Taming Multi-Agent Reinforcement Learning with Estimator Variance Reduction

Taher Jafferjee*
Huawei R&D
London, United Kingdom
taherjafferjee@gmail.com

Juliusz Ziomek
Oxford University
Oxford, United Kingdom
juliusz@robots.ox.ac.uk

Tianpei Yang
Nanjing University
Suzhou, China
tianpei.yang@nju.edu.cn

Zipeng Dai
Beijing Institute of Technology
Beijing, China
daizipeng@bit.edu.cn

Jianhong Wang
Bristol University
Bristol, United Kingdom
jianhong.wang@bristol.ac.uk

Matthew E. Taylor
University of Alberta
Edmonton, Canada
matthew.e.taylor@ualberta.ca

Kun Shao
Huawei R&D
London, United Kingdom
shaokun2@huawei.com

Jun Wang
University College London
London, United Kingdom
j.wang@ucl.ac.uk

David Mguni†
Queen Mary University London
London, United Kingdom
d.mguni@qmul.ac.uk

ABSTRACT

Multi-agent reinforcement learning (MARL) enables systems of autonomous agents to solve complex tasks from jointly gathered experiences of the environment. Many MARL algorithms perform centralized training (CT), often in a simulated environment, where at each time-step the critic makes use of a single sample of the agents' joint-action for training. Yet, as agents update their policies during training, these single samples may poorly represent the agents' joint-policy leading to high variance gradient estimates that hinder learning. In this paper, we examine the effect on MARL estimators of allowing the number of joint-action samples taken at each time-step to be greater than 1 in training. Our theoretical analysis shows that even modestly increasing the number of joint-action samples shown to the critic leads to TD updates that closely approximate the true expected value under the current joint-policy. In particular, we prove this reduces variance in value estimates similar to that of decentralized training while maintaining the learning benefits of CT. We describe how such a protocol can be seamlessly realized by sharing policy parameters between the agents during training and apply the technique to induce lower variance in estimates in MARL methods within a general apparatus which we call Performance Enhancing Reinforcement Learning Apparatus (PERLA). Lastly, we demonstrate PERLA's performance improvements and estimator variance reduction capabilities in a range of environments including *Multi-agent Mujoco*, and *StarCraft II*.

KEYWORDS

Multi-agent Reinforcement Learning; Centralised Training-Decentralised Execution; Variance Reduction

ACM Reference Format:
Taher Jafferjee, Juliusz Ziomek, Tianpei Yang, Zipeng Dai, Jianhong Wang, Matthew E. Taylor, Kun Shao, Jun Wang, and David Mguni. 2025. Taming Multi-Agent Reinforcement Learning with Estimator Variance Reduction. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 Introduction

Multi-agent reinforcement learning (MARL) has emerged to be a powerful tool to enable autonomous agents to jointly tackle difficult tasks such as complex games [7, 20], ride-sharing [28], and swarm robotics [11, 14]. Nevertheless, a key impediment to these algorithms is the high variance of the critic and policy gradient estimators. Reducing the variance of these estimators is critical since high variance estimators can lead to low sample efficiency and poor overall performance [6]. In multi-agent systems, the environment reward function and state transition dynamics function depend on the *joint-action* of all the agents in the system. As a result, for an agent in a given state, different executions of a particular action may return varying outcomes depending on the joint-actions of other agents. As agents' estimates are based on previous observations of joint actions, updates to the policies of other agents during training may result in returns that significantly deviate from current estimates. Consequently, key estimators for efficient learning can have high variance severely impairing the learning process and hence, the agents' abilities to jointly maximise performance.

Centralized Training-Decentralized Execution (CT-DE) paradigm is a popular MARL training framework in which agent's observe the joint behaviour of other agents during training. Therefore, the CT-DE paradigm has at its core a (possibly shared) critic for each agent that makes use of all available information generated by the system, including the global state and the joint action [17]. This added information can be exploited by the critic during training to

*Work was conducted while at Huawei R&D.
†Corresponding author.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

promote greater levels of coordination between the agents, which is often required to efficiently learn the optimal joint policies. Moreover, this added information can serve to reduce systemic variance. CT-DE has been shown to be highly effective in promoting high performance outcomes and thus serves as the foundation of many popular MARL methods such as MAPPO, Q-DPP [25], QMIX [19], SPOT-AC [12], and COMA [4].

In spite of these benefits, the CT-DE framework can be plagued by high variance updates during training. Central to the learning protocol of CT-DE algorithms are agent policy updates that are based on a single sample of the joint-action executed from other agents' policies at a given state. This can produce value function (VF) updates based on improbable events and result in inaccurate estimates of expected returns. This, in turn often leads to high variance VF estimates and poor sample efficiency. This is exemplified in a simple Coordination Game with the reward structure shown in Figure 1. In this game, miscoordinated actions (i.e., (l, r) or (r, l)) are penalised, and there is a sub-optimal stable (Nash equilibrium (NE)) joint strategy (r, r), and the optimal joint strategy is (l, l). In this setting, random occurrences of (l, l) are relatively improbable which can induce convergence to the joint strategy (r, r). To illustrate this, suppose the action (r, r) is sampled. A TD update towards this sample (with reward 0.5) may cause each agent to increase the policy probability of sampling r (and divert the agents to converge to the sub-optimal NE). On the other hand, if the joint-action sampled is (r, l), due to the reward of -1, the agent may reduce the policy probability of sampling r. Thus, an update following a single sample of this joint-action leads to an increase in probability weight on r, while the other decreases it producing the possibility of highly variant updates.

To investigate how to mitigate this issue, we study the problem of allowing MARL methods to make use of multiple samples of the joint action at each time step during training. A key insight is that by enabling the agents to share parameters of their individual policies, and then increasing the number of joint action samples taken during training, the variance of MARL critic estimators is significantly reduced while more accurately representing the true expected returns. Critically, in the paper we explore how this induces a much faster training process than using only a single sample of the joint action in the critic function. Additionally, we explain that this procedure dramatically reduces the variance of the critic since the critic estimate now closely approximates the *expected value* under the current joint policy of the agents. Consequently, VF updates are robust against improbable actions observed in single samples. Often, MARL training occurs in a simulated environment whereafter execution takes place in the environment. This allows for relatively inexpensive sampling during training. Nevertheless, additional joint-action sampling does incur greater computational expense, therefore a key aspect of our analysis is to understand the extent of the benefits as we vary the number of joint-action samples taken during training. We prove that employing this technique induces a vast reduction of the variance of VF estimates (Theorem 1) and that it preserves policy gradient estimators (Theorem 3) ensuring the consistency of its solution with the system objective. We also prove that applying this technique to actor-critic algorithms converges almost surely to a locally optimal joint policy profile (Theorem 6).

