Underexplored Subspace Mining for Sparse-Reward Cooperative Multi-Agent Reinforcement Learning

Yang Yu[1,2], Qiyue Yin[1,2], Junge Zhang[1,2], Hao Chen[3], Kaiqi Huang[1,2,4]
[1]*School of Artificial Intelligence, University of Chinese Academy of Sciences*
[2]*Institute of Automation, Chinese Academy of Sciences*
[3]*University of Chinese Academy of Sciences*
[4]*CAS Center for Excellence in Brain Science and Intelligence Technology*
Beijing 100049, P.R.China
yuyang2019@ia.ac.cn, chenhao915@mails.ucas.ac.cn, {qyyin, jgzhang, kqhuang}@nlpr.ia.ac.cn

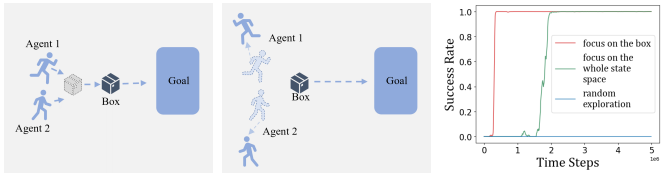
***Abstract*—Learning cooperation in sparse-reward multi-agent reinforcement learning is challenging, since agents need to explore in the large joint-state space with sparse feedback. However, in cooperative games, the cooperative target is often related to partial attributes, hence there is no need to treat the whole state space equally. Therefore, we propose Underexplored Subspace Mining (USM), a novel type of intrinsic reward that encourages agents to selectively explore partial attributes instead of wasting time on the whole state space to accelerate learning. Specially, considering that the target-related attributes are varying in different games and hard to predefine, we choose to focus on the underexplored subspace as an alternative, which is an automatic aggregation of the underexplored bottom-level dimensions without any human design or learning parameters. We evaluate our method in cooperative games with discrete and continuous state space separately. Results demonstrate that USM consistently outperforms existing state-of-the-art methods, and becomes the only method that has succeeded in sparse-reward games evaluated with larger state space or more complicated cooperation dynamics.**

***Index Terms*—sparse-reward cooperation, multi-agent system, reinforcement learning, selective exploration**

I. Introduction

Multi-agent reinforcement learning (MARL) has drawn increasing interest in recent years, since it can help address many challenging real world problems, such as cooperative games [1], [2] as well as robot fleet coordination [3], [4]. Although these MARL methods make significant progress in challenging cooperative tasks, they all rely on dense extrinsic environmental rewards or well-shaped auxiliary rewards designed by human, making the study of sparse-reward cooperative MARL largely absent. Indeed, sparse-reward cooperative setting is common in real world, where agents are required to cooperate over a long-time horizon to obtain a team reward.

More recently, there are some works studying sparse-reward multi-agent cooperative MARL. Specially, influence among agents is captured as intrinsic rewards to encourage agents to do actions that make influence on others [5], [6]. Besides, there are also works using hierarchical control to help agents learn joint exploration skills and the skill selector simultaneously [7], [8]. Although these methods have made process in sparse-reward cooperative problems, they suffer from exploration in the large joint-state space, and become more inefficient when the state space grows exponentially *w.r.t.* the number of agents.



(a) (b) (c)

Fig. 1. Two agents are exploring how to push the heavy box coordinately to the specific area to obtain a reward, where the cooperation target is only related to the position of the box. Thus the movements of the box in (a) is more important than the exploration of the position of agents in (b). And the method which only focuses on the movement of the box as in (a) instead of other unrelated exploration as in (b) will learn faster as shown in (c).

In this paper, we aim to take a step towards solving this problem. Note that in cooperative games, the cooperative target is often related to partial attributes. For instance, in a multi-agent football game, the target is only about the position of the ball, or in a multi-agent fighting game, the target is only related to the health of enemies. Thus there is no need to focus on the exploration in the whole joint-state space, where most interactions are unnecessary and unrelated to the target. Therefore we propose Underexplored Subspace Mining (USM), a novel intrinsic reward which encourages agents to only focus on the exploration in partial subspaces and avoid other unnecessary exploration to accelerate the discovery of rewards. We also demonstrate this phenomenon in Fig. 1.

Specially, considering that the target-related subspace is varying in different tasks and hard to predefine, we choose to focus on the underexplored subspace. This is inspired by the principle of optimism in the face of the uncertainty, which plays a central role in the exploration methods in many fields [9]–[12]. For example, in reinforcement learning (RL) area with sparse reward feedback, this principle encourages agents to visit novel states that are less visited to achieve a more

efficient exploration [13], [14]. Similarly, we extend this idea to the subspace level, and assume that the underexplored subspaces that are less visited are more worthy of being explored in sparse-reward cooperation problems. Concretely, when helping agents focus on the underexplored subspace, we achieve a lightweight subspace identification process, where the underexplored subspace is automatically aggregated from the bottom-level dimensions using unsupervised clustering methods, without any troublesome subspace enumeration or learning parameters.

