Off-Agent Trust Region Policy Optimization

Ruiqing Chen[1,2,*], **Xiaoyuan Zhang**[1,*], **Yali Du**[3], **Yifan Zhong**[1], **Zheng Tian**[2], **Fanglei Sun**[2] and **Yaodong Yang**[1,†]

[1]Institute for AI, Peking University, Beijing, China
[2]ShanghaiTech University, Shanghai, China
[3]King's College London, UK
yaodong.yang@pku.edu.cn

Abstract

Leveraging the experiences of other agents offers a powerful mechanism to enhance policy optimization in multi-agent reinforcement learning (MARL). However, contemporary MARL algorithms often neglect experience sharing possibilities or adopt a simple approach via direct parameter sharing. Our work explores a refined **off-agent** learning framework that allows selective integration of experience from other agents to improve policy learning. Our investigation begins with a thorough assessment of current mechanisms for reusing experiences among heterogeneous agents, revealing that direct experience transfer may result in negative consequences. Moreover, even the experience of homogeneous agents requires modification before reusing. Our approach introduces off-agent adaptations to the multi-agent policy optimization methods, enabling effective and purposeful leverage of cross-agent experiences beyond conventional parameter sharing. Accompanying this, we provide a theoretical guarantee for an approximate monotonic improvement. Experiments conducted on the StarCraftII Multi-Agent Challenge (SMAC) and Google Research Football (GRF) demonstrate that our algorithms outperform state-of-the-art (SOTA) methods and achieve faster convergence, suggesting the viability of our approach for efficient experience reusing in MARL.

1 Introduction

Multi-agent reinforcement learning (MARL) [Busoniu *et al.*, 2008; Yang and Wang, 2020] aims to develop multi-agent systems by enabling agents to co-evolve towards their respective goals of reward maximization. Recently, substantial advancements in both algorithms and testing environments have been achieved in MARL. This approach has proven to be effective in multiplayer games [Peng *et al.*, 2017; Baker *et al.*, 2020; Brown and Sandholm, 2019; Vinyals *et al.*, 2019], intelligent transportation systems [Adler and Blue, 2002], sensor networks [Zhang and Lesser, 2011], and energy networks [Glavic *et al.*, 2017; Qiu *et al.*, 2023]. Fascinating as these results are, a long-standing problem of MARL is its extremely

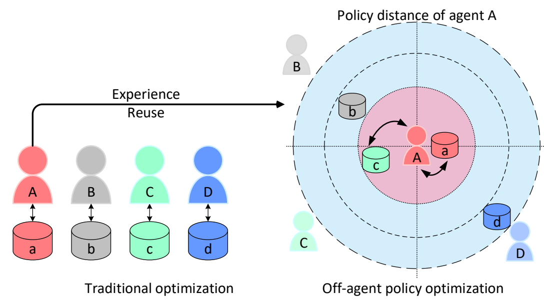


Figure 1: Experience utilization process: Independently, agents update policies using personal experiences (left). Alternatively, agent A assimilates the experiences of others, determined by policy distance; proximity within the circle correlates with policy consistency to agent A (right). Red and blue regions reflect experiences agent A adopts or omits, respectively.

high sample complexity, as the original single-agent sample complexity problem is exacerbated by multi-agent interactions. Existing methods attempt to enhance sample efficiency by designing off-policy algorithms or sharing agents' parameters. However, off-policy algorithms are direct extensions of single-agent counterparts and do not specifically address experience reuse between agents, while parameter-sharing methods only partially resolve homogeneous-agent cooperation issues. A more natural approach to improve sample efficiency remains to be proposed.

Analyzing learning patterns in human society reveals a common phenomenon of individuals subconsciously learning from their peers to improve their strategies [Jarrahi, 2018]. For instance, in soccer, players acquire effective techniques demonstrated by teammates, later employing them to achieve superior performance. Conversely, if the behaviors of the partner result in suboptimal outcomes, others will avoid repeating the same mistakes. These patterns also appear in team activities like military actions and rescue missions. This demonstrates that humans can learn not only from their own experiences but also from others, rapidly mastering complex tasks through limited trials. This observation inspires us to consider sharing valuable knowledge among agents to naturally improve sample efficiency. The significance of this lies in the fact that the improvement in sample complexity increases with the number of agents, which can be considerable in large-scale multi-agent settings [Zhou *et al.*, 2019]. To our understanding, there seems to be limited exploration into the specific conditions for agent reuse and the extent to which experiences from other agents may be leveraged.

In our study, we investigate how learning from others' experiences contributes to efficient knowledge sharing among agents. We merge on-policy and off-policy methods into a unified approach termed *off-agent policy optimization*, covering heterogeneous and homogeneous agent scenarios depicted in Figure 1. Our primary contributions include:

- We propose an off-agent framework, defining conditions for experience reuse and establishing theoretical support for approximate monotonic policy improvement. Based on these conditions, we devise a mechanism for experience sharing and a distribution of experience mappings between agents.
- We introduce two practical algorithms—Off-Agent TRPO (OATRPO) and Off-Agent PPO (OAPPO)—and validate their effectiveness and adaptability through a tailored Maze game, highlighting the benefits and effects of various mapping distributions.
- Additionally, through evaluations on the StarCraftII Multi-Agent Challenge (SMAC) and Google Research Football (GRF), we demonstrate that OAPPO and OATRPO not only achieve SOTA performance on benchmark tasks but also offer enhanced sample efficiency and stable performance improvements.

2 Related Work

Our work, grounded in the MARL policy optimization framework, relates methods of experience reuse to off-policy methods. We subsequently review the literature on MARL policy optimization and off-policy topics, distinguishing our algorithm from imitation learning.

