IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 December 2024, accepted 13 January 2025, date of publication 17 January 2025, date of current version 23 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3530974

RESEARCH ARTICLE

Rethinking Exploration and Experience Exploitation in Value-Based Multi-Agent Reinforcement Learning

ANATOLII BORZILOV[1,2], ALEXEY SKRYNNIK[1,2,3], AND ALEKSANDR PANOV[1,2,3]
[1]Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 117312 Moscow, Russia
[2]Moscow Institute of Physics and Technology, 141701 Dolgoprudny, Russia
[3]AIRI, 121170 Moscow, Russia

Corresponding author: Anatolii Borzilov (borzilov.av@gmail.com)

This work was supported by the Ministry of Science and Higher Education of the Russian Federation under Project 075-15-2024-544.

ABSTRACT Cooperative Multi-Agent Reinforcement Learning (MARL) focuses on developing strategies to effectively train multiple agents to learn and adapt policies collaboratively. Despite being a relatively new area of research, most MARL methods are based on well-established approaches used in single-agent deep learning tasks due to their proven effectiveness. In this paper, we focus on the exploration problem inherent in many MARL algorithms. These algorithms often introduce new hyperparameters and incorporate auxiliary components, such as additional models, which complicate the adaptation process of the underlying RL algorithm to better fit multi-agent environments. We aim to optimize a deep MARL algorithm with minimal modifications to the well-known QMIX approach. Our investigation of the exploitation-exploration dilemma shows that the performance of state-of-the-art MARL algorithms can be matched by a simple modification of the ϵ-greedy policy. This modification depends on the ratio of available joint actions to the number of agents. We also improve the training aspect of the replay buffer to decorrelate experiences based on recurrent rollouts rather than episodes. The improved algorithm is not only easy to implement, but also aligns with state-of-the-art methods without adding significant complexity. Our approach outperforms existing algorithms in four of seven scenarios across three distinct environments while remaining competitive in the other three.

INDEX TERMS Exploration, multi-agent reinforcement learning, value based methods.

I. INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) is an emerging field in artificial intelligence that aims to develop robust strategies for training multiple agents to learn and adapt their policies collaboratively [1], [2]. MARL methods are designed for both antagonistic and cooperative problems. Both formulations require taking into account the actions of other agents, and most practical applications also work under conditions of partial observability, where information about the goals and actions of other agents is not fully known.

Notable examples of MARL success include its application to complex tasks such as autonomous driving, where multiple vehicles must coordinate to navigate, as demonstrated by the Nocturne framework [3]. Examples of important applications of the multi-agent paradigm include IMP-MARL, which provides a platform for evaluating the scalability of cooperative MARL methods responsible for scheduling inspections and repairs of specific system components to minimize maintenance costs [4]. Another example, MATE addresses target coverage control challenges in real-world scenarios by presenting an asymmetric cooperative-competitive game with two sets of learning agents, cameras, and targets, each with opposing goals [5]. These successes highlight MARL's potential for solving real-world problems that require coordinated action among multiple agents.

Despite these successes, even for tasks that are close to real-world applications, MARL algorithms are often

The associate editor coordinating the review of this manuscript and approving it for publication was Qiang Li.

TABLE 1. List of notations.

Notation	Description
S	Set of all the environment states
s	State of the environment
A	Set of all agents
a	Agent
n	Number of agents
U	Set of all actions
u^a	Action of an agent a
$\mathbf{u}$	Joint action of agents
$P(s'\|s,u)$	Transition function
$r(s,\mathbf{u})$	Reward function
γ	Discount factor
R_t	Discounted return at a step t
Z	Set of all individual observations
$O(s,a)$	Observation function
τ^a	Action-observation history of an agent a
$\boldsymbol{\tau}$	Joint action-observation of agents
$\pi^a(u^a\|\tau^a)$	Policy of an agent a
$Q^\pi(s_t,\mathbf{u}_t)$	Action-value function
θ	Network parameters
$Q_{tot}(\boldsymbol{\tau}_t,\mathbf{u}_t,\theta)$	Joint action-value function
$L(\theta)$	TD error
$\mathcal{D}$	Replay buffer
D	Replay buffer size
$\mathcal{B}$	Batch of transitions
B	Batch size

tested in game-like environments. For example, the StarCraft Multi-agent Challenge (SMAC) and SMACv2 simulators are based on the strategy game StarCraft II,[1] in which teams of agents work together to defeat opposing groups [6], [7]. Similarly, research in Google Football demonstrates the applicability of MARL to complex, dynamic tasks [8]. Although relatively new, MARL methods often leverage basic techniques from single-agent deep reinforcement learning tasks due to their proven success.

One of the key challenges in designing MARL algorithms is to account for stochasticity in the set of experiences from the environment on which the agent is trained. In addition to changes in the agent's own policy, which updates the distribution of observations it receives, the policies of other agents change dynamically. This leads to difficulties in adapting classical statistical learning methods, including neural networks, to MARL problems. Another core challenge in MARL is the exploration problem, which becomes increasingly difficult in complex scenarios that require sophisticated cooperation among multiple agents. Agents risk falling into local optima, preventing them from acquiring the complex strategies needed to solve the problem. While the common approach to exploration in value-based MARL is the ϵ-greedy strategy [9], a variety of novel methods, such as MAVEN [10] and SMMAE [11], focus on coordinated exploration.

In our paper, we focus on the popular value-based method called QMIX [12]. This method has become the theoretical and technical foundation for many MARL approaches [11], [13], [14], [15], [16]. First, we investigate how the exploration strategy of value-based methods can be improved and propose a simple approach based on the number of available joint actions. Second, we examine the implementation of experience exploitation, which is often not well-articulated in the literature. We address how to effectively sample data from the replay buffer for methods using recurrent networks. These networks are vital in MARL, as they allow agents to retain historical information essential for decision-making in partially observable environments typical of multi-agent systems.

To summarize, we make the following **contributions**:

Contribution 1: We propose a novel exploration strategy for value-based methods, which leverages the number of available joint actions, improving the exploration-exploitation trade-off.

Contribution 2: We analyze the under-explored area of experience exploitation in value-based MARL methods, specifically focusing on how to effectively sample data from the replay buffer when recurrent networks are used.

Contribution 3: We extensively study our proposed modifications on two benchmarks, SMAC and POGEMA, demonstrating that our approach achieves comparable or even superior results to state-of-the-art value-based MARL methods.

The paper is organized as follows: Section II outlines the background of multi-agent reinforcement learning field, Section III provides a review of related literature, Section IV describes the methodology, and Sections V, VI details the experimental setup and results.

II. BACKGROUND

This paper addresses the problem of multi-agent cooperative tasks, formalized as decentralized partially observable Markov decision process (Dec-POMDP) tuple $G = \langle S, A, U, P, r, Z, O, n, \gamma \rangle$. State $s \in S$ describes the complete state of the environment at the moment. At each timestep each agent $a \in A \equiv 1, \ldots, n$ chooses an action $u^a \in U$; chosen actions of all agents form a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. These actions result in environment transition to a new state according to the transition function $P(s'|s,u) : S \times U \times S \to [0,1]$, at each timestep t. The rewards are given according to the reward function $r(s,\mathbf{u}) : S \times S \to \mathbf{R}$, which is shared by all agents, and $\gamma \in [0,1)$ is a discount factor.