To summarise our theoretical results, in this paper, we show the following:

- The variance of Q-function constructed using the joint-action sampling technique is smaller than that of the Q-function and provides an exact characterisation of the difference (Theorem 1).
- We introduce a new gradient estimator constructed using the joint-action sampling technique and show that it is an unbiased estimate of the policy gradient (Theorem 3) but has a significantly lower variance than the standard CT-DE estimator.
- We provide an exact quantification of the difference in variances between the decentralized and CT-DE estimators admits (Theorem 5).
- The method preserves convergence to local optima almost surely (Theorem 6).
- We empirically demonstrate how this procedure can be applied to multi-agent policy gradient algorithms, which are known to suffer from high variance.

We instantiate these ideas in a general technique which we call Performance Enhancing Reinforcement Learning Apparatus (PERLA), that is adaptable to any MARL method. The benefits of PERLA can be readily observed in the Coordination Game in Fig. 1 where PERLA is applied to MAPPO [26], a leading MARL algorithm. The line chart in Fig. 1 shows the probability of sampling action 'Left' averaged across both agents over 10 runs against training steps in the Coordination game. As PERLA MAPPO consistently induces lower variance through training, the policy monotonically increases the probability of playing 'Left.' As an example, whenever Agent 1 samples the action r, the VF updates calculate the expected value under the policy of Agent 2. The update is towards the expected value due to $(r, \pi_2(l))$ and $(r, \pi_2(r))$ where $\pi_2(l)$ and $\pi_2(r)$ represent the probability of actions l and r respectively by Agent 2 under its policy. In this way, the return of Agent 1 of taking r is computed more accurately, and the VF update lower variance. This enables PERLA to produce consistent convergence of the underlying MARL method MAPPO to the optimal strategy, despite its less likely occurrence (relative to the miscoordinated joint actions) under stochastic policies. On the other hand, since vanilla MAPPO is exposed to random occurrences of miscoordination, it often converges to the sub-optimal stable point due to the penalties of miscoordination. In this example, vanilla MAPPO converges to the optimal NE in 6 of 10 runs while PERLA MAPPO converges to the optimal NE in 10 out of 10 runs.

The PERLA technique imputes two key advantages: 1) PERLA improves sample efficiency and convergence properties by factoring the behaviour of other agents when querying the critic and reducing variance. This enables efficient training while imposing no restrictive VF constraints (validated empirically in Section 6.2). 2) The PERLA technique is easily incorporated into CT-DE based Actor-Critic algorithms, and significantly boosts performance over the base learners (validated empirically on MAPPO [26] in Section 6.1).

Various actor-critic methods have shown significant performance improvements over previous MARL algorithms [10, 13, 26], and are state-of-the-art in a range of MARL benchmarks. These actor-critic

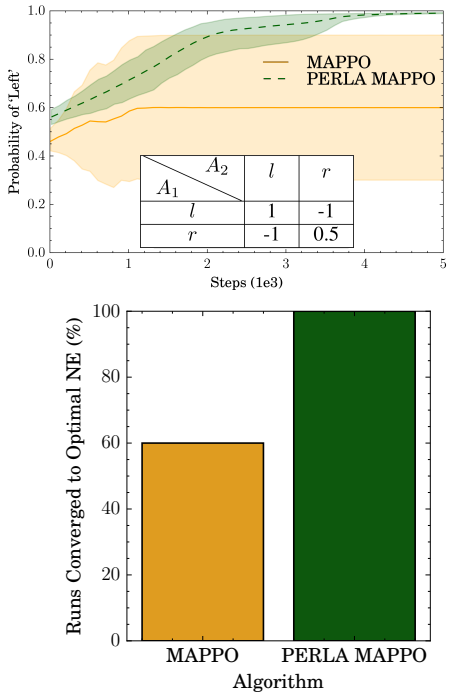

Figure 1: *Top inset table*: Payoff matrix of Coordination Game. In this game the actions ((l,r), $(r,1)$) are penalised. There is also a sub-optimal Nash Equilibrium, (r,r). The joint strategy that maximizes the agents' payoff (l,l). *Top*: policy probability of 'Left.' PERLA MAPPO learns the optimal policy and features less variance. *Bottom*: percentage of runs converged to optimal solution.

formalisms are natural candidates for PERLA as we can permit the critic to sample the joint-policy, while leaving the actor (i.e., the policy) unchanged. Thus, PERLA maintains the setup of decentralized execution (and centralised training).

2 Related Work

CT-DE MARL algorithms can be placed within two categories: *value-based* or *actor-critic* methods. In value-based methods, centralised training is assured to generate policies that are consistent with the desired system goal whenever the IGM principle [23] is satisfied.[1] To realise the IGM principle in CT-DE, QMIX, and VDN propose two sufficient conditions of IGM to factorise the joint action-value function. Such decompositions are limited by the joint action-value function class they can represent and can perform badly in systems that do not adhere to these conditions [24]. [2] Other value-based methods such as QPLEX [24] have been shown to fail in simple tasks with non-monotonic VFs [18] or in the case of QTRAN [23], scale poorly in complex MARL tasks such as the StarCraft Multi-Agent Challenge (SMAC) [16]. On the other hand, actor-critic type methods represent some of the highest performing methods such as MAPPO [26] and are among state-of-the-art. Indeed, recent work by Fu et al. [5] has shown that in particular MARL reward structures, actor-critic based CT methods are dominant as the class of algorithms that produce optimal policies. Further, empirical studies [2, 15] have shown the strength of actor-critic based CT methods over competing approaches.[4] proposed counter-factual baselines as a method to mitigate variance in MARL actor-critic methods. Their method seeks to accurately assign credit to agents for their contribution to the reward received following execution of a joint-action. [10] propose a general baseline (for the critic) applicable to all MARL actor-critic methods to mitigate variance. Moreover, the authors show that MARL variance may be dis-aggregated into the variance due to the state, the agent's own actions, and the actions of other agents. Unlike their method, which is limited to mitigating the variance from the agent's own action, we take a step further to mitigate variance due to other agents in the system which is a key impediment for MARL methods, especially with larger numbers of agents.

