Multi-Agent Reinforcement Learning with Directed Exploration and Selective Memory Reuse

Shuo Jiang
Northeastern University
Boston, Massachusetts
jiang.shuo@northeastern.edu

Christopher Amato
Northeastern University
Boston, Massachusetts
c.amato@northeastern.edu

ABSTRACT

Many tasks require cooperation and coordination of multiple agents. Multi-agent reinforcement learning (MARL) can effectively learn solutions to these problems, but exploration and local optima problems are still open research topics. In this paper, we propose a new multi-agent policy gradient method called decentralized exploration and selective memory policy gradient (DecESPG) that addresses these issues. DecESPG consists of two additional components built on policy gradient: 1) an exploration bonus component that directs agents to explore novel observations and actions and 2) a selective memory component that records past trajectories to reuse valuable experience and reinforce cooperative behavior. Experimental results verify that the proposed method learns faster and outperforms state-of-the-art MARL algorithms.

CCS CONCEPTS

• **Theory of computation** → **Multi-agent reinforcement learning**; *Gaussian processes*;

KEYWORDS

Multi-agent Reinforcement Learning, Gaussian Process Regression

ACM Reference Format:
Shuo Jiang and Christopher Amato. 2021. Multi-Agent Reinforcement Learning with Directed Exploration and Selective Memory Reuse. In *The 36th ACM/SIGAPP Symposium on Applied Computing (SAC '21), March 22–26, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3412841.3441953

1 INTRODUCTION

Recently, reinforcement learning has shown promising results in solving many challenging problems such as video games [18] and robot control problems [14]. Most reinforcement learning methods are implemented in single agent settings. However, some tasks naturally require multiple agents, which bring new challenges to reinforcement learning such as interaction, coordination, cooperation and adversarial behavior.

A number of methods have been developed for multi-agent reinforcement learning (MARL) [4, 16, 22, 24, 28]. In particular, policy gradient approaches have shown success in generating high-quality solutions for many challenging MARL problems [4, 16, 24]. Unfortunately, current MARL methods still do not perform well in many domains. Two reasons causing a lack of performance are poor exploration strategies and local optima caused by the rarity of sampling coordinated behavior.

First, effective exploration in reinforcement learning is a longstanding problem. One common approach is to use an exploration bonus: assign additional reward to less-visited states in addition to the reward provided by environment. This mechanism encourages the agents to visit novel states, so it is considered a directed exploration strategy. Several directed exploration strategies [1, 5, 11, 13, 23] have been implemented in single agent reinforcement learning, and showed improved sampling efficiency. Few have attempted to combine exploration with MARL [12, 34]. However, current exploration strategies scale poorly as the number of agents increases.

Second, MARL suffers from converging to local optima, especially in cooperative domains that need precise synchronization. Such synchronization requires all the agents to execute particular actions in particular states, the probability that such events can be sampled is inversely exponentially proportional to the number of agents. In such a context, memory reuse (such as replay buffers) has proven to be an effective way to improve performance [22]. The large volume of experiences collected do not have to be treated equally, some samples can be granted higher priority to speed up and improve convergence [6, 9, 10, 20, 26]. This idea has rarely been implemented in a multi-agent setting [7].

In this work, we propose a new multi-agent policy gradient method called decentralized exploration and selective memory policy gradient (DecESPG). DecESPG consists of two additional components over vanilla policy gradient methods: an exploration bonus component and a selective memory component. The two components work in a complementary fashion. Directed exploration prevents the algorithm from prematurely converging by encouraging agents to visit novel states and actions. This is implemented in a decentralized way making it scalable in the number of agents. Selective memory replay ensures valuable experiences are retained and incorporated into learning. Such mechanisms enable agents to explore individual state-action spaces separately while retaining synchronization and cooperative behaviors. Our contributions can be summarized as: (1) Formally define visit novelty problem in Dec-POMDP. (2) Propose an inference method of visit novelty by Gaussian process regression, which is suitable to capture the transient nature of visit novelty. (3) Design a selective memory replay mechanism that records and reinforces cooperative behaviors. (4) Our empirical results on some cooperative learning benchmarks show that DecESPG outperforms state-of-the-art MARL methods

SAC '21, March 22–26, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8104-8/21/03.
https://doi.org/10.1145/3412841.3441953

with accelerated learning speed and strong capability of escaping from local optima.

2 BACKGROUND

2.1 Dec-POMDPs

The decision making for cooperative multi agents under partial observability can be formalized as the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) framework [21]. A Dec-POMDP is defined as a tuple $\langle \mathbb{D}, \mathbb{S}, \mathbb{A}, \mathbf{T}, \mathbf{R}, \mathbf{O}, \mathbf{\Omega}, h \rangle$. $\mathbb{D} = \{1, \cdots, n\}$ is the set of n agents. $\mathbb{S}$ is a set of joint states. $\mathbb{A}$ is the set of joint actions, each is a concatenation of all individual actions $\mathbf{a} = \bigcup_i a^i$. $\mathbf{T}$ is the transition probability $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. $\mathbf{R}$ is the joint reward function. $\mathbf{O}$ is the set of all possible observations, which is the joint of all local observations of each agent. $\mathbf{\Omega}$ is a set of conditional observation probabilities $p(\mathbf{o}|\mathbf{s}', \mathbf{a})$. h is the horizon of the problem. A Dec-POMDP solution is a joint policy, which is a set of individual policies. Each agent's policy is a conditional probability mapping from local observation histories $\tau^{(i)}$ to actions $\Pi^{(i)}\left(a^{(i)}|\tau^{(i)}\right)$. The goal is to learn decentralized policies that maximize the expected total return in given horizon.

$$\Pi^* = \underset{\Pi^{(i)}}{argmax} \underset{a_t^{(i)} \sim \Pi^{(i)}}{\mathbb{E}} \left[\sum_{t=0}^{h-1} \gamma^t \cdot \mathbf{R}(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{1}$$

This symbolizes that, the agents are seeking to optimize a joint objective, but must execute decentralized policies.

