Distributional Reward Estimation for Effective Multi-Agent Deep Reinforcement Learning

Jifeng Hu[1] **Yanchao Sun**[2] **Hechang Chen**[3*] **Sili Huang**[4]

Haiyin Piao[5] **Yi Chang**[6] **Lichao Sun**[7*]

[1,3,4,6]School of Artificial Intelligence, Jlilin University, Changchun, China
[2]Department of Computer Science, University of Maryland, College Park, MD 20742, USA
[5]Northwestern Polytechnical University, Xian, China
[7]Lehigh University, Bethlehem, Pennsylvania, USA
[1,4]hujf21@mails.jlu.edu.cn, huangsl21@mails.jlu.edu.cn
[2]ycs@umd.edu
[3,6]chenhc@jlu.edu.cn, yichang@jlu.edu.cn
[5]haiyinpiao@mail.nwpu.edu.cn
[7]lis221@lehigh.edu

Abstract

Multi-agent reinforcement learning has drawn increasing attention in practice, e.g., robotics and automatic driving, as it can explore optimal policies using samples generated by interacting with the environment. However, high reward uncertainty still remains a problem when we want to train a satisfactory model, because obtaining high-quality reward feedback is usually expensive and even infeasible. To handle this issue, previous methods mainly focus on passive reward correction. At the same time, recent active reward estimation methods have proven to be a recipe for reducing the effect of reward uncertainty. In this paper, we propose a novel Distributional Reward Estimation framework for effective Multi-Agent Reinforcement Learning (DRE-MARL). Our main idea is to design the multi-action-branch reward estimation and policy-weighted reward aggregation for stabilized training. Specifically, we design the multi-action-branch reward estimation to model reward distributions on all action branches. Then we utilize reward aggregation to obtain stable updating signals during training. Our intuition is that consideration of all possible consequences of actions could be useful for learning policies. The superiority of the DRE-MARL is demonstrated using benchmark multi-agent scenarios, compared with the SOTA baselines in terms of both effectiveness and robustness.

1 Introduction

Multi-agent reinforcement learning (MARL) has achieved substantial successes in solving real-time competitive games [30], robotic manipulation [1], autonomous traffic control [32], and quantitative trading strategies [15]. Most existing works require agents to receive high-quality supervision signals, i.e., rewards, which are either infeasible or expensive to obtain in practice [61]. The rewards provided by the environment are subject to multiple kinds of randomness. For example, the reward collected from sensors on a robot will be affected by physical conditions such as temperature and lighting, which makes the reward full of bias and intrinsic randomness. The interplay among agents will result in more reward uncertainty. For instance, though an agent executes the same action under the same state, the reward can still vary because other agents can execute other actions. Thus, handling non-

*Corresponding Author. Email address: chenhc@jlu.edu.cn (H. Chen) and lis221@lehigh.edu (L. Sun).

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

stationary rewards in the learning process is a necessity for successfully learning complex behaviors in multi-agent environments.

In fact, there exist numerous works [23, 28, 61, 60] in reinforcement learning (RL) that consider the *reward uncertainty* during training. The traditional stream of research on non-stationary feedback focuses on passive reward correction. For example, Wang et al. [60] adopt the confusion matrix to model the reward uncertainty and obtain a stable reward for learning. While in [61], the authors recover the true supervision signals with peer loss, which punishes over-agreement for avoiding overfitting. However, the assumption of the reward uncertainty of the above works is restrictive. For instance, the reward flipping mechanism [60] restricts the randomness of the reward to a countable value set. Recently, another stream of research focuses on active reward estimation. To alleviate the reward uncertainty issue, some works have treated reward estimation as a point-to-point regression problem where each state-action pair is mapped to a reward [38, 64]. However, the reward uncertainty, particularly caused by agents' interplay, can not be fully resolved by the point-to-point reward estimation in MARL. This is because the regression is good at the one-to-one mapping between state and rewards. But in MARL, the same state-action pair input of one agent will lead to multiple environmental rewards (i.e., one-to-many mapping), which is intractable for regression and thus hurts the performance. Moreover, these methods do not consider the fact that the reward uncertainty comes not only from inherent environment randomness but also from the interplay among agents. These two factors cause blended influence on the received reward and increase the training difficulty. As we experimentally show later (See Figure 3 (left) for details.), such a point-to-point strategy leads to a worse suboptimal outcome with the increasing agent number and the reward uncertainty degree [31].

In this work, we aim to develop distributional reward estimation followed by policy-weighted reward aggregation for MARL. Intuitively, our idea is like one human constructs a reward blueprint of all action branches in his brain and thoughtfully makes a decision by considering all possible consequences. Traditional methods [27, 41] evaluate the policy only with environment rewards, which will bring about more uncertainty when training the critic because they only take into account the reward r_k received after taking a specific action, i.e., k-th action a_k, from the policy. While in our method, we consider not only the environmental rewards but also the potential rewards on other action branches to perform more stable critic updating and thus achieve better performance. So we propose multi-action-branch distributional reward estimation to model reward distributions $\{\tilde{R}^i(o^i, a^i_k)\}_{k=1}^K$ on all action branches, where o^i_t is the observation and $a^i_{k,t}$ is the k-th action. Then we aggregate the environment reward r_k and rewards sampled from different action-branch distributions $\{\tilde{R}^i(o^i, a^i_m)\}_{m=1,m\neq k}^K$ by weighting them according to the corresponding action selection probability of the current policy. We will obtain the aggregated mix reward $\bar{R}^i$ and lumped reward $\bar{r}^i$ for training each agent through the reward aggregation. Reward aggregation enables the agent to evaluate its historical decisions thoughtfully, thus providing a sophisticated and effective way to reduce the influence of reward uncertainty. Such a model covers the reward uncertainty in MARL environments and achieves better performance in several MARL benchmarks from small to large agent numbers.

Our contribution is three-fold. 1) We propose a novel framework, called Multi-agent Distributional Reward Estimation (DRE-MARL), to systematically characterize the reward uncertainty in MARL by modeling reward distributions for all action branches. To the best of our knowledge, this is the first effort to solve the reward uncertainty of MARL with multi-action-branch distributional reward estimation followed by reward aggregation. 2) Policy-weighted reward aggregation is developed in our framework which enables us to perform stable training of the critic and actors. Besides, DRE-MARL is a universal framework that can be expediently integrated into other MARL methods. 3) We validate the performance of our algorithm with function approximation and mini-batch update via extensive simulations in MARL scenarios with different agent numbers and reward uncertainty.

2 Related Work

Reward Uncertainty. Dealing appropriately with reward uncertainty has received quite a bit of attention in recent reinforcement learning studies [61, 60, 29, 16, 47, 24, 17, 28, 42, 59]. The core idea of this line of works is to assume access to the knowledge of the noise and define unbiased surrogate loss functions to recover the true loss or reward. A typical seminal work dates back to [28], which recovers the true loss from the noisy label distribution with the knowledge of the noise rates of labels. Follow-up works offer solutions to estimate the uncertainty level from model predictions [3, 65, 67,

52, 47] or clusterable representations [66]. A couple of recent works [60, 67, 29] have also looked into this problem in sequential settings. For example, Everitt et al. [6] analyze the potential sources of uncertainty and provide the impossibility result for training under facultative reward uncertainty. Wang et al. [60] consider modeling the reward uncertainty being caused by a confusion matrix and design a statistics-based estimation method to cover the uncertainty. Wang et al. [61] recover the true supervision signals with peer loss, which punishes over-agreement to avoid overfitting. But the method is inefficient in MARL, because the uncertainty is bigger than single agent RL.

Reward Estimation. A number of prior literature in robotic and non-robotic domains have adopted reward estimation [8, 38, 7, 44, 11, 55, 35, 56, 9, 51, 46, 57, 50, 25]. In most of these cases, the predicted reward is used for planning. They take advantage of the estimated reward in imagination augmented rollouts with a dynamic model (i.e., world model) accompanying the reward estimator. However, in our case, we aim at handling reward uncertainty rather than planning with multi-action-branch reward estimation and aggregation, thus avoiding learning system dynamics and multi-step imaginary rollouts. Reward estimation can also be performed by reward shaping [63, 12] (RS) and inverse reinforcement learning [10, 18] (IRL). But IRL doesn't account for the reward uncertainty, and RS focuses on efficient exploration. Recently, Romoff et al. [38] proposes to train reward estimator with function approximation alongside the value function. However, it adopts point-to-point reward estimation, which struggles when generalizing to multi-agent settings.

Distributional RL. Distributional RL has recently gained increasing attention due to its powerful capacity for treating RL's uncertainty [48, 49, 34, 37, 2, 39, 33, 5]. Early studies about return uncertainty can be dated back to Sobel [45]. In MARL, the uncertainty issue is a more challenging problem than single RL. Fortunately, many studies focus on it [48, 49, 34]. For example, Sun et al. [48] propose the Mean-Shape Decomposition method and quantile mixture in value decomposition, bridging the gap between distributional RL and value function factorization. Although our work also belongs to the distributional field, we focus on the reward uncertainty of the environment rather than Q-value uncertainty and model the distribution of reward rather than Q compared with [48, 49, 34]. Besides, we propose reward estimation followed by reward aggregation, which is novel and effective.

3 Preliminaries

A Markov game [20, 53], a multi-agent extension of the Markov decision process, with partial observability can be described as the tuple $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{R^i\}_{i\in\mathcal{N}}, \gamma \rangle$, where $\mathcal{N} = \{1, ..., N\}$ is the set of N agents, $\mathcal{S}$ is the state space of the environment, $\mathcal{A}^i$ is the action space of agent i, $\mathcal{O}^i$ is the observation space of agent i, $\mathcal{P} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Delta(\mathcal{S})$ ($\Delta(x)$ represents the space of distributions over x.) denotes the state transition probability that is a mapping from the current state and the joint action to the probability distribution over the state space, and $\gamma \in [0, 1)$ is the discounting factor. We consider the fact that the interplay between agents results in distributional reward feedback: instead of observing R^i for each agent i, agent i can only observe the reward sampled from the reward distribution $\widehat{R}^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Delta(\mathbb{R})$.