In the field of cooperative MARL, Independent Proximal Policy Optimization (IPPO) [de Witt *et al.*, 2020] was introduced to address the gap in Trust Region Learning (TRL). This approach views other agents as part of the environment when updating a single agent. However, this method gives rise to non-stationary issues as other agents continuously evolve. To address this problem, the centralized training with decentralized execution paradigm (CTDE) was introduced.[Lowe *et al.*, 2017; Foerster *et al.*, 2018; Zhou *et al.*, 2021]. Here, agents make decisions based on their local observations and update policies with a global value function. Building on the CTDE training paradigm, several effective multi-agent policy optimization algorithms have been proposed. MADDPG [Lowe *et al.*, 2017] provides the first general solution for cooperative and competitive scenarios, while Coordinated Proximal Policy Optimization (CoPPO) [Wu *et al.*, 2021] extends trust region methods to multi-agent systems. Multi-Agent Proximal Policy Optimization (MAPPO) [Yu *et al.*, 2021] significantly improves sample efficiency through parameter sharing. However, when faced with a heterogeneous agent scenario, MAPPO can lead to exponentially worse suboptimal outcomes [Kuba *et al.*, 2021]. Heterogeneous-Agent Trust Region Policy Optimization (HATRPO) algorithm [Kuba *et al.*, 2021] employs sequential policy update methods to avoid the joint policy stuck in local optimums. Nevertheless, it underperforms in challenging scenarios due to poor sample efficiency. [Sun *et al.*, 2023] assures monotonicity for trust regions in non-stationary settings, and [Wang *et al.*, 2023] does likewise for agent interactions. Nevertheless, their studies omit theoretical analysis of agents' experience reuse. Our off-agent algorithm designs a scheme for agents to reuse other agents' experiences, significantly enhancing sample efficiency in MARL policy optimization with guaranteed performance.

Off-policy algorithms enhance sample efficiency by reusing samples gathered from previous policies. Numerous off-policy algorithms, including Deep Deterministic Policy Gradient (DDPG) [Lillicrap *et al.*, 2015] and Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018], have been proposed within the MARL context. However, these methods may suffer from significant bias due to off-policy data. Approaches like Policy-on Policy-off Policy Optimization (P3O) [Fakoor *et al.*, 2020] and Actor Critic with Experience Replay (ACER) [Wang *et al.*, 2016], which combine on-policy and off-policy methods, can significantly mitigate distribution shifting. However, they lack a theoretical performance guarantee. Recently, Generalized Proximal Policy Optimization with Experience Reuse (GePPO) [Queeney *et al.*, 2021] proposed a PPO-based method that integrates off-policy and on-policy methods, offering a lower bound to ensure performance improvement. However, it merely considers sample reuse in the time dimension and single-RL scenarios. Shared Experience Actor Critic (SEAC) [Christianos *et al.*, 2020] has proposed a method for utilizing experiences among agents, yet it has not explored the extent to which agents can reuse experiences. Although Selective Parameter Sharing (SEPS) [Christianos *et al.*, 2021] has addressed the issue of agent selection, it has not yet investigated the conditions for possible agent reuse. To our knowledge, there appears to be a scarcity of comprehensive research and established standards on methods and conditions for agent experience reuse. Consequently, we investigated the conditions for experience reuse between agents and proposed an off-agent PG algorithm that allows an agent to update from the experiences of other agents while ensuring performance improvement.

Our algorithms facilitate learning from the experiences of other agents, a concept similar to the paradigm of imitation learning. Imitation Learning (IL) [Hussein *et al.*, 2017; Osa *et al.*, 2018; Oh *et al.*, 2018; Zare *et al.*, 2023] aims to emulate the behavior of experts in specific tasks by learning the mapping between observations and actions. While IL requires expert data, our method does not, as it updates their policy from the exchange of experiences between agents.

3 Preliminaries

We consider a *multi-agent decentralized partially observable Markov decision process* (Dec-POMDP) [Oliehoek and Amato, 2016], defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, R, Z, \mathcal{O}, p, \gamma \rangle$. Here, $\mathcal{N} = \{1, \dots, n\}$, $\mathcal{S}$ represents the finite state space, and $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}^i$ denotes the joint action space, composed of the actions of each agent. The transition probability function is $p : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$, $\Delta_{\mathcal{S}}$ means the distribution of state, while $R(s, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ stands for the reward function, and $\gamma \in [0, 1)$ is the discount factor.

In a partially observable scenario, the observation of each agent i is $o_i \in \mathcal{O}$, given by the observation mapping: $Z(s, \boldsymbol{a}) \rightarrow \mathcal{O}$. We consider the decision of each agent as a stationary policy $\pi_i : \mathcal{O} \rightarrow \Delta_{\mathcal{A}}$, where $\Delta_{\mathcal{A}}$ means the distribution of action. The agents interact with the environment as per the following protocol: at time step $t \in \mathbb{N}$, the agents are at state $s_t \in \mathcal{S}$, with agent i observing o_t^i; agent i takes an action $a_t^i \in \mathcal{A}^i$, drawn from its policy $\pi_i(\cdot \mid o_t^i)$, which, in conjunction with the actions of other agents, forms a joint action $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^n) \in \mathcal{A}$, drawn from the joint policy $\boldsymbol{\pi}(\cdot \mid s_t) = \prod_{i=1}^n \pi_i(\cdot \mid o_t^i)$; the agents receive a joint reward $r_t = R(s_t, \boldsymbol{a}_t) \in \mathbb{R}$, and transition to a state $s_{t+1} \sim p(\cdot \mid s_t, \boldsymbol{a}_t)$.

In a *fully-cooperative* setting, all agents share the same reward function and aim to maximize the expected total reward $J(\pi) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $\tau \sim \pi$ represents a process of sampling followed by $s_0 \sim \rho^0, \boldsymbol{a}_t \sim \boldsymbol{\pi}(\cdot \mid s_t)$ and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. The joint policy $\boldsymbol{\pi}$, the transition probability function p, and the initial state distribution ρ^0, induce a marginal state distribution at time t, denoted as $\rho_{\boldsymbol{\pi}}^t$. We define the marginal state distribution as $\rho_{\boldsymbol{\pi}} \triangleq \sum_{t=0}^{\infty} \gamma^t \rho_{\boldsymbol{\pi}}^t$. For an agent i, we define the observation distribution as $o_i \sim \eta_{\pi_i}(o) \triangleq \sum_{t=0}^{\infty} \gamma^t \eta_{\boldsymbol{\pi}_i}^t(o^t = o_i)$.

