MAC-PO: Multi-Agent Experience Replay via Collective Priority Optimization

Yongsheng Mei
The George Washington University
Washington, DC, United States
ysmei@gwu.edu

Hanhan Zhou
The George Washington University
Washington, DC, United States
hanhan@gwu.edu

Tian Lan
The George Washington University
Washington, DC, United States
tlan@gwu.edu

Guru Venkataramani
The George Washington University
Washington, DC, United States
guruv@gwu.edu

Peng Wei
The George Washington University
Washington, DC, United States
pwei@gwu.edu

ABSTRACT

Experience replay is crucial for off-policy reinforcement learning (RL) methods. By remembering and reusing the experiences from past different policies, experience replay significantly improves the training efficiency and stability of RL algorithms. Many decision-making problems in practice naturally involve multiple agents and require multi-agent reinforcement learning (MARL) under centralized training decentralized execution paradigm. Nevertheless, existing MARL algorithms often adopt standard experience replay where the transitions are uniformly sampled regardless of their importance. Finding prioritized sampling weights that are optimized for MARL experience replay has yet to be explored. To this end, we propose MAC-PO, which formulates optimal prioritized experience replay for multi-agent problems as a regret minimization over the sampling weights of transitions. Such optimization is relaxed and solved using the Lagrangian multiplier approach to obtain the close-form optimal sampling weights. By minimizing the resulting policy regret, we can narrow the gap between the current policy and a nominal optimal policy, thus acquiring an improved prioritization scheme for multi-agent tasks. Our experimental results on Predator-Prey and StarCraft Multi-Agent Challenge environments demonstrate the effectiveness of our method, having a better ability to replay important transitions and outperforming other state-of-the-art baselines.

KEYWORDS

Multi-Agent Reinforcement Learning; Experience Replay; Priority Optimization

ACM Reference Format:
Yongsheng Mei, Hanhan Zhou, Tian Lan, Guru Venkataramani, and Peng Wei. 2023. MAC-PO: Multi-Agent Experience Replay via Collective Priority Optimization. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

Reinforcement learning (RL) has demonstrated great success in solving challenging problems [24, 48]. For off-policy RL, experience replay mechanism [29, 38] allows utilizing history experiences in the replay buffer that stores the most recently collected transitions for training. It has been shown to significantly improve policy learning and RL algorithms' stability. Due to these benefits, various approaches [47, 49, 64] for computing priority scores of experiences have been proposed for single-agent RL. For instance, prioritized experience replay (PER) [47] leverages predefined metrics for prioritizing experience based on the temporal-difference (TD) error related to the loss of the critic network. It calculates the sampling probabilities proportional to the magnitude of TD error, resulting in a non-uniform sampling/prioritization scheme in Q-learning [63].

In practice, we often face RL tasks involving multiple agents sharing the same environment, e.g., in autonomous driving [7, 20] and robotics and planning [21, 28, 33]. To coordinate multiple agents and learn desired joint behavior from their collective experiences, we require multi-agent reinforcement learning (MARL) [2, 23, 58], such as value-based methods QMIX [44] and QPLEX [59], or policy-based methods COMA [12] and MADDPG [32]. These approaches leverage centralized training decentralized execution (CTDE) [25] and often employ standard memory replay buffers with a uniform sampling of transition history. However, in MARL problems, such a standard sampling strategy of the replay buffer cannot reflect the dynamics in the environment caused by multi-agent interactions. Therefore, indiscriminately training from past experiences will make agents less capable of using experiences optimally. Although we can impart existing single-agent prioritization schemes directly to the joint action-value function of MARL, such a naive application is oblivious to the interaction between multiple agents in the shared environment and may lead to sub-optimal performance. Thus, priority optimization for experience replay in MARL is still an open problem.

To this end, we propose MAC-PO, which formulates MARL prioritized experience replay problems as a regret minimization over the sampling weights of different state-action values. Specifically, we define policy regret as the difference between the expected discounted reward of a nominal optimal policy and that of the current policy under given sampling weights. By minimizing such a policy regret by considering its upper bound, we can narrow the gap between the optimal and current policies with respect to the sampling weights, leading to an optimal solution of sampling weights with minimum regret. We note that similar regret minimization techniques have been employed in single-agent RL settings [31].

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

Our paper expands it to analyze multi-agent prioritized experience replay and develops new solutions, e.g., to handle joint actions of multiple agents and to analyze concurrent optimality constraints. It turns out that the optimal sampling weights in MARL now depend on the collective policies of decentralized agents. To the best of our knowledge, this is the first proposal for optimizing prioritized experience replay in cooperative MARL.

In particular, we show that the proposed optimization can be solved via the Lagrangian multiplier method [4] considering an upper bound of the regret. Since we focus on multi-agent prioritized experience replay problems, the optimization objective is defined by the joint policy of all agents. Therefore, when we further analyze the Lagrangian conditions for optimality, the agents' conditions depend on each other and become a vector form. Further, by examining a weighted Bellman equation, we leverage the implicit function theorem [26] for multiple agents and apply a group of Karush–Kuhn–Tucker (KKT) [14] conditions to find the optimal sampling weights in closed form.

Our results illuminate the key principles contributing to optimal sampling weights in multi-agent prioritized experience replay. The optimal sampling weights can be interpreted to consist of four components: Bellman error, value enhancement, on-policiness of available transitions, and a new term depending on joint action probabilities. While the first three have been identified in single agent settings [27, 31], our paper shed light on a new term - as a function of joint action probabilities - to reveal that optimal sampling weights of multi-agent prioritized experience replay should depend on the interaction among all the agents within an environment. More specifically, we should assign the highest sampling weights to transitions only if one agent's action probability is small in the transition while all other agents' action probabilities are large. The result – slightly counter-intuitive since higher weights are assigned to transitions with more differentiated action probabilities (rather than similar ones) – is quantified and formalized as a new theorem in our paper. Based on this result, we also present an approximated solution for estimating sampling weights in problems involving many agents or having limited information for an exact solution.

Following the theoretical analysis, we propose a MARL algorithm, MAC-PO, for multi-agent prioritized experience replay via regret minimization. Like existing methods, MAC-PO can be plugged into any MARL algorithms with a memory replay buffer. We validate the effectiveness of MAC-PO in StarCraft Multi-Agent Challenge (SMAC) [46] and Predator-Prey [5] through comparison with other single-agent experience replay methods (adapted to MARL problems by considering all agents as a conceptual agent). Moreover, we also compared MAC-PO with state-of-the-art MARL algorithms. In the experiments, MAC-PO demonstrates improved convergence and superior empirical performance.