At each timestep, each agent a receives an individual observation $z^a \in Z$ according to the observation function $O(s,a) : S \times A \to Z$. Each agent maintains an action-observation history $\tau^a \in T \equiv (Z \times U)^*$, on which the agent's policy $\pi^a(u^a|\tau^a) : T \times U \to [0,1]$ is conditioned. The joint policy π associated with an action-value function $Q^\pi(s_t,\mathbf{u}_t) = E_{s_{t+1:\infty},\mathbf{u}_{t+1:\infty}}[R_t|s_t,\mathbf{u}_t]$, where $R_t = \sum_{i=0}^{\infty}\gamma^i r_{t+i}$ represents the discounted return. The training objective is to find the optimal action-value function.

DQN [17] is a popular algorithm for single-agent tasks, which learns agent's action-value function. For multi-agent tasks, we learn the joint action-value function $Q_{tot}(\boldsymbol{\tau}_t,\mathbf{u}_t,\theta)$, where $\boldsymbol{\tau} \in \mathbf{T}$ is a joint action-observation history and θ are

[1] StarCraft and StarCraft II are trademarks of Blizzard Entertainment™.

network parameters. During learning, the replay buffer $\mathcal{D}$, consisted of tuples $(\boldsymbol{\tau}_t, \mathbf{u}_t, r_t, \boldsymbol{\tau}_{t+1})$, is utilized. Thus, the network parameters θ will be learned by the TD error:

$$L(\theta) = \mathbf{E}_{(\boldsymbol{\tau}_t, \mathbf{u}_t, r_t, \boldsymbol{\tau}_{t+1}) \sim D} \left[r + \gamma \max_{\mathbf{u}_{t+1}} Q_{tot}(\boldsymbol{\tau}_{t+1}, \mathbf{a}_{t+1}, \theta^-) \right]^2, \tag{1}$$

where θ^- are the parameters of the target network that are periodically updated with θ.

One of the core issues of cooperative MARL is that simultaneous learning of multiple agents induces non-stationarity of the environment. That leads to the problem that decentralized learning of multiple agents is unstable. To address this issue and improve training stability under non-stationary conditions, the centralized training with decentralized execution (CTDE) paradigm was introduced. According to this approach, the execution is decentralized, which means that each agent a chooses actions according to its local action-observation history τ^a. Despite that, the training is centralized, and during training the learning algorithm has the access to the state s of the environment. In order to allow each agent a to participate in a decentralized execution, it is important to assert that:

$$\arg\max_{\mathbf{u}} Q^{\pi}(s, \mathbf{u}) = (\arg\max_{u^1} Q_1(\tau^1, u^1) \ldots \arg\max_{u^n} Q_n(\tau^n, u^n)) \tag{2}$$

One of the popular methods to solve the problem is QMIX [12]. This method is a variant of DQN [17] for multiagent tasks, and is based on the ideas of VDN [18]. For each agent, QMIX uses a DRQN network to calculate the individual value function $Q_a(\tau^a, u^a)$. These networks receive the current observation as input at each timestep. QMIX then employs a mixing network, which takes the outputs of the agents' networks as input and produces the total value $Q_{tot}(\boldsymbol{\tau}, \mathbf{u})$. The weights of the mixing network are generated by a hypernetwork based on the current state and are non-negative. The use of non-negative weights in the mixing network ensures that $\frac{\partial Q_{tot}}{\partial Q_a} >= 0, \forall a \in A$, which in turn guarantees condition (2).

III. RELATED WORK

The exploration-exploitation dilemma in MARL is closely related to similar challenges in deep RL. Many techniques originally developed for single-agent settings have been adapted for MARL. For instance, curiosity-driven exploration, a method that enhances the exploration process, has been effectively integrated into MARL to manage complexities arising from multiple interacting agents. Additionally, tools such as replay buffers and recurrent neural networks are employed to better handle and utilize data collected during agent interactions. However, while these adaptations improve exploration and data utilization, they do not directly address the non-stationarity problem inherent in MARL. Below, we provide an overview of such techniques in single-agent RL and their application in MARL, highlighting their strengths and limitations in the multi-agent context.

A. EXPLORATION IN MARL

The exploration problem is a well-studied topic in reinforcement learning. Bootstrapped DQN [19] learns several separate Q-value functions, and at the beginning of each episode samples one of these functions. Then, the agent follows the greedy policy for that sampled function. This way, the method allows the agent to use temporally-extended exploration during the whole episode. ϵz-greedy [20] modifies the ϵ-greedy method, and instead of sampling single actions, it samples options of actions, which agent follows for the number of steps that is sampled according to the distribution z. Another approach is to use an intrinsic reward to direct the exploration process. ICM [21] adopts an inverse model to extract features out of inputs that ignore uncontrollable aspects of the environment, and then uses the prediction error of these features as an intrinsic reward. VIME [22] uses Bayesian neural networks to approximate environment dynamics and then maximizes the information gain about the agent's belief of environment dynamics. VDM [23] models the stochasticity in dynamics to enhance predictions and computes the intrinsic reward using the environmental state-action transitions probabilities. RND [24] computes the exploration bonus based on state novelty, which is estimated by distilling a fixed randomly initialized network into another one. State marginal matching technique [25] (SMM) learns a policy to match its state marginal distribution with a target state distribution. NGU [26] computes intrinsic reward based on two components: exploration bonus for lifelong novelty, which is computed using RND, and episodic novelty bonus. To compute the episodic novelty bonus, it uses episodic memory, which contains all the visited states in the current episode. Then, it encourages the agent to visit as many different states as possible during a single episode. Agent57 [27], being based on NGU, also learns a family of policies with different degrees of exploration and exploitation. It uses an adaptive meta-controller to choose from these policies, which allows to control the intensity of exploration during the training process. Instead of exploring novel states, SMiRL [28] tries to minimize a surprise from new states, thus developing behavior that decreases entropy. Such an approach allows the learning agent to develop meaningful skills in unstable environments, where unexpected events happen on their own.

Generally, multi-agent methods try to adopt existing single-agent approaches for exploration. LIIR [29] uses an individual intrinsic reward for each agent, which allows the agents to be stimulated differently. The parameters of intrinsic rewards are learned using the centralized critic to maximize the team reward. EMC [30] utilizes a curiosity module, which is trained to predict individual Q-values of agents. These prediction errors are used as additional intrinsic rewards. Wang et al. [31] introduce two methods that are

based on measuring of the interactions between agents to compute intrinsic rewards: EITI uses the mutual information between agents' trajectories, and EDTI quantifies the influence of an agent on expected returns of other agents. MAVEN [10] uses a latent variable, which is generated by a hierarchical policy, to perform coordinated exploration in different modes. Then, MAVEN maximizes the mutual information between the observed trajectories to achieve diverse behavior.

CMAE [32] utilizes restricted space exploration and shared goals. It first explores goals from a low-dimensional restricted space and then trains exploration policies to reach these goals, which represent under-explored states. This method showed significant improvement in sparse-reward environments. SMMAE [11] enhances exploration in two different ways. Firstly, it introduces an intrinsic reward based on SMM. Secondly, it uses adaptive exploration, and bases each agent's probabilities of choosing random actions on correlation between agents. It predicts the actions of each agent based on other agents' observations to measure the correlation between them and increases the probability of choosing random actions if the correlation is too high.