At each time t, N agents receive different observations $(o_t^1, ..., o_t^N)$ and output certain joint action $\boldsymbol{a_t} = \{a_{k,t}^i\}_{i\in\mathcal{N}}, k \in \{1, ..., K\}$, where a^i is executed according to agent i's policy $\pi^i : \mathcal{O}^i \to \Delta(\mathcal{A}^i)$, $K = |\mathcal{A}|$ is the size of action space. The environment then transitions to s_{t+1} and rewards each agent i by $r_t^i = R^i(s_t, \boldsymbol{a_t}, s_{t+1})$, where $R^i(\cdot)$ is the reward function. The goal of agent i is to find the best policy π^i that maximizes the total reward received from the environment from a start state to an end state. The expected cumulative discounted reward is expressed as $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \cdot r_t^i]$. In this paper, we consider the N agents, represented by $\boldsymbol{\pi} = \{\pi^1, ..., \pi^N\}$, with policies (actors) parameterized by $\boldsymbol{\theta} = \{\theta_1, ..., \theta_N\}$. Given a specific policy π, we adopt state value function the $V_{\gamma,\psi}^{\pi}$ to approximate expected return, where ψ is the parameters of $V_{\gamma,\psi}^{\pi}$.

4 Problem Formulation

4.1 Reward Uncertainty in MARL

Reward uncertainty is prevalent in many real-world scenarios, but its influence is not well solved in the literature. Handling non-stationary reward during training is necessary for successfully learning complex behaviors in multi-agent environments. In many practical scenarios, the agents usually

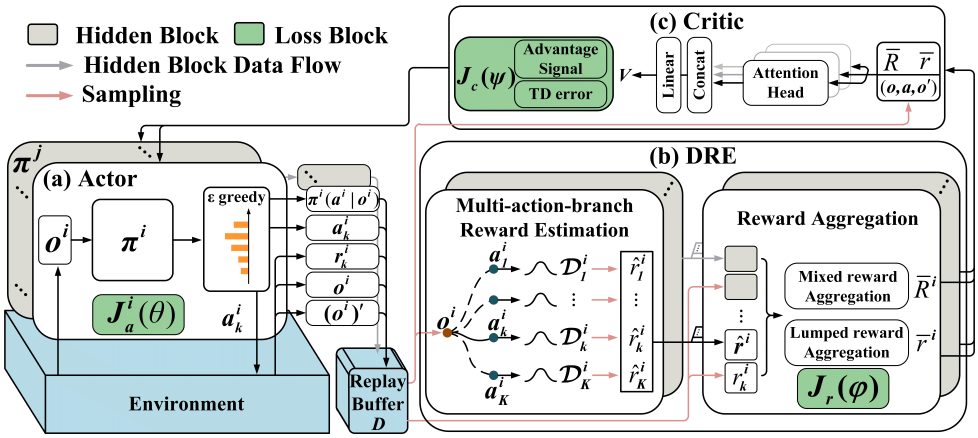

Figure 1: The overall architecture of Multi-agent Distributional Reward Estimation, which contains **(a) Decentralized Actors (b) Distributional Reward Estimation (c) Centralized Critic**. As is shown in **(a)**, we perform decentralized execution according to each agent's observation and store experience in the replay buffer. Then in **(b)**, we perform multi-action-branch reward estimation followed by policy-weighted reward aggregation. Finally, as shown in **(c)**, the centralized critic executes training with aggregated rewards and provides advantage signals for actors.

observe non-stationary reward feedback due to the interplay and inherent randomness rather than perfectly receiving the precise reward for training. Consider a multi-agent environment with high complexity and large state-action space such that it is intractable for agents to explore the entire state space. Suppose that we can only have access to the non-stationary reward. How can the agents learn cooperative behaviors under the reward uncertainty? We should figure out the factors that cause the reward uncertainty. Assume we are given a multi-agent scenario consisting of N agents. the reward contains uncertainty that is caused by multiple factors that can be classified into two aspects.

Mutual interaction. The first factor that causes reward uncertainty is mutual interaction between agents. As mentioned in Section 1, the interplay between agents can lead to the one-to-many mapping between state and reward. For example, suppose that we need two agents to complete a task cooperatively. The agent will receive various rewards because its partner executes other actions even though the agent executes the same action under the same observation.

Natural disturbance. Another aspect lies in the natural disturbance of the environment. During the training process, the environment contains inherent randomness that is ubiquitous in the real world. For instance, the sensor's feedback fluctuates due to the variation in temperature and lighting, which makes the reward feedback inaccurate.

In the following section, we focus on the two aspects and propose multi-action-branch reward estimation and reward aggregation for providing stable updating signals for agents. Here, the intuition is that the decision should be executed after carefully considering all possible consequences. Finally, in Section 6, we empirically investigate the validity of our method in different scenarios.

4.2 Reward Estimation

Usually, many MARL scenarios provide precise human-designed rewards, which deviate from the actual situation in practice. To simulate the reward uncertainty, we define the stochastic rewards that are provided by the environment according to some stochastic processes. That is, the reward is generated from some distributions with certain probability densities [60]. To capture the reward uncertainty and facilitate training, reward estimation is proposed to reduce the impact of stochastic rewards [60, 38]. The reward estimator [40], which is a core component in a scenario with stochastic rewards, evaluates s, (s, a), or (s, a, s') pairs at time step t and estimates the possible rewards that are used for guiding the agent through the user goal. Romoff et al. [38] model the task of learning the reward estimator as a simple point-to-point regression problem by function approximation: $\mathcal{L}(\varphi) = \mathbb{E}[(r - \tilde{R}(\mathcal{T};\varphi))^2]$, where $\tilde{R}(\mathcal{T};\varphi)$ is the reward estimator with parameter φ based on different inputs $\mathcal{T} \in \{s, (s, a), (s, a, s')\}$.

5 DRE-MARL

In this section, we introduce a general training framework, Multi-agent Distributional Reward Estimation (DRE-MARL), for the reward uncertainty problem in MARL, as shown in Figure 1. We adopt the architecture of centralized training and decentralized execution (CTDE) [22], which consists of N decentralized actors $\{\pi_\theta^i\}_{i\in\mathcal{N}}$ and a centralized critic $V_{\gamma,\psi}^{\pi}$, parameterized by θ and ψ, respectively. In practice, we utilize $\{\pi_\theta^i\}_{i\in\mathcal{N}}$ shown in Figure 1(a) to interact with the environment and collect experiences. Figure 1(b) shows the proposed distributional reward estimation structure, which consists of two stages: we first perform multi-action-branch reward distribution estimation from observed experiences and sample rewards from the reward distributions of action branches. Then we aggregate the environment rewards and sampled rewards to guide the training of the critic and actors. Figure 1(c) indicates that we utilize a graph attention network with multi-head attention [21, 4] to capture the global information from observations $(o^1, ..., o^N)$. The centralized critic can simultaneously produce advantage signals for all agents through one forward calculation during training. For stability, we construct target centralized critic $V_{\gamma,\tilde{\psi}}$[27, 43] and N target policies $\tilde{\pi}_{\tilde{\theta}}$, parameterized by $\tilde{\psi}$ and $\tilde{\theta}$, respectively. The target network parameters $(\tilde{\psi}, \tilde{\theta})$ are only updated with current network parameters (ψ, θ) and are held fixed between individual updates [19, 54].

5.1 Multi-action-branch Reward Estimation

Modeling reward distributions is challenging because the reward uncertainty caused by agents' interplay grows exponentially with an increase of agent number. One direct method is to model reward distributions on joint state-action space. But this method suffers from huge uncertainty caused by mutual interaction as agent number grows. Besides, in the ablation study (See Figure 3 (left) for details.), we also verify that modeling joint reward distributions cannot resolve the issue.

To handle the above challenge, we propose another method to achieve our goal. We simplify the issue by viewing other agents as a part of the environment such that we only need to focus on the reward estimation of each agent instead of all agents. Additionally, inspired by the fact that humans will imagine the potential consequences based on historical experience, we equip each agent with a reward estimator $\widehat{R}^i$ to better capture the reward uncertainty and stabilize the training process. Specifically, for agent i at time step t, we propose the multi-action-branch reward estimator $\widehat{R}^i(o_t^i, a_{k,t}^i; \varphi^i) \in \mathcal{D}$ to model the reward distribution based on agent i's observation o_t^i and k-th action $a_{k,t}^i$, where $\mathcal{D}$ represents reward distribution space, $\widehat{R}_k^i$ represents the reward distribution of agent i in k-th action branch, and φ^i is the parameters of $\widehat{R}^i$. We use $\widehat{R}^i \in \mathcal{D}^K$ to represent the K estimated reward distributions for agent i. We can achieve the above goal by optimizing the overall objective function $J_r = \sum_i J_r^i(\varphi^i)$, and the loss function of every agent is as follows:

$$J_r(\varphi) = \mathbb{E}_{(o,a_k,r_k)\sim D}\left[-\log \mathbb{P}[r_k|\widehat{R}(o, a_k; \varphi)] + \mathcal{L}_{\widehat{R}}\right], \tag{1}$$

where we omit the superscript i of $J_r(\varphi)$, D is the replay buffer, and $-\log \mathbb{P}[\cdot|\cdot]$ is negative log likelihood. $\mathcal{L}_{\widehat{R}}$ is a regular term of reward distributions. In practice, we adopt Gaussian reward distribution $\mathcal{D}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, but $\widehat{R}$ can be easily extended to other distributions. Then $\mathcal{L}_{\widehat{R}}$ can be defined by $\alpha \cdot \|\boldsymbol{\sigma}\|_1 + \beta \cdot var(\boldsymbol{\mu})$, where var is the variance of $\boldsymbol{\mu}$. α and β are hyperparameters.

Multi-action-branch reward estimation enables us to forecast possible rewards of all action branches. It is just like we only think about each specific situation separately, but in practice, we usually make a decision considering all possible consequences. Therefore, to evaluate the experience thoughtfully and stabilize training, we propose reward aggregation introduced in the following section to reduce the impact of reward uncertainty during the training process.

