Curiosity-driven Exploration in Sparse-reward Multi-agent Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sparsity of rewards while applying a deep reinforcement learning method negatively affects its sample efficiency. A viable solution to deal with the sparsity of rewards is to learn via intrinsic motivation which advocates for adding an intrinsic reward to the reward function to encourage the agent to explore the environment and expand the sample space. Though intrinsic motivation methods are widely used to improve data-efficient learning in the reinforcement learning model, they also suffer from the so-called detachment problem. In this article, we discuss the limitations of intrinsic curiosity module in sparse-reward multi-agent reinforcement learning and propose a method called I-Go-Explore that combines the intrinsic curiosity module with the Go-Explore framework to alleviate the detachment problem.

1 Introduction

In a reinforcement learning (RL) task, an agent learns from the feedback given by the environment. In contrast to other machine learning paradigms, an advantage of using reinforcement learning is the ability to solve complex learning tasks without expert domain knowledge. Particularly, in multi-agent reinforcement learning, the agent often operates in a complex environment. For instance, [LZY20] considers the problem of learning an optimal courier dispatching policy in a multi-agent dynamic environment.

A reinforcement learning problem is typically modeled using a Markov decision process (MDP) [KLM96, SB18]. An MDP is defined by a tuple that is formed by the state set, the action set, the transition function, and the reward function. The state set and the action set are typically assumed to be known to the agent while the transition function and the reward function are unknown to the agent. When an agent takes a particular action in a particular state, it receives a reward given by the reward function and the environment transitions to a state given by the transition function. In a reinforcement learning problem, the learning goal is formalized as the outcome of maximizing the obtained cumulative reward. In order to maximize the cumulative reward, a reinforcement learning agent faces a significant challenge known in the literature as the *exploration-exploitation dilemma*. An agent may choose actions tried in the past and found to be rewarding (i.e. exploitation) or it may choose unexplored actions to see if they are more rewarding (i.e. exploration). Without sufficient exploration, an agent might not be able to find the optimal solution to the reinforcement learning problem. On the other hand, excessive exploration does not contribute to the goal of maximizing the cumulative reward and hence it represents sample inefficiency. Thus, the ability to efficiently explore the environment remains key to sample-efficient reinforcement learning.

Exploration in reinforcement learning is often driven via rewards and hence the density of rewards in the environment significantly influences the efficiency and thus sustainability of the model. The efficiency is presented by the training time of the model and the execution time to achieve certain goals. Sustainability measures the feasible range of the model. Sparsity in the neural networks might

Submitted to 16th European Workshop on Reinforcement Learning (EWRL 2023). Do not distribute.

improve the learning efficiency, however, a sparse-reward environment severely reduces the efficiency and sustainability of most reinforcement learning algorithms. Consequently, the sparsity of rewards diminishes the applicability of reinforcement learning to real-life problems as shown by [WZW+20].

A feasible solution to this problem is to improve the exploration efficiency to get as much reward as possible. For example, a solution known as *reward shaping* uses some simulated positive rewards in the environment to encourage more exploration of the actual rewards from the interaction with the environment. However, reward shaping is sensitive to reward density in a sparse-reward environment. Another possible solution is to apply intrinsic motivation techniques during the learning process. Intrinsic motivation rewards the agent for exploring new states via an intrinsic reward. As explained in [AVRDS22], there are four approaches to achieve intrinsic motivation – 1) *count-based*, 2) *prediction-error*, 3) *random network distillation*, and 4) *rewarding impact-driven exploration*. These methods are able to improve exploration in most cases, though they also have some drawbacks, such as *derailment* and *detachment*. Derailment describes a situation in which the agent finds it hard to get back to the frontier exploration in the next episode since the intrinsic motivation rewards the seldom-visited states. When the intrinsic rewards run out during the exploration, the agent finishes the learning in the current episode and goes back to the starting state for the next episode. However, the agent is expected to go further from the end position in the previous episode which is not attractive to the agent since some states are already visited in the last episode. This situation is known as detachment. The detachment and derailment issues are specially discussed in intrinsic motivation methods [EHL+21]. Especially for the prediction-error approach, one of the popular implementations is via *intrinsic curiosity module* (ICM). This is due to the ability of ICM to deal with complex environments and domains [YCHL19, LWKO22, AVRDS22]. Accordingly, we use ICM as a baseline for our proposed solution.