B. EXPERIENCE EXPLOITATION IN MARL

While classic algorithms during training process replay whole episode sequences, it may create number of practical issues because of varying episode length and correlated states in trajectories. To solve this issue, R2D2 [33] trains on sequences of transitions of fixed length, which overlap by half of their length and never cross boundaries of the episode. Though that method allows to overcome some issues, which are created by learning on the whole episodes, it also creates an issue of necessity to properly initialize hidden recurrent states during training. R2D2 introduces two strategies to solve this issue: storing the recurrent state in replay buffer, and using ''burn-in'' phase during training, i.e. using the first half of the training sequence only for initialization of the recurrent states, and apply the training objective to the second half of the sequence.

Number of methods also utilizes prioritized experience replay (PER) [34]. Ape-X [35] suggests to use the absolute TD error for experience priorities. R2D2 [33] and R2D3 [36] use the mixture of the maximum absolute TD error and the mean TD error in the sequence.

Most works focused on modifying experience replay consider the single-agent domain, though some adopt this concept for multi-agent tasks. MAC-PO [37] uses weights for weighted error and samples training transitions with a uniform distribution. Reference [38] adopts PER for multi-agent tasks, but without recurrence, setting priorities for each transition. Number of methods, like QMIX [12] and its derivatives [10], [11], [13], sample for training uniformly full episodes. To the best of our knowledge, there are no works that consider modifying experience replay to use fixed-length sequences for training MARL approaches, with or without prioritization.

IV. METHOD

This section outlines the key methodological advancements introduced in our research to enhance exploration and training efficiency in reinforcement learning. The scheme of the proposed approach is sketched in Figure 1. We first present a novel modification to the traditional ϵ-greedy policy, where the exploration probability is dynamically adjusted based on the count of available joint actions.

Following this, we describe our enhanced replay buffer strategy designed to improve the training process's efficiency and stability. Instead of relying on full episodes, we utilize overlapping sequences of fixed length, allowing for more effective learning across episode boundaries.

A. MODIFICATION OF ϵ-GREEDY POLICY

As ϵ-greedy policy uses a constant value of a probability of choosing random actions ϵ, it may be hard to adapt the exploration degree to the current environment situation. The number of available to agents actions may vary in some environments, and the number of agents may change during the episode as well. The varying number of the available joint actions leads to the necessity of dynamically adaptation of the exploration extent in order to properly explore environment states.

In contrast to ϵ-greedy policy, where a constant value of ϵ is used, we compute the exploration probability ϵ using the available actions count. We assume that the more joint actions $\mathbf{U}$ are available, the more intense exploration it is required to find the optimal strategy. According to that reasoning, we introduce the following way to compute the value of ϵ_t:

$$\epsilon_t = tanh(\alpha \cdot \sqrt{log(|U_t|)}) \tag{3}$$

where $|\mathrm{U}_t|$ is a count of the available joint actions at the step t; α is a constant hyperparameter. Also, we set minimum and maximum boundaries ϵ_{min} and ϵ_{max}, so that ϵ_t would always stay in reasonable limits.

Scaling the value of ϵ_t on available joint actions count may allow adapt exploration intensity to the current state. In some environments, like SMAC [6], number of actions, available to agents, may greatly vary.

B. REPLAY BUFFER ENHANCEMENT

Replay buffer $\mathcal{D}$ consists of a fixed number of episodes' steps D. Instead of training on full episodes, we train using sequences of steps of fixed length m, in order to decrease the dependency of the learning process on the episodes length. These sequences aren't restricted by episodes boundaries and may contain steps of different episodes. At the beginning of each train iteration, recurrent state is initialized to zero. The first half of each sequence is used for the initialization of the recurrent state, and the training objective is only applied to the second half of a sequence. If a sequence contains parts of different episodes, at the step of switching between episodes the recurrent state is zero initialized again. This approach helps decorrelate experiences during training, resulting in more diverse and representative training data. As a result,