Finally, we evaluate USM in sparse-reward cooperative games with discrete and continuous state space respectively, and results demonstrate that USM outperforms existing state-of-the-art (SOTA) methods [6], [7], [15] in simple exploration games, and becomes more sample efficient in games with larger state space or more complicated cooperation dynamics.

We summarize our contributions as follow:

- We introduce a novel type of intrinsic reward for agents to do selective exploration to accelerate their learning in sparse-reward multi-agent games.
- We propose a lightweight underexplored subspace identification, which is automatically aggregated from bottom-level dimensions without any troublesome subspace enumeration or learning parameters.
- Our method outperforms existing state-of-the-art methods in both discrete and continuous state space sparse-reward cooperative games.

II. Background

We base USM on the representative multi-agent reinforcement learning algorithm monotonic value function factorization (QMIX) [16]. In this section, we will give a brief introduction to multi-agent reinforcement learning, the representative algorithm QMIX and the research of sparse-reward multi-agent cooperation.

A. Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning aims to address multi-agent sequential decision problems, especially cooperative games, which are usually modeled as Dec-POMDP, i.e., $G = \langle S, U, P, R, Z, O, n, \gamma \rangle$ [17]. S is the state space of the environment. At each time step t, every agent $i \in A \equiv \{1, \ldots, n\}$ chooses an action $u^i \in U$ which forms the joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. P is the state transition function which maps the current state s and the joint action $\mathbf{u}$ to the next state s', i.e., $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$. All agents receive a shared reward $r \in \mathbb{R}$ according to the reward function $R(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$ and $\gamma \in [0, 1)$ is the discount factor. In a partially observable setting, each agent does not have access to the full state and instead samples observations $z \in Z$ according to observation function $O(s, i) : S \times A \to Z$. The action-observation history for an agent i is $\tau^i \in T \equiv (Z \times U)^*$ on which it conditions its policy $\pi^i(u^i|\tau^i) : T \times U \to [0, 1]$. The joint action-value function is defined as $Q^\pi(s_t, \mathbf{u}_t) = E_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t, \mathbf{u}_t \right]$ where π is the joint policy. The object of MARL is to find the joint policy π to maximize the expected sum of the discounted team reward.

To address the non-stationary issue and the scalability problem in MARL, existing MARL algorithms mainly adopt centralized training and decentralized execution (CTDE) paradigm. CTDE means during training, the learning algorithm has access to the full information of all agents, while during execution each agent makes decisions on its own information. Based on CTDE paradigm, many value-based [16], [18]–[20] and policy-based [21]–[24] multi-agent reinforcement learning algorithm have been developed. In this paper, we also adopt the widely used paradigm CTDE, and base our method USM on the predominant value decomposition method QMIX.

B. Value Decomposition Methods and QMIX

As the mainstream of value-based CTDE methods, value decomposition methods try to estimate the joint-action value $Q_{\mathbf{tot}}$ by learning individual agent utilities $Q_i, i \in A$. As a classical value decomposition method, QMIX [16] factors the joint action-value Q_{tot} into a monotonic nonlinear combination of individual utilities Q_i. The individual utility is learned by each agent via a utility network, and a mixer network with nonnegative weights is used for combining agents' utilities. The nonnegativity in the mixer network ensures that $\frac{\partial Q_{tot}(s, \mathbf{u})}{\partial Q_i(\tau^i, u^i)} \geq 0$, which in turn guarantees Individual-Global-Max (IGM) Condition [25], i.e., the optimal joint actions across agents are equivalent to the collection of individual optimal actions of each agent. QMIX is effective since during execution, each agent can independently make decisions by its own utility network without any global information since IGM condition is satisfied.

C. Sparse-Reward Multi-Agent Cooperation

Compared to the vigorous development of multi-agent reinforcement learning in dense-reward cooperative games, the relevant studies in sparse-reward setting is rare until recently. These literatures can be grouped into two lines. One common approach is encouraging agents to make influence on others, where the influence is predefined like the effects on others' transition dynamics [6] or policies [5]. The other line mainly encourages agents to explore environments using combinations of different cooperative skills, which are usually predefined [8] or learnable using information theory [7]. However, both of these two lines encourage exploration in the whole state space, which makes themselves inefficient in large-scale environments.

More recently, there is a new work called cooperative multi-agent exploration (CMAE) [15], which also encourages agents to do selective exploration. Note that although both CMAE and USM propose to focus on the partial space, there are two major differences between them: (1) The focused partial spaces are different. CMAE proposes to focus on the restricted spaces, which are the subset of the global state, and whose number is growing exponentially *w.r.t.* the number of agents. While USM focuses on the underexplored subspace, which is an unsupervised aggregation of the bottom-level dimensions, and

is not sensitive to the number of agents. (2) The frameworks of these two methods are different. For exploration, CMAE needs to train explicit exploration policy in addition to the coordinated policy for each agent, while USM is a bonus-based method without introducing any additional exploration policies, and is easy to be combined with existing MARL algorithms or other advanced bonus-based methods.

