

Exploiting Semantic Epsilon Greedy Exploration Strategy in Multi-Agent Reinforcement Learning

Hon Tik Tse, Ho-fung Leung

The Chinese University of Hong Kong

1155126684@link.cuhk.edu.hk, lhf@cuhk.edu.hk

Abstract

Multi-agent reinforcement learning (MARL) can model many real-world applications. However, many MARL approaches rely on ϵ -greedy for exploration, which may discourage visiting advantageous states in hard scenarios. In this paper, we propose a new approach QMIX(SEG) for tackling MARL. It makes use of the value function factorization method QMIX to train per-agent policies and **a novel Semantic Epsilon Greedy (SEG) exploration strategy**. SEG is a simple extension to the conventional ϵ -greedy exploration strategy, yet it is experimentally shown to greatly improve the performance of MARL. We first cluster actions into groups of actions with similar effects and then **use the groups in a bi-level ϵ -greedy exploration hierarchy for action selection**. We argue that SEG facilitates semantic exploration by exploring in the space of groups of actions, which have richer semantic meanings than atomic actions. Experiments show that QMIX(SEG) largely outperforms QMIX and leads to strong performance competitive with current state-of-the-art MARL approaches on the StarCraft Multi-Agent Challenge (SMAC).

1 Introduction

Multi-agent systems (MASs) can be used to model many complex scenarios. Multi-agent reinforcement learning (MARL) has been growing in popularity and importance, and it is used in many real-world applications, such as robot swarm control [Hüttenrauch *et al.*, 2019].

Different MARL algorithms have been designed to enable multiple agents to learn to make good decisions. Among them, centralized training with decentralized execution (CTDE) has been shown to be a powerful paradigm achieving better performance than algorithms with non-centralized training [Gupta *et al.*, 2017; Foerster *et al.*, 2018]. An influential algorithm in the CTDE paradigm is QMIX [Rashid *et al.*, 2018], which estimates joint action-values as a non-linear monotonic function of per-agent values. A lot of work [Son *et al.*, 2019; Rashid *et al.*, 2020; Wang *et al.*, 2020a; Iqbal *et al.*, 2021; Wang *et al.*, 2021a; Wang

et al., 2021b] has been put into building on QMIX to develop better-performing algorithms. Recently, RODE [Wang *et al.*, 2021b], a hierarchical algorithm utilizing QMIX, has seen huge success in the StarCraft Multi-Agent Challenge [Samvelyan *et al.*, 2019] (SMAC). The novelty of RODE is twofold. First, the action space is decomposed into *roles* consisting of actions of similar functionalities. Second, the action spaces of the agents are limited by the roles they select, that is, agents must select actions from the restricted action spaces corresponding to the roles they selected.

The restriction of action spaces in RODE improves learning, but places heavy limitations on the actions agents can select. In the RODE framework, agents are required to follow a selected role for extended time steps. This reduces the flexibility of an agent and prevents the agent from reacting to rapid changes in the environment.

In fact, as Nachum *et al.* [2019] point out, in the single-agent case, hierarchical reinforcement learning algorithms attain better performance mainly due to two key improvements in exploration: *temporally extended exploration* and *semantic exploration*. They further show that these exploration benefits can actually be recreated in non-hierarchical agents employing carefully designed exploration strategies. We note that Dabney *et al.* [2021] have demonstrated in a recent paper that the *temporally extended ϵ -greedy exploration*, a simple extension of ϵ -greedy exploration, can improve the performance of reinforcement learning in hard-exploration Atari games with minimal loss in performance on the remaining games.

In this paper, we propose a novel approach QMIX(SEG) for cooperative MARL scenarios. QMIX(SEG) is a non-hierarchical learning approach. It first employs QMIX to represent the Q-values of joint actions as a nonlinear monotonic function of per-agent Q-values of actions. Then, it uses a novel **Semantic Epsilon Greedy (SEG)** exploration strategy for action selection. SEG is a simple yet effective 2-level extension to ϵ -greedy strategy achieving semantic exploration. Conceptually, at the higher level, SEG explores in the space of groups of actions, while at the lower level it explores in the space of atomic actions within the selected group.

Intuitively, QMIX(SEG) aims at recreating the exploration benefits exhibited in the hierarchical learning framework RODE in a non-hierarchical framework. Meanwhile, it lifts the restrictions RODE places on the actions agents can select, that is, they can only be selected from the currently selected

role for multiple steps. QMIX(SEG) is also an attempt to extend ε -greedy exploration to develop a still simple, yet effective exploration method that removes the inherent bias in ε -greedy.

Experimental results on SMAC show that QMIX(SEG) outperforms other state-of-the-art approaches, including QMIX, RODE, MAPPO(FP), and MAPPO(AS) [Yu *et al.*, 2021]. In particular, QMIX(SEG) can attain the highest or near-highest win rates on 4 out of 5 super hard maps. QMIX(SEG) also achieves strong results competitive with DDN [Sun *et al.*, 2021], a much more complex approach to generalize expected value function factorization methods.

