Curriculum Learning for Relative Overgeneralization

Lin Shi
University of Liverpool
Liverpool, United Kingdom
linshi1121@gmail.com

Bei Peng
University of Liverpool
Liverpool, United Kingdom
bei.peng@liverpool.ac.uk

ABSTRACT

In multi-agent reinforcement learning (MARL), many popular methods, such as VDN and QMIX, are susceptible to a critical multi-agent pathology known as relative overgeneralization (RO), which arises when the optimal joint action's utility falls below that of a sub-optimal joint action in cooperative tasks. RO can cause the agents to get stuck into local optima or fail to solve cooperative tasks that require significant coordination between agents within a given timestep. Recent value-based MARL algorithms such as QPLEX and WQMIX can overcome RO to some extent. However, our experimental results show that they can still fail to solve cooperative tasks that exhibit strong RO. In this work, we propose a novel approach called curriculum learning for relative overgeneralization (CURO) to better overcome RO. To solve a target task that exhibits strong RO, in CURO, we first fine-tune the reward function of the target task to generate source tasks that are tailored to the current ability of the learning agent and train the agent on these source tasks. Then, to effectively transfer the knowledge acquired in one task to the next, we use a transfer learning method that combines value function transfer with buffer transfer, which enables more efficient exploration in the target task. We demonstrate that, when applied to QMIX, CURO overcomes severe RO problem and significantly improves performance, yielding state-of-the-art results in a variety of cooperative multi-agent tasks, including the challenging StarCraft II micromanagement benchmarks.

KEYWORDS

Reinforcement Learning, Multi-agent Systems, Curriculum Learning, Deep Learning

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) holds great promise in solving real-world multi-agent problems, such as multi-robot search and rescue [20] and traffic light control [3]. In Many MARL settings, partial observability or practical communication constraints necessitate the learning of decentralized policies, which condition only on the local observations of each agent. The training itself, however, can be centralized in a simulated or under controlled setting, with access to additional information that might be available about the environment and other agents.

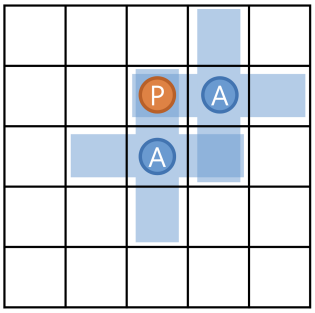
Under this paradigm of centralized training with decentralized execution (CTDE) [11, 18], VDN [26] and QMIX [23] are two popular value-based MARL algorithms that both aim to efficiently learn a centralized but factored joint action-value function. To ensure consistency between the centralized and decentralized policies, VDN and QMIX both represent the joint action-value function as a monotonic combination of per-agent utilities. However, this monotonicity constraint prevents VDN and QMIX from representing joint action-value functions that are characterized as *nonmonotonic* [14], i.e., an agent's ordering over its own actions depends on other agents' actions. Recent work [7] shows that both VDN and QMIX are prone to a multi-agent pathology called *relative overgeneralization* (RO) [30] due to their monotonicity constraints, which hinder the accurate learning of the joint action-value functions.

In cooperative tasks, RO occurs when the optimal joint action's utility falls below that of a sub-optimal joint action, which can cause the agents to get stuck into local optima or fail to solve cooperative tasks that require significant coordination between agents. An intuitive example illustrating the RO pathology is the partially-observable predator-prey task [2, 25] as shown in Figure 1. In this task, 2 agents (predators) need to coordinate with each other to capture a prey. If both agents surround the prey and execute the *catch* action simultaneously, a prey is caught and a positive reward of magnitude r is given. However, if only one surrounding agent performs the *catch* action, a negative reward of magnitude p is given. Even though this single agent executes the optimal *individual* action, the miscoordination penalty could discourage it from taking the same action in the future. When uncoordinated joint actions are much more likely than coordinated ones, this miscoordination penalty term p can dominate the average value estimated by each agent's utility. This can lead the agents to choose "safe" actions that are likely to tempt them away from the optimal joint action, resulting in failure to capture the prey. RO can therefore be detrimental to many cooperative multi-agent tasks that require extensive coordination among agents.

Rashid et al. [21] show that this problem can be addressed by introducing a weighting scheme to place more importance on the better joint actions to learn a richer class of joint action-value functions. They propose WQMIX, which infers this weighting scheme from an unrestricted joint action-value function that is learned simultaneously. While WQMIX has demonstrated some success in solving the predator-prey task exhibiting RO as mentioned above, our experimental results show that it can still fail to solve cooperative tasks that exhibit strong RO (e.g., the same predator-prey task but with a high miscoordination penalty p). We still lack reliable MARL methods for tackling RO.