III. Method

As discussed above, in USM we introduce a novel intrinsic reward, which encourages agents to only focus on the exploration of the underexplored subspace. Our work adopts the mainstream CTDE paradigm, where agents are trained with access to global information but make decisions based on their own local information in execution. We base USM on the representative CTDE algorithm QMIX, where each agent owns its utility network Q_i for decentralized execution, and there is a monotonic nonlinear mixing network combining these utility networks to estimate the joint state-action value Q_{tot} for training.

In our work, the proposed new intrinsic reward will be combined with the extrinsic reward, then factored by the mixing network together to achieve the implicit credit assignment. Therefore in Dec-POMDP, agents will receive the weighted sum of the intrinsic and extrinsic reward at each time step, $r_t = r_t^{ext} + \omega r_t^{int}$, where ω is the hyperparameter to weigh the importance of both rewards. And the algorithm will be trained by the loss:

$$L(\theta) = \mathbb{E}_{(\boldsymbol{\tau},\boldsymbol{u},r,s)\sim D}\left[\left(y^{tot} - Q_{tot}(\boldsymbol{\tau},\boldsymbol{u},s;\theta)\right)^2\right] \quad (1)$$

where θ are the network parameters of QMIX, D is the replay buffer, $\boldsymbol{\tau}, \boldsymbol{u}, r, s$ are the joint action-observation history, joint-action, intrinsic and extrinsic reward, and global state respectively, Q_{tot} is the output of QMIX, and y^{tot}, i.e., the target value estimation is

$$y^{tot} = r^{ext} + \omega r^{int} + \gamma \max_{\boldsymbol{u'}} Q_{tot}(\boldsymbol{\tau'}, \boldsymbol{u'}, s'; \theta^-) \quad (2)$$

γ is the discount factor, and θ^- are the parameters of a target network as used in DQN [26].

In the following we will introduce the calculation of this intrinsic reward in details, including the determination of the underexplored subspace and the exploration in this subspace separately. For simplicity, we will first introduce these two parts in Section III-A and Section III-B with assumption that the state space is finite and discrete, then discuss the form of USM in continuous state space in Section III-C. The framework of USM is shown in Fig. 2.

A. Underexplored Subspace Identification

USM encourages agents to focus on the underexplored subspace, however, the identification of the underexplored subspace is challenging. On the one hand, the number of all possible subspaces is growing exponentially w.r.t. the number of agents (the number of subspaces for a N-dimensional joint state is 2^N), which makes the enumeration inefficient and unaffordable with limited learning resources. On the other hand, the threshold used for selecting the underexplored one is also unknown. To address the difficulties above, we propose an automatic bottom-level aggregation mechanism achieving a lightweight subspace identification process.

In cooperative games, we observe that each dimension in the state vector describes a high-level attribute of the global state. Besides, according to the principle of optimism in the face of uncertainty, when the reward is sparse, compared to those relatively explored dimensions, the underexplored dimensions are more likely to hinder the achievement of the target, and thus focusing on them is beneficial for agents to discover useful skills and coordinated policies. Therefore, instead of enumerating all possible subspaces and selecting appropriate one, we choose to record the exploration degree of each dimension then cluster the underexplored dimensions as the identification of the focused subspace.

Concretely, agents start to explore randomly, then for a N-dimensional global state, we use N counters to help measure the exploration degree of each dimension separately. As for discrete state space, we use V_i for the i-th dimension state space and normalized entropy e_i to represent the exploration degree of this dimension. There will be counters recording the visitation counts of different states (i.e., possible values in the i-th dimension) in each dimension, and c_i counts the visitation of different states in V_i. Then the state probability distribution in V_i is:

$$p_i(*) = \frac{c_i(*)}{\sum_{v\in V_i} c_i(v)} \quad (3)$$

Thus the normalized entropy, i.e., exploration degree of the i-th dimension is:

$$e_i = \frac{H_i}{H_{max,i}} = \frac{-\sum_{v\in V_i} p_i(v)\log p_i(v)}{log(|V_i|)} \quad (4)$$

where $|V_i|$ represents the number of different states in V_i recorded during exploration. According to (4), the exploration degree of the i-th dimension space is 1 if different states in this dimension are sampled uniformly. Note that state visitation counts are accumulated across episodes, which represents the algorithm's exploration degree of different dimensions during prior training.

After obtaining the exploration degree of all dimensions, an unsupervised clustering method will be used to cluster these dimensions according to this statistic. Here we use k-means to cluster dimensions into k classes. Then the class of the least explored dimensions consists of the underexplored subspace. It is worth noting that this underexplored subspace is identified automatically without troublesome human design or learning parameters. And for a N-dimensional global state, where each dimension has M different values, the sum of all possible subspace is $\sum_{i=1}^{N}\binom{N}{i}M^i$ (where $\binom{N}{i}$ is the combination number of i out of N). However, in USM it only takes a $N \times M$ matrix recording the visitation counts to help identify the underexplored subspace, where each row represents the visitation counts of different states in the related dimension.