[1] IGM imposes an equivalence between the joint greedy action and the collection of individual greedy actions.

[2] WQMIX [18] considers a weighted projection towards better performing joint actions but does not guarantee IGM consistency.

3 Problem Setting

We formulate the MARL problem as a Markov game (MG) [22] represented by a tuple $\mathfrak{G} = \langle \mathcal{N}, \mathcal{S}, (\mathcal{A}_i)_{i\in\mathcal{N}} \coloneqq \mathcal{A}, P, R_i, \gamma \rangle$. $\mathcal{N} \in \mathbb{N}$ is the number of agents in the system, $\mathcal{S}$ is a finite set of states, $\mathcal{A}_i$ is an action set for agent $i \in \mathcal{N}$, and $R_i : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(D)$ is the reward function that agent i seeks to maximise (D is a compact subset of $\mathbb{R}$), and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the probability function describing the system dynamics We consider the fully observable setting, in which the agent observes system state $s^t \in \mathcal{S}$. To decide its actions, each agent $i \in \mathcal{N}$ samples its actions from a *Markov policy* $\pi_{i,\theta_i} : \mathcal{S} \times \mathcal{A}_i \to [0,1]$, which is parameterised by the vector $\boldsymbol{\theta}_i \in \mathbb{R}^d$. Throughout the paper, π_{i,θ_i} is abbreviated as π_i. At each time $t \in 0, 1, \ldots,$ the system is in state $s^t \in \mathcal{S}$ and each agent $i \in \mathcal{N}$ takes an action $a_i^t \in \mathcal{A}_i$, which together with the actions of other agents $\boldsymbol{a}_{-i}^t \coloneqq (a_1^t, \ldots, a_{i-1}^t, a_{i+1}^t, \ldots, a_N^t)$, produces an immediate reward $r_i \sim R(s^t, \boldsymbol{a}_i^t)$ for agent $i \in \mathcal{N}$. The system then transitions to a next state $s^{t+1} \in \mathcal{S}$ with probability $P(s^{t+1}|s^t, \boldsymbol{a}^t)$ where $\boldsymbol{a}^t = (a_1^t, \ldots, a_N^t) \in \mathcal{A}$ is the *joint action* which is sampled from the *joint policy* $\boldsymbol{\pi} \coloneqq \prod_{i=1}^N \pi_i$. The goal of each agent i is to maximise its expected returns measured by its VF $v_i(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s^t, \boldsymbol{a}^t) | s^0 = s\right]$ and the action-value function for each agent $i \in \mathcal{N}$ is given by $Q_i(s, \boldsymbol{a}) = \mathbb{E}[\sum_{t=0}^{\infty} R_i(s^t, \boldsymbol{a}^t) | \boldsymbol{a}^0 = \boldsymbol{a}]$, where $-i$ denotes the tuple of agents excluding agent i. Likewise, we denote $\prod_{j=1, j\neq i}^N \pi_j$ as $\boldsymbol{\pi}_{-i}$. In the fully cooperative case all agents share the same goal: $R_1 = \ldots R_N \coloneqq R$.

4 Theoretical Analysis

In the CT paradigm, given a state $s \in \mathcal{S}$ and the joint action $\boldsymbol{a} \in \mathcal{A}$, each agent $i \in \mathcal{N}$ computes its action-value function $Q_i(s, \boldsymbol{a})$. The action-value function provides an estimate of the agent's expected return using its policy given the behaviour of all other agents $\mathcal{N}/\{i\}$ for a given action $a_i \in \mathcal{A}_i$. Therefore, $Q_i(s, \boldsymbol{a})$ seeks to provide an estimate of the agent's own action, accounting for the actions of others. Agents use stochastic policies to explore, and therefore the

aggregated joint action $\boldsymbol{a} \sim \boldsymbol{\pi}$ may be composed of exploratory actions sampled from individual agent's policies.

In this section, we examine the effect of introducing a sampling process of the joint-policy $\boldsymbol{\pi}_{-i}$ for each agent i. This is used to compute the expected value of a_i under joint-policy of the other agents in the system. We compute the expected value of agent i's action-value function $\tilde{Q}$, as defined below:

$$\tilde{Q}_i(s, a_i) \coloneqq \mathbb{E}_{\boldsymbol{\pi}_{-i}}\left[Q_i(s, \boldsymbol{a})\right]; \quad \boldsymbol{a} \equiv (a_i, \boldsymbol{a}_{-i}) \in \boldsymbol{\mathcal{A}}, \tag{1}$$

where $s \in \mathcal{S}, a_i \sim \pi_i(\cdot|s), \boldsymbol{a}_{-i} \sim \boldsymbol{\pi}_{-i}(\cdot|s)$. This object requires some explanation; as with Q_i, the function $\tilde{Q}_i$ seeks to estimate the expected return following agent i taking action a_i. However, unlike Q_i, $\tilde{Q}_i$ builds in the expected value under the actions of other agents, $\boldsymbol{a}_{-i}$. Consequently, the critic can more accurately estimate the value of action a_i given the behaviour of the other agents in the system. In practice it may be impossible to analytically calculate (1), hence to approximate $\tilde{Q}_i(s, a_i)$, for any $\forall s \in \mathcal{S}$ and any $a_i \in \mathcal{A}_i$ we construct:

$$\hat{Q}_i(s, a_i) = \frac{1}{k}\sum_{j=1}^{k} Q_i(s, a_i, \boldsymbol{a}_i^{(j)}); \;\; \boldsymbol{a}_{-i}^{(j)} \sim \pi(\boldsymbol{a}_{-i}|s). \quad k \in 1, 2, \ldots, \tag{2}$$

We now perform a detailed theoretical analysis of increased per-state joint action sampling during training. We prove that doing so vastly reduces the variance of the key estimates used in training. We begin with a result that quantifies the reduction of variance when using $\hat{Q}_i$ instead of Q_i. We defer all proofs to the Appendix.