The main contributions of our work are as follows:

- We propose a novel method, MAC-PO, which formulates multi-agent experience replay as a policy regret minimization and solves the optimal sampling weights in closed form.
- The theoretical results illuminate a new factor in optimal sampling weights and motivate the design of new MARL experience replay algorithms with both exact and approximated weights.
- Experiment results of MAC-PO in SMAC and Predator-Prey environments demonstrate superior convergence and empirical performance over various baselines, including experience replay and state-of-the-art MARL methods.

2 BACKGROUND

2.1 Partially Observable Markov Decision Process

In this work, we consider a multi-agent sequential decision-making task as a decentralized partially observable Markov decision process (Dec-POMDP) [42] consisting of a tuple $G = \langle S, U, P, R, Z, O, n, \gamma \rangle$, where $s \in S$ describes the global state of the environment. At each time step, each agent $a \in A \equiv \{1, \dots, n\}$ selects an action $u_a \in U$, and all selected actions combine and form a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. Such a process leads to a transition in the environment based on the state transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$. All agents share the same reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$ with a discount factor $\gamma \in [0, 1)$.

In the partially observable environment, the agents' individual observations $z \in Z$ are generated by the observation function $O(s, u) : S \times A \to Z$. Each agent has an action-observation history $\tau_a \in T \equiv (Z \times U)^*$. Conditioning on the history, the policy becomes $\pi^a(u_a|\tau_a) : T \times U \to [0, 1]$. The joint policy $\boldsymbol{\pi}$ has a joint action-value function: $Q^{\pi}(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}}[R_t|s_t, \mathbf{u}_t]$, where t is the timestep and $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return. In this paper, we adopt the CTDE mechanism. The learning algorithm has access to all local action-observation histories $\boldsymbol{\tau}$ and global state s during training, yet every agent can only access its individual history in execution. Although we compute individual policy based on histories in practice, following the existing work [52], we will use $\pi^a(u_a|s)$ in analysis and proofs for simplicity.

2.2 Policy Regret

In MARL, we aim to find a joint policy $\boldsymbol{\pi}$ that can maximize the expected return: $\eta(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]$. For a fixed policy, the Markov decision process becomes a Markov reward process, where the discounted state distribution is defined as $d^{\boldsymbol{\pi}}(s)$. Similarly, the discounted state-action distribution is defined as $d^{\boldsymbol{\pi}}(s, \mathbf{u}) = d^{\boldsymbol{\pi}}(s)\boldsymbol{\pi}(\mathbf{u}|s)$. Then, we will have the expected return rewritten as $\eta(\boldsymbol{\pi}) = \frac{1}{1-\gamma}\mathbb{E}_{d^{\boldsymbol{\pi}}(s,\mathbf{u})}[r(s, \mathbf{u})]$.

We assume a nominal optimal joint policy $\boldsymbol{\pi}^*$ such that $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \eta(\boldsymbol{\pi})$. The regret of the joint policy $\boldsymbol{\pi}$ is the difference between the expected discounted reward of an optimal policy and that of the current policy as $\text{regret}(\boldsymbol{\pi}) = \eta(\boldsymbol{\pi}^*) - \eta(\boldsymbol{\pi})$. The policy regret measures the expected loss when following the current policy $\boldsymbol{\pi}$ instead of optimal policy $\boldsymbol{\pi}^*$. Since $\eta(\boldsymbol{\pi}^*)$ is a constant, minimizing the regret is consistent with maximizing of expected return $\eta(\boldsymbol{\pi})$. In this paper, we use regret as an alternative optimization objective for finding the optimal sampling weight in MARL tasks, along with multiple constraints, such as the Bellman equation. By minimizing the regret, the current joint policy $\boldsymbol{\pi}_k$ of all agents' actions will approach the optimum $\boldsymbol{\pi}^*$.

2.3 Connection of Prioritized Sampling and Weighted Loss Function

The design of prioritized sampling methods is not isolated from the loss function. Instead, the expected gradient of a loss function with non-uniform sampling is equivalent to that of a weighted loss function with uniform sampling, which facilitates the design of prioritized sampling algorithms [13]. Given a data sample set D of size d, a regular loss function L_1 where we use a specific priority scheme $pr(\cdot)$ to sample the transitions, and another loss function L_2 whose transitions are sampled uniformly, the two approaches are equivalent if we have the following requisition satisfied:

$$\nabla_Q L_1 = \frac{\chi}{pr} \nabla_Q L_2,$$

where $\chi = \frac{\sum_i pr(i)}{d}$ and $i \in D$ is the uniformly sampled instance.

We can leverage such equivalence to analyze the correctness of approaches using non-uniform sampling by transforming the loss into the uniform-sampling equivalent or considering whether the new loss is in line with the target objective. It also provides a recipe for transforming a regular loss function L_1 with a non-uniform sampling scheme into an equivalent weighted loss function L_2 with uniform sampling.

3 RELATED WORKS

3.1 MARL Algorithms

MARL algorithms have developed into neural-network-based methods that can cope with high-dimensional state and action spaces. Early methods practice finding policies for a multi-agent system by directly learning decentralized value functions or policies. For example, independent Q-learning [55] trains independent action-value functions for each agent via Q-learning. [54] extends this technique to DQN [38]. Recently, approaches for CTDE have come up as centralized learning of joint actions that can conveniently solve coordination problems without introducing non-stationary. COMA [12] uses a centralized critic to train decentralized actors to estimate a counterfactual advantage function for every agent. Similar works [17, 32] are also proposed based on such analysis. Under CTDE manner, value decomposition approaches [8, 16] are widely used in value-based MARL. Such methods integrate each agent's local action-value functions through a learnable mixing function to generate global action values. For instance, QMIX [44] estimates the optimal joint action-value function by combining mentioned utilities via a continuous state-dependent monotonic function generated by a feed-forward mixing network with non-negative weights. QTRAN [50] and QPLEX [59] further extend the class of value functions that can be represented. ReMIX [37] provides a factorization weighting scheme to find the optimal projection of an unrestricted mixing function onto monotonic function classes. PAC[67] and LAS-SAC[68] proposes to use latent assisted information [36] as extra-state information for better value factorization. Aside from methods focusing on tackling cooperative problems, other mechanisms can also solve competitive problems or mixed problems. MADDPG [32] utilizes the ensemble of policies for each agent that leads to more robust multi-agent policies, showing strength in cooperative and competitive scenarios. Beyond that, the extensions [15, 22, 51] of MADDPG have been proposed to realize further optimization towards the original algorithm. In this paper, we focus on the cooperative setting and leverage a standard QMIX with a monotonic mixing network, along with an unrestricted QMIX [43] without a monotonic function for retrieving the optimal joint policy.