2 Related Work

In recent years, there has been increasing interest in improving multi-agent reinforcement learning (MARL) algorithms by innovative exploration mechanisms. Wang *et al.* [2020b] exploit interaction between agents for exploration while Jaques *et al.* [2019] propose to reward actions that have a large influence on other agents’ behaviors. Mahajan *et al.* [2019] explore the use of a shared latent variable to coordinate exploration and encourage diverse behavior by maximizing the mutual information between the trajectories and the latent variable. Liu *et al.* [2021] encourage agents to explore the projected state space while increasing the dimension of the projected state space gradually. Gupta *et al.* [2021] propose to perform actions of similar solved tasks on the target task.

Hierarchical reinforcement learning (HRL) [Sutton *et al.*, 1999] has been shown to have superior performance, especially in sparse reward problems [Pateria *et al.*, 2021]. It has also been extensively studied in the context of MARL [Osenkoff *et al.*, 2019; Wang *et al.*, 2020a; Lee *et al.*, 2020; Yang *et al.*, 2020].

In a recent work, Nachum *et al.* [2019] show that the benefits of HRL mainly come from improved explorations: in HRL, there are both *temporally extended explorations* of the environment across multiple environment steps, and *semantic explorations* of semantically meaningful actions. They further show that such benefits can be achieved in non-hierarchical agents with certain exploration methods, allowing non-hierarchical agents to achieve competitive results with hierarchical agents.

Dabney *et al.* [2021] propose to extend ε -greedy exploration, achieving results comparable to more complex non-dithering exploration methods. They focus on extending ε -greedy exploration in the direction of temporally extended exploration by repeating a selected action for a random number of steps.

RODE [Wang *et al.*, 2021b] is a hierarchical MARL algorithm that demonstrates superiority in performance over existing algorithms in scenarios of SMAC. An innovative idea of RODE is that each agent first selects a *role* using ε -greedy exploration, and then an action available in the role action space, once again using ε -greedy exploration.

3 Methods

In this section, we propose a novel approach QMIX(SEG) for tackling cooperative multi-agent problems. The novelty comes from an exploration strategy called **Semantic Epsilon Greedy (SEG)**, which adds an extra layer of ε -greedy exploration to the conventional ε -greedy exploration. Our method is inspired by RODE, and it extends ε -greedy exploration in the direction of semantic exploration.

3.1 Background

A fully cooperative multi-agent task can be formulated as a decentralized partially observable Markov decision process (Dec-POMDP) [Oliehoek and Amato, 2016] $\langle S, A, P, R, \Omega, O, n, \gamma \rangle$, where n is the number of agents, $\gamma \in [0, 1)$ is the discount factor and $s \in S$ is the true state of the environment. At each time step, each agent i takes an action $a_i \in A$, causing the environment to move into the next state $s' \in S$ and $s' \sim P(s' | s, \mathbf{a})$ where $\mathbf{a} = (a_1, \dots, a_n) \in A^n$ is the joint action. Agents also receive a shared reward $r = R(s, \mathbf{a})$. We consider a partially observable setting so each agent i receives an observation $o_i \in \Omega$ and $o_i = O(s, i)$. Each agent has an action-observation history $\tau_i \in T \equiv (\Omega, A)^*$, on which it conditions its policy $\pi_i(a_i | \tau_i)$. The joint policy $\pi = (\pi_1, \dots, \pi_n)$ has a joint action-value function $Q_{tot}^\pi(s, \mathbf{a}) = \mathbb{E}_{s_{0:\infty}, \mathbf{a}_{0:\infty}} [\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \mathbf{a}_0 = \mathbf{a}, \pi]$.

In the centralized training with decentralized execution (CTDE) paradigm, the learning algorithm has access to the true state s of the environment and the action-observation histories $\tau = (\tau_1, \dots, \tau_n)$ of all agents during training. On the other hand, each agent’s policy π_i depends only on its local action-observation history τ_i , allowing for decentralized execution. A popular approach to achieve CTDE is *value function factorization*, which decomposes the joint Q-function into per-agent utility functions, for example as their sum [Sunehag *et al.*, 2018]. QMIX [Rashid *et al.*, 2018] factorizes the joint Q-function into a non-linear monotonic function of per-agent utility functions, and its success triggers several subsequent research [Son *et al.*, 2019; Rashid *et al.*, 2020; Wang *et al.*, 2021a].

3.2 Learning Q-values Using QMIX

Before going into how QMIX(SEG) makes use of Q-values for exploration, we first describe how Q-values are learned. We follow exactly the approach in QMIX [Rashid *et al.*, 2018] and use a Q-network to estimate the per-agent utility functions and a mixing network to estimate the joint Q-function. We use value function factorization as it allows agents to better learn coordinated behavior. We use a QMIX-style mixing network for the monotonic factorization it provides and its popularity. The mixing network is only used during centralized training, and is not used during decentralized execution. To stay self-contained, we describe the Q-network, the mixing network, and the process of learning their parameters in this subsection. Figure 1 illustrates the Q-network and the mixing network.