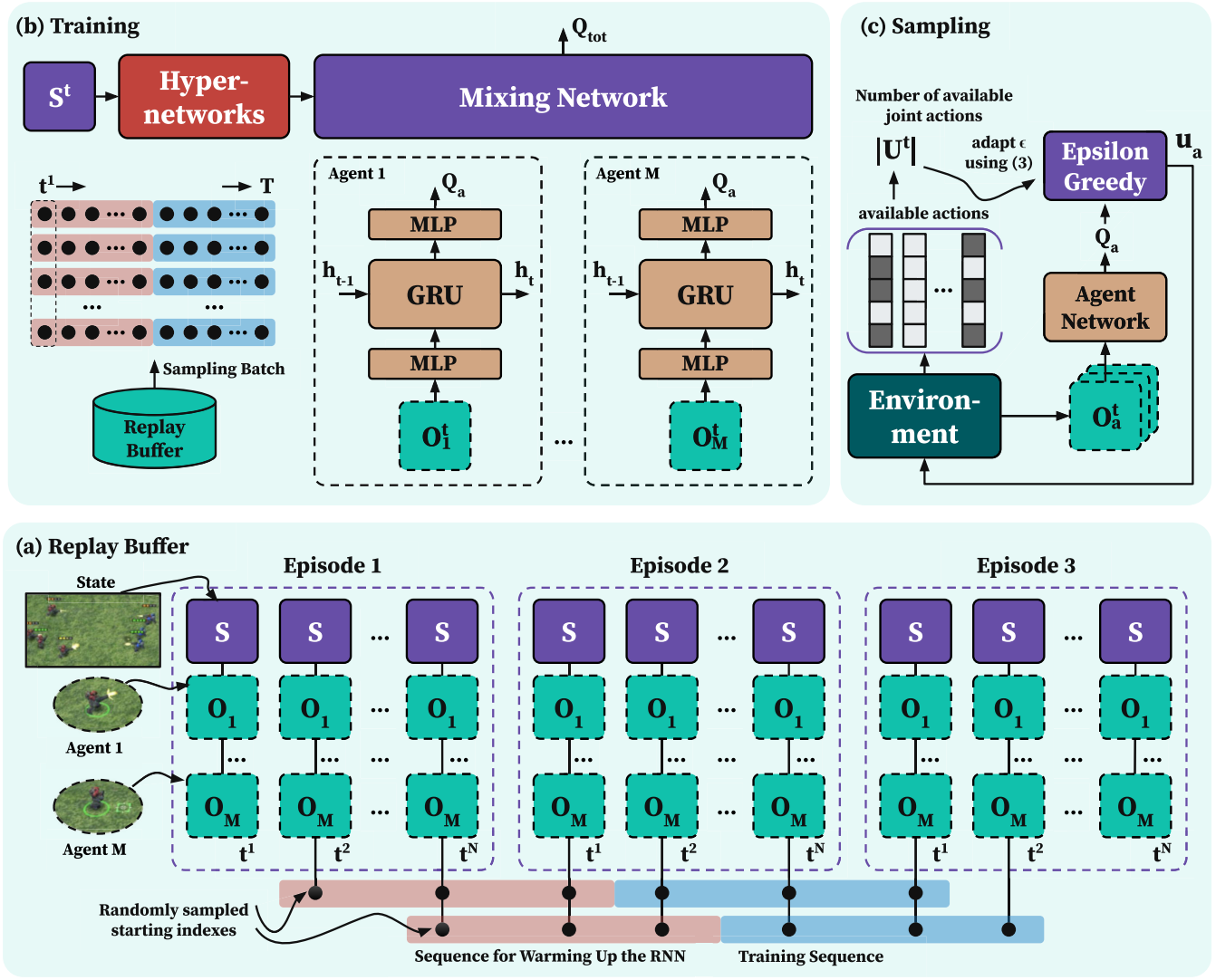

FIGURE 1. The scheme of the modified QMIX approach presents three alternating phases of learning:(a) This phase highlights how data is presented in the replay buffer. For each state, the tuple consists of agents' observations. For clarity, additional information stored in the replay buffer-such as rewards, episode termination and truncation flags, selected actions, and action masking-is excluded. Unlike the original QMIX approach, our modifications involve sampling data slices across multiple episodes (rather than training on entire episodes). This forms a batch of a specific length, consisting of two parts: a warming-up sequence (used to better initialize the recurrent network's hidden state h_t) and a training sequence (used for actual training). (b) A batch of fixed size is sampled from the replay buffer and used to train the joint value function Q_{tot} via hyper-networks through the mixing network. This process also optimizes the agent networks using the global environment state, following the CTDE paradigm. Each agent has its own observation, but all agents share the same network weights. (c) This phase describes how the Replay Buffer is filled with new experiences. It also highlights our extension to the ϵ-greedy policy, where ϵ is adjusted based on the number of available joint actions $|U^t|$ for the entire team population, as described in equation (3).

correlations in the training data are mitigated, improving the stability and generalization of the learning process. Also, to decrease dependency on the environment, we run train iterations after the constant number of rollout time-steps performed.

The process of insertion of data in replay buffer is described in Algorithm 1. As we sample a new episode of length T, we need to store it in replay buffer $\mathcal{D}$. The maximum amount of transitions that we store in $\mathcal{D}$ is D, so, if the size of $\mathcal{D}$ exceeds that limit, we remove the oldest transitions.

We sample training sequences following the Algorithm 2. As we train the network on batches of size B, we uniformly choose B starting indices. Then, for each sampled index i we put in batch $\mathcal{B}$ a sequence which consists of m transitions starting from index i up to index $i + m$. If the value $i + m$ exceeds the capacity of $\mathcal{D}$, we select in the sequence transitions from i up to the last transition stored in $\mathcal{D}$, and select the rest of transitions starting from the index 0 to fill the sequence up to the size m.

V. EXPERIMENTAL SETUP

In this section, we present the detailed description of the environments used for the evaluation of the proposed modifications and overall setup description. We consider three environments: SMAC [6], SMACv2 [7] and POGEMA [39], [40].

Algorithm 1 Inserting Transitions in Replay Buffer

Input: List of transitions $\mathcal{D}$, buffer size D
Output: List of transitions $\mathcal{D}$

Sample transition tuples $\rho \leftarrow \left\{\left(s_t, r_t \left\{\left(z_t^a, u_t^a, z_{t+1}^a\right) | a = 1, \ldots, n\right\}\right) | t = 0, \ldots, T-1\right\}$
for *each step* $t = 0, \ldots, T-1$ **do**
  if $size(\mathcal{D}) = D$ **then**
    $\mathcal{D} \leftarrow \mathcal{D}[1:]$ `// Pop oldest index`
  end
  $\mathcal{D} \leftarrow \text{concat}(\mathcal{D}, \rho_t)$
end

Algorithm 2 Sample Transitions From Replay Buffer

Input: List of transitions $\mathcal{D}$, sequence size m, batch size B
Output: Batch of transitions $\mathcal{B}$

$\mathcal{B} \leftarrow ()$ `// Initialize batch as an empty list`
while $size(\mathcal{B}) < B$ **do**
  $i \sim \mathcal{U}(0, \text{size}(\mathcal{D}) - 1)$ `// Randomly sample starting index of a sequence`
  if $i + m < size(\mathcal{D})$ **then**
    $b \leftarrow \mathcal{D}[i : i + m]$
  else
    $b \leftarrow \text{concat}(\mathcal{D}[i:], \mathcal{D}[: \text{size}(\mathcal{D}) - i])$
  end
  $\mathcal{B} \leftarrow \text{concat}(\mathcal{B}, b)$
end

We use the QMIX [12] algorithm as the base method for the proposed modifications. Our implementation is based on PyMARL [6]. Following SMMAE [11], we changed RMSProp [41] optimizer with Adam [42] optimizer with default hyper-parameters.

At the beginning of training, we linearly anneal ϵ from 1.0 to 0.05 over 50,000 steps. For methods without the replay buffer modification, the buffer size is set to 5,000 episodes, and after sampling a new episode from the environment, we select a batch of 32 episodes for training. For methods with the replay buffer modification, for every 128 steps sampled from the environment, we select a batch of 64 sequences for training. Each sequence is 128 steps long, with 64 steps used for the "burn-in" phase and the remaining 64 steps used for training. In the POGEMA environment, the replay buffer size is set to 1,000 episodes or 800,000 steps. The agents' network architecture is the same as that of QMIX, with a GRU [43] recurrent layer having a 64-dimensional hidden state. The target network is updated every 200 training episodes for methods without the replay buffer modification, and every 10,000 steps for methods with the replay buffer modification. The complete hyper-parameters setup is shown in Table 2.

TABLE 2. The hyper-parameters of proposed modifications and base version of QMIX algorithm.

Hyper-parameter	Value
Sequence length	128 steps
Length of a burn-in phase	64 steps
ϵ anneal time	50000 steps
ϵ finish	0.05
ϵ_{min}	0.05
ϵ_{max}	0.1
Learning rate	0.001
α	0.04
γ	0.99
GRU hidden size	64
Mixing network size	32
Mixer's hypernet layers	2
Mixer's hypernet hidden size	64
Optimizer	Adam
Improved Replay Buffer (w/ RB)	
Size for SMAC	200000 steps
Size for POGEMA	800000 steps
Target update interval	10000 steps
Default Replay Buffer (w/o RB)	
Size for SMAC	5000 episodes
Size for POGEMA	1000 episodes
Target update interval	200 episodes

The hyperparameters of QMIX, SMMAE, QPLEX [13] and WQMIX [16] were selected based on their default implementations. Hyperparameters, specific for our approach, including the value of α, the ϵ_{min} and ϵ_{max}, the sequence length and the target update interval, were determined through preliminary experiments. We selected the value of α within the range of 0.01 to 0.05, the sequence length from 16 to 256, the target update interval from 2,500 to 40,000, ϵ_{min} was set to 0.05, and ϵ_{max} was chosen from a range of 0.1 to 0.5.

A. SMAC AND SMACv2 ENVIRONMENTS

SMAC [6] benchmark is focused on micromanagement task of the popular game StarCraft II, where each unit is controlled by a different agent in order to defeat the opponent army controlled by a game's build-in scripted AI. The game is considered won if agents managed to defeat every enemy unit within the time limit, and the quality metrics is win rate. Initially, following SMMAE [11], we selected three scenarios for the experiments on SMAC: *6h_vs_8z*, *2c_vs_64zg*, and *corridor*. However, for *6h_vs_8z*, we observed that the agent learned to exploit the reward system. The enemies had shields that regenerated over time, and under the standard settings in SMAC, agents were rewarded for regeneration of enemies' shields as if it were damage. As a result, it was more advantageous for agents to damage the shields and then retreat out of the enemies' line of sight, which lead to low win rate. As it was a known issue, which wasn't planned to be fixed,[2] we decided to replace the map

[2]https://github.com/oxwhirl/smac/issues/72

FIGURE 2. Screenshot examples of SMAC scenarios from the StarCraft II game: *(left)* Corridor; *(middle)* 2c_vs_64zg; *(right)* MMM2. In all scenarios, the red units are controlled by RL agents, while the blue units are controlled by the game's built-in AI. The RL agents are trained jointly during learning but make independent decisions during testing, following the centralized training, decentralized execution paradigm.In the *Corridor* scenario, the RL-controlled zealots must coordinate their movement towards the lower left corner of the map, where a narrow corridor is located. This positioning allows them to defeat a large number of zerg units. In the *2c_vs_64zg* scenario, the RL agents control two colossus, exploiting the units' ability to traverse high ground to defeat a large number of zergs. In the *MMM2* scenario, a group of 2 marauders, 7 Marines, and 1 Medivac, controlled by RL agents, attempt to defeat a larger group consisting of 3 marauders, 8 Marines, and 1 Medivac.