This article focuses on improving the prediction-error intrinsic motivation method in the multi-agent reinforcement learning task to improve its sample efficiency in sparse-reward environments. In this work, we consider a multi-agent game theoretic environment to simulate a complicated spares-reward environment that is close to reality. We focus on a fully cooperative environment so that all agents share the same task. In the considered multi-agent environment, the proposed method can show the impact on exploration in cooperation as well as competition. Each agent can only observe parts of the environment and this partial observability increases the complexity of the problem. In this article, we choose *counterfactual multi-agent*(COMA) policy gradient as the learning algorithm with ICM.

COMA supports the centralized training decentralized execution framework and centralized critic benefits from the counterfactual baseline to calculate A-value for each agent within a single forward pass, it is an efficient learning algorithm in this multi-agent environment. We use ICM which provides the intrinsic reward function for the motivation of exploration to the agent team and the COMA algorithm allows the agent to get the external reward from the interaction with the environment.

The main contributions of this work are as follows :

1. We propose a novel solution to improve the training time and to find the optimal solution efficiently. Our solution also indicates a potential method to learn efficiently without using larger more complicated models.
2. We analyze the drawbacks of some existing solutions in a sparse-reward environment and compare the performance with the proposed method in a multi-agent environment.
3. Based on the provided experiment results, we provide some future directions to further improve the proposed method paving the way for improving the efficiency of the reinforcement learning methods.
4. Our work shows a way to deal with the sparsity and improve data-efficient learning in a sparse deep reinforcement learning environment by expanding the replay experience for an existing method.

1.1 Related Work

The sparse-reward environment is commonly encountered in real-world applications, especially in multi-agent reinforcement learning problems. Using intrinsic motivation shows a significant improvement in improving the sample efficiency in sparse-reward environments, for instance training a grandmaster in StarCraft [VBC+19], training the defensive escort team [SB20] etc. However,

as [DRSJ+22] demonstrated, exploration using ICM suffers from problems like detachment and inadequate exploration quality when used with limited observations. [BESK18] also show that random network distillation (RND), another intrinsic motivated method, is sufficient to motivate the agent to explore efficiently in a short-term decision process, though it has a lower performance in a long-term goal. For instance, [BESK18] consider *Montezuma's Revenge*. Montezuma's Revenge is an atari game in which the agent tries to collect useful items to escape with some actions, like jumping, running, sliding down poles, and climbing chains and ladders. However, the agent might get stuck if it focuses on short-term rewards (like collecting the items or fighting with enemies) but moves away from the exit door. RND can help the agent to get the key, however, it does not realize that the key should be saved in a long-term strategy due to the limitation of the key. Another solution is proposed in [HVP+18] which applied the pre-training with demonstration data to expand the replay buffer. The improved experiment results in the same environment, Montezuma's Revenge, indicate that expanding the replay buffer might alleviate the detachment and derailment issue. However, collecting the domain knowledge and generating demonstration data requires extra knowledge of each experiment environment which might be unavailable or not conveniently attainable.

Go-Explore [EHL+21] is another method that aims to increase the efficiency of exploration. In [EHL+21], it shows a significant exploration performance in 11 hard-exploration and unsolved games in the Atari suits. Benefiting from the exploration, Go-Exploration method outperforms the intrinsic motivation method comparing the reward in the evaluation. However, Go-Explore is more fragile in the multi-agent environment due to its complexity. In this article, we propose to combine the features of Go-Explore and the ICM framework with the aim of solving detachment and derailment issues.

2 Background

We design a decentralized partially observable environment involving multiple agents to simulate the real-world environment. The naive ICM method and the improved ICM method are implemented on a *counterfactual multi-agent*(COMA) policy gradient algorithm. To quantify the improvement, we design a complicated sparse-reward environment in which a few specific states result in positive rewards. By comparing the total reward of our proposed method, I-Go-Explore, with the total reward of the baseline naive ICM method, we showcase the performance improvement of the proposed method.

