduces some of the negative effects of partner models by using different rollout lengths of partner models.

In Fig. 9, we observe that varying the rollout length has minimal impact on the average reward achieved. Additionally, statistical analysis using the F-statistic and p value indicates no significant correlation between rollout length and final performance within each environment. This is supported by two separate ANOVA tests in both MPE and MAPDN environments, where the rollout length parameter (k) does not exert a significant statistical influence on the average reward. In MPE, F equals 0.94 with a p value of 0.43 (Fig. 9a), and in MAPDN, F equals 2.08 with a p value of 0.11 (Fig. 9b). The result proves that extending the rollout length can reduce communication instances without negatively impacting model performance. However, an excessively long rollout length can slow down convergence speed. Therefore, it is crucial to choose a suitable rollout length.

Based on the analysis above, the adaptive rollout method does avoid performance dropping by increasing the usage of partner models.

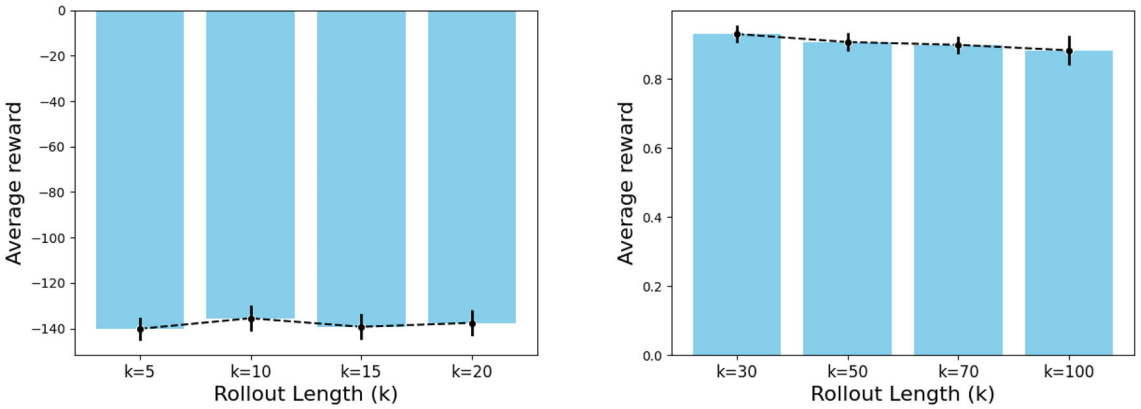
Conclusion

This study aims to contribute to the field of sample efficient partner modeling in MARL by specifically focusing on the development of sample-efficient partner modeling techniques. The intended application scenarios are real-life online learning settings that involve the coordination and collaboration of multiple robots. To investigate the benefits of partner modeling in enhancing multi-robot sample efficiency, we first examined the sample complexity associated with partner modeling and then conducted a theoretical analysis of DAPM. By leveraging partner models, DAPM is able to reduce the sample complexity. Simulation results



(a) MPE: 3 agents (b) MAPDN: 6 agents

Fig. 8 Performance curves of different rollout lengths on DAPM



(a) MPE: 3 agents (b) MAPDN: 6 agents

Fig. 9 Impact of rollout length k on final performance. Error bars represent 95% confidence intervals calculated from 20 independent runs. Result: rollout length has no influence on the average reward for both MPE: 3 agents and MAPDN: 6 agents

demonstrated that DAPM achieves comparable asymptotic performance to other model-free baselines but with lower sample complexity. Nevertheless, despite the significant reduction in sample complexity, DAPM still required a substantial number of trials to achieve satisfactory performance, emphasizing the need for further advancements in the pursuit of real-life learning in MAS. Additionally, it is crucial to highlight that the manuscript has not addressed the potential issues of model learning speed and scalability in MARL.

Future studies aim to enhance the model's learning speed and quality using the MARL framework, which includes designing a careful reward function to guide the learning process. Using well-structured rewards could speed up learning by helping agents focus on vital aspects of the task. Also, attention mechanisms could be employed to tackle scalability issues, ensuring that the proposed techniques can scale effectively to more complex scenarios. Furthermore, to explore the application of partner modeling in more complex real-world tasks. One possible approach is to merge sim-to-real [43] learning with partner modeling, enabling agents to utilize their acquired knowledge when working in real-world conditions instead of beginning from the beginning. This approach could potentially reduce the time and resources required to train an agent in a real-world setting, as well as improve its performance.