The main idea of this paper is to combine curriculum learning [1] with MARL to better overcome RO. We propose a novel approach called CUrriculum learning for Relative Overgeneralization (CURO). Our key insights is that, to solve a target task that exhibits strong RO, we can start with simpler tasks that do not exhibit RO or exhibit mild RO, and then leverage these experiences to learn harder tasks with increased severity of RO. The agent can learn faster on more difficult tasks after it has mastered simpler but similar tasks. The core research questions here are how to create a sequence of source

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.



	catch	left	right	up	down	stay
catch	+r	-p	-p	-p	-p	-p
left	-p	0	0	0	0	0
right	-p	0	0	0	0	0
up	-p	0	0	0	0	0
down	-p	0	0	0	0	0
stay	-p	0	0	0	0	0

Figure 1: The partially-observable predator-prey task, where two agents are rewarded when they execute the *catch* action simultaneously and punished when one attempt it alone, adopted from [25].

tasks (i.e., a curriculum) with increasing difficulty that is tailored to the current ability of the learning agent, and, given this capability, how to effectively transfer the knowledge acquired in one task to the next in the curriculum.

For the former question, we fine-tune the reward function of the target task to control the probability of RO occurring, in order to find suitable source tasks that are as similar as possible to the target task, while not exhibiting a strong RO problem. To achieve this, we aim to create source tasks in which the expected return under the random policy is higher than the expected return under the policy learned in the target task (i.e., the policy that fails to solve the target task due to severe RO). For the latter question, we utilises a transfer learning method that combines *value function* transfer (i.e., the parameters of a value function learned in the previous task are used to initialize the value function in the next task in the curriculum) with *buffer* transfer (i.e., the state-action-reward experience tuples stored in the replay buffer in the previous task are used to initialize the replay buffer in the next task in the curriculum). The use of buffer transfer is critical in our method since it can enable more efficient exploration in the target task. After solving the source tasks that do not exhibit RO or exhibit mild RO, we can use the learned policy to generate positive sample experiences that contain optimal joint actions, which are then used to initialize the replay buffer for the target task. Since these optimal joint actions overlap in some states with the target task, initializing the reply buffer with these sample experiences can help bias the agents towards optimal joint actions when learning the target task.

To validate CURO, we apply it to QMIX and evaluate its performance in partially-observable predator-prey tasks and challenging StarCraft Multi-Agent Challenge (SMAC) [24] benchmark tasks that both exhibit strong RO and require significant coordination among agents. Our experiments show that CURO, when applied to QMIX, can successfully overcome severe RO and outperform state-of-the-art multi-agent value-based methods. CURO is not tied to QMIX and can be readily applied to any other deep value-based MARL methods.

2 BACKGROUND AND RELATED WORK

We consider a fully cooperative multi-agent task in which a team of agents interacts with the same environment to achieve some common long-term goal. It can be modeled as a *decentralized partially observable Markov decision process* (Dec-POMDP) [17] consisting of a tuple $G = <\mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{U}}, P, R, \Omega, O, \gamma>$. Here $\mathcal{N} \equiv \{1, \ldots, n\}$ denotes the finite set of agents and $s \in \mathcal{S}$ describes the true state of the environment. At each time step, each agent $a \in \mathcal{N}$ chooses an action $u^a \in \mathcal{U}$, forming a joint action $\boldsymbol{u} \in \boldsymbol{\mathcal{U}} \equiv \mathcal{U}^n$. This causes a transition in the environment according to the state transition kernel $P(s'|s, \boldsymbol{u}) : \mathcal{S} \times \boldsymbol{\mathcal{U}} \times \mathcal{S} \rightarrow [0, 1]$. All agents are collaborative and share the same reward function $R(s, \boldsymbol{u}) : \mathcal{S} \times \boldsymbol{\mathcal{U}} \rightarrow \mathbb{R}$. $\gamma \in [0, 1)$ is a discount factor specifying how much immediate rewards are preferred to future rewards.

Due to *partial observability*, each agent a cannot observe the true state s, but receives an individual partial observation $o^a \in \Omega$ drawn from the observation kernel $o^a \sim O(s, a)$. At time t, each agent a has access to its action-observation history $\tau_t^a \in \mathcal{T}_t \equiv (\Omega \times \mathcal{U})^t \times \Omega$, on which it conditions a stochastic policy $\pi^a(u_t^a|\tau_t^a)$. $\boldsymbol{\tau}_t \in \mathcal{T}_t^n$ denotes the histories of all agents. The joint stochastic policy $\boldsymbol{\pi}(\boldsymbol{u}_t|s_t, \boldsymbol{\tau}_t) \equiv \prod_{a=1}^n \pi^a(u_t^a|\tau_t^a)$ induces a joint action-value function: $Q^{\boldsymbol{\pi}}(s_t, \boldsymbol{\tau}_t, \boldsymbol{u}_t) = \mathbb{E}[G_t|s_t, \boldsymbol{\tau}_t, \boldsymbol{u}_t]$, where $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return.