We first prove the folliowing result which is required to prove the Theorem 1.

Lemma 1. *Given N random variables $(x_i)_{i=1}^N$, where $x_i : \Omega \to \mathcal{X}_i$ and a measurable function $f : \times_{i=1}^N \mathcal{X}_i \to \mathbb{R}$, define by $\tilde{f}(x_1, \ldots, x_M) \coloneqq \mathbb{E}[f(x_1, \ldots, x_N)|x_1, \ldots, x_M]$ for some $M \leq N$ then*

$$Var(f(x_1, \ldots, x_N)) \geq Var(\tilde{f}(x_1, \ldots, x_M)). \tag{3}$$

Moreover, for a k-sample Monte-Carlo estimator of $\tilde{f}$ given by:

$$\hat{f}(x_1, \ldots, x_M) = \frac{1}{k}\sum_{i=1}^{k} f(x_1, \ldots, x_M, x_{M+1}^{(i)}, \ldots, x_N^{(i)}),$$

where $x_j^{(i)}$ is the i^{th} sample of x_j, we have:

$$Var(\hat{f}(x_1, \ldots, x_M)) = \frac{1}{k} Var(f(x_1, \ldots, x_N)) + \frac{k-1}{k} Var(\tilde{f}(x_1, \ldots, x_M)). \tag{4}$$

We are now in position to give our first key result:

Theorem 1. *The variance of marginalised Q-function $\tilde{Q}_i$ is (weakly) less than that of the non-marginalised Q-function Q_i for any $i \in \mathcal{N}$, that is to say:*

$$Var(Q_i(s, \mathbf{a})) \geq Var(\tilde{Q}_i(s, a_i)). \tag{5}$$

Moreover, for the approximation to the marginalised Q-function (c.f. Equation 2) the following relationship holds:

$$Var\left(\hat{Q}_i(s, a_i)\right) = \frac{1}{k} Var\left(Q_i(s, \boldsymbol{a}_{-i}, a_i)\right) + \frac{k-1}{k} Var\left(\tilde{Q}_i(s, a_i)\right). \tag{6}$$

Therefore for $k = 1$, we observe that the approximation has the same variance as the non-marginalised Q-function. However, for any $k > 1$ the approximation to marginalised Q-function has less variance than the non-marginalised Q-function. Therefore marginalisation procedure can essentially be used as a variance reduction technique. Let us now analyse how this framework can be applied to enhance multi-agent policy gradient algorithm, which is known to suffer from high variance in its original version.

In the policy gradient algorithms, we assume a fully cooperative game which avoids the need to add the agent indices to the state-action and state-value functions since the agents have identical rewards. The goal of each agent is therefore to maximise the expected return from the initial state defined as $\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{s_0 \sim p(s_0)}[v(s_0)]$, where $p(s_0)$ is the distribution of initial states and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T)^T$ is the concatenated vector consisting of policy parameters for all agents. The following well-known theorem establishes the gradient of $\mathcal{J}(\boldsymbol{\theta})$ with respect to the policy parameters.

Theorem 2 (MARL Policy Gradient [27])**.**

$$\nabla_{\boldsymbol{\theta}_i} \mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t Q(s^t, \boldsymbol{a}_{-i}^t, a_i^t) \nabla_{\boldsymbol{\theta}_i} \log \pi_i(a_i^t|s^t)\right].$$

Therefore, to calculate the gradient with respect to policy parameters, it is necessary to make use of the state-action-values. In practice, it can estimated by a function approximator, which gives rise to Actor-Critic methods [9]. In MARL, one can either maintain a centralised critic that provides state-action-value for the i^{th} agent using the knowledge of the other agents' actions or introduce a decentralized critic that does not take actions of others (in its inputs). This gives rise to the centralised training decentralized execution (CT-DE) $\boldsymbol{g}_i^{\mathrm{C}}$ and decentralized $\boldsymbol{g}_i^{\mathrm{D}}$ gradient estimators respectively, as defined below.

$$\boldsymbol{g}_i^{\mathrm{C}} \coloneqq \sum_{t=0}^{\infty} \gamma^t Q(s^t, \boldsymbol{a}_{-i}^t, a_i^t) \nabla_{\boldsymbol{\theta}_i} \log \pi_i(a_i^t|s^t), \tag{7}$$

$$\boldsymbol{g}_i^{\mathrm{D}} \coloneqq \sum_{t=0}^{\infty} \gamma^t \tilde{Q}(s^t, a_i^t) \nabla_{\boldsymbol{\theta}_i} \log \pi_i(a_i^t|s^t), \tag{8}$$

where $Q(s_t, a_{-i}^t, a_i^t)$ and $\tilde{Q}(s^t, a_i^t)$ are the CT-DE and decentralized critics respectively. Using our marginalisation technique (see Algorithm 1) we use a third estimator:

$$\boldsymbol{g}_i^{\mathrm{P}} \coloneqq \sum_{t=0}^{\infty} \gamma^t \hat{Q}(s^t, a_i^t) \nabla_{\boldsymbol{\theta}_i} \log \pi_i(a_i^t|s^t), \tag{9}$$

where $\hat{Q}(s^t, a_i^t)$ is the Monte-Carlo approximation of $\tilde{Q}(s^t, a_i^t)$. Note, in this approach we maintain a centralised critic $Q(s^t, a_{-i}^t, \boldsymbol{a}_i^t)$, therefore the approximation to the marginalised Q-function is obtained using $\hat{Q}(s^t, a_i^t) = \frac{1}{k}\sum_{j=1}^{k} Q(s^t, a_i^t, \boldsymbol{a}_{-i}^{t(j)})$, where $\boldsymbol{a}_{-i}^{t(j)} \sim \pi_{-i}(\boldsymbol{a}_{-i}^t|s^t)$. The estimator (9) is equal in expectation to the CT-DE estimator as stated by the following Theorem.