The Q-network consists of a fully connected layer, followed by a GRU, and further followed by a fully connected

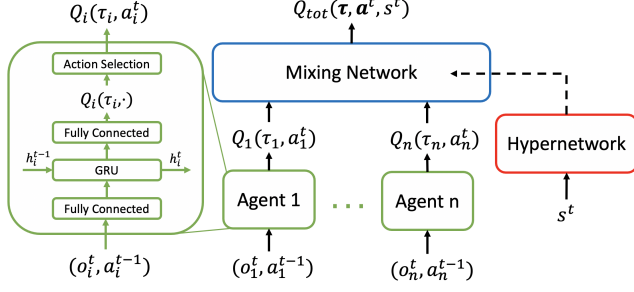


Figure 1: Q-network and mixing network same as in QMIX.

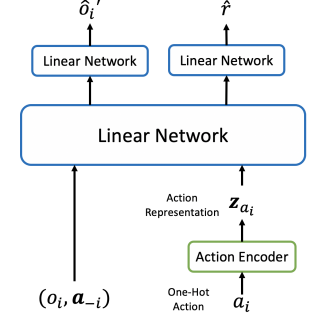


Figure 2: Forward model for learning action representations same as in RODE.

layer. The Q-network takes the local observation and the last local action selected as input. The GRU allows the agents to utilize past actions and observations for predicting the Q-values of actions, which are the outputs of the last fully connected layer. We utilize parameter sharing and agents share the same Q-network.

Suppose each agent i selects the action a_i . The Q-values of the selected actions $Q_1(\tau_1, a_1), \dots, Q_n(\tau_n, a_n)$ are passed as input to the mixing network, which outputs the joint Q-value $Q_{tot}(\tau, \mathbf{a}, s)$. The parameters of the mixing network are predicted by a hypernetwork [Ha *et al.*, 2017] taking the state s as input. In particular, the weights of the mixing network predicted by the hypernetwork are non-negative such that the joint Q-function is a monotonic function of per-agent utility functions. The biases, on the other hand, do not need to be non-negative. Passing the state s as input to the hypernetwork allows the joint Q-function to depend on the state s in non-monotonic ways. The Q-network, parameterized by θ , and the mixing network, parameterized by ϕ , are trained end-to-end by minimizing the following loss function:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathcal{D}}[(r + \gamma \max_{\mathbf{a}'} Q_{tot}^-(\tau', \mathbf{a}', s') - Q_{tot}(\tau, \mathbf{a}, s))^2].$$

Here, Q_{tot}^- is a target network, τ' is the next joint action-observation history, s' is the next state, and \mathcal{D} is a replay buffer from which uniform samples are drawn. The maximization of Q_{tot}^- in the loss function can be easily done. Since the joint Q-function is a monotonic function of per-agent utility functions, maximizing per-agent utility functions then passing the max values and the state into the mixing network gives $\max Q_{tot}^-$. After the Q-values are learnt, we use SEG for action selection and exploration.

3.3 Semantic Epsilon Greedy

In the Semantic Epsilon Greedy (SEG) exploration strategy, we first learn to cluster actions into groups of actions with similar effects. Then, we adopt a bi-level ε -greedy hierarchy to ensure each group of actions is explored with equal probability. We use "semantic" instead of "hierarchical" as the name of the exploration strategy to avoid confusion between our method, a hierarchical *exploration strategy*, and hierarchical *learning algorithms*.

The approach in SEG to cluster actions into groups is the same as that of RODE. We identify the groups of actions by

first learning the action representations of all actions and then clustering the actions into groups by their representations.

To learn the action representations that reflect the effect of actions on the environment and other agents, we use the induced reward and the change in local observations to measure the effect of an action. Figure 2 illustrates the forward model for learning action representations. First, we encode the one-hot action a_i that agent i takes as a d -dimensional representation z_{a_i} using an action encoder $f_e(\cdot; \theta_e)$. Then, the action representation z_{a_i} is passed as input to the observation predictor p_o and reward predictor p_r to predict the next local observation and the received reward respectively, given the current local observation o_i of agent i and the one-hot actions \mathbf{a}_{-i} of all other agents. The intuition is that if the representation can be used to predict the next local observation and reward, the representation captures the effect of the action on the environment and other agents. The action encoder f_e , parameterized by θ_e , and the observation predictor p_o and reward predictor p_r , jointly parameterized by ξ_e , are trained end-to-end by minimizing the following loss function:

$$\mathcal{L}_e(\theta_e, \xi_e) = \mathbb{E}_{(o, \mathbf{a}, r, o') \sim \mathcal{D}} \left[\sum_{i=1}^n \|\hat{o}'_i - o'_i\|_2^2 + \lambda_e \sum_{i=1}^n (\hat{r} - r)^2 \right],$$

where \mathcal{D} is the replay buffer, $\hat{o}'_i = p_o(z_{a_i}, o_i, \mathbf{a}_{-i})$ is the predicted next local observation of agent i , $\hat{r} = p_r(z_{a_i}, o_i, \mathbf{a}_{-i})$ is the predicted received reward, o'_i is the true next local observation of agent i , r is the true received reward, and the sum is carried out over all agents. The scaling factor λ_e balances between the error of predicting next local observations and the error of predicting the received reward. The trained f_e is then used to encode all actions into their corresponding action representations, which are fixed for the remaining time steps.