The state value function and the state action value function are defined as: $V_{\boldsymbol{\pi}}(s) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$ and $Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}\right]$. Subsequently, the advantage function is given as $A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - V_{\boldsymbol{\pi}}(s)$. $\boldsymbol{a}_{1:m}$ denote joint action $[a_1, a_2 \cdots a_m]$.

4 Off-Agent Policy Optimization

In this section, we present the framework for our Off-Agent Policy Optimization (OAPO). In the following section, we assume that all agents in the same scenario share the same state and action space. Section 4.1 outlines the theoretical framework for the reuse of experience and reviews the traditional MARL algorithm in this context. Moving to Section 4.2, we initially define the conditions required for efficient experience reuse among agents and subsequently propose the Off-Agent Multi-Agent Trust Region Method as a solution to these challenges. Ultimately, we perform a theoretical analysis of the proposed algorithm, leading to an approximate monotonic improvement guarantee.

4.1 Revisit Experience Reuse in Multi-agent Policy Optimization

To facilitate a comprehensive discussion on the mechanism of experience reuse, we will use MAPPO as the representative of parameter-sharing algorithms class, and HAPPO as the representative of parameter non-sharing algorithms class in the following theoretical framework and experimental setting. Our primary focus is on the optimization objective of MARL, which is expressed as:

$$J(\bar{\boldsymbol{\pi}}) = J(\boldsymbol{\pi}) + \mathbb{E}_{\boldsymbol{s} \sim \rho_{\boldsymbol{\pi}}, \boldsymbol{a} \sim \bar{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})\right]. \quad (1)$$

Here, $\bar{\boldsymbol{\pi}}$ is the candidate joint policy. Denote $\bar{\boldsymbol{\pi}}(\cdot|s) = \prod_{i=1}^n \bar{\pi}_i(\cdot|o_i)$, $J(\pi) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, we can extend the MDP theoretical setting to Dec-POMDP. For parameter-sharing methods in MARL, it can be seen that each agent updates its policy using the experience of all other agents.

Consider $a_i \sim \pi_{\zeta(i)}(\cdot|o_i)$ as the probability of sampling action a_i from policy $\pi_{\zeta(i)}$ given observation o_i, with ζ being a permutation of the indices such that $\cup_{i=1}^n \zeta(i) = \{1, 2, ..., n\}$. We define E_ζ to be the expected advantage when actions follow policies permuted by ζ:

$$E_\zeta = \mathbb{E}_{\boldsymbol{s} \sim \rho \boldsymbol{\pi}, a_1 \sim \pi_{\zeta(1)}(\cdot|o_1), \ldots, a_n \sim \pi_{\zeta(n)}(\cdot|o_n)}\left[A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})\right], \quad (2)$$

where ζ cycles through all policy index permutations. Consequently, we can expand formula (1) as follows:

$$J(\bar{\boldsymbol{\pi}}) = J(\boldsymbol{\pi}) + \frac{1}{\mathcal{A}_n^n} \sum_{\zeta} E_\zeta, \quad (3)$$

summing over all $\mathcal{A}_n^n$ permutations of ζ.

This equation characterizes the binding relationship between agents and experience, which can be regarded as the result of the combined action of $\mathcal{A}_n^n$ optimization objectives. Each agent uses the experience generated by another agent (possibly itself). It is imperative to maintain the uniqueness of each agent's experience within every optimization objective. The reuse binding relationship between the agents can be found in Figure 2.

For independent learning in MARL, where each agent updates the policy based on its own experience, Eq. (1) is equivalent to the following,

$$J(\bar{\boldsymbol{\pi}}) = J(\boldsymbol{\pi}) + \mathbb{E}_{\boldsymbol{s} \sim \rho_{\boldsymbol{\pi}}\ a_i \sim \bar{\pi}_i(\cdot|o_i), \forall i}\left[A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})\right], \quad (4)$$

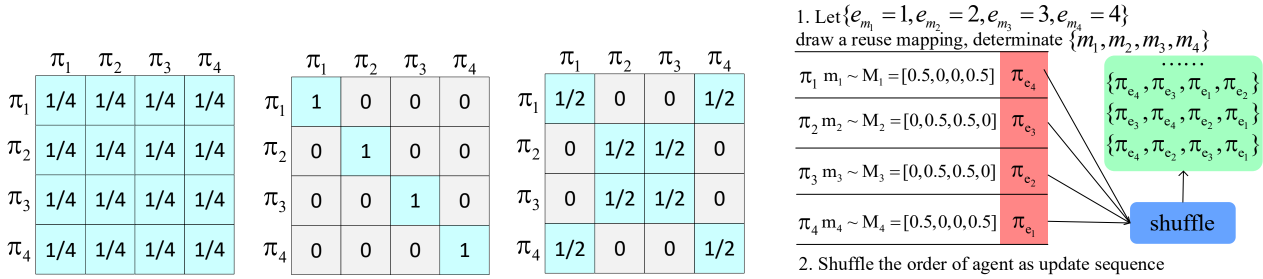
Eq.(3) and Eq.(4) correspond to the update mechanisms in MAPPO and HAPPO, respectively. In MAPPO, experience sharing may be harmful when agents have conflicting goals, despite identical action and observation spaces. This can lead to suboptimal performance. HAPPO, in contrast, ensures monotonic improvement through sequential updates, but this can result in lower sample efficiency and slower convergence when each agent learns solely from its own experiences.

In response to these issues, we propose the OAPO algorithm. OAPO slightly relaxes the monotonic improvement guarantee to boost sample efficiency and overall performance. The algorithm is designed to efficiently utilize the experiences of other agents, enhancing its applicability in both heterogeneous and homogeneous cooperative settings.