5.2 Training with Reward Aggregation

For each agent at each time step, the agent can only obtain a single reward r_k while executing k-th action a_k. But we augment r_k as a built-up reward vector, where the k-th of the vector is r_k and the following parts are the estimated rewards $\hat{\boldsymbol{r}}$ sampled from $\widehat{R}^i$. Then we aggregate the built-up reward vector with the policy-weighted operation. We mainly obtain two types of rewards after reward

aggregation: the mixed reward $\bar{R} \in \mathbb{R}$ and the lumped reward $\bar{r} \in \mathbb{R}$, where $\bar{R}$ is used to update centralized critic $V_{\gamma,\psi}$ and $\bar{r}$ is used to update decentralized actors $\{\pi_\theta^i\}_{i\in\mathcal{N}}$.

Specifically, for each agent we first define the built-up reward vector $m = h(\hat{\boldsymbol{r}}, r_k) \in \mathbb{R}^K$, which is constructed by replacing the k-th value of vector $\hat{\boldsymbol{r}}$ with true environment reward r_k because the agent only takes a_k during interaction with the environment, where $h(\cdot)$ is defined by $h(\hat{\boldsymbol{r}}, r_k) = [\hat{r}_1, ..., r_k, ..., \hat{r}_K]$. Then the mixed reward $\bar{R} = g(m^1, ..., m^N) \cdot \tilde{\pi}^i(\cdot|o)$ is calculated by policy-weighted aggregation, where $g(\cdot)$ represents two operations as below. 1) *MeanOperation* (g_{MO}): we utilize the average value of $m^1, ..., m^N$ as the output of $g(\cdot)$. 2) *SimpleSelection* (g_{SS}): we directly select m^i for agent i to calculate the mixed reward $\bar{R}$. Mathematically, $\bar{R}$ is defined by

$$\bar{R}^i = g(m^1, ..., m^N) \cdot \tilde{\pi}^i(\cdot|o) = \begin{cases} mean(m^1, ..., m^N) \cdot \tilde{\pi}^i(\cdot|o) & \text{if } g = g_{MO} \\ m^i \cdot \tilde{\pi}^i(\cdot|o) & \text{if } g = g_{SS} \end{cases}. \tag{2}$$

Sampling from the reward distributions has been proven to be beneficial for exploration in some settings [18, 60, 23]. But at the same time, it will also introduce more uncertainty during training. Here, we mainly focus on reducing the influence of reward uncertainty. Therefore we use the mean value of reward distributions to perform reward aggregation, usually achieving a more stable training process. These two schemes, the sampling option and mean option, can be chosen flexibly. It is a tradeoff: we choose the former option while the environment is hard to explore. The latter is more suitable for an environment with more uncertainty but is easy to explore.

Next, the centralized critic updates a parametric state value function $V_{\gamma,\psi}$ by minimizing the Bellman residual

$$J_c(\psi) = \mathbb{E}_{(o,r_k,o')\sim D}\left[\bar{R} + \gamma V_{\gamma,\tilde{\psi}}(o') - V_{\gamma,\psi}(o)\right]^2, \tag{3}$$

where ψ and $\tilde{\psi}$ are the parameters of current state value network and target state value network, respectively. Finally, for each agent during each training iteration, the decentralized actors follow the same training scheme of policy optimization with advantage function [41]:

$$J_a(\theta) = \mathbb{E}\left[\min\left(u(\theta), \text{clip}\left(u(\theta), 1-\epsilon, 1+\epsilon\right)\right)\hat{A}_\gamma + \eta H(\pi_\theta)\right], \tag{4}$$

where $u(\theta) = \frac{\pi_\theta(a|o)}{\pi_{\tilde{\theta}}(a|o)}$ is the importance weight, ϵ is clip hyperparameter and usually we choose $\epsilon = 0.2$. $H(\pi_\theta^i)$ is the policy entropy of agent i, and η controls the importance of entropy penalty objective. For agent i, the advantage value is defined as follows: $\hat{A}_\gamma^i = \bar{r}^i + \gamma V_{\gamma,\tilde{\psi}}(o') - V_{\gamma,\psi}(o)$, where $\bar{r}^i = l(m^1, ..., m^N, r_k)$ is the lumped reward that is defined by

$$\bar{r}^i = l(m^1, ..., m^N, r_k^i) = \begin{cases} mean(m^1, ..., m^N) & \text{if } l = l_{MO} \\ mean(m^i) & \text{if } l = l_{SMO} \\ r_k^i & \text{if } l = l_{SS} \end{cases}, \tag{5}$$

where l represents three operations as follows. 1) *MeanOperation* (l_{MO}): we average on all agents' built-up rewards ${m^i}_{i\in\mathcal{N}}$ to obtain a single output. 2) *SimpleMeanOperation* (l_{SMO}): we utilize the mean value of agent i's built-up rewards m^i as output. 3) *SimpleSelection* (l_{SS}): we directly select the environment reward r_k^i as the lumped reward.

Following the above procedure, we can update the reward estimators and policies iteratively. Further details and the full algorithm for optimizing DRE-MARL can be found in Appendix A.

6 Experiments

6.1 Environment Settings

To demonstrate the effectiveness of the proposed method, we provide experimental results in several benchmark MARL environments based on the multi-agent particle environments [22] (MPE) and several variants of MPE. Specifically, we consider cooperative navigation (CN), reference (REF), and treasure collection (TREA) environments from small to large agent numbers. In the following sections, we use CN-q, REF-q, and TREA-q to represent different environment variants, where $q \in \{2, 3, 7, 10\}$ and the entire environment variants are {CN-3, CN-7, CN-10, REF-2, REF-7,

Table 1: Performance comparison of DRE-MARL and several SOTA MARL algorithms under the r_{dete}, r_{dist}, and $r_{ac-dist}$ settings. The values represent mean episodic rewards.

	Scenario	q	DRE-MARL (**ours**)	p2p-MARL [38]	MAPPO [62]	MAAC [13]	QMIX [36]	MADDPG [22]	IQL [58]
r_{dete}	CN-q	3	**-154**$\pm$**11.90**	-167$\pm$12.80	-425$\pm$99.47	-227$\pm$43.38	-1168$\pm$77.73	-169$\pm$6.257	-738$\pm$175.2
		7	**-3070**$\pm$**89.18**	-3101$\pm$121.5	-5674$\pm$801.5	-3293$\pm$279.4	-9320$\pm$1528.	-3428$\pm$45.11	-5092$\pm$509.5
		10	**-8138**$\pm$**285.6**	-8418$\pm$277.0	-13375$\pm$1372.	-8555$\pm$873.0	-21858$\pm$4320.	-8938$\pm$67.94	-12658$\pm$864.4
	REF-q	2	-54$\pm$8.589	-58$\pm$8.638	-85$\pm$39.74	-69$\pm$28.30	-438$\pm$205.0	**-44**$\pm$**2.394**	-95$\pm$17.50
		7	**-732**$\pm$**73.85**	-808$\pm$88.99	-1187$\pm$365.0	-1120$\pm$302.5	-1359$\pm$130.5	-1133$\pm$50.92	-2400$\pm$285.2
		10	**-1524**$\pm$**140.9**	-1542$\pm$152.4	-3317$\pm$882.3	-2380$\pm$604.9	-4037$\pm$235.3	-3102$\pm$135.1	-4921$\pm$449.2
	TREA-q	3	-458$\pm$26.73	-457$\pm$28.06	-671$\pm$95.23	-480$\pm$52.52	**-390**$\pm$**37.96**	-428$\pm$11.17	-924$\pm$86.37
		7	**-2641**$\pm$**134.6**	**-2638**$\pm$**134.8**	-3648$\pm$0.005	-2809$\pm$270.0	-3324$\pm$270.9	-2678$\pm$59.59	-4733$\pm$246.7
		10	**-5076**$\pm$**190.4**	-5260$\pm$280.8	-6864$\pm$825.7	**-4931**$\pm$**438.1**	-8715$\pm$265.4	-5699$\pm$150.8	-8465$\pm$399.8
r_{dist}	CN-q	3	**-161**$\pm$**13.74**	-186$\pm$13.73	-773$\pm$103.9	-270$\pm$55.01	-1263$\pm$66.95	-213$\pm$6.300	-552$\pm$58.18
		7	**-3200**$\pm$**93.43**	-3262$\pm$132.3	-7481$\pm$1072.	-3558$\pm$419.7	-10678$\pm$2076.	-3617$\pm$49.06	-6141$\pm$1161.
		10	**-8605**$\pm$**277.0**	-8792$\pm$302.9	-19696$\pm$2607.	-8986$\pm$902.7	-22699$\pm$4151.	-9365$\pm$108.5	-12693$\pm$2046.
	REF-q	2	-57$\pm$8.504	-63$\pm$8.953	-86$\pm$27.21	-72$\pm$30.56	-280$\pm$193.1	**-43**$\pm$**2.234**	-226$\pm$30.06
		7	**-752**$\pm$**79.95**	-800$\pm$88.33	-1947$\pm$639.3	-1070$\pm$283.7	-1894$\pm$195.3	-1182$\pm$38.16	-2514$\pm$283.6
		10	**-1676**$\pm$**177.6**	**-1570**$\pm$**156.0**	-5580$\pm$1745.	-2558$\pm$588.7	-3584$\pm$338.6	-2968$\pm$121.0	-4090$\pm$1140.
	TREA-q	3	-484$\pm$25.97	-447$\pm$26.12	-691$\pm$136.3	-513$\pm$76.72	**-427**$\pm$**39.78**	-449$\pm$8.016	-889$\pm$49.29
		7	**-2733**$\pm$**131.6**	**-2745**$\pm$**150.0**	-3741$\pm$0.000	-2956$\pm$313.2	-3553$\pm$212.7	-2824$\pm$75.14	-5092$\pm$394.1
		10	**-5340**$\pm$**223.6**	-5515$\pm$305.1	-6912$\pm$876.3	**-5306**$\pm$**587.2**	-8897$\pm$285.3	-5847$\pm$147.4	-8727$\pm$464.4
$r_{ac-dist}$	CN-q	3	**477**$\pm$**59.64**	271$\pm$87.00	156$\pm$80.78	-98$\pm$254.8	367$\pm$61.66	476$\pm$67.32	-2645$\pm$740.2
		7	**888**$\pm$**398.3**	-480$\pm$484.0	-2960$\pm$1418.	-85$\pm$487.5	-7122$\pm$1167.	802$\pm$345.6	-24492$\pm$4691.
		10	**133**$\pm$**610.0**	-2900$\pm$784.4	-7379$\pm$1510.	-3036$\pm$1024.	-16977$\pm$4269.	-122$\pm$746.8	-59231$\pm$8688.
	REF-q	2	**283**$\pm$**26.21**	213$\pm$32.72	247$\pm$46.78	106$\pm$18.96	255$\pm$23.77	**287**$\pm$**25.67**	-159$\pm$128.3
		7	**3323**$\pm$**194.7**	1886$\pm$269.5	2949$\pm$264.1	215$\pm$221.7	2824$\pm$214.2	2824$\pm$297.9	-26$\pm$644.8
		10	**6340**$\pm$**435.0**	2884$\pm$575.9	5301$\pm$612.5	-230$\pm$647.3	5229$\pm$453.1	4981$\pm$729.0	181$\pm$1331.
	TREA-q	3	2567$\pm$123.9	1708$\pm$228.1	**2742**$\pm$**122.5**	1776$\pm$124.7	2384$\pm$105.7	2712$\pm$107.7	-3505$\pm$719.7
		7	14249$\pm$547.1	9627$\pm$1154.	14473$\pm$353.2	8342$\pm$463.5	14364$\pm$346.3	**15436**$\pm$**381.2**	-17986$\pm$2391.
		10	29193$\pm$1183.	20437$\pm$2679.	31878$\pm$850.5	17306$\pm$875.7	27100$\pm$1061.	**32421**$\pm$**642.6**	-32136$\pm$5006.
normalized mean performance			**9.802**	9.104	6.486	8.091	4.762	9.080	1.990