After obtaining the action representations, we cluster the actions into groups of actions using k -means clustering based on Euclidean distances between action representations. The number of clusters or groups is chosen as a hyperparameter. Again we follow RODE and use Euclidean distances as the distance metric. Actions belonging to the same group are generally similar in effect (for example, they are all *attack* actions of some kind), as they have small Euclidean distances among their action representations.

After clustering the actions into groups of actions, we further add an action that is always available to all agents called

no-op (no operation) to each group. The existence, name, and number of such actions may vary in different benchmarks. For example, in SMAC, such actions are called *no-op* and *stop*. This is to ensure that in each group of actions, at least one action is always available to all agents. Note that domain knowledge, if available, can be used to directly cluster the actions into groups without having to learn and cluster action representations.

After clustering the actions into groups, SEG makes use of the groups for bi-level exploration. The following hierarchical exploration procedure is carried out for each agent i at each time step during training. Let $Q_i(\tau_i, a)$ be the Q-value of action $a \in A$ for agent i .

1. At the higher level, with probability $1 - \varepsilon$, we select the action $\arg \max_{a \in A} Q_i(\tau_i, a)$ with the largest Q-value, just as in standard ε -greedy exploration.
2. However, the difference from standard ε -greedy exploration lies in what we do with probability ε . Specifically, with probability ε ,
 - (a) we select a group A^j uniformly randomly, and
 - (b) after selecting the group A^j , we carry out the lower level of ε -greedy exploration of actions: with probability $1 - \varepsilon$, we select the action $\arg \max_{a \in A^j} Q_i(\tau_i, a)$ with the largest Q-value that is available in the selected group A^j ; with probability ε , we select one of the actions available in the selected group A^j uniformly randomly.

This bi-level procedure is carried out for each agent, meaning that different agents, when they explore simultaneously, may explore in different groups.

SEG is a bi-level extension of ε -greedy. At the higher level, SEG can be interpreted as exploring the space of groups of actions. Suppose the number of groups is m , then the actions can be clustered into A^1, \dots, A^m . The higher level of SEG can be interpreted as selecting from the following groups: A, A^1, \dots, A^m , where the set of all available actions A is also interpreted as a group. If we associate a Q-value with each of A, A^1, \dots, A^m and assume the set of all actions A always has the highest Q-value, then using conventional ε -greedy to select one of A, A^1, \dots, A^m is very similar to what we do in SEG, which is to select the group with the highest Q-value A with probability $1 - \varepsilon$ and select one of the remaining groups A^1, \dots, A^m uniformly randomly with probability ε .

SEG achieves semantic exploration by exploring at the higher level the space of groups of actions, which has richer semantic meanings than atomic actions. The intuitive reason semantic exploration helps is that by exploring in the space of groups where elements have richer semantic meanings, the exploration carried out will have richer meaning in the environment. A good example proposed by Nachum *et al.* [2019] is that in a robot navigation task, exploring at the level of x - y coordinates is more sensible than exploring at the level of robot joint torques.

Note that SEG only limits the actions agents can select when agents explore and select groups (with probability ε) during training, in which case the actions must be selected from the selected groups. SEG does not limit the actions agents can select during test time as $\varepsilon = 0$ during test time.

As long as clustering of actions into groups of similar actions is present, SEG can be implemented easily. Classifying actions into groups of similar actions can be done easily for discrete actions, so SEG can be implemented easily for problems with discrete action space. SEG is an exploration method that only requires the Q-values of actions and clustering of actions, allowing it to be applied generally. In fact, SEG does not exploit centralized training in any way, making SEG compatible with algorithms even not in the CTDE paradigm, such as IQL [Tan, 1993].

3.4 Discussion

A more practical way of looking at why SEG gives rise to better exploration than standard ε -greedy exploration is to look at the probability of exploring states that require coordination for extended time steps to reach. Suppose there exist states that require all n agents to take actions with similar effects (or, actions in the same group) for t steps to reach. Denote s^i as the state that requires all n agents to take actions in the group A^i for t steps to reach. The probability of reaching s^i with standard ε -greedy exploration is $(\frac{\varepsilon|A^i|}{|A|})^{nt}$. Note that the probability depends on the number of actions in the group A^i . This means that standard ε -greedy is biased towards visiting states that can be reached by a large number of similar actions. This problem is present in single-agent reinforcement learning but is worsened in MARL because of the extra n in the exponent. This is undesirable if states that are critical to good performance can only be reached by jointly selecting a small number of similar actions.

SEG removes this bias in standard ε -greedy by exploring each group with equal probability. Since SEG first selects a group uniformly randomly before selecting an atomic action from the selected group, suppose the number of groups is m , the probability of reaching s^i for all i is $(\frac{\varepsilon}{m})^{nt}$. Now, the probability of reaching s^i is independent of the number of actions in the group A^i required to reach s^i . SEG, by removing the bias in standard ε -greedy, allows the agents to visit s^i 's with equal probability. In simpler settings where jointly selecting groups containing a large number of actions already give rise to optimal behavior, SEG may result in slower learning due to excessive exploration. However, in harder scenarios where advantageous states can only be reached by a small number of similar actions, SEG allows the agents to visit these states more and learn better strategy while using ε -greedy for exploration may fail to reach these states. This provides a more practical foundation for why SEG can achieve superior performance compared to standard ε -greedy exploration, especially in harder scenarios.