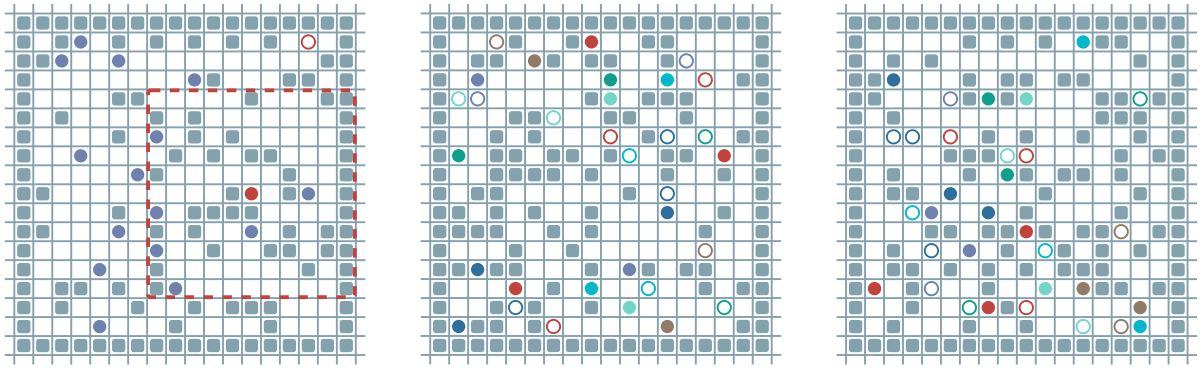

FIGURE 3. Illustration of a random POGEMA maps with a size of 16 × 16 and a population of 16 agents. The agents are represented as colored filled circles, and their targets are shown as circles of the same color. Each agent has a single, unique target. The first image shows the task from the agent's perspective, where the rectangular area represents the field of view. e study a lifelong scenario, in which an agent, upon reaching a target, is immediately assigned a new one. POGEMA is a challenging environment that requires high generalization abilities from RL algorithms, as the placement of obstacles can vary. During testing, we used scenarios different from the training set.

6h_vs_8z with a map *MMM2*, which is also considered to be ''super-hard''. Figure 2 includes screenshots of the selected maps. The version of StarCraft II used for the evaluation is SC2.4.10 (B75689).

Among the chosen scenarios the total number of actions of each agent varied from 18 to 70, and number of agents varied from 2 to 10. Given that, the maximum theoretical amount of unique joint actions is 18^{10}, though in actual experiments lots of actions are often unavailable.

While SMAC remains a popular benchmark, it has limitations, such as fixed starting positions and unit types. It has been shown [7] that certain methods can learn only sequences of actions, disregarding observations, yet still solve some SMAC scenarios. To address these limitations, the SMACv2 [7] environment was introduced. SMACv2 utilizes procedural content generation, allowing agents trained with these methods to achieve better generalization and solve a wider range of scenarios.

We selected three SMACv2 scenarios for evaluation: *zerg_10_vs_11*, *protoss_10_vs_11*, and *terran_10_vs_11*. For these scenarios, we used the default parameters, including unit type distributions and starting positions, as described in the original SMACv2 publication.

B. POGEMA ENVIRONMENT

POGEMA [39] is a grid-based multi-agent pathfinding environment where multiple agents must navigate to their respective goals, with each goal considered reached when an agent steps on it. The task of decentralized multi-agent pathfinding is particularly challenging, as highlighted by several specialized methods [44], [45], [46], [47], [48], [49]. The example of POGEMA environment is shown in Figure 3.

We consider the LifeLong scenario, where when an agent accomplishes its goal, a new goal is set for it. There are obstacles present on the map, and agents cannot pass through a cell occupied by another agent, which necessitates adopting cooperative behavior to maximize rewards. We use random generated maps for training, which means that the agents' training goal is not to memorize the map, but to be able to adapt to a new layout and find the way to the goal

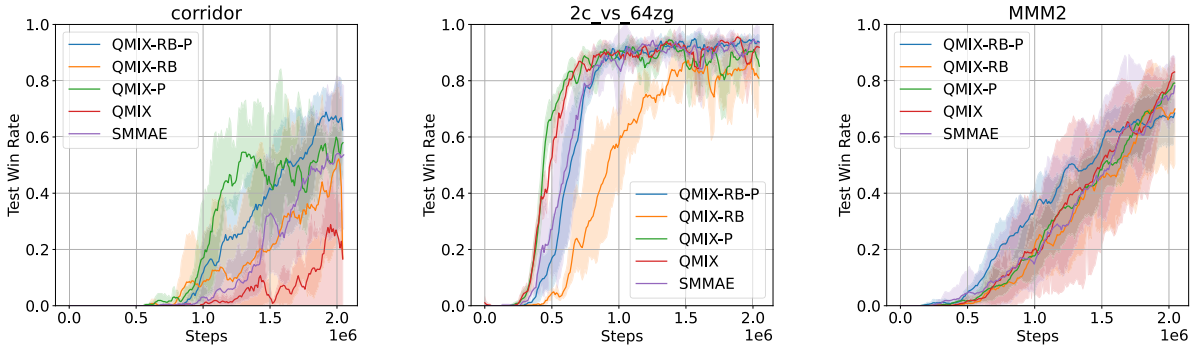

FIGURE 4. Comparison of the mean test win rate of proposed modifications with QMIX and SMMAE algorithms on SMAC. QMIX-RB-P stands for QMIX with replay buffer and exploration policy modifications; QMIX-RB stands for QMIX with replay buffer modification, and QMIX-P stands for QMIX with exploration policy modification. Plots show the mean and 95% confidence interval across five runs. For the corridor scenario, algorithms with enhanced exploration – QMIX-P, QMIX-RB-P and SMMAE – have better performance compared with QMIX; replay buffer modification also improves the results, and the best performance is achieved by QMIX-RB-P. For the 2c_vs_64zg scenario, replay buffer modification leads to worse performance, and the other algorithms have similar results. In the MMM2 scenario, QMIX-RB-P has the steepest learning curve, with similar final performance across algorithms.

on an unknown map. The quality metric used is throughput, i.e., the ratio of the number of the accomplished goals (by all agents) to the episode length.

For this experiment, we do not consider the exploration policy modification as it depends on the current number of available actions, which is constant for POGEMA environment – this means that the usage of the modified exploration policy would still result in a constant value of ϵ.