Theorem 3. *Given the same (possibly imperfect) critic, the estimators $\boldsymbol{g}_i^C$ and $\boldsymbol{g}_i^P$ have the same expectation, that is*

$$\mathbb{E}[\boldsymbol{g}_i^C] = \mathbb{E}[\boldsymbol{g}_i^P].$$

Hence, whenever the critic provides the true Q-value, the estimator (9) is an unbiased estimate of the policy gradient (which follows from Theorem 2). However, although the CT-DE and estimator (9) have the same expectations, the estimator (9) enjoys significantly lower variance. As in [10], we analyse the excess variance the two estimators have over the decentralized estimator. First define by B_i the upper bound on the gradient norm of ith agent, i.e. $B_i = \sup_{s,\boldsymbol{a}} \left\|\nabla_{\theta^i} \log \pi_i\left(a_i|s\right)\right\|$ and by C the upper bound on the Q-function, i.e. $C = \sup_{s,\boldsymbol{a}} Q(s,\boldsymbol{a})$. We now present two theorems showing the effectiveness of the estimator (9) for policy gradients.

Theorem 4. *Given true Q-values, the difference in variances between the decentralized and marginalised estimators admits the following bound:*

$$Var(\boldsymbol{g}_i^P) - Var(\boldsymbol{g}_i^D) \leq \frac{1}{k}\frac{B_i^2 C^2}{1-\gamma^2}.$$

Theorem 5. *Given true Q-values, the difference in variances between the decentralized and CT-DE estimators admits the following bound:*

$$Var(\boldsymbol{g}_i^C) - Var(\boldsymbol{g}_i^D) \leq \frac{B_i^2 C^2}{1-\gamma^2}.$$

Therefore, we can see that with $k = 1$ the bound on excess variance of the estimator (9) is the same as for the CT-DE estimator, but as $k \to \infty$, the variance of our estimator matches the one of the fully decentralized estimator. However, this is done while still maintaining a centralised critic, unlike in the fully decentralized case. The presence of a centralised critic plays an important role in guaranteeing the convergence to a local optimum almost surely (with probability 1). The result is stated by the next Theorem.

Theorem 6. *Under the standard assumptions of stochastic approximation theory [9], an Actor-Critic algorithm using $\boldsymbol{g}_i^P$ or $\boldsymbol{g}_i^C$ as a policy gradient estimator, converges to a local optimum with probability 1, i.e.*

$$P\left(\lim_{k\to\infty} \|\nabla_{\boldsymbol{\theta}_i} \mathcal{J}(\boldsymbol{\theta}^k)\| = 0\right) = 1,$$

where $\boldsymbol{\theta}^k$ is the value of vector $\boldsymbol{\theta}$ obtained after the kth update following the policy gradient.

We present a proof sketch here and defer the full proof to Appendix **??**.

Proof sketch. *The proof consists of showing that a multi-agent Actor-Critic algorithm using policy gradient estimate $\boldsymbol{g}_i^P$ or $\boldsymbol{g}_i^C$ is essentially a special case of single-agent Actor-Critic.* □

Note that because the decentralized critic does not allow us to query the state-action-value for joint action of all agents, a decentralized actor-critic using $\boldsymbol{g}_i^D$ as policy gradient estimate is not equivalent to the single-agent version and we cannot establish convergence for it. Therefore, our estimator enjoys both the low variance property of the decentralized estimator and the convergence guarantee of the CT-DE one. Additionally, having a centralised critic yields better performance in practice in environments with strong interactions between agents [8].

Algorithm 1 PERLA MAPPO

Input: Joint-policy $\boldsymbol{\pi}$, critic parameters $\boldsymbol{\rho}$, policy parameters $\boldsymbol{\theta}$, environment E, number of marginalisation samples K
Output: Optimised joint-policy $\boldsymbol{\pi}^*$
Augment MAPPO critic $V_{\boldsymbol{\rho}}$ to as input state $\boldsymbol{s}$ and joint-action $\boldsymbol{a}_{-i}$
Rollout $\boldsymbol{\pi}$ in E to obtain data $D = (\boldsymbol{s}^0, \boldsymbol{a}^1, r^1, \ldots, \boldsymbol{s}^{T-1}, \boldsymbol{a}^T, r^T)$
for $t = 0$ **to** $T-1$ **do**
  for each agent i **do**
    Generate K samples of joint-actions at the next-state observations $\{\boldsymbol{a}_{-i}^{t(j)} \sim \boldsymbol{\pi}_{-i}(s^{t+1})\}_{j=1}^K$
    Compute TD-error: $\delta_i = \boldsymbol{r} + \gamma \frac{1}{K}\sum_{j=1}^K V_\rho(\boldsymbol{s}^t, \boldsymbol{a}_{-i}^{t(j)}) - \frac{1}{K}\sum_{j=1}^K V_\rho(\boldsymbol{s}^t, \boldsymbol{a}_{-i}^{t(j)})$ over sampled joint-actions for each agent
    Update critic parameters $\boldsymbol{\rho}$ with δ_i^2 as the loss
    Update ith agent's policy parameters $\boldsymbol{\theta}_i$ with advantages given by δ_i, using PPO update
  end for
end for

5 PERLA Instantiation

We now give a concrete instantiation of PERLA on the popular MAPPO [26] algorithm. This gives rise to **PERLA MAPPO** algorithm as shown below in Algorithm 1. As MAPPO's critic function for each agent only takes $s \in \mathcal{S}$ and $a_i \in \mathcal{A}$ as input, we augment the input of the standard MAPPO critic to take $\boldsymbol{a}_{-i}$ as well. We do not make any changes to the standard MAPPO policy, and it continues to only take the agent's local observation as input. As the critic is only needed during CT and not required for execution, PERLA MAPPO operates under the CT-DE paradigm; policies are executed in a fully decentralized manner. In PERLA MAPPO we utilise a value-function style one-step critic, where $Q(s^t, a_i^t, \boldsymbol{a}_{-i}^{t(j)}) = r_t + \gamma V(s^{t+1}, \boldsymbol{a}_{-i}^{t+1(j)})$ and $V(s, \boldsymbol{a}_{-i})$ which is approximated via a deep neural network (PERLA is fully compatible with different types of critics). In this case, marginalising the behaviour of other agents is equivalent of marginalising the next step value function, as explained in more detail in Appendix **??**. During training, when performing policy and critic updates, we require to generate samples of actions from other agents. In practice, this can either be done by each agent communicating its policy parameters or samples of actions directly. Thus the communication complexity for each round would scale as $O(\min\{D, KA\})$, where D is the length of policy parameters and A is the size of action space. After the samples are communicated, the approximate expectation of the critic can be performed, which in turn can be used to compute the TD-error. The critic is trained with squared TD error as a loss and the policy is updated with the TD-error as the advantage estimate. To perform policy updates, we use PPO [21].