3.2 Single-Agent Experience Replay

Many RL algorithms adopt prioritization to increase the learning speed, initially originating from prioritized sweeping for value iteration [39, 57]. Besides, they have also been used in other modern applications, such as learning from demonstrations [18]. Prioritized experience replay [47] is one of several popular improvements to the DQN algorithms [56, 62] and has been included in many algorithms combining multiple improvements [3, 19]. Variations of PER have been proposed for considering sequences of transitions [6, 9] or optimizing the prioritization function [64]. Furthermore, to favor recent transitions without explicit prioritization, alternate replay buffers have been raised [40]. [10, 65] studied the composition and size of the replay buffer, and [30] looked into prioritization in simple environments. Other important sampling approaches also greatly improved the performance. [27] re-weights updates to reduce variance. [31] uses the regret minimization method to design the prioritized experience replay scheme for the only agent in the environment. MaPER [41] employs model learning to improve experience replay by using a model-augmented critic network and modifying the rule of priority. Also, new loss function designs can help develop prioritization schemes [53]. So far, most works about experience replay are designed for single-agent reinforcement learning, and a limited number of works [1, 11, 61] investigate the possible extensions. In this paper, we proposed MAC-PO for MARL tasks by considering the interaction among multiple agents through collective priority optimization to seek an optimal multi-agent prioritization mechanism.

4 METHODOLOGY

4.1 Problem Formulation

Let Q_k denote the action-value function at iteration k. We leverage $\mathcal{B}^* Q_{k-1}$ as the target with a Bellman operator $\mathcal{B}^*$ and update Q_k in tandem using a weighted Bellman equation: $Q_k = \arg\min_{Q \in \mathcal{Q}} \mathbb{E}_\mu[w_k(s,\mathbf{u})(Q - \mathcal{B}^* Q_{k-1})^2(s,\mathbf{u})]$, where $w_k(s,\mathbf{u})$ represent non-negative sampling weights for different transitions that need to be optimized for the experience replay.

To formulate the policy regret with respect to the joint action-value function, we consider a Boltzmann policy $\boldsymbol{\pi}_k$ corresponding to each agent's individual utilities Q_k^a, i.e., $\boldsymbol{\pi}_k = [\pi_k^1, ..., \pi_k^n]^\mathrm{T}$ and $\pi_k^a = e^{Q_k^a(\tau_a, u_a)} / \sum_{\tau_a, u'_a} e^{Q_k^a(\tau_a, u'_a)}$. Our objective is to minimize the policy regret $\eta(\boldsymbol{\pi}^*) - \eta(\boldsymbol{\pi})$ over non-negative sampling weights under relevant constraints, i.e.,

$$\begin{aligned} \min_{w_k} \quad & \eta(\boldsymbol{\pi}^*) - \eta(\boldsymbol{\pi}_k) \\ \text{s.t.} \quad & Q_k = \arg\min_{Q \in \mathcal{Q}} \mathbb{E}_\mu[w_k(s,\mathbf{u})(Q - \mathcal{B}^* Q_{k-1})^2(s,\mathbf{u})], \quad (1) \\ & \mathbb{E}_\mu[w_k(s,\mathbf{u})] = 1, \quad w_k(s,\mathbf{u}) \geq 0, \end{aligned}$$

where $\boldsymbol{\pi}_k$ and $\boldsymbol{\pi}_k^*$ are Boltzmann policies for the current and nominal optimal policy, and the latter can be obtained from another

network. The sampling weights must sum up to 1, and μ is the distribution that we uniformly sample data from the replay buffer. An additional table to summarize and explain the common notations is provided in Appendix A.

4.2 Solving Optimal Sampling Weights for Experience Replay

Our goal is to seek the optimal priority by minimizing the regret at every iteration k, with respect to the weight w used for Bellman error minimization at iteration k. For this purpose, we consider an upper bound of the relaxed regret objective and formulate its Lagrangian by introducing Lagrangian multipliers regarding the constraints. It allows us to solve the proposed regret-minimization problem and obtain optimal projection weights in closed form (albeit with a normalization factor Z^*).

Theorem 1 (Optimal sampling weight). *The optimal weight $w_k(s, \mathbf{u})$ to a relaxation of the regret minimization problem in Equation (1) with discrete action space is given by:*

$$w_k(s, \mathbf{u}) = \frac{1}{Z^*}(E_k(s, \mathbf{u}) + \epsilon_k(s, \mathbf{u})), \tag{2}$$

where we have:

$$E_k(s, \mathbf{u}) = \frac{d^{\boldsymbol{\pi}_k}(s, \mathbf{u})}{\mu(s, \mathbf{u})} |Q_k - \mathcal{B}^* Q_{k-1}| \cdot \exp(-|Q_k - Q^*|) \left(1 + \sum_{i=1}^{n} \prod_{\substack{j=1 \\ j \neq i}}^{n} \pi_k^j - n \prod_{i=1}^{n} \pi_k^i\right), \tag{3}$$

where Z^* is the normalization factor, and $\epsilon_k(s, \mathbf{u})$ is a negligible term when the probability of reversing back to the visited state is small or the number of steps agents take to revisit a previous state is large.

Proof (Sketch). We give a sketch of the steps involved for completeness below. The complete proof is provided in the Appendix B. The derivation of optimal weights consists of the following major steps: (i) Use a relaxation and Jensen's inequality to obtain a more tractable upper bound of the regret objective for minimization. (ii) Formulate the Lagrangian for the new optimization problem and analyze its KKT conditions. (iii) Compute various terms in the KKT condition and, in particular, analyze the gradient of Q_k with respect to weights p_k (defined through the weighted Bellman equation) by leveraging the implicit function theorem (IFT). (iv) Derive the optimal projection weights in closed form by setting the Lagrangian gradient to zero and applying KKT and its slackness conditions.

Step 1: Relaxing the objective and using Jensen's Inequality. To begin with, we replace the original optimization objective function, the policy regret, with a relaxed upper bound. This replacement can be achieved through the following inequality:

$$\eta(\pi^*) - \eta(\pi_k) \leq \mathbb{E}_{d^{\pi_k}(s,\mathbf{u})}[|Q_k - Q^*|(s, \mathbf{u})]. \tag{4}$$

The proof of this result is given in the appendix. The key idea is to rewrite the regret using the expectation of the action-value functions with respect to discounted state distribution d^{π_k}. After that, we adopt Jensen's inequality [34] to continue relaxing the intermediate objective function. Consider a convex function $g(x) = \exp(-x)$, a new optimization objective relaxed via Jensen's inequality generated from Equation (4) becomes:

$$\min_{w_k} \quad -\log \mathbb{E}_{d^{\boldsymbol{\pi}_k}(s,\mathbf{u})}[\exp(-|Q_k - Q^*|)(s, \mathbf{u})], \tag{5}$$

where the constraints still hold for the new optimization objective.