VI. EXPERIMENTAL RESULTS

In this section, we present our experimental results on SMAC [6] and POGEMA [39] benchmarks. As we study the effectiveness of QMIX [12] method with the proposed modifications of the exploration policy and replay buffer, we consider the version of QMIX with both of the modifications, QMIX with only replay buffer modification, and QMIX with only exploration policy modification. The source code of these methods is available at.[3]

A. COMPARISON ON SMAC

To study the impact of the modified exploration policy and replay buffer modification, we experiment on different SMAC [6] scenarious, and compare results with QMIX [12] and SMMAE [11]. Here, SMMAE is a specialized approach that enhances the exploration abilities of QMIX. We also conduct ablation experiments to study the impact of each algorithm modifications separately, and here the QMIX version with replay buffer modification is called here QMIX-RB; QMIX version with exploration policy modification is called QMIX-P, and QMIX with both of these modifications is QMIX-RB-P.

Following [11], we use in our experiments QMIX with Adam optimizer with default hyperparameters. ϵ anneal time is 50000 steps, and the exploration hyperparameter α is set to 0.04 for *corridor* and 0.02 for *2c_vs_64zg* and *MMM2*.

The results of the experiments on SMAC are shown in Figure 4. On a super-hard scenario *corridor* poor QMIX results compared to other algorithms indicate that additional exploration significantly increases learning performance. SMMAE and QMIX with adaptive ϵ achieve similar results, and QMIX with both modifications has slightly better performance. On a hard scenario *2c_vs_64zg* a replay buffer modification results in worse learning performance, meanwhile QMIX, QMIX with modified exploration policy and SMMAE achieve the similar results. We believe that the performance decrease of the replay buffer modification in this scenario may be due to inappropriate hyperparameters, as we did not adjust the parameters of the replay buffer modification for this specific scenario. On a super-hard scenario *MMM2* an algorithm with both exploration policy and replay buffer modifications have a slightly steeper learning curve, though the final winning rates of algorithms are almost identical.

Comparison of the proposed algorithm with SMMAE, which uses additional attention-based and VAE modules to enhance exploration, shows that it's possible to achieve the similar exploration effectiveness with a simple in terms of computation and implementation modification of ϵ-greedy policy.

B. ADDITIONAL COMPARISON ON SMAC

We also conducted comparison of the proposed modifications with state-of-the-art value-based MARL algorithms, as QPLEX [13] and WQMIX [16], which are not enhance idea of QMIX further in other way.

The results are presented in Figure 5. The implementation of QPLEX algorithm used in experiments was provided by the repository ,[4] and the implementation of WQMIX was provided by the repository.[5]

[3]https://github.com/tolyan3212/re-qmix

[4]https://github.com/wjh720/QPLEX

[5]https://github.com/hijkzzz/pymarl2

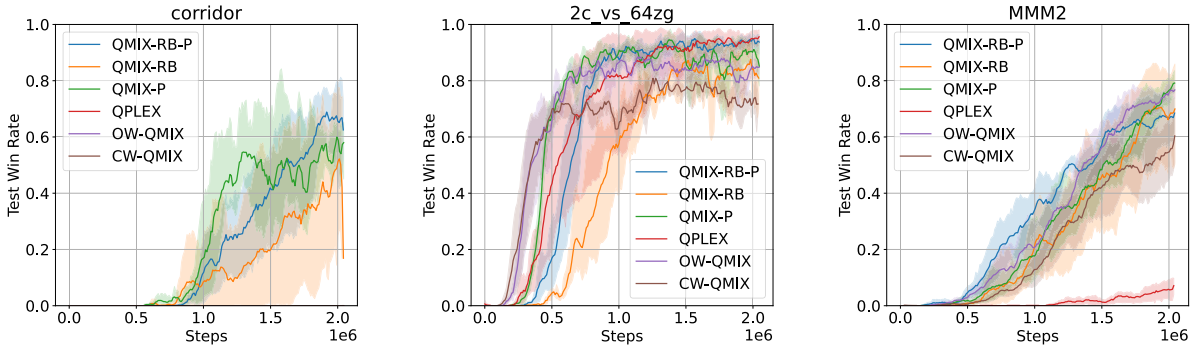

FIGURE 5. Comparison of the mean test win rate of proposed modifications with state-of-the-art MARL algorithms QPLEX and WQMIX on SMAC. QMIX-RB-P stands for QMIX with replay buffer and exploration policy modifications; QMIX-RB stands for QMIX with replay buffer modification, and QMIX-P stands for QMIX with exploration policy modification. Plots show the mean and 95% confidence interval across five runs. The corridor scenario proved to be too challenging for the QPLEX and WQMIX algorithms. In the 2c_vs_64zg scenario, CW-QMIX performed relatively close to QMIX-RB and QMIX-RB-P, and QMIX-P with QPLEX showed the best results. In the MMM2 scenario, QPLEX showed the worst results, CW-QMIX has a slightly worse performance compared to QMIX-RB and QMIX-P, while QMIX-RB-P and OW-QMIX achieved the best results.

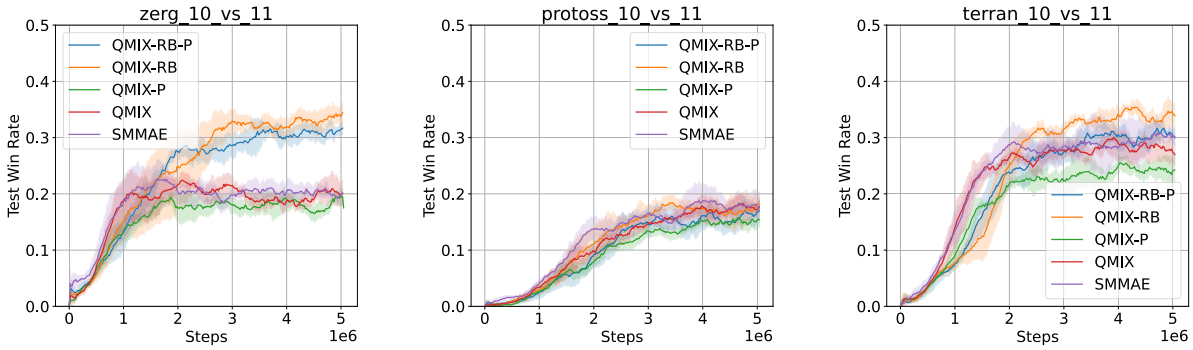

FIGURE 6. Comparison of the mean test win rate of the proposed modifications with the QMIX and SMMAE algorithms on SMACv2. QMIX-RB-P represents QMIX with replay buffer and exploration policy modifications, QMIX-RB represents QMIX with the replay buffer modification, and QMIX-P represents QMIX with the exploration policy modification. The plots display the mean and 95% confidence interval across five runs. On zerg_10_vs_11, algorithms with the replay buffer modification demonstrate significantly better performance compared to other algorithms. On protoss_10_vs_11, the results of all algorithms differ insignificantly. On terran_10_vs_11, algorithms with the replay buffer modification show slightly better performance than those without it, while the exploration policy modification slightly reduces performance; consequently, the QMIX-RB algorithm achieves the best results.

According to the results, the *corridor* map proved to be challenging for both QPLEX and WQMIX algorithms to solve during the training period. QPLEX showed competitive results on the *2c_vs_64zg* map but underperformed on the *MMM2* scenario. OW-QMIX performed similarly to QMIX-RB-P on *MMM2* and had a steep learning curve on the *2c_vs_64zg* map, although its final results on that map were worse than those of QPLEX and QMIX-P. Overall, CW-QMIX demonstrated slightly lower performance compared to the other algorithms.