2.2 Policy Gradient Algorithms

Several MARL algorithms have been developed for Dec-POMDPs [4, 16, 22, 24, 28] and policy gradient methods are promising for scaling to large action spaces [4, 16]. The methods try to maximize the future accumulated rewards $J(\theta)$ by performing gradient ascent in policy space. When the joint policy Π is parameterized by θ, the gradient can be estimated [29] as:

$$\nabla_{\theta^{(i)}} J(\theta) = \underset{a_t^{(i)} \sim \Pi_\theta^{(i)}}{\mathbb{E}} \left[\Psi_t \cdot \nabla_\theta \log \Pi_\theta \left(a_t^{(i)} | \tau_t^{(i)} \right) \right] \tag{2}$$

Where Ψ_t is the estimated value of the policy.

2.3 Gaussian Process Regression

Gaussian process regression (GPR) is a non-parametric Bayesian inference approach, and it is one of kernel based methods for solving regression tasks. Given a training set $\mathbb{D} = \{x_i, y_i\}_{i=1}^n$, general kernel-based regression methods learn a mapping function $f(\mathbf{x})$ inferred from training set. The function predicts the value $\hat{y}$ of a new query point $\hat{x}$ and can be expanded as a weighted linear combination over $\hat{x}$ and x_i.

$$\hat{y} = f(\hat{x}) = \sum_{i=1}^{n} \alpha_i \cdot k(\hat{x}, x_i) \tag{3}$$

where $k(x_1, x_2)$ is kernel function [25] and α_i are kernel weights. Different types of kernel functions can be chosen depending on applications. As we assume that $\mathbb{D}$ is a random variable under a particular distribution, specifically in GPR, $f(\mathbf{x})$ is distributed following a Gaussian process. By performing Gaussian posterior inference on $p(\hat{y}|\mathbf{x}_i, \mathbf{y}_i, \hat{x})$, the expectation of $\hat{y}$ is given by

$$\mathbb{E}[\hat{y}|\hat{x}, \mathbf{x}, \mathbf{y}] = \mathbf{k}(\hat{x}, \mathbf{x}) \left[\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_\epsilon^2 \mathbf{I} \right]^{-1} \mathbf{y} \tag{4}$$

where $\mathbf{k}(\hat{x}, \mathbf{x})$ is the kernel vector defined by the dot product of $\hat{x}$ and $\mathbf{x}$ and $\mathbf{K}(\mathbf{x}, \mathbf{x})$ is the kernel matrix defined by the dot product of $\mathbf{x}$ and $\mathbf{x}$. σ_ϵ is a noise term to prevent singularity in matrix inversion and $\mathbf{I}$ is the identity matrix.

3 METHOD

In this section, we first formalized visit novelty problem and showed how visit novelty can be inferred by GPR. We then employed such novelty to calculate exploration bonus which is provided as an intrinsic reward to the policy gradient method. After that, we detailed selective memory replay mechanism and how it is combined with exploration bonus for learning in Dec-POMDPs.

3.1 Exploration Bonus

3.1.1 Visit Counts. We first define the novelty function in observation-action space, which is also consistent with the fully observable case by simply replacing the observation history with states. For simplicity in explanation (and experiments), we only use a single observation instead of the full history, but the method could be extended to consider histories. We define $u_t = \left(\bigcup_i o_t^{(i)}, \bigcup_i a_t^{(i)} \right)$ as the joint observation-action pair of a Dec-POMDP and $\tau = \{u_0, u_1, \cdots, u_T\}$ as a sequence of joint observation-action pairs collected in an episode. Here we require that observation-action pair can be represented as a concatenated vector and the distance between observation-action pairs (or their embeddings) is well defined (e.g. Euclidean distance). We also use a smoothness assumption: once an observation-action pair is visited, another 'close' observation-action pair is more likely to be (softly) classified as visited. Then, the similarity between two joint observation-action pairs can be measured by a radial basis function (RBF) kernel.

Definition 1: The *Single Step Visit* of a new observation-action pair u_T with another visited observation-action pair u_t is defined by a radial basis function kernel as:

$$\eta(u_T, u_t) = RBF(u_T, u_t) = C \cdot exp\left(-\frac{\|u_T - u_t\|^2}{2\sigma^2} \right) \tag{5}$$

As the value of such kernel is in range $(0, 1]$, it can be seen as a soft measurement of whether the joint observation-action pair u_T has been visited in the past. If u_T is the same as u_t, then the value of kernel function is 1.

We also consider how often u_T was visited in history. In the tabular case [30], the visit count of a state is a sum of the times that the state has been visited. We can extend that idea to our soft measurement as summing over the single step visit of u_T. When T is large, the sum of all single step visits can grow to a large number, which brings difficulty to transform it to exploration bonus, thus we can bound the value by weighting factors.