Step 2: Computing the Lagrangian. In this step, we leverage the Lagrangian multiplier method to solve the new optimization problem in Equation (5). For simplicity, we use p_k that absorbs the data distribution μ into w_k. The constructed Lagrangian is:

$$\mathcal{L}(p_k; \lambda, \psi) = -\log \mathbb{E}_{d^{\boldsymbol{\pi}_k}(s,\mathbf{u})}[\exp(-|Q_k - Q^*|)(s, \mathbf{u})] + \lambda(\sum_{s,\mathbf{u}} p_k - 1) - \psi^{\mathrm{T}} p_k,$$

where p_k is the weight w_k multiplied by the data distribution μ, and λ, ψ are the Lagrange multipliers.

Step 3: Computing the Gradients Required in the Lagrangian. According to the first constraint in Equation (1), the gradient $\frac{\partial Q_k}{\partial p_k}$ can be computed via IFT given by:

$$\frac{\partial Q_k}{\partial p_k} = -[\mathrm{diag}(p_k)]^{-1}[\mathrm{diag}|Q_k - \mathcal{B}^* Q_{k-1}|].$$

We also derive the gradient $\frac{\partial d^{\boldsymbol{\pi}_k}(s,\mathbf{u})}{\partial p_k}$ for solving the Lagrangian. The derivation details are given in the appendix.

Step 4: Deriving the Optimal Weight. After having the equation for two gradients and an expression of the Lagrangian, we can compute the optimal p_k via an application of the KKT conditions, which needs to set the partial derivative of the Lagrangian equaling to zero, as:

$$\frac{\partial \mathcal{L}(p_k; \lambda, \psi)}{\partial p_k} = 0,$$

where the optimal weight w_k can be acquired from the p_k.

□

The theoretical results shed light on the key factors determining an optimal sampling weight for experience replay. Specifically, the optimal weights consist of four components relating to the Bellman error, the value enhancement, the joint action probability, and the on-policiness of available transitions. We will interpret these four components next, provide the analyses of some special cases in which the transitions will be assigned with higher weights, and develop a deep MARL algorithm through approximations of the optimal sampling weights.

Bellman error $|Q_k - \mathcal{B}^* Q_{k-1}|$: is the estimation of the action value function after the Bellman update. This term measures the distance between the estimation and the Bellman target. A significant difference in this term means higher hindsight Bellman error and will lead to higher sampling weight assignment. This character is also similar to the prioritization criterion used in PER, which nevertheless considers more about the Bellman error in the previous iterations, i.e., $|Q_{k-1} - \mathcal{B}^* Q_{k-2}|$.

Value enhancement $\exp(-|Q_k - Q^*|)$: As we compute the absolute value between the current and optimal action-value function, the value enhancement term indicates that any transitions with less accurate action values compared to the optimal value estimation (i.e., a wider gap between Q_k and Q^*) after the Bellman update should be assigned with lower weights. Conversely, a high sampling

weight will be given if the current action value is approaching the optimal one.

Joint action probability $1 + \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} \pi_k^j - n \prod_{i=1}^{n} \pi_k^i$: The agent policies determine the probabilities of choosing certain actions. This result turns out that the optimal sampling weights depend on the individual policy of each agent as well, which is unique in the MARL task. According to this term, higher sampling weights will be assigned to transitions only if one agent's action probability is small in the transition while all other agents' action probabilities are large. This is a little counter-intuitive because we give higher weights to transitions with more differentiated action probabilities rather than similar ones. We will provide a thorough analysis in section 4.3 regarding studying the condition for the highest weight assignment in the general multi-agent scenario.

Measurement of on-policy transitions $\frac{d^{\boldsymbol{\pi}_k}(s,\mathbf{u})}{\mu(s,\mathbf{u})}$*:* The efficient update of the joint action value function can be achieved by focusing on transitions that are more possibly to be visited by the current policy, i.e., with a higher $d^{\boldsymbol{\pi}_k}(s, \mathbf{u})$. Such strategy has been empirically studied in existing works [49]. Adding this term can speed up the search for the optimal Q_k close to Q^*.

4.3 Approximated Weights via Joint Action Probability Studies

Theorem 1 shows terms determining the sampling weights needed for transitions, where a function of the joint action probability is the new result for MARL tasks. Although numerical calculation for the joint action probability is available, to lower the computational complexity when the environment has many agents involved, we develop an approximated weighting scheme that can determine the joint action probability via action probabilities and action-value functions of agents. For this purpose, we present a new theorem indicating the condition for obtaining maximum probability and several special case studies.

For the environment, we consider a general MARL scenario with agent space of n, where we have $a \in A \equiv \{1, \ldots, n\}$. Every step, each agent a selects an action from its action space U^a, following $u_a^i \in U^a \equiv \{u_a^1, ..., u_a^{m_a}\}$, where m_a is the size of action space of agent a. Let the $\bar{u}_a \in U^a$ denote the selected action of agent a at the step k from the action space. Due to the CTDE manner of MARL algorithms, the joint action value function space $\boldsymbol{Q}$ contains the combinations of u_a^i (i ranges from 1 to m_a) for each agent a. For simplicity, we use Q_i to be the shorthand of $Q(s, u_1^{i_1}, \ldots, u_n^{i_n})$, which represent a random action value function from $\boldsymbol{Q}$ space. In particular, considering one selected action combination $\bar{\mathbf{u}} = (\bar{u}_1, \ldots, \bar{u}_n)$, the joint action-value function is $\bar{Q} = Q(s, \bar{\mathbf{u}})$. Since we use Boltzmann policy to compute the action probability, for one agent $a \in A$ with the action $\bar{u}_a$, its individual policy is:

$$\pi_k^a = \frac{e^{\mathbb{E}_{u_{-a}^j \sim \mu} Q(s, \bar{u}_a, u_{-a}^j)}}{\sum_{i=1}^{m_a} e^{\mathbb{E}_{u_{-a}^j \sim \mu} Q(s, u_a^i, u_{-a}^j)}} = \frac{e^{\sum_{-a} \sum_j \mu Q(s, \bar{u}_a, u_{-a}^j)}}{\sum_{i=1}^{m_a} e^{\sum_{-a} \sum_j \mu Q(s, u_a^i, u_{-a}^j)}}, \quad (6)$$

where $-a$ represents all the agents except for target agent a, and μ is short for $\mu(s, u_1^{i_1}, \ldots, u_n^{i_n})$ representing the data distribution.

Under the general environmental setting, the state s will be fixed at each iteration, and the size of the action value function space $\boldsymbol{Q}$ is $\prod_{a=1}^{n} m_a$ with the dimension of n. Let the following function denotes the joint action probability:

$$f \stackrel{\text{def}}{=} 1 + \sum_{i=1}^{n} \prod_{\substack{j=1 \\ j\neq i}}^{n} \pi_k^j - n \prod_{i=1}^{n} \pi_k^i, \quad (7)$$

and we will provide another theorem indicating the conditions where we can acquire the maximum value of joint probability f in Equation (3).