REF-10, TREA-3, TREA-7, TREA-10}. In all these environments, the reward is set to collaborative, which means the reward is the summation of individual rewards. Detailed experimental description and settings can be found in Appendix B.1.

Reward formulation. To simulate the reward uncertainty in these classic environments, we investigate several reward settings with induced uncertainty as follows. 1) *Deterministic reward* (r_{dete}): in classic environments, the reward is calculated with respect to distance [13, 22]. Suppose that the feedback is exactly precise. The deterministic reward refers to that the same observation-action pair is mapped to an identical reward. We use r^i_{dete} to represent this reward setting. 2) *Natural disturbance reward* (r_{dist}): reward uncertainty caused by natural disturbance, mentioned in Section 4.1, can be regarded as the overall impact on the received reward because the sources of dynamics are coupled with each other. Similar to prior works [38, 60], we consider the natural disturbance reward that is designed by $r^i_{dist} \sim \mathcal{N}(r^i_{dete}, 1)*0.05+r^i_{dete}$. We use r^i_{dist} to represent this reward setting. 3) *Mutual interaction reward* ($r_{ac-dist}$): in order to simulate the reward uncertainty caused by mutual interaction mentioned in Section 4.1, we add different reward distributions on different action branches such that the same observation-action pair will result in different rewards. Specifically, the action dist-reward is defined by $r^i_{ac-dist} \sim \mathcal{N}(u, \delta) + r^i_{dete}$, where $u = \text{Index}(a^i_k)$ and $\delta = 0.001$. We use $r^i_{ac-dist}$ to represent this reward setting. r_{dist} and $r_{ac-dist}$ settings are particularly challenging because agents need to conquer more dynamics.

Baselines. In order to verify the efficacy of distributional reward estimation in MARL with reward uncertainty, we select the following methods as baselines: MADDPG [22] adopts a value-based centralized Q network to concatenate and process all agents' observations. MAAC [13] uses attention mechanism in CTDE paradigm. MAPPO [62] reaches good results with on-policy method. QMIX [36] adopts monotonicity to decompose team reward. IQL [58] adopts completely independent training on each agent. Apart from the above methods, we also compare with point-to-point (p2p-MARL) reward estimation [38] and global reward estimation (GRE-MARL).

Metrics. For all methods we consider, we report the final performance under different reward settings (r_{dete},r_{dist},$r_{ac-dist}$), which are introduced above. Following prior studies [13, 22], we set the episode

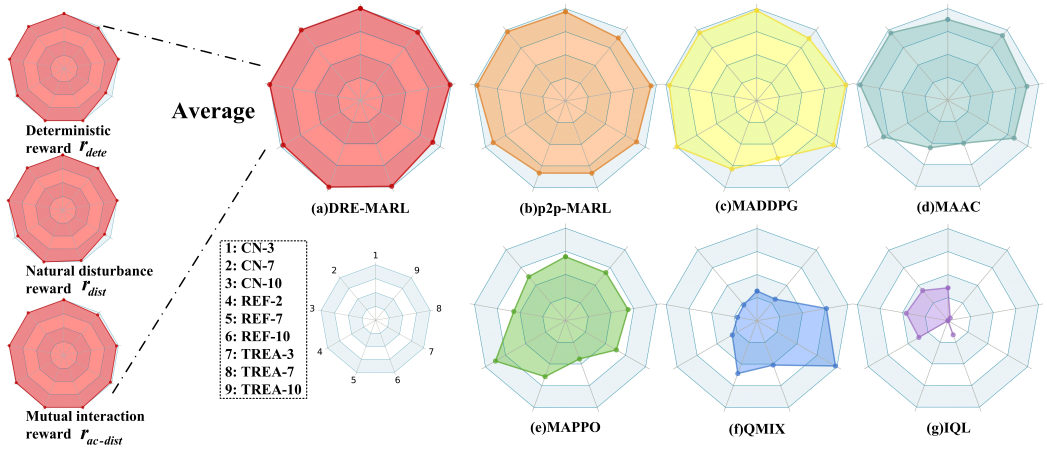

Figure 2: The normalized mean performance under different reward settings (r_{dete}, r_{dist}, $r_{ac-dist}$). Each polygon (shown right) is averaged on three sub-polygons (shown left), where each sub-polygon represents the corresponding reward setting. Each vertex of the sub-polygon denotes the normalized performance, which matches clockwise from CN-3 to TREA-10. The detailed calculation method of normalized performance can be found in Section 6.1.

length as 25 for all experiments and use the mean episode reward as the evaluation metric. Because the value of performance in different scenarios varies considerably, we also consider performance normalization based on all methods. For example, if we have performance values (M_1, M_2, M_3) of three models, then the normalized performance is defined by $M'_v = \frac{\omega(M_v - min(M_1,M_2,M_3))}{max(M_1,M_2,M_3) - min(M_1,M_2,M_3)}$, where ω enables us to distinguish the differences of methods appropriately. We set $\omega = 10$ in the experiments. More details about training and implementation are provided in Appendix C.

6.2 Results and Analysis

Table 1 shows the performance of different methods with the three reward settings. We report the true value of each experiment. The results show that in most of the environments, DRE-MARL reaches the best performance, which reveals the effectiveness of distributional reward estimation. In several scenarios, such as TREA-7 with r_{dete} and r_{dist}, DRE-MARL is not the best, but its performance is very close to the best. Under the $r_{ac-dist}$ reward uncertainty, DRE-MARL fails in TREA-3, TREA-7, and TREA-10. That is because $r_{ac-dist}$ brings more difficulties for attention-based methods in TREA: DRE-MARL and MAAC all adopt multi-head attention architecture, and they are subject to the same trend of changing. We find that in r_{dete} and r_{dist} settings, MAAC achieves the competitive performance, while our method reaches the best performance. But in the $r_{ac-dist}$ setting, the performance of MAAC is pretty bad, and also, DRE-MARL is not the best one. If we consider the utility of DRE based on attention in the setting of TREA with $r_{ac-dist}$ (i.e., Comparation between DRE-MARL and MAAC), DRE brings at least 50% improvement.

To reveal the comprehensive abilities of the above methods, we normalize the performances of all methods and plot the overall performances as shown in Figure 2. The vertexes (1$\sim$9) of the polygon represent the corresponding scenarios (CN-q,REF-q,TREA-q). To get the values on vertexes of polygon, we average the normalized values of performances under all reward settings r_{dete},r_{dist},$r_{ac-dist}$. Figure 2 suggests that DRE-MARL consistently achieves the best overall performance in almost all scenarios. In REF-2 and TREA-3, the performance of DRE-MARL is just slightly lower than MADDPG because the agent number is small, and it is relatively easy to train. Additionally, The results also illustrate that DRE-MARL has a more robust performance with respect to various reward uncertainties because the DRE can capture the reward uncertainty and stabilize training under different reward settings. Besides, exhaustive experiments are shown in Appendix B.