Definition 2: The *Bounded Visit Count* of a new observation-action pair u_T in τ at time step T is defined as a linear combination of all past Single Step Visits in between 0 and T with weights α_t, as:

$$\eta(u_T) = \sum_{t=0}^{T-1} \alpha_t \cdot \eta(u_T, u_t) \tag{6}$$

With the proper choice of α_t, the Bounded Visit Count will be bounded in a certain range, e.g. $\eta(u_T) \in (0, 1]$. By comparing Definition 2 and Equation 3, it evidences that the Bounded Visit Count can be found using a kernel based regression framework, and analytically inferred by GPR. We choose GPR over other regression methods [3] because it gives direct analytical solutions and extendable to incremental implementation that compatible with online learning. The expectation of new visit count $\eta(u_T)$ is given by:

$$\mathbb{E}\left[\eta(u_T)|u_T, \tau\right] = \mathbf{k}_{u\tau}^T \cdot \left[\mathbf{K}_{\tau\tau} + \sigma_\epsilon^2 \mathbf{I}\right]^{-1} \cdot \eta(\tau) \tag{7}$$

where $\mathbf{k}_{u\tau}$ is the kernel vector defined crossing u_T and τ, $\mathbf{K}_{\tau\tau}$ is the kernel matrix defined crossing τ and τ. The product of $\mathbf{k}_{u\tau}^T$ and inverse of $\mathbf{K}_{\tau\tau}$ jointly define α_t in Definition 2 as $\vec{\alpha}_t = \mathbf{k}_{u\tau}^T \cdot \left[\mathbf{K}_{\tau\tau} + \sigma_\epsilon^2 \mathbf{I}\right]^{-1}$.

3.1.2 Exploration Bonus. Agents should not only take interest in trajectories with high returns, which can potentially be sampled by a sub-optimal policy, but be encouraged to fully explore the observation-action space. It can be done by granting additional reward as exploration bonus when agent visits novel observation-action pair. Once we have the Visit Counts, we can calculate the exploration bonus.

Definition 3: The *Exploration Bonus (EB)* is defined as:

$$EB(u_T) = -log(\eta(u_T)) \tag{8}$$

where $\eta(u_T)$ is defined in Definition 2. We choose a negative log function, which is also adopted by previous work [5]. As $\eta(u_T)$ is bounded in $(0, 1]$, such function always provides positive Exploration Bonus. The Bonus can be directly added to the extrinsic reward to form an augmented reward as:

$$r'(u_T) = r(u_T) + \beta \cdot EB(u_T) \tag{9}$$

where β is a constant. As the exploration bonus is introduced, new observation-action pairs that are visited less often will be more valued to encourage agents to explore the whole observation-action space.

3.1.3 Visit Assignment. We have already defined Visit Counts in section 3.1.1, which is the basis to calculate the exploration bonus. From the definition, we see that the Visit Count $\eta(u_t)$ is calculated from all previous visited observation-action pairs u_t as in Equation 7. We could use a central buffer recording joint observation-action pairs u_t and Visit Counts $\eta(u_t)$ that provides the same Exploration Bonus to all agents. However, the joint observation-action space for Dec-POMDPs increases exponentially with the number of agents. Remembering all visited joint observation-action pairs and calculating the visit counts will become intractable when the exploration bonus is calculated centrally. Alternatively, we let each agent maintain a private first-in-first-out buffer that remembers past local observation-action pairs $u_t^{(i)}$ and Visit Counts $\eta^{(i)}\left(u_t^{(i)}\right)$ from local histories. Then each agent will infer the local Exploration Bonus from the private buffer by Equation 7 and Definition 3 and the agent's reward will be augmented with Exploration Bonus by Equation 9. In this way, agents are directed to explore their own local observation-action spaces independently. Though joint exploration can be lost, we found that there was no issue in practice and achieved better scalability.

3.1.4 Maintain and Update Visit Buffer. From Definition 2, Exploration Bonus is calculated from all previous visited observation-action pairs u_t. When it is implemented in decentralized way, each agent should maintain a buffer recording all past local observation-action pairs $u_t^{(i)}$ and Visit Counts $\eta^{(i)}\left(u_t^{(i)}\right)$, from which the Exploration Bonus can be calculated for new visiting observation-action pair. As the recorded trajectory grows, visit counts cannot be inferred from all observation-action pairs visited due to complexity limit. In our work, we limit the visit counts to be only calculated from recent m samples. As we employ Gaussian process regression to calculate the bounding factor α_t in Equation 7, when each new $u_t^{(i)}$ comes in, a new inverse matrix $\mathbf{K}_{\tau\tau}^{-1}$ has to be calculated for visit count estimation that takes computation complexity of $O\left(m^3\right)$. To avoid calculating this matrix anew at each step, we employed two incremental inverse matrix update methods before and after the buffer is full (we set the buffer length to m) so the inverse matrix is updated incrementally and efficiently (by complexity of $O\left(m^2\right)$).