In summary, while this manuscript contributes significantly to the understanding of sample-efficient partner modeling in MARL, we acknowledge the need for future research to address model learning speed, scalability challenges, and further explore the application of partner modeling in complex real-world tasks.

Funding This research was supported by Suzhou Science and Technology Project (Grant SYG202122), Research Development Fund of XJTLU (Grant RDF-19-02-23), Key Programme Special Fund of XJTLU (Grant KSF-A-19) and Suzhou Municipal Key Laboratory for Intelligent Virtual Engineering (Grant SZS2022004).

Data Availability Our data originates from interactions with the environment, specifically the MPE and MAPDN environments, both of

which are openly accessible on GitHub. The articles associated with these environments also include reference links.

Declarations

Conflict of interest All the authors who list in this study declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

1. Brown N, Sandholm T (2019) Superhuman ai for multiplayer poker. Science 365:885–890
2. Vinyals M, Rodriguez-Aguilar JA, Cerquides J (2011) A survey on sensor networks from a multiagent perspective. Comput J 54(3):455–70
3. Zhou M, Luo J, Villella J, Yang Y, Rusu D, Miao J, Zhang W, Alban M, FADAKAR I, Chen Z, Huang C, Wen Y, Hassanzadeh K, Graves D, Zhu Z, Ni Y, Nguyen N, Elsayed M, Ammar H, Cowen-Rivers A, Ahilan S, Tian Z, Palenicek D, Rezaee K, Yadmellat P, Shao K, chen d, Zhang B, Zhang H, Hao J, Liu W, Wang J (2021) Smarts: an open-source scalable multi-agent rl training school for autonomous driving. In: Proceedings of the 2020 conference on robot learning, vol 155. PMLR, pp 264–285
4. Long P, Fan T, Liao X, Liu W, Zhang H, Pan J (2018) Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 6252–6259
5. Tian Z, Zou S, Davies I, Warr T, Wu L, Ammar HB, Wang J (2020) Learning to communicate implicitly by actions. In: Proceedings of the AAAI conference on artificial intelligence, vol 34. AAAI Press, pp 7261–7268
6. Sukhbaatar S, Fergus R et al (2016) Learning multiagent communication with backpropagation. PMLR, pp 1556–1566
7. Iqbal S, Sha F (2019) Actor-attention-critic for multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 2961–2970
8. Wang T, Bao X, Clavera I, Hoang J, Wen Y, Langlois E, Zhang S, Zhang G, Abbeel P, Ba J (2019) Benchmarking model-based reinforcement learning. arXiv:1907.02057
9. Willemsen D, Coppola M, de Croon GC (2021) Mambpo: sample-efficient multi-robot reinforcement learning using learned world models. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 5635–5640
10. Heinrich J, Lanctot M, Silver D (2015) Fictitious self-play in extensive-form games. In: International conference on machine learning. PMLR, pp 805–813
11. Heinrich J, Silver D (2016) Deep reinforcement learning from self-play in imperfect-information games. In: NIPS deep reinforcement learning workshop. ACM. arXiv:1603.01121
12. Lowe R, Wu YI, Tamar A, Harb J, Pieter Abbeel O, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. ACM
13. Wang J, Xu W, Gu Y, Song W, Green TC (2021) Multi-agent reinforcement learning for active voltage control on power distribution networks. In: Advances in neural information processing systems. ACM, pp 3271–3284
14. Bacanin N, Stoean R, Zivkovic M, Petrovic A, Rashid TA, Bezdan T (2021) Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: application for dropout regularization. In: Mathematics. MDPI, p 2705
15. Bacanin N, Zivkovic M, Al-Turjman F, Venkatachalam K, Trojovskỳ P, Strumberger I, Bezdan T (2022) Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application. Sci Rep 12:6302
16. Sunehag P, Lever G, Gruslys A, Czarnecki WM, Zambaldi V, Jaderberg M, Lanctot M, Sonnerat N, Leibo JZ, Tuyls K et al (2018) Value-decomposition networks for cooperative multi-agent learning based on team reward. In: International conference on autonomous agents and multi-agent systems. ACM, pp 2085–2087
17. Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S (2018) Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 4295–4304
18. Rashid T, Farquhar G, Peng B, Whiteson S (2020) Weighted QMIX: expanding monotonic value function factorisation. ACM, pp 10199–10210
19. Mao H, Zhang Z, Xiao Z, Gong Z (2018) Modelling the dynamic joint policy of teammates with attention multi-agent ddpg. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems. International foundation for autonomous agents and multiagent systems. ACM, pp 1108–1116
20. Sun W, Jiang N, Krishnamurthy A, Agarwal A, Langford J (2018) Model-based reinforcement learning in contextual decision processes. arXiv:1811.08540
21. Levine S, Finn C, Darrell T, Abbeel P (2016) End-to-end training of deep visuomotor policies, pp 1334–1373. JMLR. org
22. Wilson J, Borovitskiy V, Terenin A, Mostowsky P, Deisenroth M (2020) Efficiently sampling functions from Gaussian process posteriors. In: International conference on machine learning. PMLR, pp 10292–10302
23. Rajeswaran A, Ghotra S, Ravindran B, Levine S (2017) Epopt: v neural network policies using model ensembles. In: International conference on learning representations. IEEE
24. Depeweg S, Hernández-Lobato JM, Doshi-Velez F, Udluft S (2016) Learning and policy search in stochastic dynamical systems with Bayesian neural networks. In: International conference on learning representations. IEEE
25. Sutton RS (1990) Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Machine learning proceedings 1990. Elsevier, pp 216–224
26. Tesauro G, Galperin G (1996) On-line policy improvement using Monte-Carlo search. In: Advances in neural information processing systems. ACM, pp 1068–1074
27. Tesauro G et al (1995) Temporal difference learning and td-gammon. Commun ACM 38:58–68
28. Feinberg V, Wan A, Stoica I, Jordan MI, Gonzalez JE, Levine S (2018) Model-based value estimation for efficient model-free reinforcement learning
29. Buckman J, Hafner D, Tucker G, Brevdo E, Lee H (2018) Sample-efficient reinforcement learning with stochastic ensemble value