We consider the decentralized partially observable Markov game to represent a fully cooperative multi-agent Markov game(Dec-POMDP). Dec-POMDP is defined as a tuple $< \mathcal{S}, \mathcal{U}, \mathcal{T}, R, \otimes, \mathcal{O}, \gamma >$, where :

- n agents where each agent is recognized as $i \in \mathcal{A} \equiv 1, ..., n$.
- $\mathcal{S}$ is the environment state space, with $s \in \mathcal{S}$.
- $\boldsymbol{\mathcal{U}}$ is the joint action space for each agent action set. At each time step, U is the joint action of all agents where $u^i \in U$, and $\mathbf{u} \in \boldsymbol{\mathcal{U}} \equiv U^n$.
- $\mathcal{T}$ is the state transition probability set and P is the transition probability function, $\mathcal{T}(u, s, s') = P(s'|s, \mathbf{u}) : \mathcal{S} \times \boldsymbol{\mathcal{U}} \times \mathcal{S}$.
- R is the reward function and all agents share the same reward function.
- $\mathcal{O}$ is partial observation probability set and $\boldsymbol{\mathcal{O}}$ is the joint observation space for each agent observation set. At each time step, $\mathcal{O}(s', i, o) = P(o|s', i) : \mathcal{S} \times \mathcal{A} \mapsto \otimes$.
- $\gamma \in [0, 1]$ is the discount factor.

With the definition of Dec-POMDP, the goal of this game is to find an optimal policy to guide the game so that agents can earn as many rewards as possible. The stochastic policy π^i for each agent is conditioned by its action-observation history $\tau^i \in \mathcal{T}$, where $\pi^i(u^i|\tau^i)$:$\mathcal{T} \times \mathcal{U} \rightarrow [0, 1]$. At time t, the accumulated reward $G_t = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$. The expected G_t is denoted as state value function: $V^\pi(s_t) = \mathbb{E}[R_t|s_t]$. And the action value function $Q^\pi(s_t, \mathbf{u}_t) = \mathbb{E}[R_t|s_t, \mathbf{u}_t]$. Therefore, the advantage function is calculated as $A^\pi(s_t, \mathbf{u}_t) = Q^\pi(s_t, \mathbf{u}_t) - V^\pi(s_t)$. Based on these definitions, policy gradient is a technique to find the optimal parameters of the policy. In single-agent environment, the policy parameters is denoted as θ^π and the gradient is presented as follow[SB18]:

$$g = \mathbb{E}[\sum_{t=0}^{\infty} R_t \nabla_{\theta^\pi} \log \pi(u_t|s_t)]. \tag{1}$$

Actor-critic algorithms combine value-based and policy-based methods so that the actor learns the policy by the policy gradient technique and the critic estimates the value function. And a baseline $b(s_t)$ is introduced during the reward function R_t calculation to reduce the variance. Temporal difference(TD) error is one of the common methods to implement baseline for the reward function, where $G_t = R_t + \gamma V(s_{t+1}) - V(s)$. For a multi-agent environment, actor-critic structure also supports centralized training-decentralized execution. Agent's information can be shared by using the same parameters in the actor-network or training the actors without sharing the parameters. In this paper, the critic network in COMA shares the agents' information during the training to accelerate the process and each agent has an independent actor network to execute the action according to its action-observation history. More details are provided in the next section.

3 Methodology

In this section, we describe the methodology used in our article.

3.1 Counterfactual Multi-agent policy gradient(COMA)

The most contribution of COMA is adding a counterfactual baseline to advantage function to deal with the credit assignment in centralized critic and accelerate the evaluation during the training. COMA has the actor-critic framework to perform centralized training and decentralized execution. The actor-network is distributed to execute the action for each agent and all agent's information is collected by the same critic network to communicate the information among agents and evaluate the actor networks' performances in the learning phase. Following the definitions in the previous section, the policy in each actor is denoted as $\pi^i(u^i|\tau^i)$, the action value function is $Q(\tau^i, u^i)$ and the state value function is $V(\tau^i)$. τ is the action-observation history that is implemented by GRU.