Augmented Update (before buffer is full) Assuming the buffer before an update is $\tau_t = [u_0, u_1, \cdots, u_{k-1}]$, the updated buffer with a new observation-action pair added at the end of the buffer is $\tau_{t+1} = [u_0, u_1, \cdots, u_k]$. The corresponding kernel matrices are $\mathbf{K}_k$ and $\mathbf{K}_{k+1}$. The first k columns and rows of $\mathbf{K}_k$ and $\mathbf{K}_{k+1}$ are the same. We partition $\mathbf{K}_{k+1}$ as:

$$\mathbf{K}_{k+1} = \begin{bmatrix} \mathbf{K}_k & \vec{b} \\ \vec{b}^T & d \end{bmatrix} \tag{10}$$

$\vec{b}$ is a $k \times 1$ size vector, d is a scalar. Here we also have known the old inverse matrix $\mathbf{K}_k^{-1}$. Then the new inverse matrix can be calculated by [19]:

$$\mathbf{K}_{k+1}^{-1} = \begin{bmatrix} \mathbf{K}_k^{-1} + g\vec{e}\vec{e}^T & -g\vec{e} \\ -g\vec{e}^T & g \end{bmatrix} \tag{11}$$

where $\vec{e} = \mathbf{K}_k^{-1}\vec{b}$, $g = \left(d - \vec{b}^T\vec{e}\right)^{-1}$.

Sliding Window Update (after buffer is full) Assuming the buffer before updating is $\tau_t = [u_t, u_{t+1}, \cdots, u_{t+m-1}]$, the updated buffer with a new observation-action pair is $\tau_{t+1} = [u_{t+1}, \cdots, u_{t+m-1}, u_{t+m}]$. We denote the kernel matrix before and after incorporating new sample by $\mathbf{K}_{old}$ and $\mathbf{K}_{new}$ respectively. We partition $\mathbf{K}_{old}$, $\mathbf{K}_{old}^{-1}$ and $\mathbf{K}_{new}$ as [15]:

$$\mathbf{K}_{old} = \begin{bmatrix} a & \vec{b}^T \\ \vec{b} & \mathbf{D} \end{bmatrix}, \mathbf{K}_{old}^{-1} = \begin{bmatrix} e & \vec{f}^T \\ \vec{f} & \mathbf{H} \end{bmatrix}, \mathbf{K}_{new} = \begin{bmatrix} \mathbf{D} & \vec{h} \\ \vec{h}^T & g \end{bmatrix} \tag{12}$$

$\vec{b}$, $\vec{f}$, $\vec{h}$ are $m \times 1$ size vectors, a, e, g are scalars, $\mathbf{D}$ and $\mathbf{H}$ are $m - 1 \times m - 1$ matrices, the new inverse matrix can be calculated as:

$$\mathbf{K}_{new}^{-1} = \begin{bmatrix} \mathbf{B} + \left(\mathbf{B}\vec{h}\right)\left(\mathbf{B}\vec{h}\right)^T s & -\left(\mathbf{B}\vec{h}\right)s \\ -\left(\mathbf{B}\vec{h}\right)^T s & s \end{bmatrix} \tag{13}$$

where $\mathbf{B} = \mathbf{H} - \frac{\vec{f}\vec{f}^T}{e}$, $s = \frac{1}{g - \vec{h}^T\mathbf{B}\vec{h}}$.

3.2 Selective Replay Memory

The rarity of optimal trajectories may cause insufficient gradient estimates, and potentially converge to a sub-optimum. We expect the trajectories with high returns can produce long-lasting effect

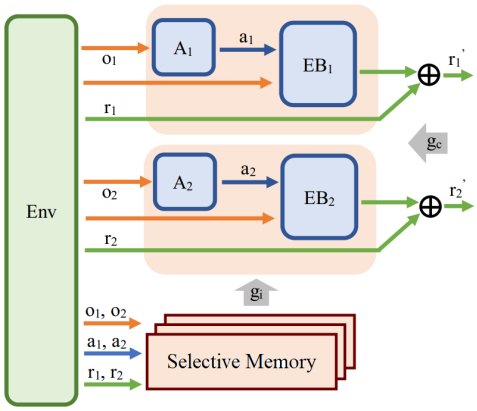

Figure 1: System structure. Two agents have private policy networks (A_1, A_2) and exploration buffer (EB_1, EB_2). Individual rewards r_1, r_2 are augmented by Exploration bonuses and used for on-policy gradient g_c estimation. Selective memory pool recording past good experiences are used for off-policy gradient g_i estimation

on gradient estimation. That is, we can retain high-performing trajectories and train on them in an off-policy manner as well as training on-policy with new trajectories. For episodic problems, considering we have several terminal states $\phi_n \in U$ (this can also be applied to domains with no terminal states simply assuming there's one virtual terminal state at time limit). For each terminal state ϕ_i we keep several trajectories $\tau^{(i)}$ with the highest historical return G_i, and it will be replaced when a better trajectory is sampled. High-performance trajectories will provide additional off-policy gradient estimates. Thus, the total gradient in policy gradient estimate consists of two parts.

$$g_c = \frac{1}{h}\sum_{t=0}^{h-1} \Psi_t \cdot \nabla_\theta \log \Pi_\theta \left(\mathbf{a}_t|\mathbf{o}_t\right) \tag{14}$$

$$g_i = \frac{1}{T}\sum_{t=0}^{T-1} \Psi_t^{(i)*} \cdot \nabla_\theta \log \Pi_\theta \left(\mathbf{a}_t^*|\mathbf{o}_t^*\right) \tag{15}$$

g_c is an on-policy gradient as shown in Equation 2, and g_i is an off-policy gradient estimator [20]. $\mathbf{a}_t$ and $\mathbf{o}_t$ are samples from current policy, $\mathbf{a}_t^*$ and $\mathbf{o}_t^*$ are sampled from past good trajectory. Ψ and $\Psi^{(i)*}$ are values of current policy and past good trajectories respectively. Some other off-policy gradient estimator can also be utilized as [32, 33], but we found this estimator is simple and gets good performances in experiments. The total gradient is calculated as:

$$g = g_c + \frac{\Psi^{(i)*}}{\sum_i \Psi^{(i)*}} \cdot g_i \tag{16}$$

Equation 16 suggests that in addition to the on-policy gradient estimator, the gradient terms from past good trajectories are consistently injected. We expect such valuable experiences will record valid coordination strategies with high return that can be reinforced.