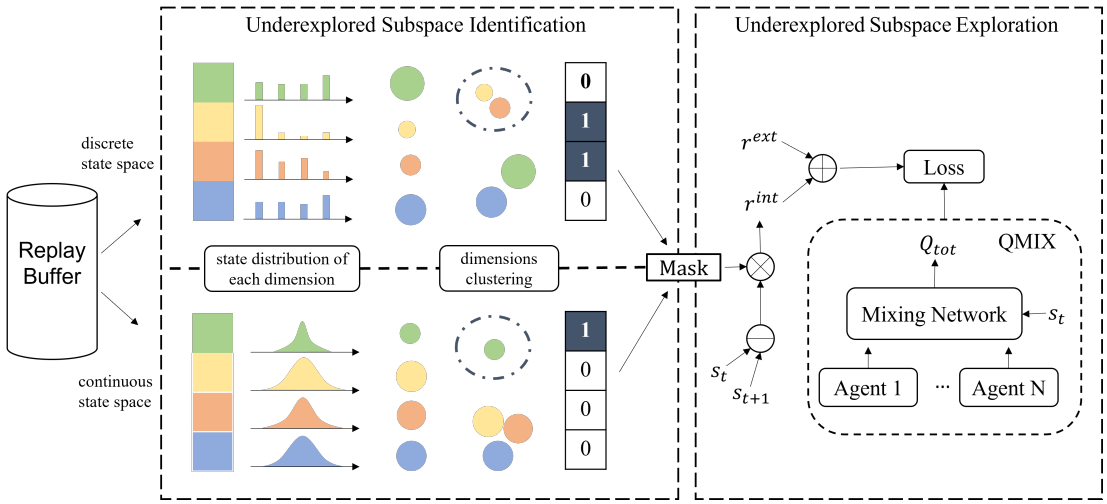


Fig. 2. The framework of USM, which identifies the underexplored subspace and encourages the exploration in this subspace. On the left, the underexplored subspace is identified by clustering bottom-level dimensions according the exploration degree of each dimension, based on the observation that each dimension has its own meaning and exploration difficulty, and underexplored dimensions are more likely to hinder the cooperation. On the right, the identified subspace is used as a mask to help calculate the effects of agents on this focused area, which can be combined with other advanced MARL algorithms easily.

B. Underexplored Subspace Exploration

After identifying the underexplored subspace using the mechanism introduced in Section III-A, the algorithm will know where to pay attention. In other words, the algorithm get a mask vector $\boldsymbol{m}$ made up of 0 and 1, whose dimension is the same as the global state. In the mask, 1 means the related dimension is included in the underexplored subspace and 0 means not. Now the question becomes how to make agents only focus on this part of space. In order to fulfill this demand, the intrinsic reward is designed as:

$$r_t^{int}(s_t, s_{t+1}) = \frac{\sum_{i=1}^{N} \mathbb{I}\left[s_t^i \neq s_{t+1}^i\right] m_i}{\sum_{i=1}^{N} m_i} \tag{5}$$

where N is the dimension of the global state, $\mathbb{I}$ is an indicator function (1 if the i-th dimension of the global state changes; 0 otherwise), and m_i is the i-th dimension of the mask $\boldsymbol{m}$. Equation (5) means this intrinsic reward encourages agents to explore different states, but this difference needs to happen in the underexplored subspace. In other words, only behaviors affecting the underexplored subspace will be encouraged, while others are not, even though they also have an influence on the environment. In addition, the dimension of the focused subspace is also considered in the denominator, thus the reward will be only in relation to the average exploration degree of the focused subspace, no matter what its dimension is.

Furthermore, in order to ensure that agents do not go back and forth between consecutive states to gain intrinsic rewards, which is highly likely to happen especially in grid world environment, we discount the intrinsic reward in (5) by $\sqrt{N(s_{t+1})}$ referring to [13], [27], where $N(s_{t+1})$ is the number of times the global state s_{t+1} has been visited during the past episodes. Hence the overall intrinsic reward used in USM in discrete environment is:

$$r_t^{int}(s_t, s_{t+1}) = \frac{\sum_{i=1}^{N} \mathbb{I}\left[s_t^i \neq s_{t+1}^i\right] m_i}{\left(\sum_{i=1}^{N} m_i\right)\sqrt{N(s_{t+1})}} \tag{6}$$