4.2 Approximate Monotonic Improvement for Trust-Region Method

For each agent indexed by j within a set of n agents, let $e_j \in 1, \ldots, n$. Here, e_j signifies that agent indexed by j reuses the experience of agent e_j. We define the vector $\boldsymbol{e} = [e_1, \ldots, e_n]$ to represent a specific mapping of experience reuse among the n agents. The distribution $\omega = \Delta_{\boldsymbol{e}}$ encompasses all these possible mappings. An illustration of selecting a mapping from ω is shown in Figure 2, aligning with the procedure in lines 5-6 of Algorithm 1.

As agents update their policies in a sequential manner, $\tilde{\boldsymbol{\pi}}^j$ denotes the resulting joint policy after the j-th update, which



(a) MAPPO (b) HAPPO (c) Ours (d) Sampling pipeline for ours.

Figure 2: A reuse matrix example for four agents with MAPPO, HAPPO, and our method shows joint policy $\boldsymbol{\pi} = \pi_1, \pi_2, \pi_3, \pi_4$ and matrix $\boldsymbol{M}$ for mapping reuses, where $\boldsymbol{M}_{ij}$ is the probability of agent i using agent j's experience. Figure (2d) displays sampling the update sequence and mapping, starting with a binding mapping $\pi_{e_4} = \pi_1, \pi_{e_3} = \pi_2, \pi_{e_2} = \pi_3, \pi_{e_1} = \pi_4$ from $\boldsymbol{M}$. The red indicates reuse relations, and agents are shuffled for an update order, then updated accordingly.

Algorithm 1 Off-agent Policy Iteration with Approximate Monotonic Improvement

Initialize the joint policy $\boldsymbol{\pi}_0$.
for each iteration $k = 0, 1, \ldots$ **do**
  Collect data with $\boldsymbol{\pi}_k$ in the environment.
  Compute the advantage function $A_{\boldsymbol{\pi}_k}(s, a)$, $\epsilon = \max_{s,\boldsymbol{a}} |A_{\boldsymbol{\pi}_k}(s, \boldsymbol{a})|$ and $C = \frac{4\gamma\epsilon}{(1-\gamma)^2}$.
  Draw a reuse sequence $\boldsymbol{e} = [e_1, \cdots, e_n]$ from ω.
  Shuffle an update sequence $\tilde{\boldsymbol{e}} = \textit{shuffle}(\boldsymbol{e})$ randomly.
  for e_m in $\tilde{\boldsymbol{e}}$ **do**
    Update the agent using

$$\pi_{k+1}^{e_m} = \arg\max_{\pi_{e_m}} \left[\mathcal{L}_{\boldsymbol{\pi}_k}^{e_{1:m}} \left(\tilde{\boldsymbol{\pi}}_{k+1}^{m-1}, \pi_{e_m}\right) - C\mathbf{D}_{\mathrm{KL}}^{\max}\left(\pi_k^m, \pi_{e_m}\right)\right]$$

  end for
end for

is constructed as $\tilde{\boldsymbol{\pi}}^j = \bar{\pi}_{e_1} \times \cdots \times \bar{\pi}_{e_j}$. We define the joint policy as $\boldsymbol{\pi} = \prod_{k=1}^{n} \pi_{e_k}$ with its associated joint advantage function $A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})$. During each update, $\bar{\pi}_{e_m}$ represents the candidate policy for the m-th agent, then we define,

$$\mathcal{L}_{\boldsymbol{\pi}}^{\boldsymbol{e}_{1:m}}(\tilde{\boldsymbol{\pi}}^{m-1}, \bar{\pi}_{e_m}) = \mathbb{E}_{\boldsymbol{s}\sim\rho_{\boldsymbol{\pi}}, a_{1:m-1}\sim\tilde{\boldsymbol{\pi}}^{m-1}, a_m\sim\bar{\pi}_{e_m}}\left[A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}_{1:m})\right].$$

Where $a_{1:m-1}$ represents the joint actions taken by the first $m-1$ agents, and the term $A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}_{1:m})$ can be broken down as specified by the Multi-Agent Advantage Decomposition (Appendix B). The function $\mathcal{L}$ serves as our optimization target within a multi-agent cooperative framework.

To ensure stable performance improvement, we impose constraints on our optimization objectives. The following lemma provides a lower bound on the potential increase in expected returns when switching from a current joint policy $\boldsymbol{\pi}$ to any new joint policy $\bar{\boldsymbol{\pi}}$:

Lemma 1. *Given $\boldsymbol{\pi}$ as a joint policy, for any joint policy $\bar{\boldsymbol{\pi}}$, we have*

$$J(\bar{\boldsymbol{\pi}}) - J(\boldsymbol{\pi}) \geq \mathop{\mathbb{E}}_{e\sim\omega} \sum_{m=1}^{n} \left[\mathcal{L}_{\pi}^{e_{1:m}}(\tilde{\boldsymbol{\pi}}^m) - C\mathbf{D}_{KL}^{\max}\left(\pi_{e_m}, \bar{\pi}_{e_m}\right)\right],$$

where $\tilde{\pi}^i = \bar{\pi}_{e_1} \times \cdots \times \bar{\pi}_{e_i}$ is defined as the policy update up to the i-th agent, ω is a distribution of reuse mapping satisfying Condition 1, and $J(\boldsymbol{\pi}) = \mathbb{E}_{\tau\sim\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$. $C = \frac{4\gamma\epsilon}{(1-\gamma)^2}$ and $\epsilon = \max_{s,\boldsymbol{a}} |A_{\boldsymbol{\pi}_k}(s, \boldsymbol{a})|$. The proof is shown in Appendix D.