Algorithm 1: Decentralized Exploratory Selective Policy Gradient (DecESPG)

Input: Decentralized policies $\left\{\pi^{(1)}, \pi^{(2)}, \cdots, \pi^{(N)}\right\}$, N individual EB components for decentralized exploration.
for epochs of iteration **do**
  Sample a trajectory τ
  for each element $\{\mathbf{s}_t, \mathbf{a}_t, r_t\}$ in τ **do**
    Calculate EB by Definition 3
    Augment reward with EB by Equation 9
    Remember the visit of $\left\{u_t^{(i)}, \eta^{(i)}\left(u_t^{(i)}\right)\right\}$ by EB components by Equation 11 or Equation 13
  end for
  Update selective memory if return of τ is better than old trajectories.
  Backpropagate on- and off-policy gradients by Equation 16
end for

3.3 Decentralized Control

The system, following the widely adopted centralized training and decentralized execution framework, is shown in Figure (1). For each agent, there is a unique policy network A_i parameterized by θ_i modeling the policy which is a conditional probability $\pi_{\theta_i}\left(a^{(i)}|o^{(i)}\right)$. The local observation o_i and action a_i will be fed into individual Exploration Bonus buffers EB_i to calculate decentralized exploration bonuses by Equation 8, which is then added to extrinsic rewards provided by the environment by Equation 9. Such rewards are used for on-policy gradient g_c estimation by Equation 14. Due to the transient nature of exploration bonus, the off-policy gradient in Equation 15 does not incorporate such exploration bonus and is trained with original reward signal. We also set a selective memory

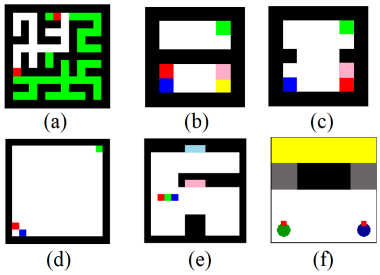

Figure 2: Environments (a) Cleaner, (b, c) FindTreasure, (d) GoTogether, (e) MoveBox, (f) PushBox

pool that records high-performing experiences for terminal states (for episodic tasks) jointly and concurrently for all agents. Each time the team gets a new experience, it will replace the corresponding experience with the same terminal state when the new experience has higher return. Such experiences provide off-policy gradient calculated by Equation 15. The pseudo code for our algorithm is shown in Algorithm 1.

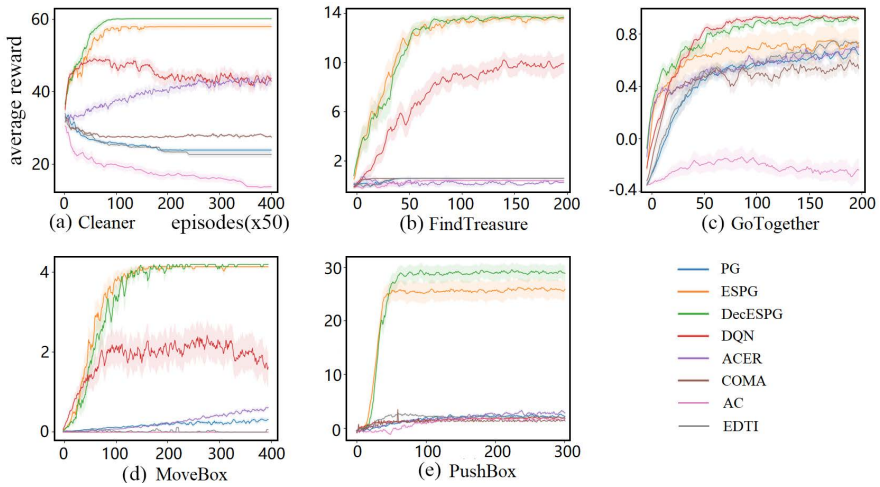

Figure 3: Performance of DecESPG in five domains compared to the baseline methods

4 EXPERIMENTS

4.1 Environments

To show the capability of escaping local optima, we adopted several multi-agent tasks as our benchmarks, some of them have obvious local optima that are hard to escape by previous methods.

Cleaner As Figure 2a shows, agents marked with red will clean a randomly generated maze. The black blocks are walls that agent cannot go through, while green cells are dirty floors that agents should clean. At each step, agent will clean the cell it is standing on, and turn it to white cell (means it is clean), then it will be rewarded +1. Agents start at the left top corner of the room and the observations are agents positions. Actions are going up/down/left/right.

FindTreasure The environment consists of two rooms. Two agents (red and blue) run in the bottom room and a treasure (green) is in the top room. Initially, there is no access to the top room as Figure 2b, but a lever (yellow) can open the secret door Figure 2c at the center wall in the bottom room. The door is open only when one of the agent is standing on the lever cell. Agents get reward +100 when any one stands on the treasure cell, which terminates the episode. Actions are going up/down/left/right/wait. The observations are agents positions. An alternative terminal state is when both agents are at the right two cells on the right edge of bottom room, this results in a reward of +3.