C. USM in Continuous State Space

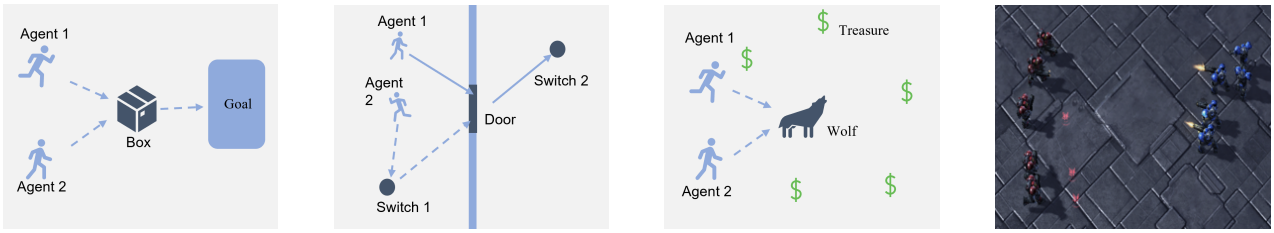
Now we introduce how to identify and explore the underexplored subspace in continuous state space. First for the subspace identification, we still use k-means clustering to divide dimensions of the global state, and the only difference is that the statistic used here is the variance of each dimension not the entropy. Similarly, in order to ensure the comparability among dimensions, all dimensions are normalized according to their own maximum and minimum values, which are recorded and updated periodically. Hence the i-th dimension clustering feature used in continuous state spaces is:

$$var_i = \sigma_i^2 = \frac{1}{n} \sum_{d \in D_i} (d - \overline{D_i})^2 \tag{7}$$

where n is the number of global states sampled during prior training, D_i represents the related i-th dimension data, $\overline{D_i}$ is the average value of data in D_i. Then the clustering result will be used to help determine the underexplored subspace, i.e., the mask.

As for the intrinsic reward form in continuous state spaces, it is nearly the same as the form in (5) except that the change of consecutive states is calculated by the difference not the comparison, therefore the intrinsic reward here is:

$$r_t^{int}(s_t, s_{t+1}) = \frac{\sum_{i=1}^{N} |s_t^i - s_{t+1}^i| m_i}{\sum_{i=1}^{N} m_i} \tag{8}$$



(a) (b) (c) (d)

Fig. 3. The descriptions of tasks in PWE and SMAC. (a) PushBox. (b) Pass. (c) Island. (d) $5m$. A task of SMAC. In these tasks, the extrinsic reward or the largest extrinsic reward will not be given unless the cooperation target is achieved, which encourages agents to explore long-term cooperative strategies.

IV. Experiments

We evaluate USM with SOTA baselines in sparse-reward cooperative games with discrete or continuous state spaces respectively. In the following we will introduce the environments we study, the baselines we compare with, and finally the network architecture and training details in our method.

A. Environments

We evaluate our method on two types of challenging sparse-reward multi-agent cooperative games: (1) a discrete version of multi-agent particle world environment (PWE) [6], [15], [21]; (2) a sparse reward version of the StarCraft multi-agent challenge (SMAC) [2]. These two environments are described below, and the sketch of them are shown in Fig. 3. Table I lists the state space of different tasks.

PWE: We consider 3 tasks in this environment, which are PushBox, Pass and Island. In PushBox, agents are required to coordinately push a heavy box to a specific position. In Pass, agents need to move from left room to right room, and the door between rooms will only open when one of agents occupies the switch in any room. In Island, agents are coordinated to kill the wolf, and there are also treasures to distract agents from cooperation. In these tasks, the extrinsic reward or the largest extrinsic reward will not be given unless the cooperation target is achieved, which encourages agents to explore long-term cooperative strategies.

SMAC: This is a two-team battle game, and each team controls many agent units. The goal of the team is to eliminate all enemy units to obtain the only team reward. This environment also requires agents to learn long-term cooperative strategies, except that the state in this environment is continuous and higher-dimensional. In particular we consider 4 representative tasks of different difficulties in SMAC: 3m, 2m_vs_1z, 3s_vs_5z, 25m.

B. Baselines

In USM, the clustering method in all experiments is k-means, and k is 2. For PWE tasks, there is no common basic algorithms and existing methods adopt different basic algorithms like q-learning method with q-table and prioritized replay buffer [28] used in CMAE [15], or on-policy PPO method [29] with GAE [30] in EITI and EDTI [6]. Here we adopt the mainstream MARL algorithm QMIX combined with TD(λ) and prioritized replay buffer (PER) as our basic algorithm. We call this basic algorithm QMIX(+), and base USM on it. For SMAC tasks, we directly base USM on QMIX, which is popular in dense-reward SMAC tasks.

TABLE I
State Space of PWE and SMAC

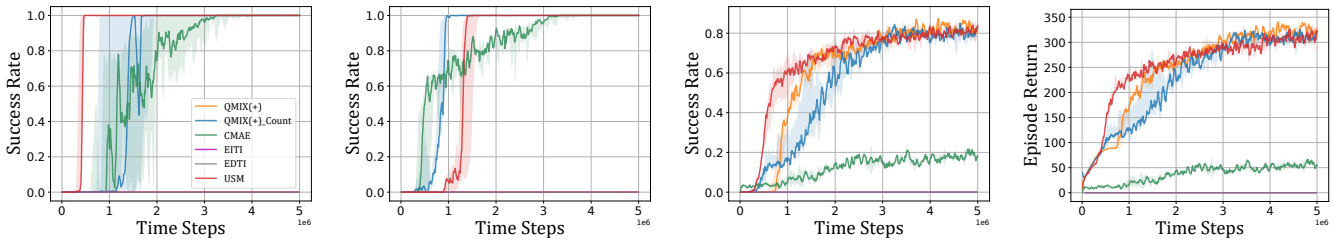
Name	Agents	Dimensions	State Space
PushBox	2	6, discrete	$\approx 1.1 \times 10^7$
Pass	2	5, discrete	$\approx 1.6 \times 10^6$
Island	2	10, discrete	$\approx 1.1 \times 10^{10}$
3m	3_vs_3	21, continuous	-
2m_vs_1z	2_vs_1	12, continuous	-
3s_vs_5z	3_vs_5	35, continuous	-
25m	25_vs_25	175, continuous	-