Theorem 2 (Maximum probability conditions). *Considering a selected action value $\bar{Q}$ with action combination $\bar{\mathbf{u}}$ of the step k, the joint action probability function reaches its maximum $f_{\max}$ if and only if the value of each action probability π_k^i is on the boundary (i.e., either 0 or 1) as well as at least one probability π_k^a equals to 0.*

Proof. See Appendix C. □

Based on Theorem 2, a higher joint action probability will be assigned to $\bar{Q}$ whose agents' action probabilities are on the boundary of the interval $[0, 1]$ and at least one of the agents have its probability equaling 0. This conclusion casts light on determining the approximated sampling weights for MARL tasks. To better illustrate such an idea, we introduce several special case studies with respect to the selected $\bar{Q}$ at step k.

Case 1: single large value $\bar{Q}$. In this case, we assume only one action value $\bar{Q}$ out of the action value function space $\boldsymbol{Q}$ is large, and values Q_i of other action combinations elsewhere are negligibly small, represented by v. These small values obey $v \approx 0$ and $v \ll Q_m$. Therefore, according to Equation (6) and Theorem 2, the joint action probability for the selected action combination of $\bar{Q}$ is lower since the action probability π_k^a for each agent is similarly large. In contrast, the action combinations with only one action difference (e.g., $Q(s, \bar{\mathbf{u}}_{-a_1}, u_{a_1}^i)$) will be given with high weights α_h. The remaining position, such as the action combinations with two or more different actions, along with $\bar{Q}$, will be assigned with low weight α_l.

Case 2: dual large values $\bar{Q}, \bar{Q}'$. We propose only two large values $\bar{Q} \approx \bar{Q}'$ under this setting. Other positions are filled with negligible value v. $\bar{Q}'$ is the very same as $\bar{Q}$ except that one agent's action is different, i.e., $Q_m = Q(s, \bar{\mathbf{u}})$ and $\bar{Q}' = Q(s, \bar{\mathbf{u}}_{-a_1}, u_{a_1}^i)$. Since two large values equally share the importance over the action value space, based on given equations/conditions, we can drive those action combinations with only one different agent's action other than agent a will be assigned with medium weights α_m, e.g., $Q(s, \bar{\mathbf{u}}_{-(a_1,a_2)}, u_{a_1}^i, u_{a_2}^j)$. The positions where action combination with one action difference over agent a_1 will receive high weights α_h, e.g., $Q(s, \bar{\mathbf{u}}_{-a_1}, u_{a_1}^{i'})$. Besides, we will give low weight α_l for other locations.

Case 3: isolated large value $\bar{Q}''$ and $\bar{Q}$. Apart from given $\bar{Q}$, we assume an isolated large value with two or more actions different from $\bar{Q}$, i.e., $\bar{Q}'' = Q(s, \bar{\mathbf{u}}_{-(a_1,a_2)}, u_{a_1}^i, u_{a_2}^j)$, which exists somewhere in the action value function space, and satisfies $\bar{Q} \approx \bar{Q}''$. Other action values are v. In this situation, both two large values share the same importance, and we will assign the medium weight α_m to $\bar{Q}$, $\bar{Q}''$, and the action combinations having one action different over agent a_1 or a_2, such as $Q(s, \bar{\mathbf{u}}_{-(a_1,a_2)}, u_{a_1}^{i'}, u_{a_2}^j)$. The rest of the action combination values will be allocated with low weight α_l. This special case demonstrates that if one or more action values

are extraordinarily large, indicating another joint policy candidate with latent high joint action probability, we should also heed such equivalent competitor and its local search.

We can establish the approximation structure by studying from mentioned special cases. The scaled weights α_l, α_m, and α_h provide an alternative solution that spares us from directly using the numerically computed sampling weights to solve the latent computational cost, yet the performance remains mainly impervious.

4.4 Proposed Algorithms

Our analytical results in Theorem 1 identify four key factors determining the optimal projection weights. The first term, relating to the Bellman error, recovers the designs in classic prioritized experience replay. Specifically, when the Bellman error of a particular transition is high, which indicates a wide hindsight gap between Q_k and the Bellman target, we may consider assigning a larger weight to this transition. Besides, the value enhancement term selectively emphasizes the importance of incoming transitions. Based on the difference between current Q_k and ideal Q^*, it will compensate the near-optimal Q_k with larger importance while penalizing non-optimal Q_k with a smaller weighting modifier. Moreover, similar to previous studies, the measurement of on-policy transitions in the weighting expression underlines the useful information carried by more current, on-policy transitions.

Our analysis also identifies a new term reflecting the interaction among agents in the MARL scenario: the joint action probability of multiple agents, which is crucial in obtaining optimal sampling weights for specific transitions. We interpret the joint action probability term in optimal weights constrained by the given condition: one agent's action probability is small in the transition, while all other agents' action probabilities are large. We increase the weights for transitions satisfying this condition. On the contrary, we decrease the weight if the condition fails to be satisfied.

Following these theoretical results, we propose a MARL algorithm for collective priority optimization, MAC-PO, with regret-minimizing joint policy in multi-agent environments. We consider a new loss function with respect to the optimal sampling weights w_k applied to the Bellman equation of, i.e.,

$$L_{\text{MAC-PO}} = \sum_{i=1}^{b} w_k(s, \mathbf{u})(Q_k - y_i)^2(s, \mathbf{u}), \tag{8}$$

where b is the batch size, and $y_i = \mathcal{B}^* Q_{k-1}$ is a fixed target that can be obtained through a target network.

The Bellman error and joint action probability of all agents in the environment can be directly computed using Theorem 1. To compute the sampling weights for value enhancement term in practice, we use the backbone of the classical value factorization MARL algorithm QMIX and leverage the unrestricted joint action-value function Q^* to compute the approximated optimal action-value function quantitatively. Ideally, we could have included measurement of on-policy transitions term in the computation, but it is not readily available since the distribution $d^{\boldsymbol{\pi}_k}(s, \mathbf{u})$ in the numerator cannot be directly obtained. It is also worth mentioning that such term can be dismissed and the other terms in the weight expression are enough to provide a good estimate and lead to performance improvements, as shown in existing work [27]. Furthermore, based on our previous discussion, we designed an approximated counterpart, MAC-PO Approximation, by setting the threshold and scaling sampling weights values into low, medium, and high ones. The pseudo-codes are provided in Appendix D.