GoTogether In the environment shown in Figure 2d, agents marked with red and blue should find a way to the goal position. When agents are both at the green position, it gets reward of +10. The two agents should keep in certain range, when the agents are too close or too far apart, they will get -0.5 reward. The states are agents positions. Actions are going up/down/left/right.

MoveBox In the environment shown in Figure 2e, two agents (red and blue) have to firstly find the way to be at the right or left cell close to the box (green). Only when both agents are at the side of the box and take the same action (go up/down/left/right), they will carry the box to the corresponding direction. The goal is to move the box to the light blue area in Figure 2e, and each agent will be rewarded for +100. When the box is pushed in the pink area, they will be rewarded +10.

PushBox In the environment shown in Figure 2f [36], two agents are trying to push boxes to the border of the area. There are two types of boxes, large box can only be moved by jointly pushing by two agents, and small boxes can be moved by single agent. Pushing large box to the destination is rewarded +100 and small boxes for +10. Only one agent pushes large box will be punished -5. Actions include move forward, turn left, turn right and wait. the agent can only observe its front cells as empty, teammate, boundary, small box, or large box.

4.2 Baselines

For the baselines, we compared vanilla policy gradient methods using REINFORCE [35] as PG, vanilla Actor-Critic as AC [35] and ACER [32], which is an off-policy policy gradient method. We also tested Q-learning in the form of decentralized DQN with a concurrent replay buffer as introduced in [22] without hysteretic learning rates. Because our environments have discrete actions, MADDPG [16] cannot be directly applied. Instead, we compared with COMA [4], which is a common baseline using a centralized critic (similar to MADDPG). EDTI [31], a recent influence based learning method, is also included in the comparison. Our method, which is based on decentralized REINFORCE combined with Exploration Bonus and Selective Memory components is termed DecESPG. The centralized Exploration Bonus version where all agents share one Exploration Bonus buffer that records the joint observation-action pairs is termed ESPG. All the methods have independent actors (or Q-networks in decentralized DQN) for each agent to execute in a decentralized way. Off-policy methods as DQN and ACER employ concurrent replay buffer which can also be used for decentralized learning. COMA and AC adopt a centralized critic, so represent centralized training for decentralized execution. We use average reward per step in the current episode as the metric. The training details for five domains are shown in Table 1.

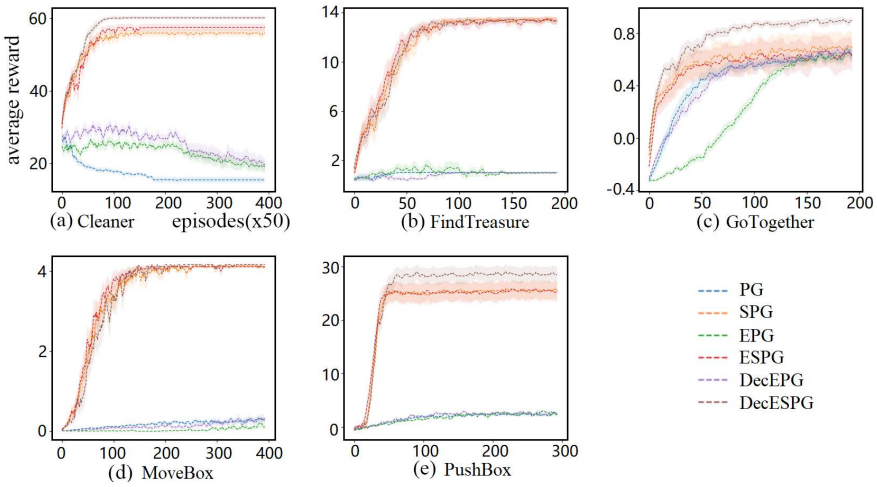

Figure 4: Ablation study of DecESPG in five domains

4.3 Results

Figure 3 shows the training curves with mean and standard error of 20 trials. DecESPG reached the best performance in the most domains followed closely by ESPG. In Cleaner, policy gradient methods (PG and COMA) got worse with training. We observed 'lazy agent' issues in this domain where only one agent is trying to clean the floor due to a joint reward for actions done by the other agent. The two agents share a common path at the beginning and the shared reward possibly caused a credit assignment problem as discussed in [28]. Off-policy methods (DQN and ACER) converged to some sub-optimal paths likely due to lack of exploration. In FindTreasure, there is a local optimum at 1.0 and a global optimum at 14.25. We can see most policy gradient methods converged to a sub-optimum while our methods (ESPG, DecESPG) converge to the global optimum. Similarly, in MoveBox, there is one sub-optimum at 0.77 (pushing box to near goal). Policy gradient methods usually could not exceed it while our methods escape from the local optimum (pushing box to far goal). Also, in PushBox, all baselines learn pushing small boxes with return 0.94-1.88 (push one or two small boxes), but our methods learned to jointly push the large box, approaching global optimum at 33.3. The sampled trajectories of PG and DecESPG are shown in Figure 5. From the visualization, we show DecESPG effectively explores the environment and escapes from local optima.

4.4 Ablation Study

We conducted additional experiments in Figure 4 to show the contributions of Exploration Bonuses and Selective Memory. The baselines are REINFORCE as PG, and PG with only centralized Exploration Bonus buffer as EPG and only Selective Memory as as SPG. The full version with both centralized exploration is ESPG and DecESPG is used for decentralized exploration buffers. We see from the Cleaner and PushBox domains that both mechanisms improved the performance. In the GoTogether domain, the exploration bonus encourages agents to visit novel observation-action pairs, which can violate the distance restriction in this environment. Thus we observe the deteriorated performance of EPG. We also see selective memory is essential in escaping local optima, which is shown in FindTreasure, MoveBox and PushBox domains, which have obvious sub-optimal solutions.