QMIX(SEG) bears similarity with RODE [Wang *et al.*, 2021b]. In particular, the procedures for clustering actions are the same. Just as QMIX(SEG), RODE achieves semantic exploration by exploring at the higher level the space of roles. By the novel SEG exploration strategy, we recreate semantic exploration from RODE in QMIX. QMIX(SEG) shows in the MARL setting that exploration benefits in hierarchical algorithms (such as RODE) can be attained in non-hierarchical agents with certain exploration techniques, allowing non-hierarchical agents to achieve competitive results

with hierarchical agents.

In addition to capturing the exploration benefits of RODE, QMIX(SEG) removes a restriction in RODE. RODE requires that after a role is selected, an agent follows the selected role for c time steps, that is, for c time steps it will only take actions in the restricted action space corresponding to the role, even during test time.¹ This greatly reduces the flexibility and level of freedom with which agents can select actions, and it can be disastrous for an agent that is unable to change its role promptly in response to some rapid changes in the environment. This restriction is removed in QMIX(SEG).

QMIX(SEG) further differs from RODE in three aspects. First, RODE focuses on efficiently learning a set of roles to decompose complex tasks while QMIX(SEG) focuses on improving the exploration for MARL. Second, RODE achieves improved exploration through its bi-level hierarchical agents while QMIX(SEG) makes use of non-hierarchical agents and employs hierarchy only in action selection, which makes QMIX(SEG) computationally less expensive. Third, RODE uses Q-values of roles to assign roles to agents while QMIX(SEG) assigns groups to agents uniformly randomly.

4 Experiments

In our experiments, we have two goals. First, we would like to provide empirical evidence for our discussion in Section 3.4. We mentioned that in scenarios where advantageous states can only be reached by jointly selecting actions in groups containing a small number of actions, using SEG for exploration gives rise to better performance than using ϵ -greedy for exploration. We prove this by comparing SEG and ϵ -greedy on a simple coordination game we designed. Second, we would like to find out whether improved exploration with SEG leads to better performance in complicated scenarios. We study this problem by performing experiments on the popular SMAC benchmark [Samvelyan *et al.*, 2019].

4.1 Coordination Game

To find out whether SEG allows agents to learn better strategy than ϵ -greedy in scenarios where advantageous states are hard to reach, we compare SEG and ϵ -greedy on a proof-of-concept coordination game. The game is specified by a 3-tuple (N, K, M) , where N is the number of agents, K is the number of steps agents must coordinate to reach the advantageous state, and M is the number of actions in the action space. The states of the game are $\{s_0, \dots, s_K\}$, where s_K is the advantageous state that agents should cooperate to visit. All N agents start at state s_0 and have action space $A = \{a_0, \dots, a_{M-1}\}$. There is only one way to receive a positive reward in this game, which is to reach s_K . In s_K , any joint action selected gives a reward of 100 and causes the state to transition to s_0 . Any joint action selected in any state other than s_K gives a reward of 0. The only way to reach s_K is to repeatedly take the joint action (a_0, \dots, a_0) , which causes

¹As a consequence, RODE removes roles with only a single action and adds these actions to each of the remaining roles. This is because being assigned a singleton role forces an agent to take the same action for c steps, which is undesirable. The removal of the singleton groups is not necessary in QMIX(SEG).

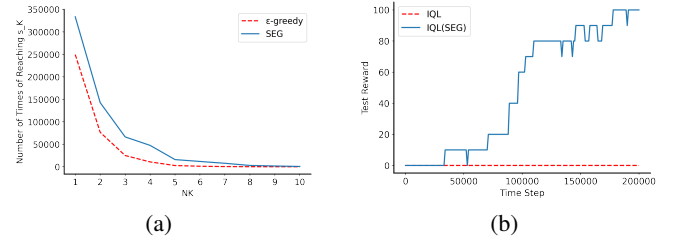


Figure 3: (a) The number of times s_K is reached in one million steps as NK increases. We choose $M = 3$ and $\epsilon = 0.5$. (b) Test reward of IQL(SEG) and IQL on the coordination game with $N = 5$, $K = 4$, and $M = 3$.

the state to transition from s_j to s_{j+1} for $0 \leq j < K$. Taking any joint action other than (a_0, \dots, a_0) in s_j for $0 \leq j < K$ causes the state to transition to s_0 .

The game poses a difficult challenge to agents. To receive the reward, the N agents must coordinate and all select a_0 for consecutive K steps to reach s_K . The parameters N and K control how difficult it is to reach s_K . When N increases, more agents are required to coordinate. When K increases, agents are required to coordinate for a larger number of time steps. We initialize $Q_i(s, a_0) = 0 \forall i, s$ and $Q_i(s, a_j) = 0.1 \forall i, s, j \neq 0$ such that the agents do not select (a_0, \dots, a_0) at the beginning. The agents must reach s_K via exploration to learn the optimal strategy of visiting s_K .