expansion. In: Advances in neural information processing systems. ACM, pp 8224–8234
30. Luo Y, Xu H, Li Y, Tian Y, Darrell T, Ma T (2019) Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In: International conference on learning representations. IEEE
31. Janner M, Fu J, Zhang M, Levine S (2019) When to trust your model: model-based policy optimization. In: Advances in neural information processing systems. ACM, pp 12519–12530
32. Park YJ, Cho YS, Kim SB (2019) Multi-agent reinforcement learning with approximate model learning for competitive games. PLoS One 14:0222215
33. Krupnik O, Mordatch I, Tamar A (2020) Multi-agent reinforcement learning with multi-step generative models. In: Conference on robot learning. PMLR, pp 776–790
34. Bargiacchi E, Verstraeten T, Roijers DM (2021) Cooperative prioritized sweeping. In: AAMAS. ACM, pp 160–168
35. Brown GW (1951) Iterative solution of games by fictitious play, pp 374–376
36. He H, Boyd-Graber J, Kwok K, Daumé III H (2016) Opponent modeling in deep reinforcement learning. In: International conference on machine learning. PMLR, pp 1804–1813
37. Papoudakis G, Albrecht SV (2020) Variational autoencoders for opponent modeling in multi-agent systems. In: The Association for the Advancement of Artificial Intelligence. AAAI Press
38. Raileanu R, Denton E, Szlam A, Fergus R (2018) Modeling others using oneself in multi-agent reinforcement learning. PMLR
39. Rabinowitz N, Perbet F, Song F, Zhang C, Eslami SA, Botvinick M (2018) Machine theory of mind. In: International conference on machine learning. PMLR, pp 4218–4227
40. Zhang K, Yang Z, Liu H, Zhang T, Basar T (2018) Fully decentralized multi-agent reinforcement learning with networked agents. In: International conference on machine learning. PMLR, pp 5872–5881
41. Fisher RA (1970) Statistical methods for research workers. In: Breakthroughs in statistics: methodology and distribution. Springer, pp 66–70
42. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. In: Swarm and evolutionary computation. Elsevier, pp 3–18
43. Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 23–30