6.3 Ablation Study

Firstly, we perform the ablation study to investigate the impact of different reward estimation methods on the proposed model: 1)DRE-MARL: distributional reward estimation. 2) GRE-MARL:

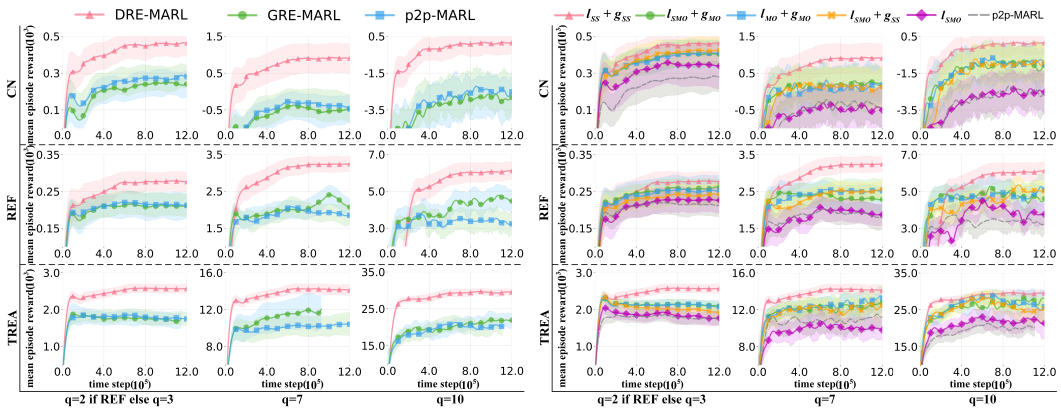

Figure 3: *Left*: Ablation study under reward setting $r_{ac-dist}$. We change the reward estimation strategy, such as global reward estimation (GRE-MARL) and point-to-point reward estimation (p2p-MARL), and probe the influence of different reward estimation methods. *Right*: Reward aggregation analysis on different aggregated schemes. We evaluate our model with different and frequently-used reward aggregation strategies, which enable us to assess our method comprehensively.

global reward estimation based on joint state-action pairs. 3) p2p-MARL: point-to-point reward estimation. As shown in Figure 3 (left), we report the learning curves of the above methods under the reward setting of $r_{ac-dist}$. The results show that DRE-MARL is able to achieve better asymptotic performance under different scenarios. GRE-MARL fails to capture the reward uncertainty because the joint state-action space of MARL is huge, which brings more difficulties for global reward estimation. p2p-MARL does not consider the reward uncertainty caused by mutual interactions among agents as the one-to-many mapping problem, so it performs badly by adopting the one-to-one reward estimation method in MARL scenarios.

To probe the effect of different reward aggregation schemes on the performance, we report the performance of several aggregation schemes in Figure 3 (right) under the $r_{ac-dist}$ setting. Recall the reward aggregation introduced in Section 5.2, we select the following aggregation schemes. 1) $l_{SS}+g_{SS}$, 2) $l_{SMO}+g_{MO}$, 3) $l_{MO}+g_{MO}$, 4) $l_{SMO}+g_{SS}$, 5) l_{SMO} (no aggregation). In Figure 3 (right), we remark that the utility of reward aggregation, which makes the agent make decisions thoughtfully, consistently improves performance in various scenarios compared with GRE-MARL and p2p-MARL. Among all aggregation schemes, $l_{SS}+g_{SS}$ reaches the best performance, and the reported results are also based on this aggregation scheme. More details are shown in Appendix B.2.

7 Conclusion, Limitations, and Broader Impact

In this paper, we present DRE-MARL, a general and effective reward estimation method for reward uncertainty in MARL. To capture the reward uncertainty and stabilize the training process, we investigate the benefits of distributional reward estimation followed by reward aggregation in MARL when reward uncertainty is present. The intuition is that careful consideration of all possible consequences is useful for learning policies. For our proposed DRE-MARL, we propose two stages to perform distributional reward estimation: we first design the multi-action-branch reward estimation to model reward distributions on all action branches. Then we propose policy-weighted reward aggregation with environment rewards and sampled rewards. Here we emphasize that reward aggregation augments the reward from one action branch to all action branches and thus provides more precise updating signals for the critic and actors. Experimental results verify that agents learned by our framework can achieve the best performance and show strong robustness on different types of reward uncertainty. In conclusion, we hope that our work could demonstrate the potential of distributional reward estimation to improve the capacity of MARL when reward uncertainty is present and encourage future works to develop novel reward estimation methods.

In terms of limitations, DRE-MARL requires a prior assumption about the form of reward distribution. Just like tanglesome signals can be factorized into the superposition of simple but elementary

sinusoidal signals, we may consider using a cluster of basic distributions as reward distribution in future work. Another limitation is that reward aggregation is only used in discrete action space. Two solutions may be investigated in the future to solve this limitation: 1) Discretizing of the range of the available action value. 2) Learning a network with actions as inputs, just like how we convert discrete-action DQN [26] into continuous-action Q network in DDPG [19]. We do not see any negative societal impacts of our work while using our method in practice.

Acknowledgments and Disclosure of Funding

We thank all anonymous reviewers for their constructive suggestions. This work is partially supported in part by the National Natural Science Foundation of China under grants Nos.61902145, 61976102, and U19A2065; the International Cooperation Project under grant No. 20220402009GH; and the National Key R&D Program of China under grants Nos. 2021ZD0112501 and 2021ZD0112502.

References

[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

[3] Gilles Blanchard, Marek Flaska, Gregory Handy, Sara Pozzi, and Clayton Scott. Classification with asymmetric label noise: Consistency and maximal denoising. *Electronic Journal of Statistics*, 10(2):2780–2824, 2016.

[4] Haoqiang Chen, Yadong Liu, Zongtan Zhou, Dewen Hu, and Ming Zhang. Gama: Graph attention multi-agent reinforcement learning algorithm for cooperation. *AI*, 50(12):4195–4205, 2020.

[5] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[6] Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement learning with a corrupted reward channel. *arXiv preprint arXiv:1705.08417*, 2017.

[7] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[8] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3582–3589, 2019.

[9] Marek Grzes and Daniel Kudenko. Learning shaping rewards in model-based reinforcement learning. In *Proc. AAMAS 2009 Workshop on Adaptive Learning Agents*, volume 115, page 30. Citeseer, 2009.

[10] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

[11] Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning in discrete action spaces. 2017.

[12] Hlynur Davíð Hlynsson and Laurenz Wiskott. Reward prediction for representation learning and reward shaping. *arXiv preprint arXiv:2105.03172*, 2021.

[13] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR, 2019.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Jinho Lee, Raehyun Kim, Seok-Won Yi, and Jaewoo Kang. Maps: multi-agent reinforcement learning-based portfolio management system. *arXiv preprint arXiv:2007.05402*, 2020.

[16] Lisa Lee, Ben Eysenbach, Russ R Salakhutdinov, Shixiang Shane Gu, and Chelsea Finn. Weakly-supervised reinforcement learning for controllable behavior. *Advances in Neural Information Processing Systems*, 33:2661–2673, 2020.

[17] Xuefeng Li, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. Provably end-to-end label-noise learning without anchor points. In *International Conference on Machine Learning*, pages 6403–6413. PMLR, 2021.

[18] Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward uncertainty for exploration in preference-based reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021.

[19] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[20] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[21] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *AAAI*, volume 34, pages 7211–7218, 2020.

[22] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NIPS*, pages 6379–6390, 2017.

[23] Rubén Majadas, Javier García, and Fernando Fernández. Disturbing reinforcement learning agents with corrupted rewards. *arXiv preprint arXiv:2102.06587*, 2021.

[24] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International conference on machine learning*, pages 125–134. PMLR, 2015.

[25] Alberto Maria Metelli, Matteo Pirotta, and Marcello Restelli. Compatible reward inverse reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] 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. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[28] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

[29] Jan Corazza1 Ivan Gavran2 Daniel Neider. Reinforcement learning with stochastic reward machines. 2022.

[30] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

[31] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[32] Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[33] Jorion Philippe. Value at risk: the new benchmark for managing financial risk. *NY: McGraw-Hill Professional*, 2001.

[34] Wei Qiu, Xinrun Wang, Runsheng Yu, Rundong Wang, Xu He, Bo An, Svetlana Obraztsova, and Zinovi Rabinovich. Rmix: Learning risk-sensitive policies for cooperative reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:23049–23062, 2021.

[35] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[36] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[37] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[38] Joshua Romoff, Peter Henderson, Alexandre Piché, Vincent Francois-Lavet, and Joelle Pineau. Reward estimation for variance reduction in deep reinforcement learning. *arXiv preprint arXiv:1805.03359*, 2018.

[39] Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. In *International Conference on Machine Learning*, pages 5528–5536. PMLR, 2019.

[40] Tim GJ Rudner, Vitchyr Pong, Rowan McAllister, Yarin Gal, and Sergey Levine. Outcome-driven reinforcement learning via variational inference. *Advances in Neural Information Processing Systems*, 34, 2021.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[42] Clayton Scott. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Artificial Intelligence and Statistics*, pages 838–846. PMLR, 2015.

[43] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[44] David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International conference on machine learning*, pages 3191–3199. PMLR, 2017.

[45] Matthew J Sobel. The variance of discounted markov decision processes. *Journal of Applied Probability*, 19(4):794–802, 1982.

[46] Jonathan Sorg, Satinder Singh, and Richard L Lewis. Variance-based rewards for approximate bayesian reinforcement learning. *arXiv preprint arXiv:1203.3518*, 2012.

[47] Sainbayar Sukhbaatar and Rob Fergus. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2(3):4, 2014.

[48] Wei-Fang Sun, Cheng-Kuang Lee, and Chun-Yi Lee. Dfac framework: Factorizing the value function via quantile mixture for multi-agent distributional q-learning. In *International Conference on Machine Learning*, pages 9945–9954. PMLR, 2021.

[49] Wei-Fang Sun, Cheng-Kuang Lee, and Chun-Yi Lee. A distributional perspective on value function factorization methods for multi-agent reinforcement learning. In *AAMAS*, pages 1671–1673, 2021.

[50] Yanchao Sun and Furong Huang. Can agents learn by analogy? an inferable model for pac reinforcement learning. *arXiv preprint arXiv:1912.10329*, 2019.

[51] Yanchao Sun, Cong Qian, Ning Yang, and S Yu Philip. Collaborative inference of coexisting information diffusions. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1093–1098. IEEE, 2017.

[52] Yanchao Sun, Da Huo, and Furong Huang. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. *arXiv preprint arXiv:2009.00774*, 2020.

[53] Yanchao Sun, Xiangyu Yin, and Furong Huang. Temple: Learning template of transitions for sample efficient multi-task rl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9765–9773, 2021.

[54] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087*, 2021.

[55] Yanchao Sun, Ruijie Zheng, Xiyao Wang, Andrew Cohen, and Furong Huang. Transfer rl across observation feature spaces via model-based regularization. *arXiv preprint arXiv:2201.00248*, 2022.