As described in Algorithm 1, OAPO proceeds by collecting data, computing the advantage function, and update the policy of each agent sequentially based on a specific order and experience reuse mapping. To enable effective experience reuse, we introduce a constraint on the similarity between policies, which is formalized in the following:

Condition 1. *Let $\bar{\pi}_i$ be the candidate policy of agent i. For agent i to reuse the experience produced by agent j, the following condition must be satisfied:*

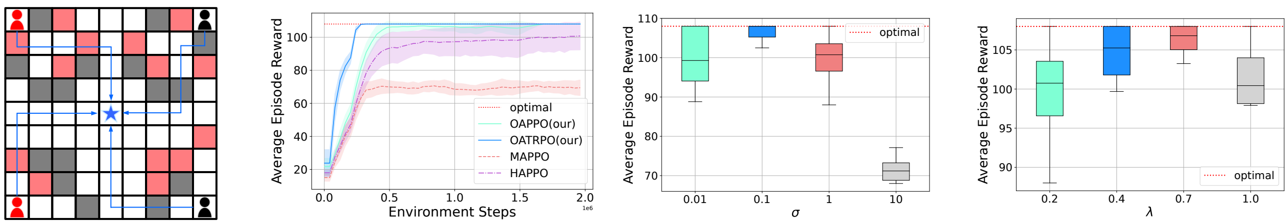
$$\bar{\pi}_i \in \mathfrak{B}_\sigma(\pi_j),\ i, j \in \mathcal{N}, \tag{5}$$

here, $\mathfrak{B}_\sigma(\pi_j)$ represents a sphere defined by the KL divergence, and σ is a small positive threshold. The candidate policy $\bar{\pi}_i$ must stay within this KL divergence sphere around π_j to utilize its experience. This condition enables us to construct an appropriate distribution for reuse mapping, and a complete explanation is available in Section 5.3.

Building on Lemma 1, we propose a theorem guaranteeing the approximate monotonic improvement of the OAPO algorithm. The theorem is stated as follows:

Theorem 1. *Let $U(x_0, \epsilon)$ denote the neighborhood of a point x_0 within distance ϵ, where ϵ is a positive number. For a sequence of joint policies $\boldsymbol{\pi}_{k=0}^{\infty}$ that satisfies Condition 1, after applying updates via Algorithm 1, we find $J(\boldsymbol{\pi}_{k+1}) - J(\boldsymbol{\pi}_k) \geq \max\left(U(0, \epsilon)\right)$ when $k > K$, $K \in \mathbb{N}^+$. This result demonstrates the approximate monotonic improvement of the policies, with a comprehensive proof provided in Appendix E.*

Remark. *Given the sequential updating approach, the agent only employs experiences that satisfy the reuse condition during the reuse phase. This methodology ensures a stable convergence of expected returns and provides a guarantee of approximate monotonicity, ensuring that our policy exhibits consistent iterative improvement.*



(a) Map for Black and Red (b) Results on Black and Red. (c) The effect of policy distance (σ). (d) The influence of reuse mapping distribution (λ).

Figure 3: (a) Black and Red maze game map: ★ symbolizes the target, the red agent, and the black agent. (b) Algorithmic performance in the maze game, each tested with 5 seeds—mean shown by solid line, standard deviation by shadow. Consistent seed number used in (c) and (d). Effects of distribution probability changes and policy distance on algorithm performance examined in (c) and (d).

5 Practical Algorithm Implementation

Direct computation of $\mathcal{L}$ is not practical. As such, we propose the following transformation:

Proposition 1. *Assume $\boldsymbol{\pi} = \prod_{k=1}^{n} \pi_{e^k}$ is a joint policy, $\boldsymbol{e}$ represents the binding relationship between experience reuse for all agents, $A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})$ denotes joint advantage function, and $\hat{\pi}_{e_m}$ is a candidate policy conditioned on the experience of agent m. Then,*

$$\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}^{\boldsymbol{e}_{1:m}}(\tilde{\boldsymbol{\pi}}^{m-1}, \hat{\pi}_{e_m}) \\
&= \mathbb{E}_{\boldsymbol{s}\sim\rho_{\boldsymbol{\pi}}, a_{1:m-1}\sim\tilde{\pi}^{m-1}, a_m\sim\hat{\pi}_{e_m}}\left[A_{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}_{1:m})\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\hat{\pi}_{e_m}}{\pi_m} - 1\right)\frac{\tilde{\boldsymbol{\pi}}^{m-1}}{\boldsymbol{\pi}_{1:m-1}} A_{\boldsymbol{\pi}}(s, \boldsymbol{a})\right].
\end{aligned} \tag{6}$$

Through this transformation, we convert an intractable optimization objective into one leveraging the existing joint advantage function, modifying it as needed with the updated joint policy factor. Balancing computational complexity and performance enhancement, we devise two MARL algorithms using the trust region method to substantiate our method. Each agent's policy π_i is parameterized by θ_i, and the collective policy $\boldsymbol{\pi}_\theta$ is denoted by $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$.

5.1 Off-Agent Trust Region Policy Optimization (OATRPO)

In line with TRPO, we replace the challenging maximum KL-divergence with the expected KL-divergence constraint $\mathbb{E}_{o\sim\eta_{\boldsymbol{\pi}_{e_m}}}\left[\mathbf{D}_{\text{KL}}\left(\pi_m(\cdot|o), \bar{\pi}_{e_m}(\cdot|o)\right)\right] \leq \delta$, where δ is a pre-set threshold, and Monte Carlo method is used for approximation.

Our proposed OATRPO algorithm updates the policy parameters $\theta_{e_m}^{k+1}$ for agent e_m at each iteration $k+1$, utilizing the experience from agent m. The refined optimization objective for OATRPO is:

$$\theta_{e_m}^{k+1} = \arg\max_{\tilde{\theta}} \mathbb{E}_{s,\boldsymbol{a}\sim p}\left[\left(\frac{\bar{\pi}_{\tilde{\theta}}^{e_m}}{\pi_{\theta^k}^{m}} - 1\right)\rho_{\theta^{k+1}} A_{\boldsymbol{\pi}_{\theta^k}}(s, \boldsymbol{a})\right],$$

$$\text{subject to } \mathbb{E}_{o\sim\eta_{\boldsymbol{\pi}_{\theta^k}^m}}\left[\mathbf{D}_{\text{KL}}\left(\pi_{\theta^k}^m(\cdot|o), \bar{\pi}_{\tilde{\theta}}^{e_m}(\cdot|o)\right)\right] \leq \delta. \tag{7}$$

Where $\rho_{\theta^{k+1}} = \frac{\tilde{\boldsymbol{\pi}}_{\theta^{k+1}}^{m-1}}{\boldsymbol{\pi}_{\theta^k}^{1:m-1}}$, $\tilde{\theta}$ denotes the currently optimized policy, and $\mathbb{E}_{s,\boldsymbol{a}\sim p}[\cdot]$ is $\mathbb{E}_{\boldsymbol{s}\sim\rho_{\boldsymbol{\pi}_{\theta^k}}, a_{1:m-1}\sim\tilde{\pi}_{\theta^{k+1}}^{m-1}, a_m\sim\pi_{\tilde{\theta}}^{e_m}}[\cdot]$, the detailed of OATRPO show in Appendix G.