In order to evaluate the performance of USM, we compare it with SOTA methods that are suitable for discrete or continuous or both environments. For PWE, we compare USM with QMIX(+), CMAE, EITI and EDTI. Besides, considering that count-based method is popular in single-agent exploration problems [13], [31], [32], we provide the results of QMIX(+) combined with count-based exploration, i.e., QMIX(+)_Count, where the intrinsic reward is given when a novel global state is visited. For SMAC tasks, we compare USM with other value-based CTDE methods, QMIX, MAVEN [7] and CMAE. Similarly, we also provide the results of QMIX combined with count-based method RND [14], which evaluates the global state novelty in continuous environment, i.e., QMIX_RND.

Results of baselines we compare with are obtained using the publicly available code released by their authors. We evaluate USM and baselines with the evaluation protocol used in existing methods [2], [16]. Each experiment is repeated using 5 independent training runs with different random seeds, and the resulting plots include the median performance shown in dark color as well as the 25%-75% percentiles shown in the shaded area.

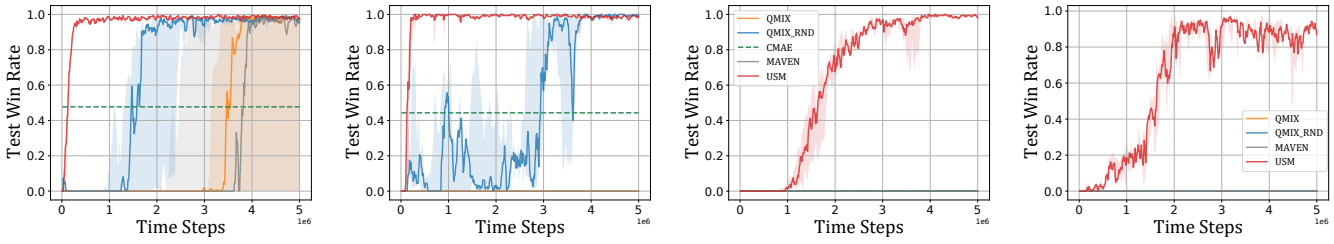
C. Architecture and Training

For all QMIX-based algorithms used in PWE and SMAC, most parameters follow the default setting used in QMIX [16]. As for the hyperparameters of the modules specially



(a) PushBox (b) Pass (c) Island (d) Island

Fig. 4. Learning curves of different methods during training in PWE tasks. (c) and (d) demonstrate the success rate and the episode return of different methods in Island respectively.



(a) $3m$ (b) $2m\_vs\_1z$ (c) $3s\_vs\_5z$ (d) $25m$

Fig. 5. Test win rate of different methods in SMAC tasks. USM becomes the only method succeeding in games with larger state space like $25m$ or complicated cooperation dynamics like $3s\_vs\_5z$

designed in QMIX(+), i.e., TD(λ) and PER, λ in TD(λ) target value estimation is 0.9, and α in PER is 1. In USM, at the beginning, agents will make a random exploration for 10 thousand time steps before doing subspace exploration (about 32 and 128 episodes in PWE and SMAC tasks), and the underexplored subspace is updated every 200 episodes for both environments. The hyperparameter ω used in weighing the intrinsic and extrinsic reward is important, and it is 1 in USM, QMIX(+)_Count and QMIX_RND in all tested tasks.

V. Results

In the following, we will first demonstrate the comparison between USM and other SOTA methods, then provide a visualization to help further analyze USM. At last, an ablation study about the clustering method used in Section III-A will also be discussed to justify this module.