2.1 CTDE

We adopt the *centralized training with decentralized execution* (CTDE) paradigm [11, 18]. During training, our learning algorithm has access to the true state s of the environment and every agent's action-observation history τ^a. However, during execution, each agent's decentralized policy π^a can only condition on its own local action-observation history τ^a. This approach can take advantage of extra information that is not available during execution and also enable agents to freely share their observations and internal states during training, which can greatly improve the efficiency of learning [4, 5].

2.2 Value Function Factorization

Value function factorization [10] has been widely employed in value-based MARL algorithms. VDN [26] and QMIX [23] are two representative examples that both learn a centralized but factored action-value function Q_{tot}, using CTDE. They are both Q-learning algorithms for cooperative MARL tasks with discrete actions. To ensure consistency between the centralized and decentralized policies, VDN and QMIX factor the joint action-value function Q_{tot} assuming additivity and monotonicity, respectively. More specifically, VDN factors Q_{tot} into a sum of the per-agent utilities: $Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}; \boldsymbol{\theta}) = \sum_{a=1}^n Q^a(\tau^a, u^a; \theta^a)$. QMIX, however, represents Q_{tot} as a continuous monotonic mixing function of each agent's utilities: $Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}, s; \boldsymbol{\theta}, \phi) = f_\phi\big(s, Q^1(\tau^1, u^1; \theta^1), \ldots, Q^n(\tau^n, u^n; \theta^n)\big)$, where $\frac{\partial f_\phi}{\partial Q^a} \geq 0, \forall a \in \mathcal{N}$. Here f_ϕ is approximated by a monotonic mixing network, parameterized by ϕ. The weights of the mixing network are constrained to be non-negative to guarantee monotonicity. These weights are produced by separate *hypernetworks*, which condition on the full state s. This can allow Q_{tot} to depend on the extra state information in nonmonotonic ways. The joint action-value function Q_{tot} can be trained using Deep Q-Networks (DQN) [15]. The

greedy joint action in both VDN and QMIX can be computed in a decentralized fashion by individually maximizing each agent's utility Q^a.

2.3 Relative Overgeneralization

In deep MARL, there are only a sparse number of existing works that explicitly address the relative overgeneralization (RO) problem. One example is the multi-agent soft Q-learning method proposed by Wei et al. [31], which extends soft Q-learning [8] to multi-agent settings using the CTDE paradigm. However, multi-agent soft Q-learning is only evaluated in simple single state games. It is not clear whether it can perform well in sequential continuous games or more complex cooperative tasks. In addition, it cannot scale well to multi-agent systems with a large number of agents and/or actions.

Peng et al. [19] recently employ value function factorization in the multi-agent actor-critic framework to learn decentralized policies with a centralized but factored critic. In contrast to value-based approaches such as VDN and QMIX, adopting value function factorization in an actor-critic framework allows for the direct use of a nonmonotonic value function factorization with full representational capacity. It also enables scalable learning of a centralized critic, leading to better performance in cooperative tasks with a larger number of agents and/or actions. Their work shows that, when using a nonmonotonic value function factorization, their multi-agent actor-critic approach can overcome RO to some extent and solve some cooperative tasks that cannot be solved using monotonic value function factorization. However, it can still fail to solve cooperative tasks that exhibit *strong* RO. We still lack reliable and scalable MARL methods for tackling RO.

2.4 Curriculum Learning

In the context of machine learning, curriculum learning [1] is a methodology to optimize the order in which training examples is presented to the learner, so as to increase performance or training speed on one or more final tasks. Rather than considering all training examples at once, the training data can be introduced in a meaningful order based on their apparent simplicity to the learner, such that the learner can build up a complex model step by step. Through generalization, knowledge acquired quickly from simple training examples can be exploited to speed up learning on more difficult examples.

Curriculum learning has been largely used in reinforcement learning (RL) to train the agent to solve tasks that may otherwise be too difficult to learn from scratch [16]. For RL domains, when creating a curriculum (i.e., a sequence of intermediate tasks), the intermediate tasks [1] may differ in state/action space, reward function, or transition function from the target task (i.e., the task we are interested in solving). Most existing works focus on investigating how to create and sequence a set of intermediate tasks with increasing difficulty that is tailored to the current ability of the learning agent, and how to develop better transfer learning methods that enable the agent to effectively transfer reusable knowledge acquired in one task to the next in the curriculum. See Taylor and Stone [27] for an in-depth overview of transfer learning in reinforcement learning, and Narvekar et al. [16] for an in-depth overview of curriculum learning in reinforcement learning.