6 Experiments

We ran a series of experiments in *Large-scale Matrix Games* [23], *Level-based Foraging* (LBF) [1], *Multi-agent Mujoco* [3] and the *StarCraft II Multi-agent Challenge* (SMAC) [20][3] to test if PERLA: **1.** Improves overall performance of MARL learners. **2.** Enables sample efficiency when the the number of agents is scaled up. **3.** Reduces variance of value function estimates. In all tasks, we compared the performance of PERLA MAPPO against MAPPO [26]. We report average training results across multiple scenarios/maps in LBF and SMAC. Detailed performance comparisons are deferred to the Appendix. Lastly, we ran a suite of ablation studies which we deferred to the Appendix. We implemented PERLA on top of the MAPPO implementation provided in the codebase accompanying the MARL benchmark study of Papoudakis et al. [15]. Hyperparameters were tuned using simple grid-search, the values over which we tuned the hyperparameters are presented in Table **??** in the Appendix. All results are means over 3 random seeds unless otherwise stated. In plots dark lines represent the mean across the seeds while shaded areas represent 95% confidence intervals.

6.1 Performance Analysis

Large-Scale Matrix Games. To demonstrate PERLA's ability to handle various reward structures and scale efficiently we first tested its performance in a set of variants of the hard matrix game proposed in [23] (Appendix **??**). This game contains multiple stable points and a strongly attracting equilibrium [23] (all agents selecting action A). In our variants, we scaled up the number of agents, or we increased the size of the action space (Appendix **??**). In both cases, it is crucial to accurately account for the behaviour of other agents in the system. For instance, even if the joint-policy has converged to the optimal solution, if even one agent samples an exploratory action, it can be strongly destabilising due to its high penalty (reward of 0 or −12 as opposed reward of 8 for optimal joint-action). To avoid these issues, it is crucial to base updates on the joint-policy rather than samples of the joint-action.

We used 500 training iterations and averaged the results of 10 random seeds in each method. As shown in Figure 2, policy-based methods (MAPPO and IPPO) and leading value-based methods (MAIC, QPLEX [24] and WQMIX [18]) achieve optimal performance in the initial settings (2 agents with 3 available actions), while other algorithms achieve suboptimal outcomes. When scaling with more agents and larger action space, only PERLA can maintain optimal performance across almost all variations. As shown, MAPPO completely fails when we scale the actions-space, going from a return of 8 to 0.5. PERLA MAPPO, however, is robust and attains the highest return of all tested algorithms at about 7.75.

Level-based Foraging. Figure 2 shows learning curves averaged across all LBF maps that we ran (the full list of maps is given in the Appendix **??**). As shown in the plot, PERLA significantly improves base MAPPO, both in learning speed and the quality of the policy at the end of training. For example, it takes PERLA MAPPO about 800, 000 interactions with the environment to achieve a mean evaluation return of 0.8, whereas vanilla MAPPO does not achieve

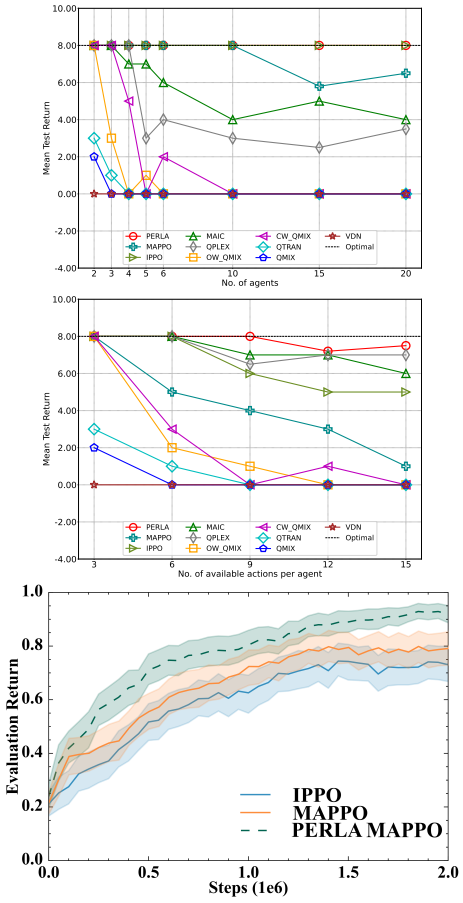

Figure 2: *Top* and *centre*: scaling with more agents and larger action space, respectively, in Cooperative Matrix Game; in both cases PERLA is able to maintain optimal performance while other algorithms' performances degrade. *Bottom:* Learning curves of mean evaluation return of MAPPO and PERLA MAPPO averaged over all tested LBF maps. PERLA improves sample efficiency (better performance faster) and quality of the final policy.

such a perfomance level even by the end of training. Moreover, PERLA MAPPO is able to attain an mean evaluation return of 0.9 by the end of training, vanilla MAPPO attains an mean evaluation return just under 0.8. Furthermore, as shown in Figure **??** (Appendix **??**)) in the map requiring the highest level of coordination between agents `Foraging-15x15-8p-1f-coop-v2` (8 agents must cooperate to attain reward), PERLA MAPPO manages to learn, and achieves an evaluation return of 0.7 while MAPPO fails entirely. This shows that PERLA enables vanilla MAPPO to scale to a high number of agents in an environment with sparse and a high variance reward function.

StarCraft II Multi-agent Challenge. Figure 3 shows performance of MAPPO and PERLA MAPPO over a wide range of SMAC maps from all difficulty levels. At regular intervals during training, we ran 10 evaluation episodes and tracked the median win-rate.