COMA updated the Q-values with on-policy style with TD error, and the gradient is $g = \nabla_{\theta^\pi} \log \pi(u|\tau_t^i)(r + \gamma V(s_{t+1}) - V(s_t))$. However, the r is the total returns for all agents. Hence the counterfactual baseline is introduced to solve the credit assignment. Counterfactual baseline is inspired by difference rewards[WT01, TA07] that $D^i = r(s, \mathbf{u}) - r(s, (\mathbf{u^{-i}}, \mathbf{c^i}))$, where $r(s, \mathbf{u})$ is the total rewards of all agents at state t, $\mathbf{u^{-i}}$ presents the joint actions of all agents except agent i and c^i is the default action for agent i. COMA can use the centralized critic to estimate Q-values for the joint actions and compute the Q-value for the individual by advantage function. The advantage function computes the state value by marginalizing one agent's action and fixing the rest joint actions, for instance,

$$A^i(s, \mathbf{u}) = Q(s, \mathbf{u}) - \sum_{u'^i} \pi^i(u'^i|\tau^i) Q(s, (\mathbf{u}^{-i}), u'^i), \tag{2}$$

where u'^i is the default action for agent i. With the counterfactual baseline, at each iteration k, the policy gradient is

$$g_k = \mathbb{E}[\sum_i^n \nabla_{\theta_k} \log \pi^i(u^i|\tau^i) A^i(s, \mathbf{u})], \lim_{k\to\infty} \parallel \nabla J \parallel = 0 \wp 1. \tag{3}$$

where J is the expected rewards. The structure of the critic network is shown in Figure1.

The complete COMA framework is shown in figure 2

3.2 Intrinsic Curiosity Module

The ICM method [PAED17] motivates the agent to explore in a multi-agent environment by giving an additional reward signal as an intrinsic reward. The intrinsic reward is the prediction error which is the difference between the predicted next state and the actual next state. The more disparate the prediction and the reality, the higher the returned intrinsic reward. For the decentralized execution, each agent has a private intrinsic reward module. ICM is composed of three parts: *encoder*, *forward model*, and *inverse model*.

The encoder abstracts the feature from the input state which is denoted as $\phi(S_t)$. Let f be a function representing the forward model which generates a prediction of the next state with the current state feature and the action taken as input i.e.,

$$o'_{t+1} = f(\phi(S_t), \mathbf{u}_t), \tag{4}$$

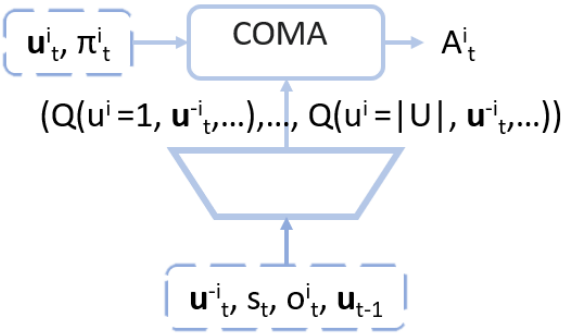


Figure 1: The critic network calculates for the value function for each agent and joints the results for advantage function.

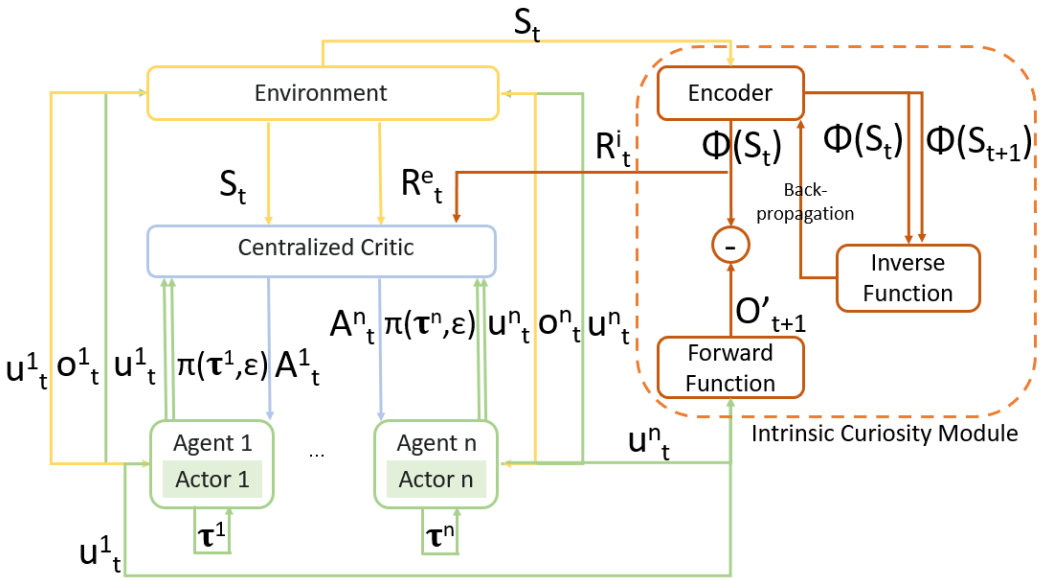


Figure 2: COMA structure and intrinsic reward function. COMA is structured in 3 parts(green, yellow, and blue). The green part presents the actor networks, the yellow part presents the environment, and the blue part is the critic network. The red part is the intrinsic reward function.