While curriculum learning has been extensively used in single-agent RL settings to improve learning and generalization, it has received comparatively less attention from the MARL community. Recent works mostly focus on leveraging curriculum learning to scale MARL methods to more complex multi-agent problems with a large population of agents [6, 12]. In this work, we focus on utilising curriculum learning to enable MARL algorithms to overcome RO in a more reliable way than existing methods.

3 METHODOLOGY

In this section, we present our approach called CUrriculum learning for Relative Overgeneralization (CURO) that aims to combine curriculum learning with MARL to better overcome relative overgeneralization (RO) in cooperative multi-agent tasks. We start by describing how we generate source tasks that are tailored to the current ability of the learning agent in CURO. We then discuss the transfer learning method used in CURO, which combines value function transfer with buffer transfer to enable more efficient knowledge transfer between intermediate tasks in the curriculum.

3.1 Source Task Generation in CURO

Our key insight behind CURO is that, to solve a target task that exhibits strong RO, we can start with simpler tasks that do not exhibit RO or exhibit mild RO, and then leverage these experiences to learn harder tasks with increased severity of RO. One main problem we need to address here is how to create a sequence of source tasks (i.e., a curriculum) with increasing difficulty that is tailored to the current ability of the learning agent (i.e., neither too hard nor too easy to the learning agent).

Consider the partially-observable predator-prey task as mentioned above, the probability that RO occurs in this task increases if we increase the magnitude of the penalty p associated with each miscoordination, since the agent is more discouraged from taking the optimal individual action (i.e., the *catch* action) in the future after being punished more. If we remove the penalty associated with each miscoordination, i.e., $p = 0$, the corresponding task is free from RO. Existing MARL methods such as VDN and QMIX can then reliably solve this task since the monotonic value function can efficiently represent the optimal joint action-values. Inspired by this, to create a curriculum, we change only the reward function of the target task to control the probability of RO occurring, without modifying any other components (e.g., the state space and action space) of the target task Dec-POMDP.

Property 1 *If agent histories are truncated to an arbitrary finite length, there is a stationary distribution of Markov chain that is guaranteed to exist under any agent policy π_θ.*

Algorithm 1 describes how we generate the source tasks for the agent to learn first. Our main idea is to fine-tune the reward function of the target task Dec-POMDP to find suitable source tasks that are not too easy nor too hard for the agent to solve. In other words, the source task should be as similar as possible to the target task without exhibiting a strong RO problem. More specifically, given a

[1] We use "intermediate tasks" and "source tasks" interchangeably to refer to the tasks in the curriculum.

Algorithm 1 Source Task Generation in CURO

Input: Target task M, a sequence of candidate reward functions $< R_1, R_2, \ldots, R_n >$, and a random policy π_{random}

Train a MARL method on the target task M until it converges and save the policy learned, π_M.
$M' \leftarrow M$
for R' in $< R_1, R_2, \ldots, R_n >$ **do**
  $M'.R \leftarrow M'.R'$
  if AveragedTestReturn(M', π_{random}) > AveragedTestReturn(M', π_{M}) **then**
    return M'
  end if
end for

Algorithm 2 Transfer Learning in CURO

Input: Target task M, source task M', and the replay buffer for the source task $\mathcal{D}^{M'} \leftarrow \emptyset$

Train a value-based MARL method on the source task M' until it converges and save the action-value function learned, $Q(s, a|\theta^{M'})$
for episode= 1 to n_{buffer} **do**
  Receive initial state $s_1 \sim p(s_1)$
  for t = 1 to max-episode-length **do**
    Select action $a_t \leftarrow \arg\max_a Q(s_t, a_t|\theta^{M'})$
    Execute action a_t and observe r_t and s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}^{M'}$
  end for
end for
Initialize the Q network for M with weight $\theta^M \leftarrow \theta^{M'}$
Initialize the replay buffer for M as $\mathcal{D}^M \leftarrow \mathcal{D}^{M'}$
Train the same value-based MARL method on M

target task M we are interested in solving, we first use an existing MARL algorithm to train a policy π_M on M, [2] which should fail to learn a good coordination strategy due to the strong RO problem in M. To determine the reward function R of the source task M', we generate a sequence of candidate reward functions, by changing the magnitude of the miscoordination penalty term in the reward function of the target task M, while keeping other components of the reward function the same. $< R_1, R_2, \ldots, R_n >$ is the resulting sequence of candidate reward functions, which are ordered from the smallest miscoordination penalty term to the largest one. Due to Property 1, for any task with a fixed-length episode, there is a stationary distribution on the set of all trajectories under any given policy and a relatively stable averaged test return can be obtained under the given policy [13].[3] We therefore evaluate if the random policy π_{random} is better than π_M in the candidate source task M' by estimating their averaged test returns. If the averaged test return of π_{random} is higher than the averaged test return of π_M in task M', then M' is returned as the source task we aim to find. [4]

3.2 Transfer Learning in CURO

Given that we know how to generate source tasks to build a curriculum for the agent, the next question is how to effectively transfer knowledge acquired in one task to the next while learning through a curriculum. One naive method is to use *value function* transfer [28], where the parameters of a value function learned in the previous task are used to initialize the value function in the next task in the curriculum, such that the agent learns the next task with some initial policy that is better than random exploration. However, when learning the target task exhibiting strong RO, the good initial policy obtained from the previously learned task can be quickly overwritten due to the large number of negative sample experiences (i.e., the experiences where the agents get punished for miscoordination) added to the reply buffer during early exploration stage of training.