[56] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[57] Erik Talvitie. Learning the reward function for a misspecified model. In *International Conference on Machine Learning*, pages 4838–4847. PMLR, 2018.

[58] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[59] Brendan van Rooyen and Robert C Williamson. Learning in the presence of corruption. *arXiv preprint arXiv:1504.00091*, 2015.

[60] Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6202–6209, 2020.

[61] Jingkang Wang, Hongyi Guo, Zhaowei Zhu, and Yang Liu. Policy learning using weak supervision. *Advances in Neural Information Processing Systems*, 34, 2021.

[62] C Yu, A Velu, E Yinitsky, et al. The surprising effectiveness of mappo in cooperative, multi-agent games [j/ol]. 2021.

[63] Mingqi Yuan, Mon-on Pun, Dong Wang, Yi Chen, and Haojun Li. Multimodal reward shaping for efficient exploration in reinforcement learning. *arXiv preprint arXiv:2107.08888*, 2021.

[64] Kaiqing Zhang, Tao Sun, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. Robust multi-agent reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 33:10571–10583, 2020.

[65] Zhaowei Zhu, Tongliang Liu, and Yang Liu. A second-order approach to learning with instance-dependent label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10113–10123, 2021.

[66] Zhaowei Zhu, Yiwen Song, and Yang Liu. Clusterability as an alternative to anchor points when learning with noisy labels. In *International Conference on Machine Learning*, pages 12912–12923. PMLR, 2021.

[67] Zhaowei Zhu, Jingxuan Zhu, Ji Liu, and Yang Liu. Federated bandit: A gossiping approach. In *Abstract Proceedings of the 2021 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 3–4, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please refer to Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please refer to Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to the Appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please refer to the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please refer to Table 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] The environment code used in the experiments is public.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Please refer to the Appendix.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Please refer to the Section 6.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The adopted scenarios are public benchmark environments.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not use crowdsourcing or conducted research with human subjects.

Appendix

A Pseudocode of DRE-MARL

The pseudocode for DRE-MARL training is shown in Algorithm 1, which takes the following steps. 1) We perform several interactions with the environment and collect experiences in advance. 2) As shown in lines $4-9$ (Algorithm 1), we collect transitions and deposit them in replay buffer D. 3) During the training process, i.e., lines $10-17$ (Algorithm 1), we update the centralized critic, decentralized actors, and reward estimators every 100 time steps following previous methods [13, 22] when the episode ends. 4) We will evaluate our model and refresh the replay buffer periodically. Source code is available at https://github.com/JF-Hu/DRE-MARL.git.

Algorithm 1: Multi-agent Distributional Reward Estimation (DRE-MARL)

Input: N policies $\{\pi^i\}_{i\in\mathcal{N}}$ and target policies $\{\tilde{\pi}^i\}_{i\in\mathcal{N}}$ parameterized by $\boldsymbol{\theta}=\{\theta_i\}_{i\in\mathcal{N}}$ and $\tilde{\boldsymbol{\theta}}=\{\tilde{\theta}_i\}_{i\in\mathcal{N}}$, respectively; Centralized critic and target centralized critic parameterized by ψ and $\tilde{\psi}$, respectively; N reward estimators parameterized by $\boldsymbol{\varphi}=\{\varphi_i\}_{i\in\mathcal{N}}$; The environment with reward uncertainty

Output: ψ, $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$

1 **Initialize:** ψ, $\tilde{\psi}$, $\boldsymbol{\theta}$, $\tilde{\boldsymbol{\theta}}$, $\boldsymbol{\varphi}$, reply buffer D
2 Pre-interact with environment
3 **for** *each iteration* **do**
4 **for** *each time step* t **do**
5   Sample action $a^i_{k,t}\sim\pi^i(\cdot|o^i_t)$ for each agent i
6   Execute joint action $\boldsymbol{a_t}$ in the environment
7   Observe next observation o^i_{t+1} and reward $r^i_{k,t}$ of each agent i
8   Store $\Big(\{o^i_t\},\{\pi^i(\cdot|o^i_t)\},\{a^i_{k,t}\},\{r^i_k\},\{o^i_{t+1}\}\Big)$ in replay buffer D
9 **end**
10 **if** *it's time to update* **then**
11   Sample a batch of transitions $\mathcal{B}$ from D
12   Update distributional reward estimators with Equation 1
13   Sampling rewards $\hat{\boldsymbol{r}}^{\boldsymbol{i}}$ from multi-action-branch reward distributions $\widehat{R}^i$
14   Perform reward aggregation with Equation 2 and Equation 5
15   Update centralized critic and decentralized actors
16   Update target networks $\{\pi_{\tilde{\theta}_i}\}_{i\in\mathcal{N}}$, $V_{\gamma,\tilde{\psi}}$: $\tilde{\theta}_i\leftarrow\tau\theta_i+(1-\tau)\tilde{\theta}_i$,$\tilde{\psi}\leftarrow\tau\psi+(1-\tau)\tilde{\psi}$
17 **end**
18 Evaluate individual policies periodically
19 Refresh replay buffer periodically
20 **end**

B Additional Experiments

B.1 Environmental description

Cooperation Navigation. In this environment, three agents must collaborate to cover all landmarks and avoid colliding with each other simultaneously. The property of the received reward in this environment is set to be collaborative. For more complex environmental settings, we increase the number of agents and landmarks (i.e., Evaluation on 7 and 10 agents). We use the abbreviation CN to denote this environment.

Reference. It is a scenario with two agents and three landmarks. The difference between Cooperative Navigation and Reference is that the target landmark of each agent is only known to its partner. Thus every agent must convey correct information to each other to accomplish the task. To evaluate the capacity of our method in complicated environments, likewise the Cooperative Navigation, we also select 7 and 10 agents for evaluation. We use the abbreviation REF to denote this environment.

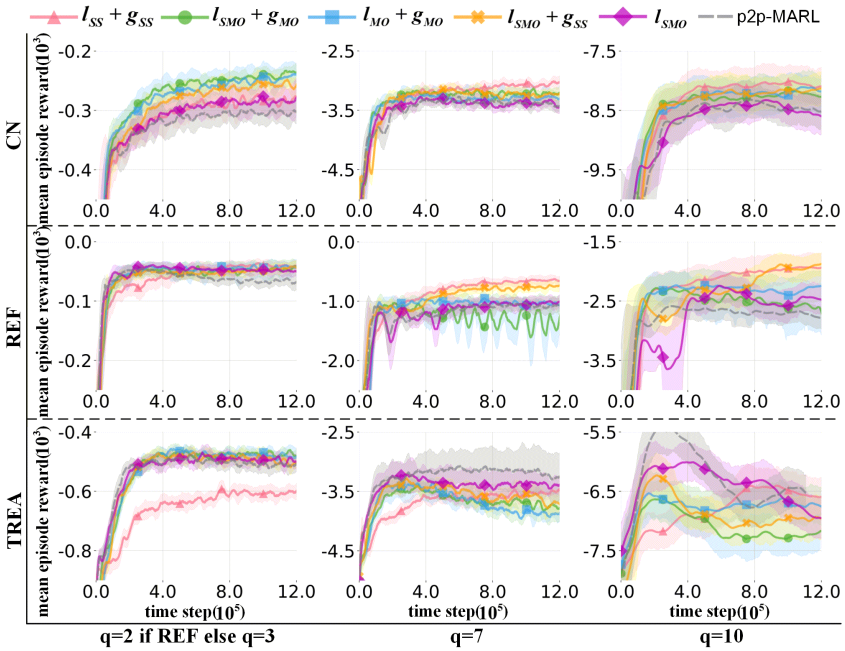

Figure 4: Comparison of the performance on different aggregated schemes while training with the $r_{ac-dist}$ setting and evaluating without the $r_{ac-dist}$ setting.

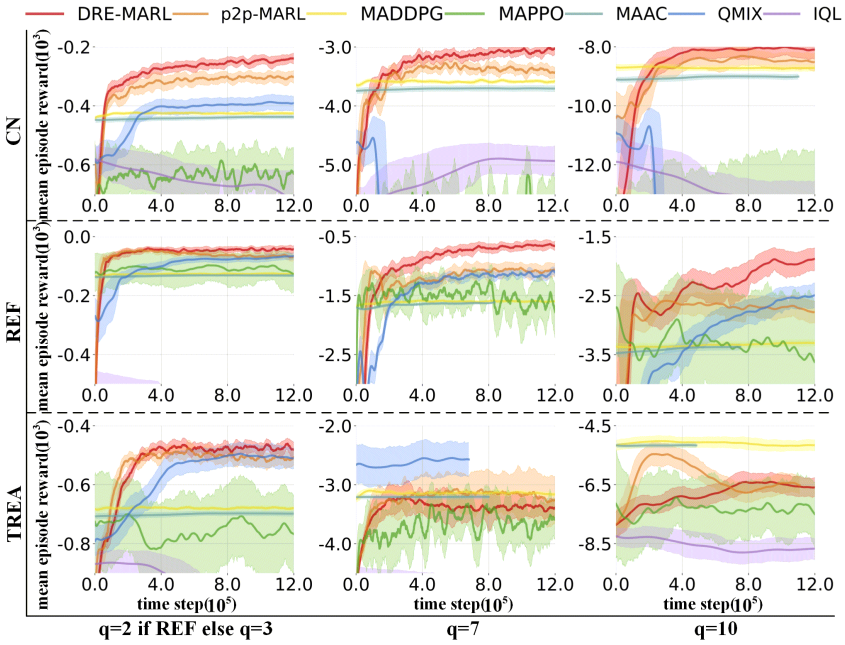

Figure 5: Comparison of DRE-MARL and different baselines while training with the $r_{ac-dist}$ setting and evaluating without the $r_{ac-dist}$ setting.

Treasure Collection. In this scenario, a successful process of treasure collection contains two stages: the first stage is that collectors cover the correct landmarks (i.e., collecting treasure), and the second is that the bank agents successfully receive the treasure. In this scenario, the per episode length is set to 25 because fewer steps bring more challenges to accomplish the two-stage task. The number of collectors and banks is the same, so the collectors should find their unique bank from all banks. Analogously, we choose 3c3b, 7c7b, 10c10b as measurements.

