Disentangling Sources of Risk for Distributional Multi-Agent Reinforcement Learning

Kyunghwan Son [1] Junsu Kim [1] Sungsoo Ahn [2] Roben Delos Reyes [1] Yung Yi [1] Jinwoo Shin [1]

Abstract

In cooperative multi-agent reinforcement learning, the outcomes of agent-wise policies are highly stochastic due to the two sources of risk: (a) random actions taken by teammates and (b) random transition and rewards. Although the two sources have very distinct characteristics, existing frameworks are insufficient to control the risk-sensitivity of agent-wise policies in a disentangled manner. To this end, we propose **D**isentangled **RI**sk-sensitive **M**ulti-**A**gent reinforcement learning (DRIMA) to separately access the risk sources. For example, our framework allows an agent to be optimistic with respect to teammates (who can prosocially adapt) but more risk-neutral with respect to the environment (which does not adapt). Our experiments demonstrate that DRIMA significantly outperforms prior state-of-the-art methods across various scenarios in the StarCraft Multi-agent Challenge environment. Notably, DRIMA shows robust performance where prior methods learn only a highly suboptimal policy, regardless of reward shaping, exploration scheduling, and noisy (random or adversarial) agents.

1. Introduction

In cooperative multi-agent reinforcement learning (MARL), the paradigm of centralized training with decentralized execution (CTDE) has shown great success in recent years (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019; Rashid et al., 2020a; Wang et al., 2020b). Under such a paradigm, researchers execute agent-wise policies without inter-agent communication, i.e., use decentralized execution, while they train the agents with access to the full information, i.e., apply centralized training.

[1]Korea Advanced Institute of Science and Technology (KAIST) [2]Pohang University of Science and Technology (POSTECH). Correspondence to: Kyunghwan Son <kevinson9473@kaist.ac.kr>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Table 1: Payoff matrix of the two-step game. $\{e_1, e_2\}$ denotes a reward set in which one element of this set is uniformly sampled as a reward. The payoffs of the first and the second step are designed to assess whether an algorithm is able to handle cooperative and environmental uncertainty, respectively.

	A	B	C
A	7	-12	-12
B	-12	0	0
C	-12	0	0

(a) Payoff in the first step.

	A	B	C
A	$\{-10, 10\}$	$\{-11, 9\}$	$\{-11, 9\}$
B	$\{-11, 9\}$	$\{-8, -6\}$	$\{-11, 9\}$
C	$\{-11, 9\}$	$\{-11, 9\}$	$\{-26, 12\}$

(b) Payoff in the second step.

However, despite the progress on value-based CTDE, agents still often fail to cooperate due to environment randomness, limited observation of agents, and time-varying policies of other agents, especially in highly stochastic environments. For such environments in single-agent settings, risk-sensitive reinforcement learning (RL) (Chow & Ghavamzadeh, 2014) has shown remarkable results by using policies that consider *risk* rather than simple expectations for return distribution caused by state transitions, rewards, and actions. Here, risk refers to the uncertainty over possible outcomes, and risk-sensitive policies act with a risk measure, such as variance or conditional value at risk (CVaR) (Chow & Ghavamzadeh, 2014). The main goal of this paper is to apply this risk-sensitive technique to value-based CTDE to learn more robust policies against various factors of uncertainty.

In this regard, our motivating observation is that in MARL, the returns are affected by two types of uncertainty: (a) *cooperative uncertainty* and (b) *environmental uncertainty*. Here, the cooperative uncertainty stems from how the agents cannot communicate with each other. Furthermore, environmental uncertainty is caused by stochastic transition and the rewarding mechanism of the environment. See Figure 1 for an illustration of such uncertainties.

The existing distributional reinforcement learning frameworks are, however, unable to disentangle such sources of risk. This is important since the cooperative risk-seeking yet environmental risk-neutral policies are likely to be of favorable choices in many practical scenarios (i.e., surviving

Table 2: Test rewards in the stochastic two-step matrix game for DRIMA, DMIX with varying risk-sensitivity across twelve random seeds.