To address this problem, in CURO, we use a transfer learning method as shown in Algorithm 2, which combines *value function* transfer with *buffer* transfer to enable more efficient exploration in the target task. In buffer transfer, the state-action-reward experience tuples generated in the previous task are used to initialize the replay buffer in the next task in the curriculum. Specifically, after training the agent on the source task, the converged policy is saved and used to generate n_{buffer} episodes of experiences, which are then used to initialize the replay buffer $\mathcal{D}^M$ for the target task M. When training the agent on the target task, the old episodes consisting of a sequence of transitions generated in the source task in the replay buffer are gradually replaced by new episodes consisting of a sequence of transitions generated in the target task. This can be seen as an approximation environment changing gradually from the source task to the target task. When optimizing the policy, we sample a mini-batch of n_{batch} episodes uniformly from the replay buffer and train on fully unrolled episodes. The use of buffer transfer is critical since it can enable more efficient exploration in the target task. When learning the target task, the initial replay buffer is filled with positive sample experiences that contain optimal joint actions. Since these optimal joint actions overlap in some states with the target task, initializing the reply buffer with these sample experiences can help bias the agents towards optimal joint actions when learning the target task.

4 EXPERIMENTAL RESULTS

In this section, we present our experimental results on the cooperative predator-prey tasks similar to one proposed by Son et al. [25] and on the challenging StarCraft Multi-Agent Challenge (SMAC) [24] benchmark. To validate our approach CURO, we apply it to QMIX and refer to our method as CURO-QMIX. We compare CURO-QMIX against different state-of-the-art multi-agent value-based methods, including QMIX [22], WQMIX [21], and QPLEX [29]. For evaluation, all experiments in this section are carried out with 5 different random seeds. In each figure, the median performance across 5 random seeds is plotted and the 25 − 75% percentiles is shown shaded, similar to [24]. More details about the reward functions of the source tasks selected by CURO in each domain are included in Appendix A.

[2] We use QMIX to train the policy π_M on the target task in our experiments.
[3] Formal justification for Proposition 1 can be found in [13].
[4] In practice, it is easy to find a random policy π_{random} that achieves higher averaged test return than policy π_M, since the assumption here is the target task M exhibits severe RO and thus the converged policy π_M performs very poorly.

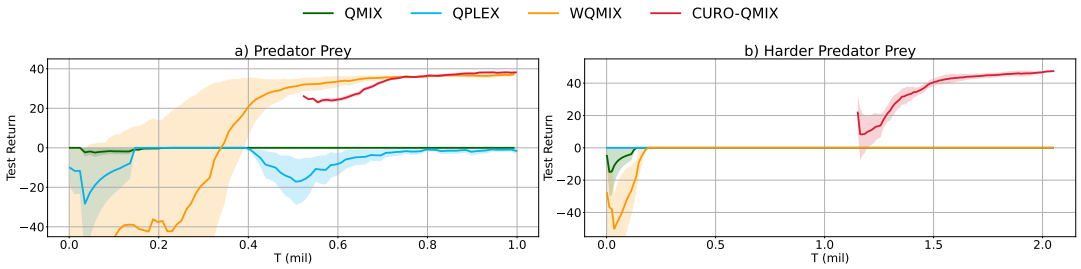


Figure 2: Median test return for QMIX, QPLEX, WQMIX, and CURO-QMIX on predator-prey with different levels of difficulty. For CURO-QMIX, the learning curve is offset to reflect time spent in source tasks. (a) Predator-prey task with 8 predators and 8 prey in a 10×10 grid, where each prey needs to be captured by at least *two* surrounding predators. (b) Predator-prey task with 16 predators and 16 prey in a 16×16 grid, where each prey needs to be captured by at least *three* surrounding predators.