4.5 Single Component Comparison

We also compare with previous exploration strategies and selective memory reuse strategies in Figure 6. Our comparison is based on replacing single components of our method by previous exploration strategy or memory reuse schemes. We adopted exploration strategies EX2 [5], ICM [23] and EMI [13]. EX2 is implemented by replacing GPR components in DecESPG with EX2. We implemented ICM and EMI by replacing GPR in ESPG. ICM and EMI learn the transition dynamics of the domain and translate the prediction error to an exploration bonus. It would be possible to implement them in a decentralized way, but we expect performance to be poor in that case. Thus, we only consider them in centralized learning. EX2, ICM and EMI will provide an exploration bonus based on a given observation-action pair which can be used by Equation 9. Another selective memory component considered is GASIL [7]. We see DecESPG reached the best performance. As EX2, ICM and EMI are neural networks based methods, the drawback is that learning the novelty of state-action pairs requires multiple iterations of gradient propagation. Neural networks can learn the transient nature of observation-action novelty with delay. Conversely, GPR is an inference method that outputs novelties without delay, which can capture such novelty. Also, DecESPG outperforms GASIL in all domains, which converges to local optimum. We believe it is because the discriminator takes too long to converge so the generator converges to a local optimum.

5 RELATED WORKS

Recent literature on directed exploration mainly focuses on how to extract effective visit novelty from high dimensional observations and translate them to an exploration bonus. [1] proposed pseudo-counts to estimate visit counts in high dimensional state space with

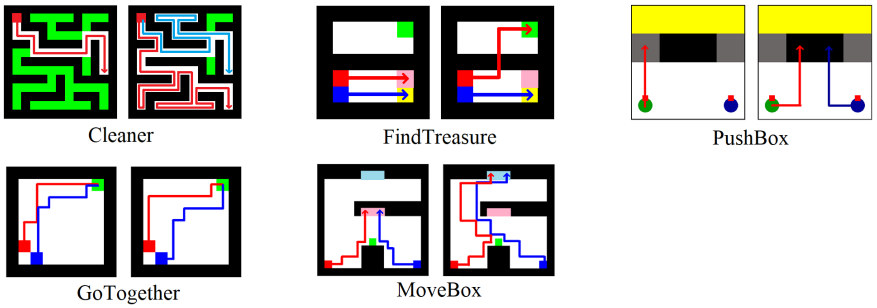

Figure 5: Sampled trajectories of five environments. Left in each pair:vanilla policy gradient. Right: DecDSPG

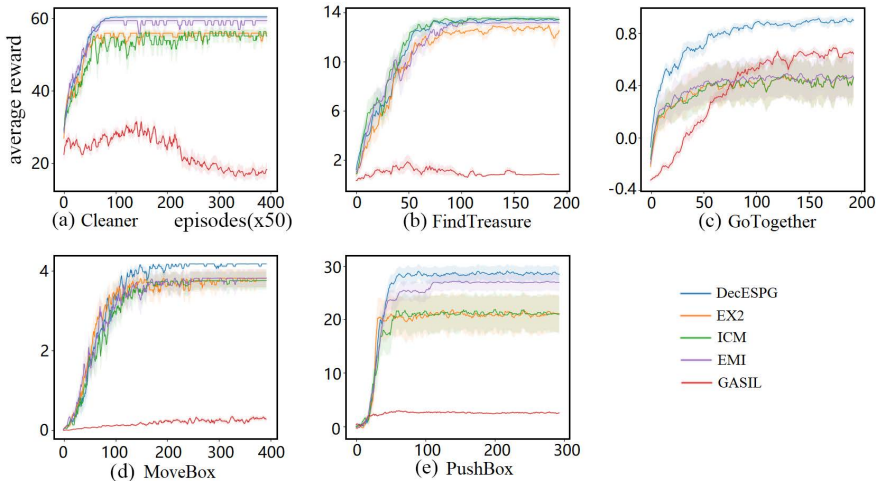

Figure 6: Comparison to previous exploration strategies and memory reuse methods

a density model. [5] proposed a discriminative exemplar model that discriminates the novelty of newly visited states and translates to exploration bonus. Some other works focus on deriving exploration bonuses directly from model dynamic prediction [2, 13, 23, 27]. [31] proposed an influence based exploration strategy, however it also requires learning the transition model and such model is learnt in count-based way which can be hard to generalize in high dimensional state space. Model learning in multi-agent domains is much harder than in single agent settings. Also, models learned in a decentralized way only based on local information can be poor, limiting the efficacy of implementing these methods in MARL. In contrast to previous methods, our work encourages exploration without such model learning and employs inference to overcome the delayed novelty capturing problem with neural networks.

Sample efficiency also affects training performance to a large extent, which is a pain point in on-policy algorithms such as policy gradient methods. [26] proposed prioritized experience replay assigning significance to each instance in replay buffer of Deep Q-Network [17] which is proportional to the temporal difference error. They showed such a mechanism leads to substantial improvements in most Atari games [9, 10]. [20] proposed to exploit the agent's past good experiences to improve learning, which shares similarity with our selective memory. [7] combined such a method with GAIL [8] in multi-agent settings to record and reinforce valuable skills found. Our work incorporates selective memory reuse in multi-agent setting to record rare synchronized and cooperative experiences. Such experiences are learnt by simple off-policy gradient which provides robust and solid improvement.