Table 2: Performance comparison of DRE-MARL, DRE-MARL variants, and several SOTA MARL algorithms while training with the individual rewards under the $r_{ac-dist}$ setting. The values represent mean episodic rewards.

Reward Setting	$r_{ac-dist}$						Normalized Performance
Scenario	CN-q			REF-q			
q	3	7	10	2	7	10	
DRE-MARL ($l_{SS} + g_{SS}$)	129$\pm$26.93	36$\pm$77.89	-122$\pm$113.8	23$\pm$64.11	-13$\pm$201.3	402$\pm$108.3	7.893
DRE-MARL ($l_{SMO} + g_{MO}$)	126$\pm$26.54	11$\pm$75.25	-172$\pm$116.4	82$\pm$40.48	258$\pm$68.25	278$\pm$83.82	8.467
DRE-MARL ($l_{MO} + g_{MO}$)	144$\pm$22.80	43$\pm$66.72	-163$\pm$78.67	113$\pm$17.16	310$\pm$55.84	454$\pm$72.59	9.360
DRE-MARL ($l_{SMO} + g_{SS}$)	133$\pm$22.79	39$\pm$70.88	-86$\pm$113.8	11$\pm$61.19	177$\pm$120.8	74$\pm$133.6	7.904
DRE-MARL (l_{SMO})	99$\pm$24.99	1$\pm$69.99	-186$\pm$120.6	44$\pm$45.23	171$\pm$87.49	300$\pm$89.35	7.737
p2p-MARL	92$\pm$25.27	-15$\pm$72.02	-171$\pm$116.9	35$\pm$50.32	192$\pm$101.8	227$\pm$87.09	7.517
GRE-MARL	113$\pm$26.66	-18$\pm$60.79	-249$\pm$98.28	116$\pm$16.30	244$\pm$67.66	231$\pm$98.08	8.171
MADDPG	138$\pm$30.59	47$\pm$84.41	-217$\pm$125.42	-108$\pm$102.8	-371$\pm$166.0	-609$\pm$196.7	4.608
MAPPO	10$\pm$43.89	-313$\pm$168.2	-686$\pm$299.0	-50$\pm$89.86	-103$\pm$243.2	-61$\pm$312.4	2.457
MAAC	132$\pm$17.02	34$\pm$47.49	-146$\pm$70.89	11$\pm$25.14	26$\pm$53.52	31$\pm$66.36	7.324
QMIX	-4$\pm$70.57	-209$\pm$170.6	-715$\pm$195.83	125$\pm$15.78	380$\pm$42.38	577$\pm$45.12	5.970
IQL	-66$\pm$42.65	-309$\pm$115.3	-705$\pm$151.5	-62$\pm$76.95	-210$\pm$137.1	-271$\pm$173.8	1.204

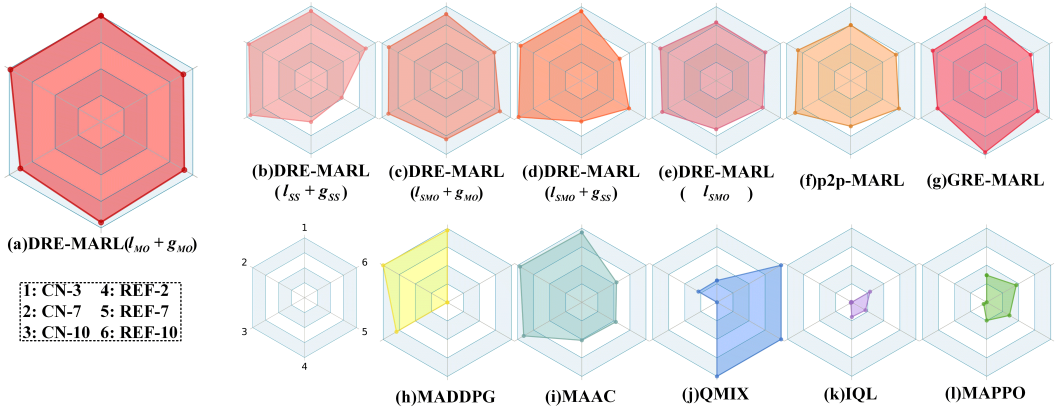

Figure 6: The normalized performance while training with the individual rewards and evaluating with the $r_{ac-dist}$ setting.

B.2 Additional aggregation analysis

There are two choices when evaluating the proposed method. 1) Evaluating in the environment with the reward uncertainty setting as same as training. 2) Evaluating the environment without the setting of reward uncertainty. We usually select the first choice in practice because the training and testing environments are the same. The second choice is also important because it can show whether our model is stuck in the setting of reward uncertainty. We have report the results of first choice in Table 1, Figure 2, and Figure 3. In the next parts, we report the performance of different reward aggregation schemes and other baselines based on the second choice.

We evaluate the performance under the team reward setting in the environments where the $r_{ac-dist}$ is not added. As shown in Figure 4, in most environments, the proposed reward aggregation schemes achieve better performance than p2p-MARL. The aggregation scheme $l_{SMO} + g_{SS}$ is better than $l_{SS} + g_{SS}$ while Figure 3 (right) shows that $l_{SS} + g_{SS}$ is the best. This enlightens us different aggregation schemes are suitable for various purposes. In other words, if we want to train a model in

Table 3: Performance comparison of DRE-MARL, DRE-MARL variants, and several SOTA MARL algorithms while training with the individual rewards and evaluating without the $r_{ac-dist}$ setting. The values represent mean episodic rewards.

Reward Setting	$r_{ac-dist}$						Normalized Performance
Scenario	CN-q			REF-q			
q	3	7	10	2	7	10	
DRE-MARL ($l_{SS} + g_{SS}$)	-403$\pm$33.82	-4148$\pm$416.8	-9283$\pm$442.9	-358$\pm$112.2	-5663$\pm$2102.	-4711$\pm$996.3	7.016
DRE-MARL ($l_{SMO} + g_{MO}$)	-320$\pm$32.84	-3904$\pm$204.1	-9775$\pm$396.8	-166$\pm$81.60	-2183$\pm$300.2	-5744$\pm$668.7	8.457
DRE-MARL ($l_{MO} + g_{MO}$)	-230$\pm$15.54	-3260$\pm$126.8	-8491$\pm$223.9	-70$\pm$13.92	-1361$\pm$175.7	-2882$\pm$384.3	9.797
DRE-MARL ($l_{SMO} + g_{SS}$)	-283$\pm$23.55	-3587$\pm$158.7	-8791$\pm$388.5	-310$\pm$113.5	-2825$\pm$800.7	-7847$\pm$1756.	7.920
DRE-MARL (l_{SMO})	-350$\pm$22.56	-3672$\pm$161.9	-9543$\pm$428.2	-195$\pm$49.99	-2161$\pm$382.7	-4421$\pm$536.6	8.574
p2p-MARL	-367$\pm$23.72	-3717$\pm$160.7	-9241$\pm$397.3	-212$\pm$51.14	-2213$\pm$452.2	-4315$\pm$489.4	8.591
GRE-MARL	-467$\pm$29.11	-4522$\pm$165.5	-11408$\pm$351.5	-103$\pm$16.75	-2604$\pm$435.7	-6964$\pm$773.8	7.715
MADDPG	-429$\pm$6.242	-3589$\pm$42.26	-8693$\pm$66.01	-133$\pm$3.568	-1635$\pm$21.71	-3348$\pm$35.76	8.986
MAPPO	-1074$\pm$240.4	-9349$\pm$2315.	-21475$\pm$5276.	-603$\pm$216.0	-6357$\pm$2021.	-11532$\pm$3721.	1.007
MAAC	-441$\pm$5.511	-3732$\pm$35.19	-9076$\pm$77.81	-140$\pm$2.933	-1712$\pm$23.21	-3504$\pm$40.08	8.817
QMIX	-671$\pm$234.8	-4870$\pm$1128.	-14964$\pm$3096.	-66$\pm$10.64	-1128$\pm$100.0	-2327$\pm$145.4	7.790
IQL	-973$\pm$38.33	-6053$\pm$918.8	-13836$\pm$1331.	-631$\pm$75.07	-7905$\pm$381.7	-16048$\pm$744.0	2.081

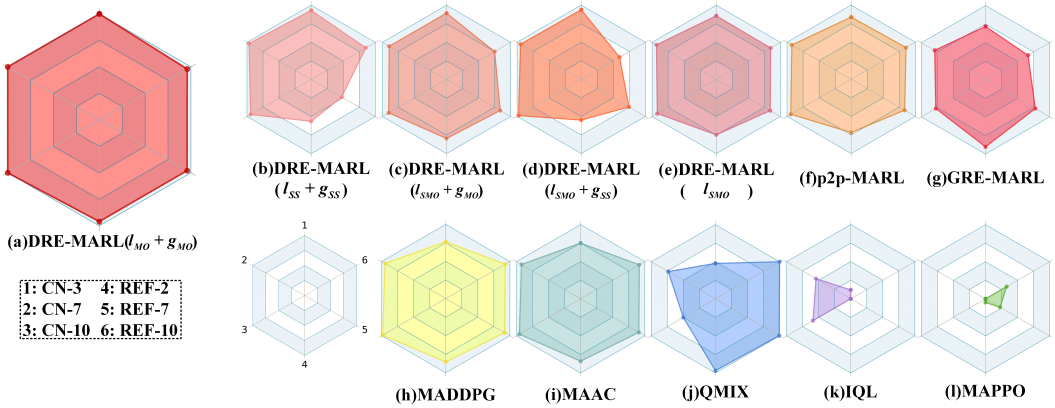

Figure 7: The normalized performance while training with the individual rewards and evaluating without the $r_{ac-dist}$ setting.

precise environments designed by humans, we should choose $l_{SMO} + g_{SS}$ rather than $l_{SS} + g_{SS}$. But if we are given an environment with reward uncertainty and the evaluation is also performed in it, the aggregation scheme $l_{SS} + g_{SS}$ is a better choice.