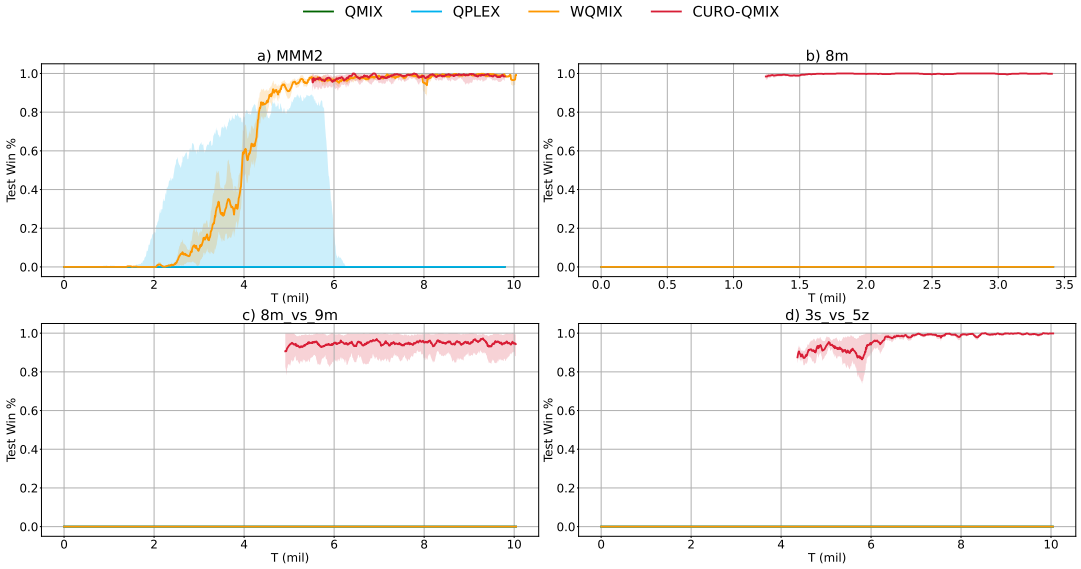


Figure 3: Median test win rates for QMIX, QPLEX, WQMIX and CURO-QMIX on four different SMAC maps with negative reward scaling $p = 1.0$. For CURO-QMIX, the learning curve is offset to reflect time spent in source tasks.

4.1 Domain 1: Cooperative Predator-Prey

We first evaluate CURO-QMIX on cooperative predator-prey tasks similar to one proposed by Son et al. [25], but significantly more complex in terms of the coordination required among agents. We consider a partially-observable predator-prey task involving 8 agents (predators) and 8 prey in a 10×10 grid. Each agent can either move in one of the four coordinate directions, stay still, or try to catch any adjacent prey. The prey moves by randomly selecting an available action or stays still if all surrounding positions are occupied. If two agents surround the prey and execute the catch action simultaneously, a prey is caught and a positive reward of +10 is given. However, if only one surrounding agent performs the catch action, a negative reward of -2 is given. Otherwise, no reward is given. An agent's observation is a 5×5 sub-grid centered around it. The punishment for miscoordination makes this task exhibit RO.

Figure 2(a) illustrates the median test return attained by different methods on this cooperative task during testing. We can see that QMIX fails to learn a policy that achieves any positive reward due to its monotonicity constraint, which hinders the accurate learning of the values of different joint actions. Interestingly, QPLEX also fails to solve the task despite not having any restrictions on the joint action-value functions it can represent. By contrast, both CURO-QMIX and WQMIX can successfully solve this cooperative task that exhibits RO.

We then increase the difficulty of the predator-prey task by making the capture condition stricter: each prey needs to be captured by at least three surrounding agents (rather than two surrounding agents) with a simultaneous capture action. The miscoordination penalty term remains to be -2. This makes the task suffer from more severe RO problem since, compared to the previous task, there is a higher probability of the agents receiving the miscoordination penalty through random exploration, leading to a higher probability of the agents choosing "safe" individual actions that are not part of the optimal joint action. In addition, we increase the size of the grid