6 CONCLUSION

This paper proposed a novel framework for policy gradient based multi-agent reinforcement learning. Two mechanisms were introduced to address effective exploration and the experience reuse problem. We firstly formalized the definition of visit novelty in the Dec-POMDP framework. Then we proposed a new way of calculating an exploration bonus by incremental GPR, which can be deployed in both centralized and decentralized manners. The exploration bonus is used as an intrinsic reward in the policy gradient method. We also proposed a selective memory mechanism that reuses valuable past trajectories in multi-agent learning. The selective replay memory injects additional off-policy gradients in the policy gradient method. The two mechanisms complement each other: the exploration encourages agents to fully explore the observation-action space and selective memory reuses past experiences to encourage convergence to a global optima. Results in the

Table 1: Implementation details for DecESPG

environment	Cleaner	FindTreasure	GoTogether	MoveBox	Push box
network structure	256 neurons, ReLU 128 neurons, ReLU 4 neurons, Softmax	64 neurons, ReLU 5 neurons, Softmax	64 neurons, ReLU 4 neurons, Softmax	256 neurons, ReLU 128 neurons, ReLU 4 neurons, Softmax	64 neurons, ReLU 32 neurons, ReLU 4 neurons, Softmax
discount factor	τ = 0.99	τ = 0.98	τ = 0.98	τ = 0.999	τ = 0.95
maximal episode length	l = 200	l = 200	l = 2000	l = 200	l = 200
number of episodes	e = 20000	l = 10000	l = 10000	l = 20000	l = 20000

experiments showed that the proposed method outperformed some state of the art algorithms with accelerated training speed and an ability to escape from local optimum.

ACKNOWLEDGEMENTS

This work was partially funded by the U.S. Air Force Research Laboratory (AFRL), BAA Number: FA8750-18-S-7007.

REFERENCES

[1] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying Count-based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*. 1471–1479.

[2] Gino Brunner, Manuel Fritsche, Oliver Richter, and Roger Wattenhofer. 2018. Using State Predictions for Value Regularization in Curiosity Driven Deep Reinforcement Learning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 25–29.

[3] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. 1997. Support Vector Regression Machines. In *Advances in Neural Information Processing Systems*. 155–161.

[4] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[5] Justin Fu, John Co-Reyes, and Sergey Levine. 2017. Ex2: Exploration with Exemplar Models for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*. 2577–2587.

[6] Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. 2018. Generative Adversarial Self-imitation Learning. *arXiv preprint arXiv:1812.00950* (2018).

[7] Xiaotian Hao, Weixun Wang, Jianye Hao, and Yaodong Yang. 2019. Independent Generative Adversarial Self-Imitation Learning in Cooperative Multiagent Systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1315–1323.

[8] Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

[9] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. 2018. Distributed Prioritized Experience Replay. *6th International Conference on Learning Representations (ICLR 2018)* (2018).

[10] Yuenan Hou, Lifeng Liu, Qing Wei, Xudong Xu, and Chunlin Chen. 2017. A Novel DDPG Method with Prioritized Experience Replay. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 316–321.

[11] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*. 1109–1117.

[12] Shariq Iqbal and Fei Sha. 2019. Coordinated Exploration via Intrinsic Rewards for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1905.12127* (2019).

[13] Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. 2019. EMI: Exploration with Mutual Information. In *International Conference on Machine Learning*. 3360–3369.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous Control with Deep Reinforcement Learning. *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (2016).

[15] Weifeng Liu, Jose C Principe, and Simon Haykin. 2011. *Kernel Adaptive Filtering: A Comprehensive Introduction*. Vol. 57. John Wiley & Sons.

[16] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602* (2013).

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529.

[19] Duy Nguyen-Tuong and Jan Peters. 2010. Incremental Sparsification for Real-Time Online Model Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 557–564.

[20] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-Imitation Learning. *Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.* (2018).

[21] Frans A Oliehoek, Christopher Amato, et al. 2016. *A Concise Introduction to Decentralized POMDPs*. Vol. 1. Springer.

[22] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep Decentralized Multi-Task Multi-Agent Reinforcement Learning under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2681–2690.

[23] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-Driven Exploration by Self-Supervised Prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 16–17.

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Proceedings of the 35-th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018* (2018).

[25] Carl Edward Rasmussen. 2003. Gaussian Processes in Machine Learning. In *Summer School on Machine Learning*. Springer, 63–71.

[26] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952* (2015).

[27] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. 2015. Incentivizing Exploration in Reinforcement Learning with Deep Predictive Models. *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. 2016* (2015).

[28] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. *Sunehag, Peter, et al. "Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward." AAMAS. 2018.* (2018).

[29] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

[30] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*. 2753–2762.

[31] Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. 2019. Influence-Based Multi-Agent Exploration. *8th International Conference on Learning Representations (ICLR 2020)* (2019).

[32] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample Efficient Actor-Critic with Experience Replay. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (2016).

[33] Paweł Wawrzyński. 2009. Real-Time Reinforcement Learning by Sequential Actor–Critics and Experience Replay. *Neural Networks* 22, 10 (2009), 1484–1497.

[34] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent Soft Q-learning. In *2018 AAAI Spring Symposium Series*.

[35] Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine learning* 8, 3-4 (1992), 229–256.

[36] Yuchen Xiao, Joshua Hoffman, and Christopher Amato. 2019. Macro-Action-Based Deep Multi-Agent Reinforcement Learning. In *3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.*