[3]The specific maps/variants used of each of these environments in given in Sections **??** and **??** of the Appendix)

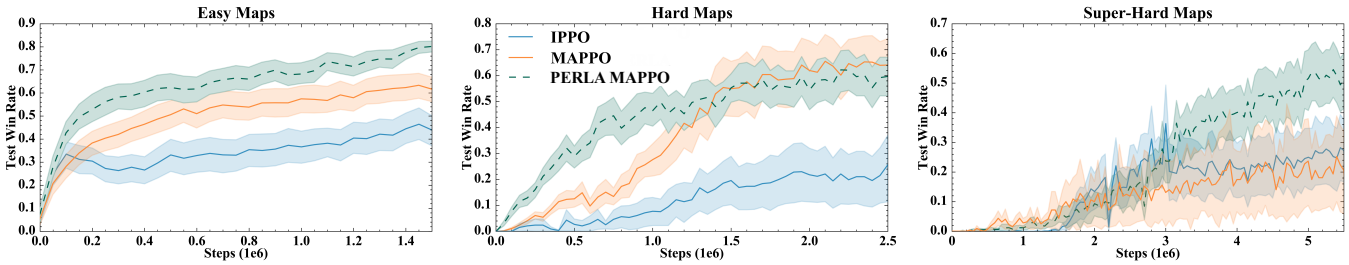

Figure 3: Average performance over *all* SMAC maps for IPPO, MAPPO, PERLA_MAPPO. PERLA enhances the base method both in terms of sample efficiency (better performance, faster) and final performance.

The learning curves are then generated by computing the mean of these win rates (disaggregated curves for each map are available in Figure **??** in Appendix **??**). SMAC maps are richly diverse and vary along several dimensions such as the number of agents to control, density of environment reward, degree of coordination required, and (partial)-observability. Therefore, aggregated and averaged performance over all maps gives us a fairly robust understanding of the effects of PERLA. As with LBF, PERLA enhances the performance MAPPO. PERLA MAPPO is more sample efficient and converges to better overall policies (within the training budget).

Multi-agent Mujoco. To study PERLA's capabilities in complex settings that require both scalability and coordination, we compared its performance with vanilla MAPPO on three tasks in Multi-agent Mujoco: Walker 2×3, Hopper 3×1, and Swimmer 2×1. In Figure 4, we report learning curves averaged over 6 seeds of each algorithm. As can be seen, PERLA MAPPO outperforms or equals MAPPO on all three tasks. By enabling agents to maintain estimates that account for other agents' actions, PERLA achieves more accurate value estimation with the variance reduction, therefore establishing more efficient learning.

6.2 Scaling Analysis

Scaling efficiently to large systems (i.e. systems with many agents and large action spaces) is a major challenge in MARL. In Section 1, we claimed the PERLA framework enables MARL to scale efficiently in terms of the number of samples. To test this claim, we investigated PERLA's scaling ability in our large scale matrix games (described above) and LBF. In matrix games, we demonstrated PERLA's ability to efficiently scale across both dimensions, namely, games varied by (1) the number of agents $N = 2, 3, 4, 5, 6, 10, 15, 20$ and (ii) the cardinality of the agents' action sets $|\mathcal{A}_i| = 3, 6, 9, 12, 15$. In each case, we retained the setup that reward agents with a score of 8 only when all agents choose the first action.

We further tested this claim in LBF scenarios with 2, 5, and 8 agent, respectively. We expected to see PERLA MAPPO's advantage increase with the number of agents as the algorithm allows each agent to better account for the actions of other agents in the system. Figure 5 shows PERLA MAPPO's over MAPPO. As shown, PERLA enables monotonic performance gains with the number of agents, yielding over 1000% improvement in systems with 8 agents.

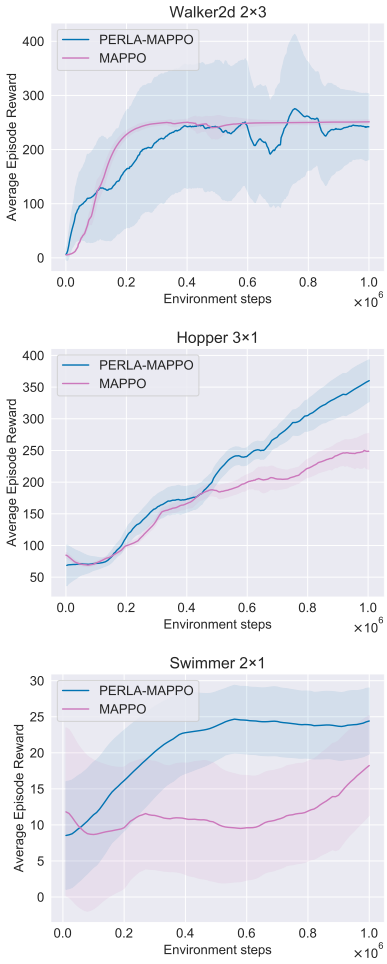

Figure 4: Comparisons of PERLA MAPPO versus MAPPO on three Multi-agent Mujoco tasks. PERLA MAPPO learns faster and converges to superior policies.

6.3 Variance Analysis

In Sec. 4, we proved that PERLA reduces the variance of VF estimators. To show this empirically, we constructed a toy problem with three agents. Each agent has a binary action space of $\{0, 1\}$

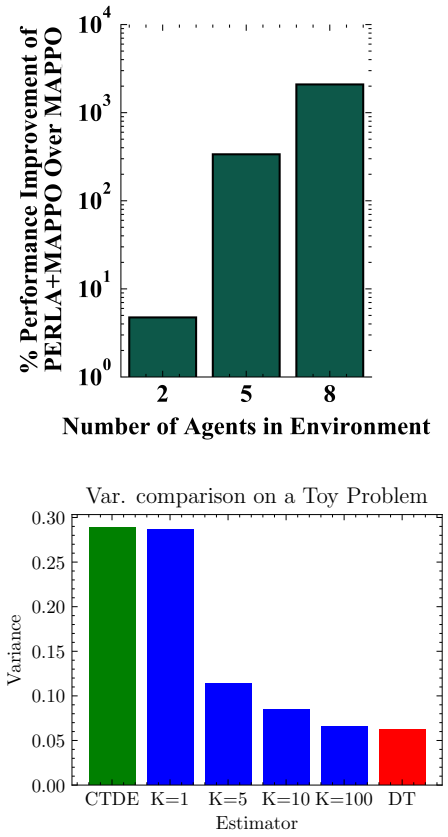

Figure 5: Top: % performance increase of policy trained by PERLA MAPPO over policy trained by MAPPO as a function number of agents (N). For $N = 8$, PERLA MAPPO yields over +1000% performance gains over MAPPO. Bottom: Comparison of Policy Gradient Variance on a Toy Problem for CTDE, DT and PERLA estimators. $K = n$ denotes a PERLA estimator using n samples.