Algorithm	Risk sensitivity		Reward
	Cooperative	Environmental	
DRIMA (Ours)	Seeking	Neutral	**6.98**
DMIX (Sun et al., 2021)	Neutral		-0.02
	Seeking		0.03

together as a team for a long time under the expectation that every teammate will cooperate for the sake of the team). In Table 1, we design a simple matrix game showing the importance of disentangling such risk sources. At the first step of the game, cooperative risk-seeking is crucial for the agents to take the optimal action A. Otherwise, an agent 1 will take action B or C to consider the risk of teammate agent 2 taking a non-cooperative action B or C. However, for the second step, both the agents should take an environmental risk-neutral decision A to maximize the expected reward. DMIX (Sun et al., 2021), a representative distributional MARL method, has limited capability in adjusting risk-sensitivity in this simple example. As shown in Table 2, we observe that changing risk-sensitivity in DMIX affects two sources of risk simultaneously. Namely, risk-seeking adjustment makes both cooperative and environmental risk-sensitivity become seeking. Therefore, agents can obtain a high mean reward by selecting action A in the first step, but get a low mean reward by selecting action C in the second step. More details about the implementation of the matrix game are included in Appendix E.

Contribution. In this paper, we present **D**isentangled **RI**sk-sensitive **M**ulti-**A**gent reinforcement learning (DRIMA), a novel framework on disentangling sources of uncertainty for distributional MARL. The main idea is to separate risk levels in both centralized training and decentralized execution with a hierarchical quantile structure and quantile regression. Unlike prior risk-sensitive MARL frameworks (Qiu et al., 2021; Sun et al., 2021), DRIMA considers cooperative risk into each utility function and environmental risk into a joint true action-value estimator. Therefore, each utility function is encouraged to learn the action-value distribution with respect to other agents' cooperative policy, and the joint-action value estimator is trained to learn action-value distribution with respect to the environment stochasticity. As illustrated in Table 2, our DRIMA can obtain a higher reward compared to the prior risk-sensitive MARL approach by learning a cooperative risk-seeking and environmental risk-neutral policy.

We also demonstrate the effectiveness of DRIMA on various scenarios in the StarCraft[1] Multi-Agent Challenge (SMAC) environment which is widely used as the environment for many MARL research (Samvelyan et al., 2019). DRIMA significantly outperforms state-of-the-art methods, including distributional MARL methods and non-distributional ones. Notably, by disentangling sources of risk, DRIMA shows impressive results regardless of reward shaping and exploration schedule, whereas existing works learn only a suboptimal policy (see Figure 3).

Finally, we demonstrate the importance of risk-sensitivity control through experiments in the presence of random and adversarial agents. In the real world, a multi-agent system can be easily broken due to the presence of a malicious or a broken agent, and agents are necessary to mitigate these potential dangers. DRIMA, which is capable of disentangled sources of risk, can learn a robust policy even in these situations. We hope that our idea will motivate new future research directions such as safe MARL.

2. Preliminaries

2.1. Centralized training with decentralized execution

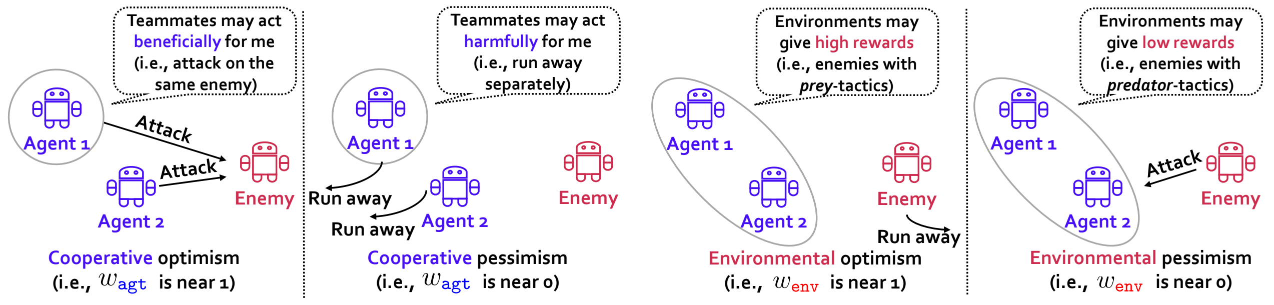
In this paper, we consider a decentralized partially observable Markov decision process (Oliehoek et al., 2016) represented by a tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{U}, P, r, O, N, \gamma \rangle$. Here, $s \in \mathcal{S}$ denotes the true state of the environment. At each step, an agent $i \in \mathcal{N} := \{1, ..., N\}$ selects an action $u_i \in \mathcal{U}$ as an element of the joint action vector $\boldsymbol{u} := [u_1, \cdots, u_N]$. Then the next state s' is determined by a stochastic transition dynamic $P(s'|s, \boldsymbol{u})$. All the agents cooperate to maximize the reward $r(s, \boldsymbol{u})$ that is discounted by a factor of γ for each step. Each agent i is associated with a partial observation defined by the observation function $O(s, i) : \mathcal{S} \times \mathcal{N} \mapsto \mathcal{O}$, and an action-observation history τ_i. Concatenation of all the agent-wise action-observation histories is coined as the overall history $\boldsymbol{\tau}$.