5 EXPERIMENTS

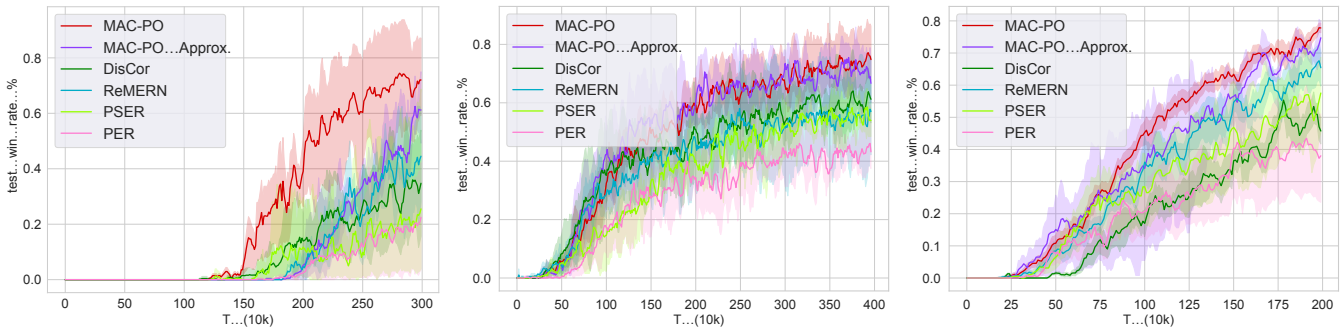
In this section, we present our experimental results on Predator-Prey and SMAC benchmarks and demonstrate the effectiveness of MAC-PO by comparing the results with several state-of-the-art MARL baselines. Additionally, we compare MAC-PO and MAC-PO Approximation with other experience replay methods adapted from single-agent RL to multi-agent environments. Each comparison is implemented independently with fixed and optimized [35] hyperparameters. We also conduct the ablation experiments to discuss the contribution of each term mentioned in Theorem 1. More implementation details are provided in Appendix E. The code has been made available at: https://github.com/ysmei97/MAC-PO.

5.1 Comparison with Existing Experience Replay Methods

In this experiment, we compare MAC-PO with other experience replay methods in the multi-agent environment SMAC. Since existing experience replay methods are designed for the single-agent scenario, we borrow their core designs and transplant them to multi-agent environments by considering all agents as the conceptual agent to match the single-agent target in their original settings. Such transplanting will recover the most important ingredients from RL to MARL to the greatest extent. It is worth mentioning that many other algorithms are also introducing a variety of experience replay schemes. Some of them [41, 53] depend on new components, and others [45] have different algorithm architectures. Since the backbone MARL algorithm of our choice in this experiment is QMIX, we do not expect a significant change over the algorithm architecture (e.g., actor-network) or major components (e.g., loss structure) as presented in other approaches to realize a relatively fair comparison.

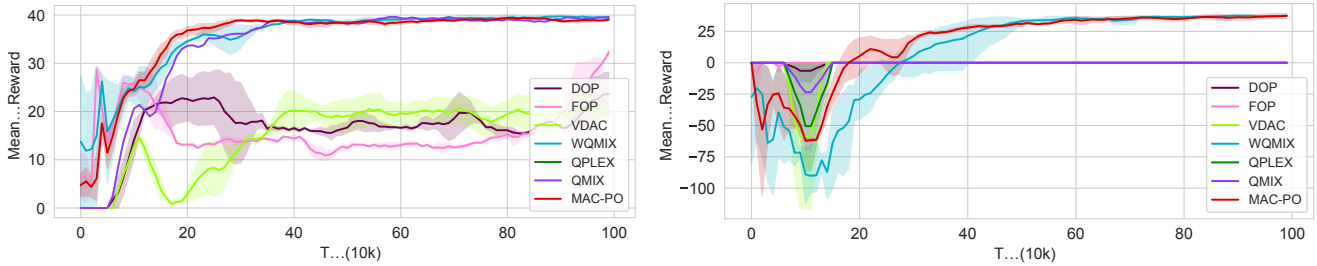
The first approach for comparing is PER [47]. Due to the equivalence between loss functions and non-uniform sampling for experience replay [13], we reconstruct PER scheme by computing weights only related to the current TD error regarding the joint action-value function. We also compare our method with the one mentioned in DisCor [27], where the weights are calculated from the production of Bellman error and value enhancement terms, and ReMERN [31], which has an additional term describing action likelihood, and we extend it to the multi-agent case. Besides, we transplanted the mechanism from PSER [6], which is another extension of PER, by applying an additional decay factor and window size on the weights for coming transitions. For this experiment, we set the decay factor as 0.4 and the window size as 5.

Figure 1 shows the performance comparison among MAC-PO and other experience replay algorithms on three maps of SMAC benchmark, which are *3s_vs_5z*, *5m_vs_6m*, and *MMM2*. Compared to MAC-PO, other experience replay schemes underperform in improving learning performance. DisCor and ReMERN have higher final winning rates than the regular PER, demonstrating the effectiveness of additional terms. PSER also acts better than PER



(a) 3s_vs_5z (hard) **(b) 5m_vs_6m (super hard)** **(c) MMM2 (super hard)**

Figure 1: Comparison between MAC-PO, MAC-PO Approximation and other experience replay methods on three SMAC maps (from hard to super hard), where MAC-PO outperforms the second best one – MAC-PO Approximation by 10%, 6%, and 4% on each map, respectively.



(a) No punishment **(b) Punishment = −1.5**

Figure 2: Average reward per episode on the Predator-Prey tasks for MAC-PO and other MARL algorithms of two settings, where MAC-PO shows the profoundly better convergence speed and pretty good results.

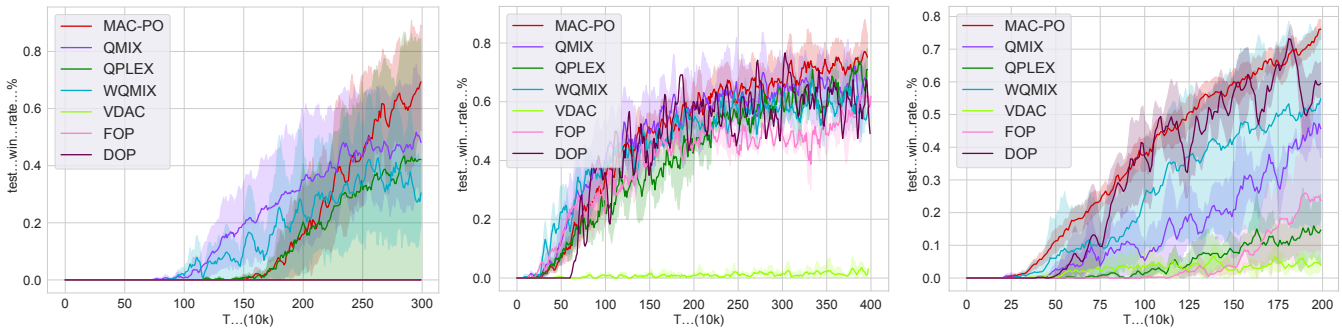
owing to its decaying mechanism for selecting history transitions. All transplanted algorithms from single-agent scenario act unstably in the multi-agent environment, as we can notice the variance reflected by the shaded area in Figure 1.