5.2 Off-Agent Proximal Policy Optimization (OAPPO)

To avoid the significant computational cost associated with the second-order gradient, we adopt the same method used in PPO. Here, we employ a clipping operation to transform the optimization objective (7) into an unconstrained first-order optimization problem. The clipped objective for updating the parameter of agent e_m is given by:

$$\mathbb{E}_{s,\boldsymbol{a}\sim p}\left[\min\left(\rho_{\tilde{\theta}}, \text{clip}\left(\rho_{\tilde{\theta}}, 1\pm\epsilon\right)\right)\rho_{\theta^{k+1}}^{e_{1:m-1}} A_{\boldsymbol{\pi}_{\theta^k}}(s, \boldsymbol{a})\right], \tag{8}$$

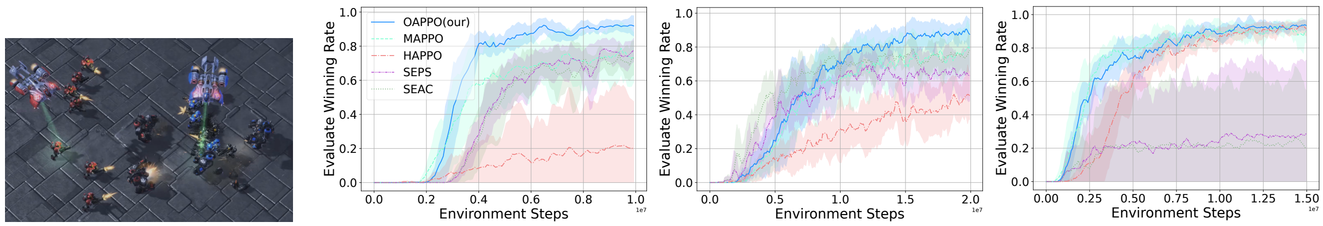
where, $\rho_{\tilde{\theta}} = \frac{\bar{\pi}_{\tilde{\theta}}^{e_m}}{\pi_{\theta^k}^{m}}$, and the remaining notation is consistent with optimization objective (7), the detailed of OAPPO show in Appendix H.

5.3 Reuse Mapping Distribution

The choice of reuse mappings influences the optimization trajectory. We represent the frequency of reuse mappings with a distribution $\boldsymbol{e} \sim \omega$. Mappings that violate the reuse condition $\text{D}_{\text{KL}}^{\max}(\pi_j, \bar{\pi}_i) > \sigma$ can impede the approximate monotonic improvements and cause update inconsistencies when sharing experiences, thus they are assigned a zero probability. For mappings meeting the reuse condition, we define distribution ω to allocate occurrence probabilities:

$$p(e_m) = \begin{cases} \lambda, & \text{D}_{\text{KL}}^{\max}(\pi_m, \bar{\pi}_{e_m}) = 0 \\ \frac{1-\lambda}{\gamma(m)-1}, & 0 < \text{D}_{\text{KL}}^{\max}(\pi_m, \bar{\pi}_{e_m}) \leq \sigma \\ 0, & \text{D}_{\text{KL}}^{\max}(\pi_m, \bar{\pi}_{e_m}) > \sigma \end{cases}. \tag{9}$$

Where $\gamma(m)$ denotes the count of agents whose experience agent m can draw on. The hyperparameter λ balances the experience exchange between agent m and others, while σ sets the policy distance threshold for experience reuse. The probability distribution $p(e_m)$ must integrate into one over all experiences e_m. As outlined in Algorithm 1, each policy iteration employs a single reuse mapping. Using all possible mappings exponentially increases the computation and potential invalid updates for an agent. To improve sampling efficiency and lower complexity, we translate reuse mapping probabilities into weights, indicating how often an agent's experience



(a) SMAC (b) 6h vs 8z (c) 3s5z vs 3s6z (d) MMM2

Figure 4: (a) The *MMM2* scenario of SMAC, the red team uses our algorithm and the blue team uses the built-in algorithm. (b)-(d) Winning rates compared on SMAC's **super hard** map, each algorithm run 5 seeds; solid lines is mean and shaded areas show standard deviation.

is utilized. The complete reuse batch update procedure is in Appendix G and H.

6 Experiments

We designed two maze-based games to assess the efficacy of various reuse strategies in mixed environments. Subsequently, we concentrated on two prevalent environments: the StarCraft II Multi-Agent Challenge (SMAC) and Google Research Football (GRF). Within these settings, we evaluated the performance of our algorithms, both with and without an off-agent. To ensure a thorough evaluation of our algorithm's capabilities, we also conducted comparisons with several established baseline algorithms, including: MAPPO, which uses parameter sharing; HAPPO, which relies exclusively on individual experiences for updates; SEPS, which adopts selective parameter sharing; and SEAC, which incorporates experiences from other agents for updates.

6.1 Maze Game

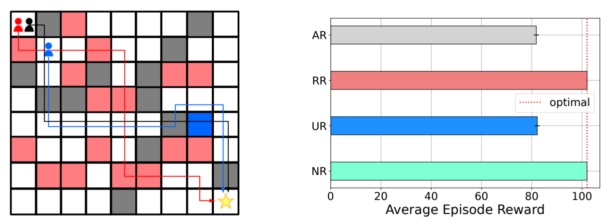
Black and Red Game

Figure 3a illustrates the **Black and Red** game scenario: red agents can move through red grids and open spaces; black agents through black grids and open spaces. Both strive to reach the terminal (★) via possible optimal routes (blue lines). The global state comprises a 9×9 map matrix and agent locations, with each agent's 3×3 view matrix and location forming its observation. Agents can move: *up, down, left, right,* or *nop*. Experiment details are in Appendix I.