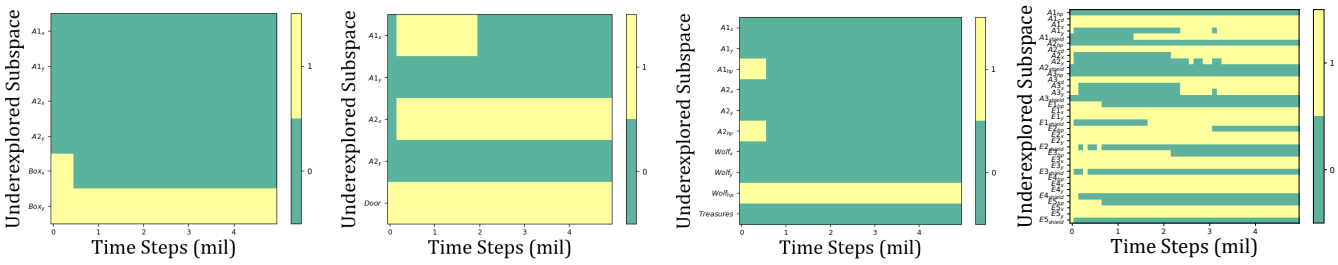
A. Comparison with State-of-the-Art

The comparison of different methods in PWE tasks is shown in Fig. 4, which demonstrates the higher sample efficiency of USM in all tasks. It is worth noting that in PushBox and Island, USM learns faster than QMIX(+)_Count, while in Pass the opposite is true. We suspect this comes from the different characteristics of different tasks. Pass requires agents to find a new room, thus the novel state is needed, making the novelty-seeking method QMIX(+)_Count learn a little faster. While in tasks PushBox and Island, where seeking novelty in the whole state space is more inefficient and unrelated to the target, the advantage of USM with selective exploration is exerted.

Then we compare USM with other baselines in SMAC, and results are shown in Fig. 5. Note that we plot the performance of CMAE with a dotted line using the results presented in its own paper. According to the results we find that in simple exploration tasks like $3m$ and $2m\_vs\_1z$, USM learns at least twice as fast as others. Besides, it is worth noting that in tasks $3s\_vs\_5z$ and $25m$, USM becomes the only method that can solve the task successfully. In fact, $25m$ has a larger state space than $3m$ or $2m\_vs\_1z$ as shown in Table I, hence the performance of USM in $25m$ illustrates the advantage of USM in addressing games with larger state space. Moreover, the task in $3s\_vs\_5z$ is more complicated than that in $3m$ or $2m\_vs\_1z$, where there is a disparity in strength between the two teams. In this game, if agents in team $3s$ want to win, they must learn to scatter enemies and eliminate them one by one instead of only attacking enemies one after another jointly, where the latter strategy is easier to explore. Hence the performance of USM in $3s\_vs\_5z$ shows that it indeed provides agents with more efficient exploration ability, which helps agents not only learn faster in simple exploration tasks, but also succeed in finding more sophisticated coordinated policies in tasks with more complicated cooperation dynamics.

B. Visualization

In order to further explain the performance of USM, we visualize the underexplored subspace identified during training, and results are shown in Fig. 6. Fig. 6(a) shows in PushBox, USM will encourage agents to focus more on the box which is rarely pushed, then help agents learn to push the



(a) PushBox (b) Pass (c) Island (d) 3s_vs_5z

Fig. 6. Identified underexplored subspace (shown in yellow) of USM in different PWE tasks and SMAC task. In SMAC tasks, the state space describes the health, cooldown, x, y locations, shield of each ally unit and the health, x, y locations, shield of each enemy unit.

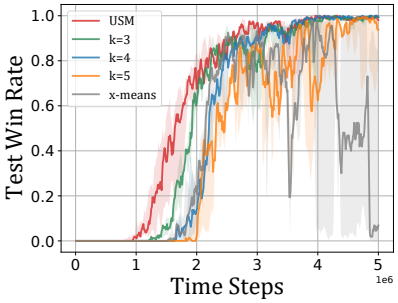
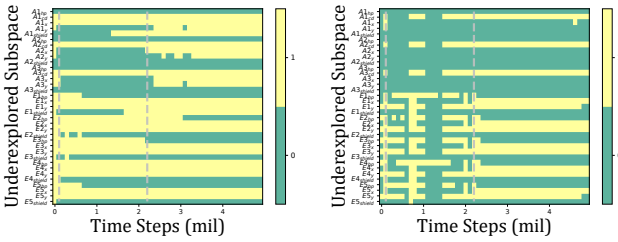


Fig. 7. Ablation study about the clustering method used in USM. Compared with USM, the learning of other numbers of cluster centers is slow and unstable.



(a) USM's focus (b) x-means' focus

Fig. 8. The focused subspace of USM and x-means. There is a difference of the focused area between these two method in different stages divided by gray dotted lines.

box cooridinately. In Pass, Fig. 6(b) shows that USM helps agents choose to explore the subspace of the door at first, after that agents start to visit the right room and entropies of x-axis related dimensions decrease sharply. Then USM encourages agents to explore the subspace consists of x-axis coordinates of agents and the door status, which determines whether both agents can reach the right room. As for in Island shown in Fig. 6(c), compared with other dimensions that are easy to explore, like locations of agents and the wolf, USM first encourages agents to affect the health of agents and the wolf, and the exploration in this subspace will encourages them to meet. Then because the wolf can attack agents automatically while the agents cannot, the health of agents begins to change dramatically, then drops out of the underexplored subspace. After that agents begin to focus on the health of wolf then learn how to kill the wolf gradually. From the development of the underexplored subspace in PWE, we found that in these tasks the underexplored subspace includes important target-related dimensions, and due to the focus on them, agents learn necessary skills for cooperation and their exploration is accelerated greatly.