Besides, we also test our approximated sampling weight approach, shown as MAC-PO Approximation in Figure 1. We set the higher weight α_h as 0.75, medium weight α_m as 0.5, and lower weight α_l as 0.25. The final result is almost identical to the original MAC-PO with small nuance. For the original MAC-PO, the computational complexity of obtaining sampling weights will increase if more agents get involved. Since the approximated MAC-PO uses scaled weights instead of numerical results, it will improve the computational efficiency of the original MAC-PO at the price of slightly sacrificing the overall performance.

5.2 Comparison with MARL Algorithms

5.2.1 Predator-Prey. We compare MAC-PO with MARL algorithms on a complex partially-observable multi-agent cooperative environment, Predator-Prey, that involves eight agents in cooperation as predators to catch eight prey on a 10×10 grid. In this task, a successful capture with the positive reward of 1 must include two or more predator agents surrounding and catching the same prey simultaneously, requiring a high level of cooperation. A failed coordination between agents to capture the prey, which happens when only one predator catches the prey, will receive a negative punishment reward. We select multiple state-of-the-art MARL algorithms for comparison, which include value-based factorization MARL algorithm (i.e., QMIX, WQMIX [43], and QPLEX), decomposed policy gradient method (i.e., VDAC [51]), and decomposed actor-critic approaches (i.e., FOP [66] and DOP [60]). All mentioned baselines have shown strength in handling MARL tasks in existing works.

Figure 2 shows the performance of seven algorithms with different punishments, where all results show the effectiveness of MAC-PO. Besides, regarding efficiency, we can spot that MAC-PO has the fastest convergence speed in seeking the best policy. In Figures 2b, MAC-PO significantly outperforms other state-of-the-art algorithms in a hard setting requiring a higher level of coordination among agents as learning the best policy. Most MARL algorithms learn a sub-optimal policy where agents learn to work together with limited coordination. Although the performance of MAC-PO and WQMIX are similar, compared to the latter, MAC-PO converges to the optimal policy profoundly faster, demonstrating that our multi-agent optimal weighting scheme can efficiently learn from specific existing transitions.



(a) 3s_vs_5z (hard) (b) 5m_vs_6m (hard) (c) MMM2 (super hard)

Figure 3: Comparison between MAC-PO and other MARL algorithms on three SMAC maps (from hard to super hard). MAC-PO achieves the best results with the optimal weighting scheme alone, outperforming the second best result by 11%, 4%, and 16% on each map, respectively.

5.2.2 SMAC. Next, we evaluate MAC-PO on the SMAC benchmark. We report the experiments on three maps consisting of two hard maps and one super-hard map. The selected baselines for this experiment are consistent with those in the Predator-Prey environment. The empirical results are provided in Figure 3, demonstrating that MAC-PO can effectively generate optimal weight transitions on SMAC for achieving a higher win rate, especially when the environment becomes substantially complicated and harder, such as *MMM2*. We can see that several state-of-the-art algorithms are brittle when significant exploration is undergoing without finding optimal sampling weights.

Specifically, MAC-PO performs well on hard maps, such as *3s_vs_5z*, the best policy found by our optimal weighting approach significantly outperforms the remaining baseline algorithms regarding winning rate. For super-hard map *MMM2*, MAC-PO, along with QMIX, WQMIX, and QPLEX, can learn a better policy than VDAC, DOP, and FOP. We achieve the highest winning rate by adopting our algorithm on *MMM2*, showing the superiority of the optimal weighting scheme in utilizing past transitions.

5.3 Ablation Experiments

For ablations, we conduct experiments by disabling one term (mentioned in Theorem 1) every trial to investigate their contribution to finding optimal sampling weights, respectively. The terms considered in these experiments are Bellman error, value enhancement, and joint action probability. Figure 4 shows the results on *MMM2*. Compared to the original result, missing any of the terms will be detrimental to the performance, and the tests without joint action probability have the lowest final winning rate, which is around 60%. Such a phenomenon demonstrates that the interaction among agents is the critical factor in MARL tasks. The designing of the optimal weighting scheme without taking joint action probability into account will be less capable of achieving ideal results. Furthermore, the contributions of Bellman error and value enhancement terms are similar according to the given trend in Figure 4.

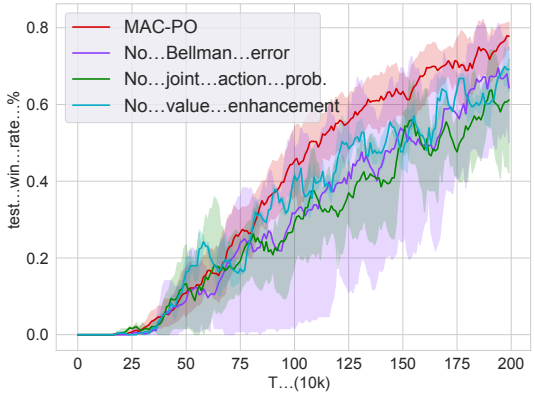


Figure 4: Ablations by disabling one term each for MAC-PO on MMM2 (super hard). The final winning rates decrease by 18% for disabling the joint action probability term, 15% for disabling the Bellman error term, and 10% for disabling the value enhancement term.

6 CONCLUSION

In this paper, we formulate multi-agent experience replay as a regret minimization problem and solve the optimal sampling weights in close form. The theoretical results illustrate key ingredients for an optimal experience replay in MARL settings. The results enable us to propose MAC-PO (with both exact and approximated weights) as a new MARL experience replay algorithm with optimized experience replay weights. Our experiment results in multiple MARL environments show the effectiveness of MAC-PO by demonstrating superior convergence and empirical performance over other experience replay solutions (adapted from single-agent RL) as well as state-of-the-art MARL methods.

ACKNOWLEDGMENTS

This research is based on work supported by the National Science Foundation under grant CCF-2114415 and partially by research gifts from CISCO and Meta.

REFERENCES

[1] S Ahilan and P Dayan. 2021. Correcting Experience Replay for Multi-Agent Communication. In *Ninth International Conference on Learning Representations (ICLR 2021)*.

[2] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528* (2019).

[3] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. 2018. Distributed Distributional Deterministic Policy Gradients. In *International Conference on Learning Representations*.

[4] Dimitri P Bertsekas. 2014. *Constrained optimization and Lagrange multiplier methods*. Academic press.

[5] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. 2020. Deep coordination graphs. In *International Conference on Machine Learning*. PMLR, 980–991.