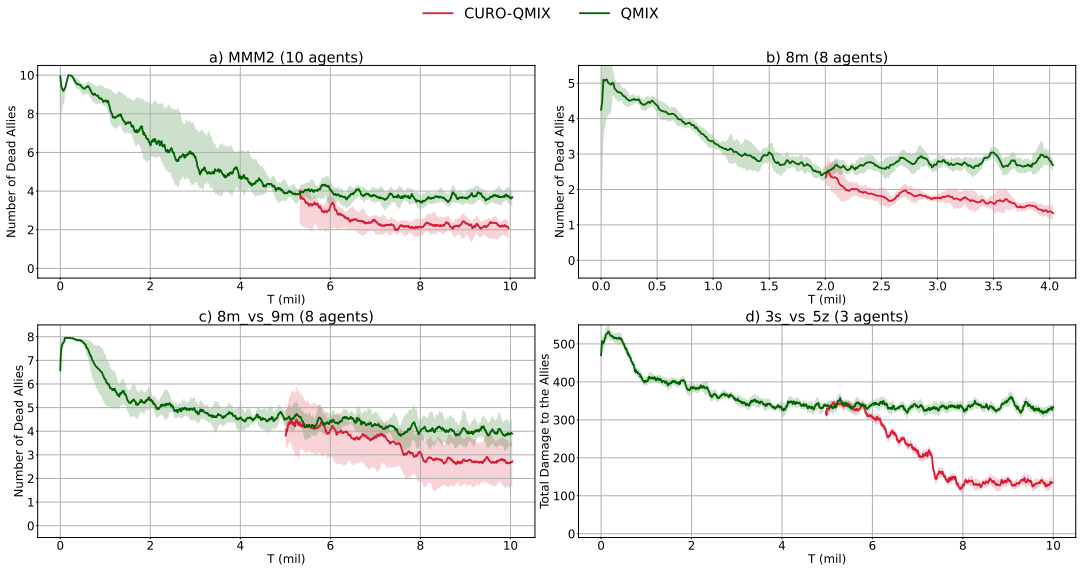


Figure 4: Number of dead allies ((a)-(c)) or total damage to the allies (d) for CURO-QMIX (on SMAC maps with $p = 1.0$) and QMIX (on SMAC maps with $p = 0$ during testing). For CURO-QMIX, the learning curve is offset to reflect time spent in source tasks.

world and the number of agents, with 16 predators and 16 prey in a 16×16 grid, to test the robustness of our method to RO.

As shown in Figure 2(b), CURO-QMIX can still successfully solve this task that exhibits stronger RO, whereas all other three multi-agent value-based methods fail to learn anything useful in this task. This suggests that WQMIX can struggle with cooperative tasks that exhibit strong RO, while our method can overcome RO more reliably. We emphasize that while both QPLEX and WQMIX can represent an unrestricted joint action-value function, which may allow them to overcome RO, they are not guaranteed to do so in any reasonable amount of time. This is mainly due to the simple noise-based exploration (i.e., ε-greedy action-selection strategy) used in these algorithms, which is shown to be sub-optimal in a MARL setting and can result in slow exploration and sub-optimal solutions in complex environments [14]. By contrast, our approach can enable more efficient exploration in the target task that exhibits strong RO, by leveraging the positive sample experiences collected from previously solved simpler tasks to bias the agents towards optimal joint actions.

4.2 Domain 2: StarCraft Multi-Agent Challenge

We also evaluate our method on the challenging StarCraft Multi-Agent Challenge (SMAC) [24] benchmark. SMAC consists of a set of complex StarCraft II micromanagement tasks that are carefully designed to study decentralized multi-agent control. The tasks in SMAC involve combat between two armies of units. The first army is controlled by a group of learned allied agents. The second army consists of enemy units controlled by the built-in heuristic AI. The goal of the allied agents is to defeat the enemy units in battle, to maximize the win rate. As shown by Gupta et al. [7], the default reward function used in SMAC does not suffer from RO as it has been designed with QMIX in mind. Specifically, each ally agent unit receives positive reward for killing/damage on enemy unites, but does not receive punishment for being killed or suffering damage from the enemy (i.e., negative reward scaling $p = 0$). Previous work [32] has established that a number of value-based MARL methods such as VDN and QMIX can solve almost all SMAC maps with this default reward setting.

To better evaluate our method, which aims to overcome RO that arises in many cooperative games, we change the reward function as in Gupta et al. [7], such that ally agents are additionally penalized for losing their ally units, inducing RO in the same way as in the above predator-prey tasks. We set $p = 1.0$ to equally weight the lives of ally and enemy units. Figure 3 shows the test win rate attained by different MARL methods on four different SMAC maps. We can see that both QMIX and QPLEX fail to learn anything useful in all four maps tested. WQMIX successfully learns an effective strategy that achieves a 100% win rate on *MMM2*. However, it fails to learn anything useful in other three maps tested. By contrast, CURO-QMIX can successfully learn a good policy that consistently defeats the enemy, resulting in (nearly) 100% test win rate in all four scenarios. This demonstrates that our method can overcome severe RO reliably and achieve efficient learning in challenging coordination problems.

We also examine the casualties condition of ally agents resulted from the policies learned by CURO-QMIX (on SMAC maps with $p = 1.0$) and QMIX (on SMAC maps with $p = 0$), respectively. Figures 4(a)-(c) illustrate the number of dead allies in each episode during testing on maps *MMM2*, *8m*, and *8m_vs_9m*. For CURO-QMIX, we evaluate the number of dead allies on SMAC maps with $p = 1.0$. For QMIX, we use the original SMAC setting where $p = 0$ since QMIX fails to learn anything in SMAC maps with $p = 1.0$. We can see that, compared to QMIX, CURO-QMIX can learn a more conservative policy that keeps more ally agents alive, while achieving nearly 100% test win rate. For map *3s_vs_5z*, we evaluate the total

damage value to the allies in each episode as in Figure 4(d), due to the low number of dead agents in this map. We see that, compared to QMIX, CURO-QMIX learns a more conservative policy with less total damage taken by the ally agents, while achieving nearly 100% test win rate. These results demonstrate that our method CURO, when applied to QMIX, can solve challenging cooperative multi-agent tasks that exhibit strong RO and achieve more efficient and robust learning.