The number of available actions M determines how difficult it is for an agent using ϵ -greedy for exploration to select a_0 . Since ϵ -greedy exploration explores each action with equal probability, when M increases, ϵ -greedy exploration has a lower probability of exploring a_0 . When $M > 2$, ϵ -greedy is biased towards selecting one of a_1, \dots, a_{M-1} and it is difficult for agents using ϵ -greedy for exploration to reach s_K . On the other hand, using SEG for exploration does not have this problem. In this game, we choose the number of groups to be 2. Since a_1, \dots, a_{M-1} have the same effects, the actions are split into two groups: $A^1 = \{a_0\}$ and $A^2 = \{a_1, \dots, a_{M-1}\}$. Since SEG first selects a group uniformly randomly before selecting an action, using SEG for exploration gives the same probability of selecting a_0 and selecting one of a_1, \dots, a_{M-1} , thus giving a higher probability of visiting state s_K . To see whether this is true empirically, we count the number of times of visiting s_K by exploration for SEG and ϵ -greedy. Since we are interested in the probability s_K is visited by exploration, we initialize the Q-values as mentioned and do not update the Q-values. We present the number of times of visiting s_K by exploration for SEG and ϵ -greedy in one million steps as NK increases in Figure 3(a).

To see whether using SEG for exploration allows agents to learn a better strategy on this game, we train IQL [Tan, 1993] using SEG for exploration (IQL(SEG)) and IQL using ϵ -greedy for exploration (IQL) on this game with $N = 5$, $K = 4$ and $M = 3$. We anneal ϵ from 1.0 to 0.05 in 30,000 steps. We run 10 trials and in each trial, we evaluate the algorithms for $K + 1$ steps on a regular interval. We present the mean reward of the 10 trials in Figure 3(b).

From Figure 3(a), it can be observed that SEG allows the

Map	Map Difficulty	QMIX	RODE	MAPPO(FP)	MAPPO(AS)	DDN	QMIX(SEG)
MMM2	Super Hard	82.8(4.0)	89.8(6.7)	51.6(21.9)	28.1(29.6)	97.2	96.1(2.3)
3s5z_vs_3s6z		56.2(11.3)	96.8(25.11)	75.0(36.3)	18.8(37.4)	94.0	99.2(1.8)
27m_vs_30m		34.4(5.4)	96.8(1.5)	93.8(3.8)	89.1(6.5)	91.5	89.8(5.3)
6h_vs_8z		3.1(1.5)	78.1(37.0)	78.1(5.6)	81.2(31.8)	83.9	82.8(4.9)
corridor		64.1(14.3)	65.6(32.1)	93.8(3.5)	93.8(2.8)	95.4	93.8(23.5)

Table 1: Median evaluation win rate and standard deviation on 5 super hard maps. Numbers for QMIX, RODE, MAPPO(FP), and MAPPO(AS) are obtained from the MAPPO paper [Yu *et al.*, 2021]. Numbers for DDN are obtained from the DFAC framework paper [Sun *et al.*, 2021]. The highest win rates for each map are bolded. For each map, each algorithm is trained for the same number of time steps as RODE.

agents to explore s_K for a larger number of times than ϵ -greedy. When $NK = 10$, using SEG for exploration still allows the agents to reach s_K for around 700 times while using ϵ -greedy only allows the agents to reach s_K for less than 10 times. This proves that SEG, by removing the bias in ϵ -greedy, allows agents to visit advantageous states more when such states can only be reached by a small number of similar actions. From Figure 3(b), it can be observed that SEG, by allowing agents to visit the advantageous state s_K for a much larger number of times, allows the IQL algorithm to learn the optimal behavior of reaching s_K , getting a reward of 100. On the other hand, using ϵ -greedy for exploration fails to give rise to the optimal behavior. Therefore, in scenarios where agents have to jointly select from a small number of actions with similar effects for extended time steps to reach advantageous states, SEG allows agents to better reach these states and thus allows the agents to learn better strategies.

4.2 StarCraft Multi-Agent Challenge

To find out whether improved exploration with SEG can lead to better performance in complicated scenarios, we evaluate our method QMIX(SEG) on SMAC [Samvelyan *et al.*, 2019]. We choose SMAC since it allows algorithms to be compared directly by their win rates on different maps. In SMAC, maps can be classified into three difficulties: easy, hard, and super hard. Hard maps pose different challenges such as handling large action space and kiting. Super hard maps are hard-exploration problems, making them appropriate scenarios to test our method.

We compare QMIX(SEG) with QMIX to directly see whether the improved exploration over ϵ -greedy can provide better performance. We also compare QMIX(SEG) with RODE [Wang *et al.*, 2021b] to see whether QMIX(SEG), capturing the exploration benefits and avoiding the limitations of RODE, can outperform RODE. We further compare QMIX(SEG) with state-of-the-art algorithms on SMAC: MAPPO [Yu *et al.*, 2021] and DDN [Sun *et al.*, 2021]. We follow the evaluation procedure outlined in the MAPPO paper [Yu *et al.*, 2021].