[6] Marc Brittain, Josh Bertram, Xuxi Yang, and Peng Wei. 2019. Prioritized sequence experience replay. *arXiv preprint arXiv:1905.12726* (2019).

[7] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2012. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2012), 427–438.

[8] Jacopo Castellini, Frans A Oliehoek, Rahul Savani, and Shimon Whiteson. 2019. The representational capacity of action-value networks for multi-agent reinforcement learning. *arXiv preprint arXiv:1902.07497* (2019).

[9] Brett Daley and Christopher Amato. 2019. Reconciling λ-returns with experience replay. *Advances in Neural Information Processing Systems* 32 (2019).

[10] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. 2015. The importance of experience replay database composition in deep reinforcement learning. In *Deep reinforcement learning workshop, NIPS*.

[11] Sheng Fan, Guanghua Song, Bowei Yang, and Xiaohong Jiang. 2020. Prioritized Experience Replay in Multi-Actor-Attention-Critic for Reinforcement Learning. In *Journal of Physics: Conference Series*, Vol. 1631. IOP Publishing, 012040.

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

[13] Scott Fujimoto, David Meger, and Doina Precup. 2020. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in neural information processing systems* 33 (2020), 14219–14230.

[14] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. 2021. KKT Conditions, First-Order and Second-Order Optimization, and Distributed Optimization: Tutorial and Survey. *arXiv preprint arXiv:2110.01858* (2021).

[15] Kailash Gogineni, Peng Wei, Tian Lan, and Guru Venkataramani. 2023. Scalability Bottlenecks in Multi-Agent Reinforcement Learning Systems. *arXiv preprint arXiv:2302.05007* (2023).

[16] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. 2002. Coordinated reinforcement learning. In *ICML*, Vol. 2. Citeseer, 227–234.

[17] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*. Springer, 66–83.

[18] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[19] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. 2018. Distributed Prioritized Experience Replay. In *International Conference on Learning Representations*.

[20] Yeping Hu, Alireza Nakhaei, Masayoshi Tomizuka, and Kikuo Fujimura. 2019. Interaction-aware decision making with adaptive strategies under merging scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 151–158.

[21] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. 2017. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011* (2017).

[22] Shariq Iqbal and Fei Sha. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 2961–2970.

[23] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. 2019. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*. PMLR, 3040–3049.

[24] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*. PMLR, 651–673.

[25] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[26] Steven George Krantz and Harold R Parks. 2002. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

[27] Aviral Kumar, Abhishek Gupta, and Sergey Levine. 2020. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems* 33 (2020), 18560–18572.

[28] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[29] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3 (1992), 293–321.

[30] Ruishan Liu and James Zou. 2018. The effects of memory replay in reinforcement learning. In *2018 56th annual allerton conference on communication, control, and computing (Allerton)*. IEEE, 478–485.

[31] Xu-Hui Liu, Zhenghai Xue, Jingcheng Pang, Shengyi Jiang, Feng Xu, and Yang Yu. 2021. Regret Minimization Experience Replay in Off-Policy Reinforcement Learning. *Advances in Neural Information Processing Systems* 34 (2021), 17604–17615.

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

[33] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib. 2012. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *Twenty-sixth AAAI conference on artificial intelligence*.

[34] Edward James McShane. 1937. Jensen's inequality. *Bull. Amer. Math. Soc.* 43, 8 (1937), 521–527.

[35] Yongsheng Mei, Tian Lan, Mahdi Imani, and Suresh Subramaniam. 2022. A Bayesian Optimization Framework for Finding Local Optima in Expensive Multi-Modal Functions. *arXiv preprint arXiv:2210.06635* (2022).

[36] Yongsheng Mei, Tian Lan, and Guru Venkataramani. 2023. Exploiting Partial Common Information Microstructure for Multi-Modal Brain Tumor Segmentation. *arXiv preprint arXiv:2302.02521* (2023).

[37] Yongsheng Mei, Hanhan Zhou, and Tian Lan. 2023. ReMIX: Regret Minimization for Monotonic Value Function Factorization in Multiagent Reinforcement Learning. *arXiv preprint arXiv:2302.05593* (2023).

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[39] Andrew W Moore and Christopher G Atkeson. 1993. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning* 13, 1 (1993), 103–130.

[40] Guido Novati and Petros Koumoutsakos. 2019. Remember and forget for experience replay. In *International Conference on Machine Learning*. PMLR, 4851–4860.

[41] Youngmin Oh, Jinwoo Shin, Eunho Yang, and Sung Ju Hwang. 2021. Model-augmented Prioritized Experience Replay. In *International Conference on Learning Representations*.

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

[43] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:2006.10800 [cs.LG]

[44] 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.

[45] Baturay Saglam, Furkan B Mutlu, Dogan C Cicek, and Suleyman S Kozat. 2022. Actor Prioritized Experience Replay. *arXiv preprint arXiv:2209.00532* (2022).

[46] 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).

[47] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *ICLR*.

[48] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[49] Samarth Sinha, Jiaming Song, Animesh Garg, and Stefano Ermon. 2022. Experience replay with likelihood-free importance weights. In *Learning for Dynamics and Control Conference*. PMLR, 110–123.

[50] 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.

[51] Jianyu Su, Stephen Adams, and Peter Beling. 2021. Value-decomposition multi-agent actor-critics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11352–11360.

[52] Kefan Su and Zongqing Lu. 2022. Divergence-regularized multi-agent actor-critic. In *International Conference on Machine Learning*. PMLR, 20580–20603.

[53] Shivakanth Sujit, Somjit Nath, Pedro HM Braga, and Samira Ebrahimi Kahou. 2022. Prioritizing Samples in Reinforcement Learning with Reducible Loss. *arXiv preprint arXiv:2208.10483* (2022).

[54] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

[55] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[56] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[57] Harm Van Seijen and Rich Sutton. 2013. Planning by prioritized sweeping with small backups. In *International Conference on Machine Learning*. PMLR, 361–369.

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

[59] 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).

[60] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2020. Dop: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*.

[61] Yishen Wang and Zongzhang Zhang. 2019. Experience selection in multi-agent deep reinforcement learning. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 864–870.

[62] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.

[63] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3 (1992), 279–292.

[64] Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. 2019. Experience Replay Optimization. In *IJCAI*.

[65] Shangtong Zhang and Richard S Sutton. 2017. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275* (2017).

[66] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. 2021. FOP: Factorizing Optimal Joint Policy of Maximum-Entropy Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. PMLR, 12491–12500.

[67] Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. 2022. PAC: Assisted Value Factorisation with Counterfactual Predictions in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2206.11420* (2022).

[68] Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. 2022. Value functions factorization with latent state information sharing in decentralized multi-agent policy gradients. *arXiv preprint arXiv:2201.01247* (2022).