5 CONCLUSION AND FUTURE WORK

In this work, we proposed CURO that leverages curriculum learning to better overcome the RO problem in cooperative tasks. In CURO, we first fine-tune the reward function of the target task to generate suitable source tasks that are tailored to the current ability of the learning agent and train the agent on these source tasks. We then use a transfer learning method which combines value function transfer with buffer transfer, to effectively transfer the knowledge acquired in one task to the next. To validate CURO, we applied it to QMIX and evaluated its peformance in cooperative predator-prey tasks and challenging SMAC benchmark tasks that both exhibit strong RO. Our experimental results demonstrated that CURO, when applied to QMIX, can successfully overcome severe RO and outperform state-of-the-art MARL algorithms. CURO is general and can be applied to other deep value-based MARL methods. One main limitation of our method is the uncertainty surrounding how to effectively fine-tune the reward function to generate source tasks for target tasks that suffer from RO, but lack an explicit miscoordination penalty in the reward function. Interesting directions for future work include an automatic generation of the source tasks (e.g., without manually defining the candidate reward functions) and the application of CURO to other deep MARL methods.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*. 41–48.

[2] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. 2020. Deep coordination graphs. In *International Conference on Machine Learning*. PMLR, 980–991.

[3] Jeancarlo Arguello Calvo and Ivana Dusparic. 2018. Heterogeneous Multi-Agent Deep Reinforcement Learning for Traffic Lights Control.. In *AICS*. 2–13.

[4] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems* 29 (2016).

[5] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[6] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 66–83.

[7] Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Böhmer, and Shimon Whiteson. 2021. UneVEn: Universal value exploration for multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3930–3941.

[8] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*.

[9] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. 2021. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv e-prints* (2021), arXiv–2102.

[10] Daphne Koller and Ronald Parr. 1999. Computing factored value functions for policies in structured MDPs. In *International Joint Conference on Artificial Intelligence*. 1332–1339.

[11] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[12] Qian Long, Zihan Zhou, Abhibav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. 2020. Evolutionary population curriculum for scaling multi-agent reinforcement learning. In *International Conference on Learning Representations*.

[13] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. 2021. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and Multi-agent Systems*.

[14] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, Vol. 32.

[15] 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), 529–533.

[16] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2021. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research* (2021).

[17] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[18] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[19] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. FACMAC: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*.

[20] Jorge Pena Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. 2020. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *Ieee Access* 8 (2020), 191617–191643.

[21] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 33. 10199–10210.

[22] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

[23] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. 4295–4304.

[24] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

[25] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5887–5896.

[26] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[27] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, 7 (2009).

[28] Matthew E Taylor, Peter Stone, and Yaxin Liu. 2007. Transfer Learning via Inter-Task Mappings for Temporal Difference Learning. *Journal of Machine Learning Research* 8, 9 (2007).

[29] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex dueling multi-agent Q-learning. In *International Conference on Learning Representations*.

[30] Ermo Wei and Sean Luke. 2016. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research* 17, 1 (2016), 2914–2955.

[31] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent soft Q-learning. In *AAAI Symposium*.

[32] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The surprising effectiveness of PPO in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).

A EXPERIMENTAL SETUP

All algorithms are implemented in the PyMARL2 framework [9]. For SMAC, we use the latest version SC2.4.10. Note that performance is not always comparable across versions. Our experiments can collect 10 million samples within 6 hours with a Core i7-11700K CPU and a RTX 3090 GPU.

All our experiments use ε-greedy action exploration strategy. In CURO-QMIX, ε is annealed linearly from 1.0 to 0.05 over $400k$ time steps. For SMAC, the sequence of candidate reward function selected by CURO is $< 0.5, 0 >$. The replay buffer contains the most recent 5000 episodes. For predator-prey, the sequence of candidate reward function selected by CURO is $< -1, 0 >$. The replay buffer contains the most recent 1000 episodes. For all experiments, 8 rollouts for parallel sampling are used to obtain samples from the environments. We sample batches of 128 episodes uniformly from the replay buffer and train on fully unrolled episodes. All the agent networks are the same as those in QMIX [22]. For QMIX, QPLEX, and WQMIX, the mixing network settings are the same as those in PyMARL2 [9] respectively. All target networks are periodically updated every 200 training steps. All neural networks are trained using Adam optimizer with learning rate 0.001. We set the discount factor $\gamma = 0.99$ for all experiments.