where o'_{t+1} is the prediction of the next state. The inputs in the forward function are the encoded state and the joint actions. The loss function used is the prediction error

$$\mathcal{L} = \frac{1}{2}||o'_{t+1} - o_{t+1}||_2^2. \tag{5}$$

The intrinsic reward is calculated using the following function h which quantifies the novelty:

$$R_t^i = h(o'_{t+1}, o_{t+1}), \tag{6}$$

where R_t^i is the intrinsic reward at time t for agents. The last part in the intrinsic curiosity module is the inverse model which uses the feature of the current observation and the feature of the next observation to predict a possible action at the current state. Then the inverse prediction error is used to train the inverse model and the encoder. The complete details of the ICM are shown in Figure 2.

Reward Function in COMA The reward function is divided into two parts – external reward signal and intrinsic reward signal. We focus on the fully cooperative task in this paper and the aim of the intrinsic reward is to drive the agent team to explore. Therefore, the reward can be denoted at time t

as:

$$R_t = R_t^i + R_t^e, \tag{7}$$

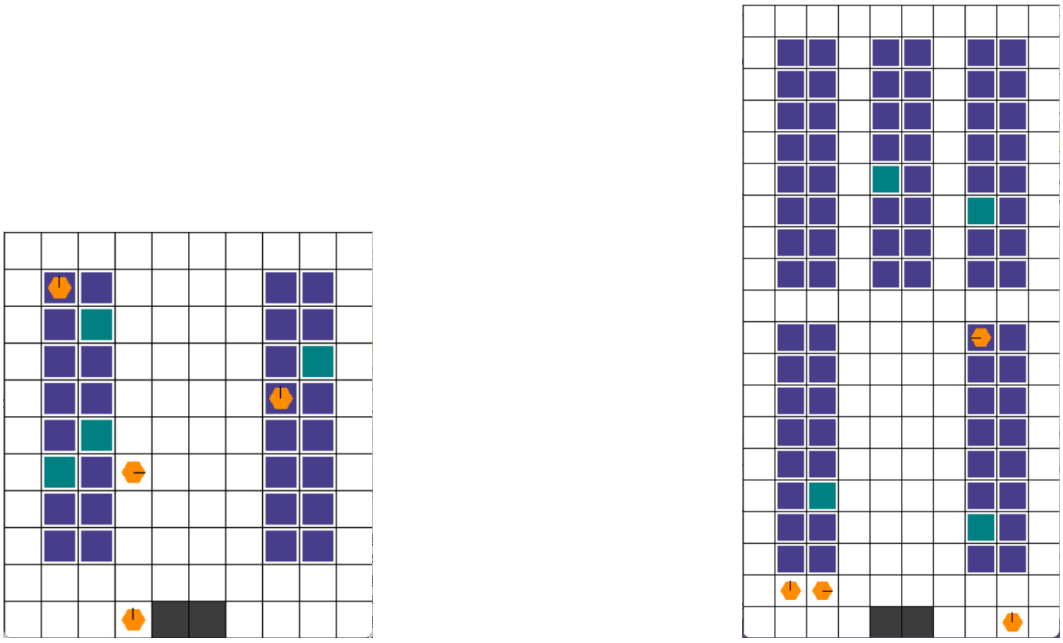
where R_t is the total reward, R_t^i is the intrinsic reward and R_t^e is the external reward. The intrinsic reward is generated using the ICM method. The external reward is decided by the environment. This reward function is used with the COMA method.