We consider value-based centralized training with decentralized execution (CTDE) (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019). Here, agents are trained in a centralized manner and executed in parallel without access to the global state s. Under value-based CTDE, we aim to train each agent-wise utility function $q_i(\tau_i, u_i)$ whose greedy actions are consistent with the greedy actions from the joint action-value estimator $Q_{\texttt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u})$. Formally, the following *decentralization* condition should be satisfied:

$$\arg\max_{\boldsymbol{u}} Q_{\texttt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}) = \begin{pmatrix} \arg\max_{u_1} q_1(\tau_1, u_1) \\ \vdots \\ \arg\max_{u_N} q_N(\tau_N, u_N) \end{pmatrix}. \quad (1)$$

To meet this condition, value-decomposition networks (VDN, Sunehag et al. 2018) learns a centralized yet factored joint action-value estimator by representing the joint action-value estimator as a sum of individual agent-wise

[1]StarCraft is a trademark of Blizzard Entertainment$^{\text{TM}}$.


Teammates may act beneficially for me (i.e., attack on the same enemy)
Agent 1
Attack
Attack
Agent 2
Enemy
Cooperative optimism (i.e., $w_{\texttt{agt}}$ is near 1)
Teammates may act harmfully for me (i.e., run away separately)
Run away
Run away
Cooperative pessimism (i.e., $w_{\texttt{agt}}$ is near 0)
Environments may give high rewards (i.e., enemies with prey-tactics)
Run away
Environmental optimism (i.e., $w_{\texttt{env}}$ is near 1)
Environments may give low rewards (i.e., enemies with predator-tactics)
Environmental pessimism (i.e., $w_{\texttt{env}}$ is near 0)

(a) cooperative uncertainty (b) Environmental uncertainty

Figure 1: Two types of uncertainty in MARL: (a) cooperative uncertainty and (b) environmental uncertainty. Note that friendly agents learn a policy over time, while enemies in the environment act through a stationary distribution. Current value-based CTDE methods do not consider two sources of risk explicitly or tackle them in an entangled way which may lead to a suboptimal solution.

utility functions. QMIX (Rashid et al., 2018) extends VDN by employing a *mixing network* to express a non-linear monotonic relationship among individual agent-wise utility functions in the joint action-value estimator. QTRAN (Son et al., 2019) has been proposed to eliminate the monotonic assumption on the joint action-value estimator in QMIX (Rashid et al., 2018). Instead of directly decomposing the joint action-value estimator into utility functions, QTRAN proposes a training objective that enforces the decentralization of the joint action-value estimator into the summation of individual utility functions.

Recently, several other methods have been proposed to solve the limitations of QMIX. QPLEX (Wang et al., 2020b) takes a duplex dueling network architecture to factorize the joint value function. Unlike QTRAN, QPLEX learns using only one joint action-value network and a single loss function. Also, Rashid et al. (2020a) proposed CW-QMIX and OW-QMIX, which use a weighted projection that allows more emphasis to be placed on better joint actions. We provide a more detailed description of other CTDE algorithms in Appendix C.

2.2. Distributional reinforcement learning

Instead of training a scalar state-action estimator $Q^\pi(s,u)$, distributional RL represents an action-value as a random variable $Z^\pi(s,u)$. Then the distributional Bellman operator for policy evaluation in single-agent RL can then be expressed as follows:

$$Z^\pi(s,u) \overset{D}{=} R(s,u) + \gamma Z^\pi(S',U'), \tag{2}$$

where random variables R, S', and U' are distributed according to $P_R(\cdot|s,u), P_{S'}(\cdot|s,u)$, and $\pi(\cdot|s')$, respectively. Furthermore, $A \overset{D}{=} B$ denotes that the two random variables a and b follow the same probability distribution.

Implicit quantile network (IQN). IQN (Dabney et al., 2018a) is a popular framework for distributional RL. It approximates the (true) action-value as a distribution using a quantile-based representation. To be specific, a deterministic parametric function $Z(s,u,\omega)$ is trained to reparametize samples ω drawn from a base distribution $\mathcal{U}(0,1)$, to the respective quantile values of a distribution.