Figure 3b reveals that in heterogeneous tasks, the shared parameters of MAPPO often produce suboptimal results. HAPPO performs better, but requires more samples and converges slowly. Homogeneous scenarios (e.g. Ma-Mujoco) show that direct experience reuse can degrade performance (Appendix A). Our methods surpass MAPPO and HAPPO, avoiding MAPPO's local optima and converging faster than HAPPO through inter-agent reuse. In TRPO's tighter constraints, OATRPO converges quicker than OAPPO.

Modifying the parameter σ significantly affects the scope of experience that agents can reuse, as depicted in Figure 3c. With a small σ ($\sigma = 0.01$), agents are constrained to their own experiences, causing policy updates and convergence to be slower. A larger σ ($\sigma = 0.1$) incorporates more relevant experiences, enhancing the convergence rate. However, as σ expands further ($\sigma = 1$), the inclusion of irrelevant experiences confuses the direction of gradient update. An overly large σ ($\sigma = 10$) causes the agent to integrate experiences from all peers, which may hinder performance due to the assimilation of divergent policies.

When assessing the effect of reuse mapping distributions on performance through adjustment of λ, as illustrated in Figure 3d, we find that a smaller λ ($\lambda = 0.2$) promotes agents to prioritize external experiences, potentially leading to local optima due to policy fluctuations in initial training. A larger λ ($\lambda = 1$) restricts agents to their own experiences, which is less effective and delays convergence compared to intermediate values of λ ($\lambda = 0.4/0.7$), which strike a balance in experience reusing. Among these, $\lambda = 0.7$ demonstrates marginally improved performance relative to $\lambda = 0.4$. Optimal selection of λ is thus critical for advancing agent performance and accelerating convergence.



(a) Map for Black Blue Red (b) Results on Black Blue Red

Figure 5: (a) Black Blue Red maze map: blue agent blocked by blue grids, star as target. Red, black, and blue lines show optimal routes. (b) Reuse scheme comparison in Black Blue Red maze with 5 seeds; thin black lines on bars represent standard variance.

Black Blue Red Game

To assess how different experience reuse methods affect performance, we created another maze game with three humanoid icons representing red, black, and blue agents. A blue trap grid, accessible to only black and red agents, helps demonstrate reuse effects among similar agents. The red agent cannot enter black grids, the black agent cannot enter red grids, and the blue agent cannot enter both. Figure 5a shows the game map and agent roles, revealing numerous decision conflicts between the red and black agents, while black and blue agents have fewer differences.

We tested four reuse schemes to examine inter-agent reuse effects. No Reuse (NR) involves agents updating based solely on their experiences (HAPPO). Unreasonable Reuse

(a) Google research football

Algorithms	pass and shoot	3_vs_1	CA-Easy	CA-Hard
MAPPO	**93.8**(5.3)	88.7(2.5)	59.7(12.6)	58.5(16.6)
HAPPO	89.8(4.0)	**93.7**(5.0)	25.8(12.4)	13.5(10.3)
SEAC	**91.4**(2.1)	**94.9**(3.7)	52.1(20.2)	36.4(5.3)
SEPS	**95.3**(3.4)	**96.8**(2.6)	76.3(9.1)	36.2(9.2)
OAPPO(our)	**94.4**(2.1)	**95.7**(3.1)	**88.1**(6.6)	62.5(2.5)
OATRPO(our)	**93.3**(3.3)	**94.5**(3.3)	**94.9**(5.6)	**89.1**(4.8)
Steps	5e6	5e6	1e7	1.5e7

(b) Results on GRF cooperative multi-agent scenarios.

Figure 6: (a)The *3_vs_1* scenario of GRF. (b)Performance of the evaluation score in cooperative scenarios of GRF. Each method was run with 5 different seeds, first number denote mean value and number in bracket represent standard deviation.

(UR) has black and red agents sharing experiences. Reasonable Reuse (RR) involves sharing between black and blue agents. All Reuse (AR) indicates all agents share experiences (MAPPO). We modified OAPPO parameters to implement these strategies, as shown in Figure 5b. Results indicate that experience reuse is suboptimal when agents' policies differ significantly, potentially causing a local optimum trap. In UR and AR, the black agent's experience sharing with the red, leading them to become stuck in a local optimum. In RR, the blue agent's slightly different optimal policy, with the help of TRL, they can still reach an optimal solution as in NR.

6.2 Results on SMAC

SMAC [Samvelyan *et al.*, 2019] is a widely used multi-agent cooperative environment where teams of same or different agents work together to defeat he opposing team. HAPPO excels on Hard and some Super Hard maps but struggles on others due to its reliance on individual training samples, limiting its ability to achieve the performance of MAPPO with the same number of samples, particularly when the agent count rises. Thus, we focus on the **Super Hard** maps and PPO-based algorithms to illustrate performance differences. The global state includes all map cells, agent coordinates relative to the map center, and unit features in view. SMAC's local observation encompasses a circular area around each unit and visible surviving units. Experiment details are in Appendix K.

In scenarios with heterogeneous agents (*3s5z vs 3s6z* and *MMM2*, referenced in Figures 4c and 4d), OAPPO outperforms MAPPO and HAPPO. Unlike MAPPO, OAPPO maintains parameter sharing aligned with Proposition 1, and its experience reuse mechanism enhances convergence. Furthermore, OAPPO can construct a suitable distribution for the agent by adjusting both the distribution and the policy thresholds. This method consequently achieves better results in contrast to the conventional approach of directly selecting the experience reuse methods of SEPS and SEAC. In the homogeneous agent setup *6h vs 8z* (Figure 4b), OAPPO leverages reuse mapping distribution to balance the degree of experience reuse of other agents, achieves more stable performance enhancements without sacrificing experience reuse, This stability results in superior outcomes compared to MAPPO. Overall, OAPPO surpasses HAPPO in convergence speed and performance across all SMAC tasks at the same epochs.