3.3 I-Go-Explore

Go-Explore [EHL+21] starts the exploration from the achieved state which alleviates the problems of detachment and derailment for intrinsic motivation methods. The Go-Explore method has two phases: exploration and robustify. In the exploration phase, it first selects a state from the archive, simulates the chosen state, explores from the simulated state, and finally updates the exploration result to the archive. The archive is the visited state's space and the selection is based on the probability distribution or a heuristic method. As mentioned in [EHL+21], detachment might happen when the intrinsic reward guides the agent to the frontier of the exploration and it loses interest in going back to the exploration state. Therefore, we propose a solution to add an exploration phase at the end of each episode to give additional motivation so that the experience in the replay buffer is expanded. This exploration is decentralized and each agent explores with fixed steps. As for the simulation step, the trajectory, the action sequence, and the reward signal also need to be replayed from the storage. The newly visited states will be updated to the achievement of agent i. The existing state will be updated with fewer trajectory steps in the achievement. In our experiments, we show that the performance and training efficiency of reinforcement learning can be improved by our proposed solution. The pseudocode is shown is Algorithm 1.

Algorithm 1: I-Go-Explore

```
Initialise environment, parameters for actor θ^π, parameters for critic θ^c, Go-Exploration
  settings, buffer
for each training episode n do
    for m = 1, ..., BatchSize do
        Sample (S, u, S', R^e) from π_θ
        Add episode to buffer
    end for
    Collate episodes in buffer into single batch
    for t=1,...,T do
        Batch unroll RNN using S, u, R
        Compute intrinsic curiosity module R^i
        Compute TD(λ) y_t^i by θ̂^c
    end for
    for t = T, ..., 1 do
        ΔQ_t = y_t − Q(S, u)
        Compute critic gradient ∇θ^c
        Update θ^c for the fix steps
    end for
    for t = T, ..., 1 do
        A^i(S_t, u) = Q(S_t, u) − Σ_u Q(S_t, u, u^{−i}) π(u|τ^i)
        Compute actor policy gradients by A^i(S_t, u)
    end for
    Update actor weights
    Start k-steps exploration from the last state S_t
    Update exploration information to archive
    Update exploration result to buffer
end for
```



(a) tiny map. (b) small map.

Figure 3: Figure 3a is tiny map with 4 agents and Figure 3b is small map with 4 agents. The green block is the required shelf, the yellow hexagon presents the agent and the black blocks are the goal workshops.

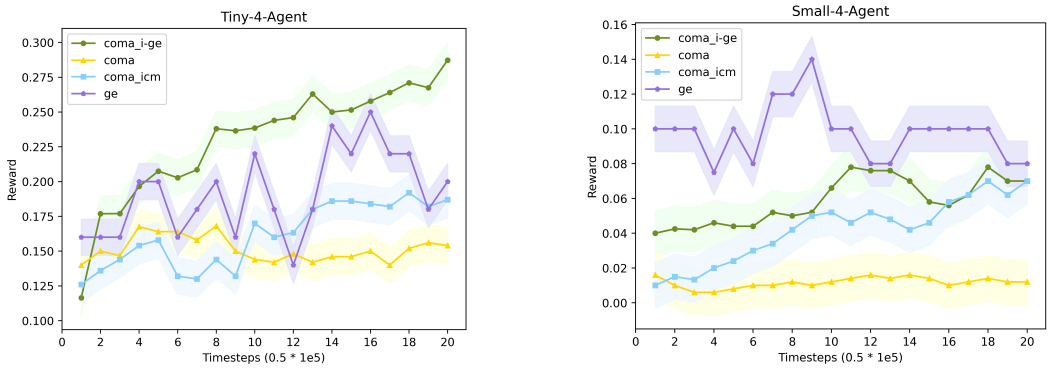
4 Experimental Analysis

This article is extended based on the success of the ICM method in multi-agent reinforcement learning sparse reward environment and the success of Go-Explore in a single-agent sparse reward environment. The initial multi-agent reinforcement learning (MARL) environment implements *Multi-Robot Warehouse (RWARE)* which is proposed in [PCSA20]. This environment was created for simulating a sparse-reward hard exploration environment for fully cooperative tasks [1]. RWARE simulates a grid-world warehouse where agents must fully cooperate to locate and deliver the requested shelves to the workstations and return them after delivery. The agents are only rewarded after successful delivery which requires multiple steps. In this paper, we focus on the tiny map and small map for a 4 agents' fully cooperative task. The illustrations of two maps are shown in Figure3.