and plays a uniform policy. The team receives a reward of 1 if all players play action 0 and a reward of 3 if all players play action 1. If at least one player plays an action different from others, the whole team receives a reward of 0. As this is a state-less game, the Q-function takes only actions as inputs. We consider the case, where the policy of agent 1 is defined by a sigmoid function i.e. $\pi(a_1 = 1; \theta) = \frac{1}{1+\exp(-\theta)}$. We repeat the experiment 1000 times and measure the variance of the policy gradient for $\theta = 0$ calculated using PERLA (Eq. 9), CTDE (Eq. 7) and DT (Eq. 8) estimators. In the results shown in Figure 5 we see that as number of samples k increases, the variance of PERLA estimator sharply decreases and almost matches the variance of DT estimator for very large k. This is consistent with Theorem 4, where we have proven that as k increases variance of PERLA estimator approaches the variance of DT estimator at a rate of $1/k$.

6.4 Ablation on number of agents

An important question is the scaling behaviour of the PERLA method with the number of agents. To answer this question, we examined the performance of PERLA MAPPO while varying the number of agents from 2 to 8 for different per-state joint action samples. We ran this experiment in the LBF environment. Figure 6 displays the overall return when the PERLA method is allowed the make 5, 25 and 125 per-state joint action samples. The results indicate that the performance remains stable when the number of per-state joint action samples is varied while, as expected, the performance diminishes only slightly when the number of agents increases.

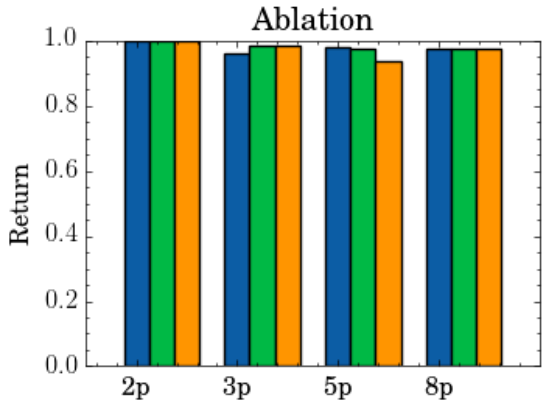

Figure 6: Final mean return against the number of players increases in LBF environment. Blue indicates 5 samples, green indicates 25 and orange corresponds to 125 samples.

7 Conclusion

Centralised training is a fundamental paradigm in performant, modern actor-critic MARL algorithms. While it enables agents to coordinate to solve challenging problems, MARL estimators still suffer from high variance. This hinders learning and reduces the sample efficiency of MARL methods. Scalability and efficient learning are key challenges in MARL research. In this paper, we introduced PERLA, an enhancement tool that induces sample efficient, coordinated learning among MARL agents. Our theory and empirical analyses show that PERLA reduces the variance of VF estimators which is critical for efficient learning. In this way, PERLA enables MARL algorithms to exhibit sample efficient learning and a high degree of scalability.

Acknowledgements

Jianhong Wang was supported by the Engineering and Physical Sciences Research Council [Grant Ref: EP/Y028732/1].

REFERENCES

[1] Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[2] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is Independent

Learning All You Need in the StarCraft Multi-Agent Challenge? *arXiv preprint arXiv:2011.09533* (2020).

[3] Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2020. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709* (2020).

[4] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*.

[5] Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. 2022. Revisiting Some Common Practices in Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 6863–6877. https://proceedings.mlr.press/v162/fu22d.html

[6] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. 2016. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247* (2016).

[7] Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. 2024. Exploration in Deep Reinforcement Learning: From Single-Agent to Multiagent Domain. *IEEE Trans. Neural Networks Learn. Syst.* 35, 7 (2024), 8762–8782.

[8] Jelle R Kok and Nikos Vlassis. 2004. Sparse cooperative Q-learning. In *Proceedings of the twenty-first international conference on Machine learning*. 61.

[9] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[10] Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, Yaodong Yang, et al. 2021. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems* 34 (2021), 13458–13470.

[11] David Mguni, Joel Jennings, and Enrique Munoz de Cote. 2018. Decentralised learning in systems with many, many strategic agents. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[12] David Mguni, Yutong Wu, Yali Du, Yaodong Yang, Ziyi Wang, Minne Li, Ying Wen, Joel Jennings, and Jun Wang. 2021. Learning in Nonzero-Sum Stochastic Games with Potentials. *arXiv preprint arXiv:2103.09284* (2021).

[13] David Henry Mguni, Taher Jafferjee, Jianhong Wang, Nicolas Perez-Nieves, Oliver Slumbers, Feifei Tong, Yang Li, Jiangcheng Zhu, Yaodong Yang, and Jun Wang. 2021. LIGS: Learnable Intrinsic-Reward Generation Selection for Multi-Agent Learning. *arXiv preprint arXiv:2112.02618* (2021).

[14] Zepeng Ning and Lihua Xie. 2024. A survey on multi-agent reinforcement learning and its application. *Journal of Automation and Intelligence* (2024).

[15] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*. http://arxiv.org/abs/2006.07869

[16] Bei Peng, Tabish Rashid, Christian A Schroeder de Witt, Pierre-Alexandre Kamienny, Philip HS Torr, Wendelin Böhmer, and Shimon Whiteson. 2020. FACMAC: Factored Multi-Agent Centralised Policy Gradients. *arXiv preprint arXiv:2003.06709* (2020).

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

[18] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2006.10800* (2020).

[19] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 4295–4304.

[20] 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. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).

[22] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

[23] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5887–5896.

[24] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2020. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062* (2020).

[25] Yaodong Yang, Ying Wen, Jun Wang, Liheng Chen, Kun Shao, David Mguni, and Weinan Zhang. 2020. Multi-agent determinantal q-learning. In *International Conference on Machine Learning*. PMLR, 10757–10766.

[26] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).

[27] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. 2018. Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*. PMLR, 5872–5881.

[28] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, et al. 2020. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776* (2020).