Furthermore, we also evaluate the performance while training with individual rewards, i.e., the agents can only receive the individual rewards rather than the team rewards during the training process. This environment setting is more complicated than the team reward setting because the agents can cause invalid updating on the centralized critic. The reward aggregation can provide a consistent updating direction for the centralized critic, so as illustrated in Table 2 and Table 3, our method achieves a better performance than the other baselines. The aggregation scheme $l_{MO} + g_{MO}$ reaches the best performance because the mean operation offers a relatively consistent direction of updating.

B.3 Additional experiment results

In order to verify whether our model will be stuck in the setting of reward uncertainty, we evaluate the proposed method and various baselines in the environments without reward uncertainty. We report the learning curves in Figure 5. DRE-MARL performs better than different baselines, which illustrates

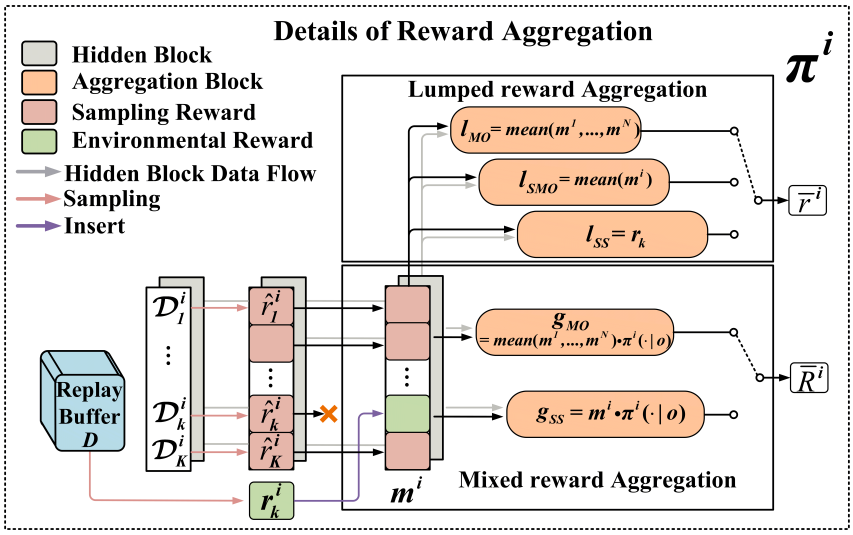

Figure 8: Graphical description of reward aggregation.

that DRE-MARL can reduce the impact of reward uncertainty by adopting multi-action-branch reward estimation and policy-weighted reward aggregation during the training process.

Multi-action-branch reward estimation followed by reward aggregation not only provides a solution to reduce reward uncertainty by augmenting reward from one action branch to all action branches but also offers a good way to improve collaboration by considering the aggregated rewards of all agents. In order to evaluate the performance just given the individual rewards, we set the environment to be non-collaborative during the training process, i.e., the agents can only receive their individual rewards rather than the team reward. Table 2 and Table 3 show the performance of baselines and different aggregation schemes in several scenario variants. The comprehensive abilities of the above methods are shown in Figure 6 and Figure 7. DRE-MARL ($l_{MO} + g_{MO}$) achieves better performance than other aggregation schemes and baselines in most environments. The reason is that reward aggregation provides a good solution to consider the holistic impact of individual agents when updating the centralized critic so as to achieve the common goal. Although QMIX reaches higher performance in REF, the difference is slight, and the overall performance of DRE-MARL ($l_{MO} + g_{MO}$) is obviously better than QMIX. Besides, in order to train the QMIX model, we actually calculate the team reward for QMIX during training, which violates the environment setting of training with the individual rewards. The value decomposition adopted by QMIX can be regarded as one backward method from the team reward to the individual rewards for solving MARL tasks. Our method provides a brand-new forward solution from the individual rewards to the team reward by using multi-action-branch reward estimation followed by policy-weighted reward aggregation.

C Implement Details

Details of Reward Aggregation. For each agent, we first sample rewards $\hat{r}^i$ from estimated reward distributions $\mathcal{D}^i$. Then, we construct the built-up reward vector m^i by replacing the value of $\hat{r}^i$ on the k-th action branch with the environment reward r_k^i. Next, we aggregate $\{m^i\}_{i\in\mathcal{N}}$ with different operations, as shown in Figure 8. Finally, we obtain the mixed reward $\bar{R}$ and the lumped reward $\bar{r}$ to train the centralized critic and the decentralized actors, respectively. The details of reward aggregation are shown in Figure 8.

Network Architecture. The decentralized actors and distributional reward estimation networks adopt the simple fully-connected feedforward neural network with three layers in our framework. The two hidden layers' units are 64. $|\mathcal{A}|$ and $2|\mathcal{A}|$ are the number of output units in actors and distributional reward estimation networks, respectively, where $|\mathcal{A}|$ represents the size of action space. The centralized critic uses a graph attention neural network with eight attention heads, and each head's hidden unit is set to 8 to capture the dynamic relationship between agents. After the graph

attention neural network, a two-hidden-layer fully-connected feedforward neural network is used to get the state value. Besides, we choose leaky relu as nonlinear activation for all networks.

Compute. Experiments are carried out on NVIDIA GeForce RTX 1080Ti GPUs and with fixed hyperparameter settings, which are described in the following. Each run of the experiments spanned about 2-12 hours, depending on the algorithm and the agents' number in the environment.

Hyperparameters. For all scenarios, the per-episode length is set to 25. The periodical replay buffer learning rate is 0.4, the learning interval is set to 4 episodes, and the entropy scale is set to 0.3. Detailed hyperparameters can be found in Table 4.

Table 4: The hyperparameters of DRE-MARL.

Hyperparameter	Value
optimizer	Adam [14]
learning rate	$1 \cdot 10^{-3}$
entropy scale	0.1/0.3
number of hidden layers (all networks)	2
nonlinearity of hidden layers (all networks)	Leaky ReLU
number of hidden units per layer (all networks)	64
number of attention head	8
time difference (TD)	1
buffer clear rate	0.4
discount (γ)	0.95
batch size	1024
ε-greedy	0.7$\rightarrow$0.9
tau (τ)	0.01
α	0.1
β	10
η	0.3

D More discussion of limitations

Continuous action space. In our method, the number of action branches matches the number of available discrete actions, so we can set the value of K to be equal to the number of available discrete actions such as "move forward", "move backward", "move left", "move right", and "motionless" in MPE. But in continuous action space, for example, we want to manipulate a robot arm. The "grasping" action needs us to assign a continuous value such as rotation angle to the robot arm. Right now, the available action is a range, so we can not define the number of K. We discuss the following possible solutions to solve the problem of continuous action space.

- One possible solution may be the discretization of the range of the available action value. More sophisticated discretization will bring better manipulation, but it will consume more computational resources at the same time. Although coarse discretization reduces the consumption of physical time, it may hurt the performance.
- Another possible solution is learning a network with actions as inputs, just like how we convert discrete-action DQN into continuous-action Q network in DDPG. The implicit reward function makes it possible to perform reward aggregation by sampling several discrete action points.

Prior Gaussian distribution. The reward distribution is indeed a bit hard to select in practice. However, just like tanglesome signals can be factorized into the superposition of simple but elementary sinusoidal signals, we may consider using a cluster of basic distributions as reward distribution in future work.

Table 5: The comparison of the computational cost about different models based on the CN-3 scenario. The items of the comparison contain three aspects: parameters, physical training time (min), and max memory consumption (MB).

comparison items		model					
		DRE-MARL	MAPPO	MAAC	QMIX	MADDPG	IQL
para-meters	total	198440	72854	450364	307330	172884	564642
	trainable	54126	72854	450364	123488	86436	141147
	untrainable	144314	0	0	183842	86448	423495
physical training time (min)		256.0$_{\pm 1.859}$	336.0$_{\pm 9.623}$	915.9$_{\pm 6.270}$	867.1$_{\pm 16.54}$	166.9$_{\pm 1.976}$	217.4$_{\pm 2.311}$
max memory consumption (MB)		1354.$_{\pm 60.18}$	310.6$_{\pm 0.297}$	229590.1$_{\pm 65.69}$	2146.5$_{\pm 22.95}$	3754.$_{\pm 227.3}$	668.4$_{\pm 20.18}$

Table 6: Performance comparison of DRE-MARL with and without regularization term L_R while training with the team rewards and evaluating without the $r_{ac\text{-}dist}$ setting. The values represent mean episodic rewards.

comparison items		model
		DRE-MARL
$l_{SS} + g_{SS}$	with L_R	-272.7$_{\pm 25.22}$
	without L_R	-274.6$_{\pm 19.13}$
$l_{SMO} + g_{SS}$	with L_R	-235.1$_{\pm 16.38}$
	without L_R	-365.0$_{\pm 25.19}$
$l_{MO} + g_{MO}$	with L_R	-242.1$_{\pm 16.15}$
	without L_R	-258.9$_{\pm 17.36}$
$l_{MO} + g_{SS}$	with L_R	-252.6$_{\pm 17.49}$
	without L_R	-354.2$_{\pm 23.47}$
no reward estimation		-258.2$_{\pm 20.75}$

Table 7: Performance comparison of DRE-MARL variants and DFAC variants. We conduct the experiment in CN-3 with DFAC variants which is based on SC II originally, and adopt the original hyperparameters of DFAC variants. The DFAC-diql(128) denotes the number of hidden neural is 128.

model	performance
$l_{SS} + g_{SS}$	-272.7$_{\pm 25.22}$
$l_{SMO} + g_{SS}$	-235.1$_{\pm 16.38}$
$l_{MO} + g_{MO}$	-242.1$_{\pm 16.15}$
$l_{MO} + g_{SS}$	-252.6$_{\pm 17.49}$
DFAC-diql(128)	-786.9$_{\pm 180.2}$
DFAC-diql(256)	-1117.9$_{\pm 21.38}$
DFAC-dmix(128)	-1121.6$_{\pm 21.51}$