In SMAC, the development of the underexplored subspace is similar to that in PWE, and we take this process happened in $3s\_vs\_5z$ as an example, which is shown in Fig. 6(d). The global state in this task includes the health, shield, location and cooldown (the minimum delay between attacks) of allies and enemies. The result shows that from the beginning, USM focuses on all dimensions except for the health of allies, since enemies can attack allies automatically and therefore, the health of allies are always changing. After focusing on this underexplored subspace, agents first learn to change their own locations, since later their location-related parts drop out of the underexplored subspace. Then agents start to attack enemies' shield and health, which indeed helps them discover the successful policy for eliminating all enemies. The development of underexplored subspace in $3s\_vs\_5z$ also illustrates that USM will help agents focus on the subspace that is not fully explored, avoid unnecessary interaction and accelerate the learning of cooperation.

C. Ablation Study

In Section III-A, we use k-means where k is 2 to cluster underexplored dimensions. In the following we will perform an ablation study to verify this choice. Specifically, we provide the results of other number of cluster centers (k = 3, 4, 5) and x-means, which can determine the cluster centers automatically. Here we choose a high-dimensional state space $3s\_vs\_5z$ to make the comparison, and the performance of different variations is similar in other tasks. Results are shown in Fig. 7, which demonstrate that with the number of cluster centers growing, the performance is poorer and the performance

of x-means is especially unsatisfactory. We attribute this result to the different inclusiveness in different clustering methods. Note that sometimes the target-related subspace is not fully learned but may not be the most underexplored, therefore the cluster method need to find as many underexplored dimensions as possible, thus k is 2 is conservative but appropriate.

Additionally, we provide the underexplored subspace identified by USM (k=2) and x-means during training in $3s\_vs\_5z$ in Fig. 8. It shows that because x-means always focuses on the most underexplored subspace, it does not include some necessary relatively underexplored dimensions at the beginning, and discards necessary dimensions easily, making the learning process deficient and fluctuant. On the contrary, USM clusters dimensions into two classes will include as more underexplored dimensions as possible at the beginning, and updates the focused subspace gradually, therefore it achieves a more robust performance.

VI. Conclusion

In this paper, we have proposed USM, a novel type of intrinsic reward that encourages agents to do selective exploration, alleviating the inefficient exploration problem in large state space, and achieving a significant performance especially in challenging sparse-reward cooperative games. However, USM currently can only be used in games with state vector input, and how to apply it to pixel-input tasks is worth studying.

VII. Acknowledgments

This work is supported in part by Basic Cultivation Fund project, CAS (JCPYJJ-22017) and the Youth Innovation Promotion Association CAS.

References

[1] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

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

[3] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2012.

[4] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[5] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3040–3049.

[6] T. Wang, J. Wang, Y. Wu, and C. Zhang, "Influence-based multi-agent exploration," in *International Conference on Learning Representations*, 2019.

[7] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "Maven: Multi-agent variational exploration," *Advances in Neural Information Processing Systems*, vol. 32, pp. 7613–7624, 2019.

[8] S. Iqbal and F. Sha, "Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning," *arXiv preprint arXiv:1905.12127*, 2019.

[9] S. Agrawal and R. Jia, "Optimistic posterior sampling for reinforcement learning: worst-case regret bounds," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10] R. Keramati, C. Dann, A. Tamkin, and E. Brunskill, "Being optimistic to be conservative: Quickly learning a cvar policy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4436–4443.

[11] T. M. Moldovan, S. Levine, M. I. Jordan, and P. Abbeel, "Optimism-driven exploration for nonlinear systems," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3239–3246.

[12] I. Szita and A. Lőrincz, "The many faces of optimism: a unifying approach," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1048–1055.

[13] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *Advances in Neural Information Processing Systems*, vol. 29, pp. 1471–1479, 2016.

[14] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *International Conference on Learning Representations*, 2018.

[15] I.-J. Liu, U. Jain, R. A. Yeh, and A. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6826–6836.

[16] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.

[17] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[18] C. Peng, M. Kim, Z. Zhang, and H. Lei, "Vdn: Virtual machine image distribution network for cloud data centers," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 181–189.

[19] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[20] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "Qplex: Duplex dueling multi-agent q-learning," *arXiv preprint arXiv:2008.01062*, 2020.

[21] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, vol. 30, pp. 6379–6390, 2017.

[22] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[23] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for multi-agent actor-critic," *arXiv preprint arXiv:2007.02529*, 2020.

[24] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of mappo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.

[25] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5887–5896.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] R. Raileanu and T. Rocktäschel, "Ride: Rewarding impact-driven exploration for procedurally-generated environments," *arXiv preprint arXiv:2002.12292*, 2020.

[28] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[30] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[31] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[32] H. Kim, J. Kim, Y. Jeong, S. Levine, and H. O. Song, "Emi: Exploration with mutual information," *arXiv preprint arXiv:1810.01176*, 2018.