State and Action Feature There are global state and local observations for the critic network and actor-network. Each agent takes a 3×3 square centered on the agent as its local observation. The action space contains four types of discrete actions: turn left, turn right, move forward, and load or unload the shelf. Furthermore, the agent can't cross the occupied cell and it is forced to move when it collides by the default environment setting. Agents are rewarded 1 when the requested shelf is delivered and the requested shelves are ensured during the running.

Training setting We extend the framework from [PCSA20] and the parameters setting for COMA as the baseline model. The actor-network is composed of Recurrent Neural Networks(RNN). And the action probabilities are generated by the softmax distribution. The critic network applies multiple-layer perceptron with the full connection. As for the intrinsic curiosity module part, the feature function takes 4 convolution layers to encode the state information and the inverse function concatenates two outputs from the encoder to a MLP with 256 units and outputs the action probability. I-Go-Explore takes 100 steps during the post-exploration phase due to its time-consuming.

[1]The code used for the experiments will be made available via a Github repository on acceptance of this submission.



(a) tiny map.

(b) small map.

Figure 4: The total rewards for four algorithms in two maps.

Ablations We compare the I-Go-Explore performance with COMA, COMA with ICM, and Go-Explore. The maximum number of steps for each episode is 500. And the parameters are shared between agents for COMA. As the baseline model, the parameters for COMA, COMA with ICM, and I-Go-Explore are the same. In I-Go-Explore, we set 500 steps for post-exploration. The test results are shown in Figure 4. The tiny map environment is easier than the other map and the agents get more rewards. It's worth noting that all algorithms take 500 steps as the step limitation for each episode which limited the performance for Go-Explore since it benefits from finding more cells. From the results, we found out that I-Go-Explore outperforms the rest of the baselines which has 0.22947 as the average return for $1e6$ episode test. And I-Go-Explore gets the same among rewards with fewer timesteps. With post-exploration, the proposed method outperforms the hard-exploration task more than the baselines. On the other hand, combining the post-exploration with the intrinsic curiosity method takes less execution time than Go-Explore method.

5 Concluding Remarks and Future Work

In this article, we showed that the proposed method I-Go-Explore shows promising results for the task of improving the performance of the existing reinforcement learning methods in a data-sparse environment. This article indicates a promising research direction to improve the model efficiency with sparsity.

Our experimental results showcase the better performance of I-Go-Explore compared to the baseline of the ICM method.

Interesting directions for future work include the following.

- For learning algorithm: ICM method has been implemented with some other centralized critic algorithms, like MADDPG [LWT+17], MAPPO [YVV+22]. We could implement I-Go-Explore with such different learning algorithms.
- To show the robustness of the I-Go-Explore method, we could test more partially observable multi-agent environments.

References

[AVRDS22] Alain Andres, Esther Villar-Rodriguez, and Javier Del Ser. An evaluation study of intrinsic motivation techniques applied to reinforcement learning over hard exploration environments. *arXiv preprint arXiv:2205.11184*, 2022.

[BESK18] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[DRSJ+22] Roben Delos Reyes, Kyunghwan Son, Jinhwan Jung, Wan Ju Kang, and Yung Yi. Curiosity-driven multi-agent exploration with mixed objectives. *arXiv e-prints*, pages arXiv–2210, 2022.

[EHL+21] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

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

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[LWKO22] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022.

[LWT+17] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[LZY20] Yexin Li, Yu Zheng, and Qiang Yang. Cooperative multi-agent reinforcement learning in express system. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 805–814, 2020.

[PAED17] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[PCSA20] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SB20] Hassam Ullah Sheikh and Ladislau Bölöni. Multi-agent reinforcement learning for problems with combined individual and team reward. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[TA07] Kagan Tumer and Adrian Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, 2007.

[VBC+19] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[WT01] David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03):265–279, 2001.

[WZW+20] Tong Wu, Pan Zhou, Binghui Wang, Ang Li, Xueming Tang, Zichuan Xu, Kai Chen, and Xiaofeng Ding. Joint traffic control and multi-channel reassignment for core backbone network in sdn-iot: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Network Science and Engineering*, 8(1):231–245, 2020.

[YCHL19] Hsuan-Kung Yang, Po-Han Chiang, Min-Fong Hong, and Chun-Yi Lee. Flow-based intrinsic curiosity module. *arXiv preprint arXiv:1905.10071*, 2019.

[YVV+22] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.