In summary, while algorithms such as QPLEX and WQMIX may outperform QMIX with the proposed modifications on certain scenarios, their performance is less stable across various scenarios, and these methods struggle to succeed in more challenging scenarios, such as the corridor.

C. COMPARISON ON SMACv2

To further investigate the impact of the proposed modifications, we conducted experiments in the SMACv2 environment, comparing our approach with QMIX and SMMAE. For this purpose, we selected three SMACv2 scenarios: *zerg_10_vs_11*, *protoss_10_vs_11*, and *terran_10_vs_11*, using the default configuration. As outlined in the previous sections, QMIX with the replay buffer modification is abbreviated as QMIX-RB, QMIX with the exploration policy modification as QMIX-P, and QMIX with both modifications as QMIX-RB-P.

The evaluation results on SMACv2 are shown in Figure 6. The selected scenarios are asymmetric, meaning the enemy has additional units, placing the agents at a disadvantage. Consequently, these scenarios are particularly challenging for the algorithms. On *zerg_10_vs_11*, QMIX, QMIX-P, and SMMAE perform relatively similarly, while the algorithms with the replay buffer modification (QMIX-RB and QMIX-RB-P) demonstrate significantly better performance. On *protoss_10_vs_11*, the performance of all algorithms is relatively similar. On *terran_10_vs_11*, the modified replay buffer slightly improves results, while the modified

exploration policy slightly reduces performance. SMMAE performs comparably to QMIX, and the best performance on this scenario is achieved by QMIX-RB.

In summary, algorithms with increased exploration, such as SMMAE, QMIX-P, and QMIX-RB-P, without specific parameter adjustments, perform the same or worse than their counterparts using the plain epsilon-greedy policy (QMIX and QMIX-RB). This suggests that the exploration enforcement provided by SMMAE and the modified exploration policy is too aggressive for the SMACv2 environment, where the default epsilon-greedy policy provides sufficient exploration. Conversely, the replay buffer modification improves performance in the *zerg_10_vs_11* and *terran_10_vs_11* scenarios, while having minimal impact on *protoss_10_vs_11*.

D. COMPARISON ON POGEMA

To further study the impact of the replay buffer modification, we conduct experiments on POGEMA environment with large episode length of 1000 steps and with batch size 64. The goal of that experiment is to simulate the situation, when the environment's episodes are very large and contain an amount of information which is hard to process during the training.

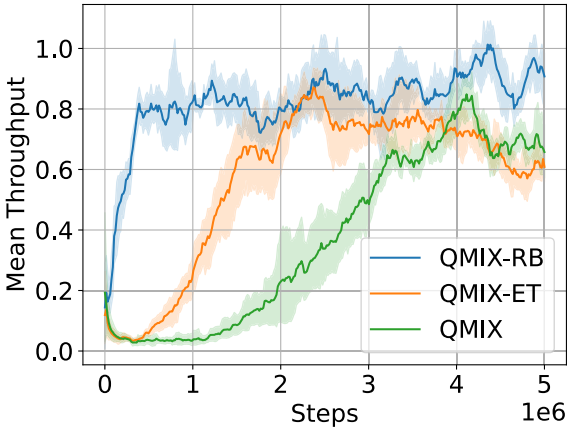

FIGURE 7. Comparison of the mean average throughput on POGEMA with large episodes. QMIX-RB stands for QMIX with replay buffer modification; QMIX-ET stands for QMIX with early training start. Plots show the mean and 95% confidence interval across five runs.QMIX-RB starts quickly improving its performance from the beginning of training, achieving the best results. QMIX-ET has a steeper learning curve compared to QMIX, but it still takes a lot of time to achieve competitive results. Please note that the average throughput can exceed 1.0.

We compared QMIX with proposed replay buffer modification with other two versions of QMIX: one is a usual QMIX implementation, where training starts when enough episodes are sampled to form a full batch of the given size. The second version of QMIX has no restrictions on the start of training, so that training starts right after the first episode was sampled, but with each next sampled episode the batch size is increased up to 64. The results of that comparison are shown in Figure 7.

In this experiment, we use QMIX with standard parameters in an environment featuring long episodes. This leads to poor training performance because a large amount of data is sampled for each batch. We attempted to address this issue by removing the limitation on the number of sampled episodes in the replay buffer, but the results were still worse compared to proposed QMIX-RB approach. In contrast, our algorithm demonstrated strong performance without task-specific parameter tuning, highlighting its reduced dependency on environment characteristics such as episode length. This suggests that the proposed replay buffer modification simplifies hyperparameter selection, including batch size, by mitigating episode length dependence.

TABLE 3. The average throughput of the trained models was evaluated on random POGEMA environment maps with varying agent counts. Throughput was averaged over 500 episodes for each method. QMIX-ET refers to QMIX with an early training start, QMIX-RB denotes QMIX with replay buffer modifications. The QMIX-RB algorithm demonstrated better performance across all scenarios.

Number of Agents	Average Throughput ↑		
	QMIX	QMIX-ET	QMIX-RB
4	0.311	0.228	**0.350**
8	0.492	0.425	**0.612**
16	0.705	0.632	**0.811**
32	0.724	0.684	**1.103**
64	0.393	0.295	**0.890**

We also evaluated the trained models on random maps with varying agent counts to test their generalization ability. The evaluation results are presented in Table 3. Initially, as the number of agents increases, the average throughput also increases because more agents are available to reach their goals. However, when the number of agents becomes too high, the map becomes overcrowded, making it more difficult for agents to reach their goals, leading to a decrease in throughput. Across all scenarios, QMIX-RB demonstrates better performance, indicating that it has learned to generalize more effectively.

VII. CONCLUSION AND LIMITATIONS

In this paper, we have investigated the impact of a modified exploration policy and replay buffer modification in the context of cooperative MARL, using the SMAC and POGEMA environments as benchmarks. Our results show that these modifications can significantly improve the performance of the QMIX algorithm without introducing significant complexity. Across seven different scenarios in three distinct environments, our proposed approach outperforms other algorithms in four scenarios while remaining competitive in the other three. Our enhancements provide a streamlined alternative to complex MARL methods, achieving results comparable to state-of-the-art methods with minimal changes to the original algorithm, thereby simplifying the adaptation process for diverse multi-agent environments.

A promising direction for future research is the study of prioritized replay buffers for MARL, with our proposed modification serving as a potential first step in this area. Additionally, we see large-scale MARL setups-where agents

train for over 1 billion steps in the environment-as an underexplored area within the community.

A potential limitation of our work is the reliance on the centralized training and decentralized execution paradigm, which may be restrictive in settings where full state information is unavailable or the state space is large. While our current approach does not explicitly use local views, it can be adapted to decentralized settings by leveraging local agent actions, making it flexible for broader applications. Additionally, our replay buffer modification can be used independently of CTDE and is compatible with decentralized approaches like Independent Q-Learning.

A further limitation is the lack of formal theoretical guarantees regarding the exploration component. This could be addressed by framing the exploration process as a multi-armed bandit problem, where the number of available arms changes at each time step. While this theoretical perspective could provide more rigorous guarantees, fully developing it within the scope of this paper would be challenging.

REFERENCES

[1] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 24611–24624.

[2] A. Skrynnik, A. Yakovleva, V. Davydov, K. Yakovlev, and A. I. Panov, "Hybrid policy learning for multi-agent pathfinding," *IEEE Access*, vol. 9, pp. 126034–126047, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9532001/

[3] E. Vinitsky, N. Lichtlé, X. Yang, B. Amos, and J. Foerster, "Nocturne: A scalable driving benchmark for bringing multi-agent learning one step closer to the real world," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, Jan. 2022, pp. 3962–3974.