6.3 Results on GRF

We evaluate OAPPO and OATRPO across several GRF [Kurach *et al.*, 2020] academy scenarios, including `academy_pass_and_shoot_with_keeper` (pass and shoot), `academy_3_vs_1_with_keeper` (3_vs_1), `academy_counterattack_easy` (CA-Easy), and `academy_counterattack_hard` (CA-Hard). In these scenarios, a team of agents competes against an opponent team controlled by built-in algorithm to score. While all agents have the same action space, position-based role differences exist. The global state refers to the complete set of data returned by the environment after actions are performed. Local observations include player coordinates, ball possession and direction, active player, or game mode. Configuration of the experiment show in Appendix J.

Table 6b presents the results. For simple tasks (pass and shoot, 3_vs_1), agents achieve nearly perfect scores solely through their individual experiences, without sharing. However, as tasks become complex (CA-Easy), the reliance on individual experience is inadequate to obtain high performance. In these scenarios, selective reuse of the experiences of other agents yields satisfactory results, clearly demonstrating the superiority of OAPPO and OATRPO. When tasks gradually increase in complexity and difficulty (CA-Hard), the effective reuse of experiences continues to exhibit a faster convergence speed. This maintains the superior performance of OAPPO and OATRPO over other baseline algorithms.

7 Conclusion

To tackle low sample efficiency in MARL, we developed a method enhancing inter-agent experience reuse, and designed Maze Game to investigate conditions requisite for such reuse. Our approach yielded two innovative algorithms, OATRPO and OAPPO, based on inter-agent experience sharing with guarantees of approximate monotonic improvement. Experimental results on SMAC and GRF show our algorithms allow selective experience reuse by agents, leading to superior sample efficiency and improved performance in these tasks. Nevertheless, the efficacy of our experience reuse is dependent upon the hyperparameter λ and the distribution threshold σ, suggesting future work to adaptively fine-tune these parameters with respect to agent roles and experience quality for enhanced performance.

Contribution Statement

Ruiqing Chen and Xiaoyuan Zhang contribute equally to this work, work done when Ruiqing Chen visited Peking University. The corresponding author is Yaodong Yang.

References

[Adler and Blue, 2002] Jeffrey L Adler and Victor J Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5-6):433–454, 2002.

[Baker *et al.*, 2020] Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *ICLR*, 2020.

[Brown and Sandholm, 2019] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[Busoniu *et al.*, 2008] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(2):156–172, 2008.

[Christianos *et al.*, 2020] Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10707–10717, 2020.

[Christianos *et al.*, 2021] Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, pages 1989–1998. PMLR, 2021.

[de Witt *et al.*, 2020] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[Fakoor *et al.*, 2020] Rasool Fakoor, Pratik Chaudhari, and Alexander J Smola. P3o: Policy-on policy-off policy optimization. In *Uncertainty in Artificial Intelligence*, pages 1017–1027. PMLR, 2020.

[Foerster *et al.*, 2018] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Glavic *et al.*, 2017] Mevludin Glavic, Raphaël Fonteneau, and Damien Ernst. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, 50(1):6918–6927, 2017.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[Hussein *et al.*, 2017] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[Jarrahi, 2018] Mohammad Hossein Jarrahi. Artificial intelligence and the future of work: Human-ai symbiosis in organizational decision making. *Business horizons*, 61(4):577–586, 2018.

[Kuba *et al.*, 2021] Jakub Grudzien Kuba, Ruiqing Chen, Munning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021.

[Kurach *et al.*, 2020] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4510, 2020.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Lowe *et al.*, 2017] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[Oh *et al.*, 2018] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018.

[Oliehoek and Amato, 2016] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[Osa *et al.*, 2018] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[Peng *et al.*, 2017] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[Qiu *et al.*, 2023] Dawei Qiu, Yi Wang, Jianhong Wang, Ning Zhang, Goran Strbac, and Chongqing Kang. Resilience-oriented coordination of networked microgrids: a shapley q-value learning approach. *IEEE Transactions on Power Systems*, pages 1–15, 2023.

[Queeney *et al.*, 2021] James Queeney, Yannis Paschalidis, and Christos G Cassandras. Generalized proximal policy optimization with sample reuse. *Advances in Neural Information Processing Systems*, 34:11909–11919, 2021.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[Sun *et al.*, 2023] Mingfei Sun, Sam Devlin, Jacob Beck, Katja Hofmann, and Shimon Whiteson. Trust region bounds for decentralized ppo under non-stationarity. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 5–13, 2023.

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[Wang *et al.*, 2016] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[Wang *et al.*, 2023] Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. *arXiv preprint arXiv:2302.06205*, 2023.

[Wu *et al.*, 2021] Zifan Wu, Chao Yu, Deheng Ye, Junge Zhang, Hankz Hankui Zhuo, et al. Coordinated proximal policy optimization. *Advances in Neural Information Processing Systems*, 34:26437–26448, 2021.

[Yang and Wang, 2020] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

[Yu *et al.*, 2021] C Yu, A Velu, E Yinitsky, et al. The surprising effectiveness of mappo in cooperative, multi-agent games [j/ol]. 2021.

[Zare *et al.*, 2023] Maryam Zare, Parham M Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *arXiv preprint arXiv:2309.02473*, 2023.

[Zhang and Lesser, 2011] Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[Zhou *et al.*, 2019] Ming Zhou, Yong Chen, Ying Wen, Yaodong Yang, Yufeng Su, Weinan Zhang, Dell Zhang, and Jun Wang. Factorized q-learning for large-scale multi-agent systems. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*, pages 1–7, 2019.

[Zhou *et al.*, 2021] Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Weinan Zhang, and Jun Wang. Malib: A parallel framework for population-based multi-agent reinforcement learning. *arXiv preprint arXiv:2106.07551*, 2021.