We present the results of QMIX(SEG) and baselines on all 5 super hard maps in Table 1. QMIX(SEG) outperforms most state-of-the-art algorithms, including QMIX, RODE, and MAPPO. Compared with MAPPO(FP) and MAPPO(AS), QMIX(SEG) achieves higher win rates on 2 maps and similar win rates on the remaining 3 maps. DDN makes use of distributional reinforcement learning and is much more com-

plex than our approach. Still, QMIX(SEG) is able to obtain a higher win rate than DDN on 1 map and obtain win rates within 2% of that of DDN on all remaining maps.

By comparing QMIX(SEG) and QMIX, it can be observed that using SEG instead of ϵ -greedy for exploration significantly improves the performance on all maps. On harder maps, agents are usually required to jointly move for extended time steps to reach advantageous positions on the map. Also, movement actions are outnumbered by attack actions. SEG, by removing the bias in ϵ -greedy towards selecting attack actions, allows agents to jointly explore movement actions more and reach advantageous positions with higher probability, leading to better performance on harder maps.

QMIX(SEG) has higher win rates than RODE on 4 out of 5 super hard maps. Removing the limitations of RODE may be the reason QMIX(SEG) can achieve better performance on these maps. A difference between QMIX(SEG) and RODE is that RODE uses the Q-values of roles to determine which role to assign to an agent while we assign groups to agents uniformly randomly with probability ϵ . From the close performance of QMIX(SEG) and RODE, it can be concluded that how we assign roles or groups, and correspondingly action spaces, to agents do not matter. We just need to make sure that the action spaces of agents are restricted with a certain probability to gain the exploration benefits.

5 Conclusion

We propose a novel non-hierarchical learning approach QMIX(SEG), which utilizes QMIX for value function factorization, and a novel hierarchical dithering exploration method SEG. SEG is a simple, general, and effective way to improve exploration by semantic exploration in the space of groups of similar actions and removing the bias in ϵ -greedy. From the experiments, we show that SEG significantly outperforms ϵ -greedy, especially in hard scenarios. QMIX(SEG) can achieve state-of-the-art performance on SMAC. Our work also provides yet another compelling example of extracting exploration methods from hierarchical reinforcement learning algorithms. Future work can focus on combining our work with temporally extended ϵ -greedy and applying SEG on single agent reinforcement learning problems.

References

- [Dabney *et al.*, 2021] Will Dabney, Georg Ostrovski, and Andre Barreto. Temporally-extended ε -greedy exploration. In *ICLR*, 2021.
- [Foerster *et al.*, 2018] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, 2018.
- [Gupta *et al.*, 2017] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS*, 2017.
- [Gupta *et al.*, 2021] Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Boehmer, and Shimon Whiteson. Uneven: Universal value exploration for multi-agent reinforcement learning. In *ICML*, 2021.
- [Ha *et al.*, 2017] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *ICLR*, 2017.
- [Hüttenrauch *et al.*, 2019] Maximilian Hüttenrauch, Adrian Sosic, and Gerhard Neumann. Deep reinforcement learning for swarm systems. *J. Mach. Learn. Res.*, 20:54:1–54:31, 2019.
- [Iqbal *et al.*, 2021] Shariq Iqbal, Christian A. Schröder de Witt, Bei Peng, Wendelin Boehmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *ICML*, 2021.
- [Jaques *et al.*, 2019] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Çağlar Gülçehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *ICML*, 2019.
- [Lee *et al.*, 2020] Youngwoon Lee, Jingyun Yang, and Joseph J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *ICLR*, 2020.
- [Liu *et al.*, 2021] Iou-Jen Liu, Unnat Jain, Raymond A. Yeh, and Alexander G. Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *ICML*, 2021.
- [Mahajan *et al.*, 2019] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: multi-agent variational exploration. In *NIPS*, 2019.
- [Nachum *et al.*, 2019] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *CoRR*, abs/1909.10618, 2019.
- [Oliehoek and Amato, 2016] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer, 2016.
- [Ossenkopf *et al.*, 2019] Marie Ossenkopf, Mackenzie Jorgensen, and Kurt Geihs. When does communication learning need hierarchical multi-agent deep reinforcement learning. *Cybernetics and Systems*, 50(8):672–692, 2019.
- [Pateria *et al.*, 2021] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Comput. Surv.*, 54(5), jun 2021.
- [Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, 2018.
- [Rashid *et al.*, 2020] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *NIPS*, 2020.
- [Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *AAMAS*, 2019.
- [Son *et al.*, 2019] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, 2019.
- [Sun *et al.*, 2021] Wei-Fang Sun, Cheng-Kuang Lee, and Chun-Yi Lee. DFAC framework: Factorizing the value function via quantile mixture for multi-agent distributional q-learning. In *ICML*, 2021.
- [Sunehag *et al.*, 2018] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, 2018.
- [Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.
- [Tan, 1993] Ming Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *ICML*, 1993.
- [Wang *et al.*, 2020a] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. In *ICML*, 2020.
- [Wang *et al.*, 2020b] Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. In *ICLR*, 2020.
- [Wang *et al.*, 2021a] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent q-learning. In *ICLR*, 2021.
- [Wang *et al.*, 2021b] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: learning roles to decompose multi-agent tasks. In *ICLR*, 2021.
- [Yang *et al.*, 2020] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In *AAMAS*, 2020.
- [Yu *et al.*, 2021] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of MAPPO in cooperative, multi-agent games. *CoRR*, abs/2103.01955, 2021.