[4] P. Leroy, P. G. Morato, J. Pisane, A. Kolios, and D. Ernst, "IMP-MARL: A suite of environments for large-scale infrastructure management planning via Marl," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, Jan. 2023, pp. 1–11.

[5] X. Pan, M. Liu, F. Zhong, Y. Yang, S.-C. Zhu, and Y. Wang, "MATE: Benchmarking multi-agent reinforcement learning in distributed target coverage control," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27862–27879.

[6] M. Samvelyan, T. Rashid, C. Schroeder de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft multi-agent challenge," 2019, *arXiv:1902.04043*.

[7] B. J. Ellis, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. Foerster, and S. Whiteson, "SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, Jan. 2022, pp. 1–16.

[8] Y. Song, H. Jiang, H. Zhang, Z. Tian, W. Zhang, and J. Wang, "Boosting studies of multi-agent reinforcement learning on Google research football environment: The past, present, and future," 2023, *arXiv:2309.12951*.

[9] J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang, "Exploration in deep reinforcement learning: From single-agent to multiagent domain," 2021, *arXiv:2109.06668*.

[10] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "MAVEN: Multi-agent variational exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, Jan. 2019, pp. 1–20.

[11] S. Zhang, J. Cao, L. Yuan, Y. Yu, and D. Zhan, "Self-motivated multi-agent exploration," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, Jan. 2023, pp. 476–484.

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

[13] J. Wang, Z. Ren, T. Z. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2020, pp. 1–7.

[14] H. M. R. U. Rehman, B.-W. On, D. D. Ningombam, S. Yi, and G. S. Choi, "QSOD: Hybrid policy gradient for deep multi-agent reinforcement learning," *IEEE Access*, vol. 9, pp. 129728–129741, 2021.

[15] K. Son, D. W. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2019, pp. 5887–5896.

[16] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Jan. 2020, pp. 10199–10210.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, vol. 3, Jul. 2018, pp. 2085–2087.

[19] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, Feb. 2016, pp. 1–11.

[20] W. Dabney, G. Ostrovski, and A. Barreto, "Temporally-extended ϵ-greedy exploration," 2020, *arXiv:2006.01782*.

[21] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2778–2787.

[22] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, "VIME: Variational information maximizing exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, Jan. 2016, pp. 1–16.

[23] C. Bai, P. Liu, K. Liu, L. Wang, Y. Zhao, L. Han, and Z. Wang, "Variational dynamic for self-supervised exploration in deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 4776–4790, Aug. 2023.

[24] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2018, pp. 1–14.

[25] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov, "Efficient exploration via state marginal matching," 2019, *arXiv:1906.05274*.

[26] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, and C. Blundell, "Never give up: Learning directed exploration strategies," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2020, pp. 1–17.

[27] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, O. Vitvitskyi, Z. D. Guo, and C. Blundell, "Agent57: Outperforming the Atari human benchmark," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, Jul. 2020, pp. 507–517.

[28] G. Berseth, D. Geng, C. Devin, N. Rhinehart, C. Finn, D. Jayaraman, and S. Levine, "SMiRL: Surprise minimizing reinforcement learning in unstable environments," in *Proc. Int. Conf. Learn. Represent.*, Dec. 2019, pp. 1–18.

[29] Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao, "LIIR: Learning individual intrinsic reward in multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–9.

[30] L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, Jan. 2021, pp. 3757–3769.

[31] T. Wang, J. Wang, Y. Wu, and C. Zhang, "Influence-based multi-agent exploration," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2019, pp. 1–12.

[32] I.-J. Liu, U. Jain, R. A. Yeh, and A. G. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2021, pp. 6826–6836.

[33] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, Sep. 2018, pp. 1–12.

[34] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.
[35] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018, *arXiv:1803.00933*.
[36] T. Le Paine, C. Gulcehre, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, G. Barth-Maron, Z. Wang, N. de Freitas, and W. Team, "Making efficient use of demonstrations to solve hard exploration problems," 2019, *arXiv:1909.01387*.
[37] Y. Mei, H. Zhou, T. Lan, G. Venkataramani, and P. Wei, "MAC-PO: Multi-agent experience replay via collective priority optimization," 2023, *arXiv:2302.10418*.
[38] Y. Wang and Z. Zhang, "Experience selection in multi-agent deep reinforcement learning," in *Proc. IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2019, pp. 864–870.
[39] A. Skrynnik, A. Andreychuk, A. Borzilov, A. Chernyavskiy, K. Yakovlev, and A. Panov, "POGEMA: A benchmark platform for cooperative multi-agent navigation," 2024, *arXiv:2407.14931*.
[40] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "POGEMA: Partially observable grid environment for multiple agents," 2022, *arXiv:2206.10944*.
[41] T. Tieleman, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. for Mach. Learn.*, vol. 4, no. 2, p. 26, 2012.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[43] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.
[44] A. Andreychuk, K. Yakovlev, A. Panov, and A. Skrynnik, "MAPF-GPT: Imitation learning for multi-agent pathfinding at scale," 2024, *arXiv:2409.00134*.
[45] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "SCRIMP: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 9301–9308.
[46] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "Decentralized Monte Carlo tree search for partially observable multi-agent pathfinding," in *Proc. AAAI Conf. Artif. Intell.*, Jan. 2023, vol. 38, no. 16, pp. 17531–17540.
[47] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "When to switch: Planning and learning for partially observable multi-agent pathfinding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 17411–17424, Dec. 2024.
[48] A. Skrynnik, A. Andreychuk, M. V. Nesterova, K. Yakovlev, and A. I. Panov, "Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, Jan. 2023, pp. 17541–17549.
[49] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. S. Kumar, S. Koenig, and H. Choset, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

ANATOLII BORZILOV received the M.S. degree in computer science from MIREA–Russian Technological University, Moscow, Russia, in 2024, and the Ph.D. degree from Moscow Institute of Physics and Technology, Moscow.

Since 2023, he has been a Junior Researcher with the Cognitive Dynamic System Laboratory, Moscow Institute of Physics and Technology, focusing on reinforcement learning and multi-agent systems.

ALEXEY SKRYNNIK received the M.S. degree in computer science from Rybinsk State Aviation Technical University, Rybinsk, Russia, in 2017. He defended his Ph.D. thesis in the field of artificial intelligence and machine learning, in 2023.

Since 2021, he has been a Senior Researcher with the AIRI Institute, Cognitive AI Systems Laboratory. His current research interests include reinforcement learning, learning and planning, and multi-agent systems.

ALEKSANDR PANOV received the M.S. degree in computer science from Moscow Institute of Physics and Technology, Moscow, Russia, in 2011, and the Ph.D. degree in theoretical computer science from the Institute for Systems Analysis, Moscow, in 2015.

Since 2010, he has been a Research Fellow with the Federal Research Center "Computer Science and Control," Russian Academy of Sciences. Since 2018, he has headed the Cognitive Dynamic System Laboratory, Moscow Institute of Physics and Technology. He authored three books and more than 100 research articles. In 2021, he joined the research group on neurosymbolic integration with the Artificial Intelligence Research Institute. His academic interests include behavior planning, reinforcement learning, embodied AI, and cognitive robotics.