Map	Map Difficulty	QMIX	RODE	MAPPO(FP)	MAPPO(AS)	QMIX(SEG)
2s_vs_1sc	Easy	96.9(1.2)	100.0(0.0)	100.0(0.0)	100.0(0.0)	100.0(1.2)
2s3z		95.3(3.9)	100.0(0.0)	96.9(1.5)	96.9(1.5)	100.0(1.2)
3s5z		85.9(4.6)	93.75(2.0)	71.9(11.8)	53.1(15.4)	97.7(1.9)
1c3s5z		95.3(1.2)	100.0(0.0)	100.0(0.0)	96.9(2.6)	100.0(0.6)
10m_vs_11m		82.8(4.1)	95.3(2.2)	81.2(8.3)	89.1(5.5)	98.4(1.4)
2c_vs_64zg	Hard	70.3(3.8)	100.0(0.0)	96.9(3.1)	95.3(3.5)	96.9(1.5)
bane_vs_bane		100.0(0.0)	100.0(46.4)	100.0(0.0)	100.0(0.0)	100.0(5.2)
5m_vs_6m		54.7(3.5)	71.1(9.2)	65.6(14.1)	68.8(8.2)	94.5(2.2)
3s_vs_5z		56.2(8.8)	78.9(4.2)	98.4(5.5)	100.0(1.2)	36.7(29.6)

Table 2: Median evaluation win rate and standard deviation on 5 easy and 4 hard maps. Numbers for QMIX, RODE, MAPPO(FP), and MAPPO(AS) are obtained from the MAPPO paper [Yu *et al.*, 2021]. The highest win rates for each map are bolded. For each map, each algorithm is trained for the same number of time steps as RODE.

Map	ε anneal time (Thousand Steps)
2s_vs_1sc	50
2s3z	50
3s5z	50
1c3s5z	50
10m_vs_11m	50
2c_vs_64zg	50
bane_vs_bane	50
5m_vs_6m	50
3s_vs_5z	300
MMM2	50
3s5z_vs_3s6z	500
27m_vs_30m	50
6h_vs_8z	300
corridor	500

Table 3: ε anneal time of QMIX(SEG) on different maps.

A Hyperparameters

For learning action representations, we choose the dimension of action representations d to be 20 and $\lambda_e = 10$. Following RODE, we train the encoder f_e and the predictors p_o , and p_r in only the first $50K$ steps. It is shown in the RODE paper [Wang *et al.*, 2021b] that $50K$ steps are enough to learn an action encoder that transforms a one-hot action to a representation vector that captures the effect of the action on the environment and other agents. For k -means clustering, same as RODE, we choose the number of clusters to be 3 for homogeneous enemies, 5 for heterogeneous enemies, and 2 for single enemy.

The architecture of the Q-network and the mixing network in QMIX(SEG) follows that of QMIX [Rashid *et al.*, 2018] and are trained using Adam with learning rate $5e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 1e-5$, and with no weight decay. The implementation of QMIX(SEG) is modified from the implementation of QMIX by the Whiteson Research Lab and does not include the implementation tricks in the MAPPO paper. We expect the performance of QMIX(SEG) to further improve if these tricks are implemented. The batch size used is 32. All experiments on the SMAC benchmark follow the default settings [Samvelyan *et al.*, 2019].

The ε for SEG is the same across two levels and annealed from 1 to 0.05 over different time steps for different maps, ranging from $50K$ steps to $500K$ steps. We find that the ε anneal time is an important hyperparameter affecting the performance of QMIX(SEG) on harder maps. We present the ε anneal time of QMIX(SEG) on different maps used in our experiments in Table 3.

B Additional Experiment Results

The performances of QMIX(SEG) and baselines on 5 easy maps and 4 hard maps of SMAC are presented in Table 2. QMIX(SEG) can achieve the highest win rates on all of the easy maps. QMIX(SEG) achieves the highest win rates on 2 hard maps and a win rate close to the highest on another hard map. QMIX(SEG) outperforms QMIX, MAPPO(FP), and MAPPO(AS) on easy and hard maps, obtaining similar or higher win rates than these baselines on all but one map. QMIX(SEG) has close performance as RODE on easy and hard maps.

The only map where QMIX(SEG) struggles to obtain good performance is **3s_vs_5z**. On this map, agents are required to alternate between attacking for one step and moving for several steps. Since QMIX(SEG) does not repeat a group for multiple steps as in RODE, QMIX(SEG) requires more time than RODE to learn to move for multiple steps after attacking once. When we extend the number of steps to train to $5M$, QMIX(SEG) achieves a median win rate of 98.4% with standard deviation 8.92% on **3s_vs_5z**.