To train the IQN, one can use the following Huber quantile regression loss (Huber, 1964):

$$\mathcal{L}_{\texttt{IQN}} = \begin{cases} |\omega - \mathbb{1}_{\delta>0}| \times \frac{1}{2}\delta^2, & \text{if } |\delta| \le 1, \\ |\omega - \mathbb{1}_{\delta>0}| \times (|\delta| - \frac{1}{2}), & \text{otherwise,} \end{cases}$$
$$\delta = Z(s,u,\omega) - (r + \gamma Z^{\texttt{tar}}(s', u'_{\texttt{opt}}, \omega')),$$
$$u'_{\texttt{opt}} = \arg\max_{u'} \mathbb{E}_w[Z^{\texttt{tar}}(s',u',w)],$$

where $Z^{\texttt{tar}}$ is the target network whose parameters are updated periodically from the original action-value estimator Z. Next, (s,u,r,s') is a tuple of experience transitions from a replay buffer. In distributional RL, weights w are used to express a level of rewards and next state transitions are generated from the randomness of the environment.

Distributional MARL. Recently, distributional RL has also been applied to CTDE regime (Qiu et al., 2021; Sun et al., 2021), but they lack disentanglement of risk sources. RMIX (Qiu et al., 2021) and DFAC (Sun et al., 2021) showed promising results by extending agent-wise utility functions from deterministic variables to random variables. RMIX demonstrates the effectiveness of risk-sensitive MARL via distributional RL, but it is limited since they cannot represent policies, which have different risk levels relying on sources (i.e., cooperative risk-seeking yet environmental risk-averse policies). DFAC is another distributional MARL framework with proposed mean-shape decomposition while employing IQN network for agent-wise utility function, but they only showcase risk-neutral policies. We further describe distributional MARL methods in Appendix C.

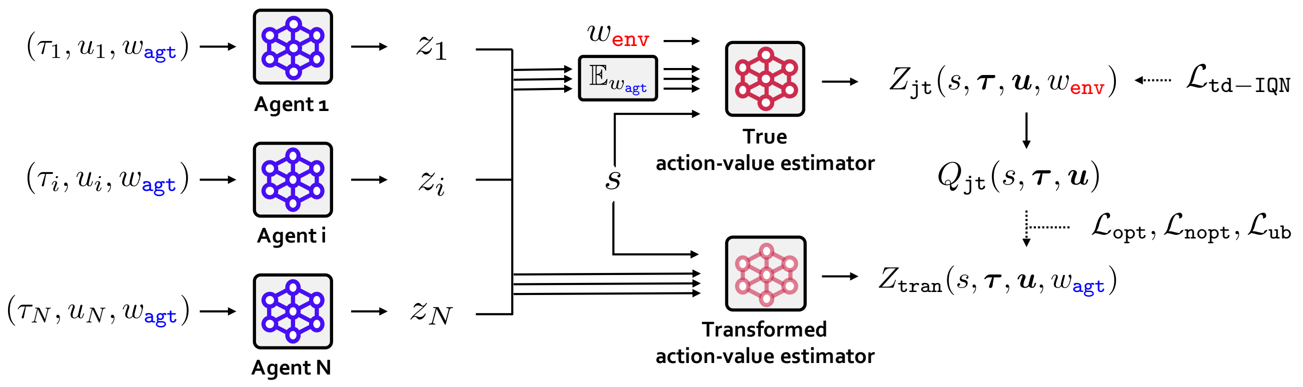

$(\tau_1, u_1, w_{\mathtt{agt}})$
Agent 1
z_1
$(\tau_i, u_i, w_{\mathtt{agt}})$
Agent i
z_i
$(\tau_N, u_N, w_{\mathtt{agt}})$
Agent N
z_N
$w_{\mathtt{env}}$
$\mathbb{E}_{w_{\mathtt{agt}}}$
s
True action-value estimator
Transformed action-value estimator
$Z_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}, w_{\mathtt{env}})$
$\mathcal{L}_{\mathtt{td-IQN}}$
$Q_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u})$
$\mathcal{L}_{\mathtt{opt}}, \mathcal{L}_{\mathtt{nopt}}, \mathcal{L}_{\mathtt{ub}}$
$Z_{\mathtt{tran}}(s, \boldsymbol{\tau}, \boldsymbol{u}, w_{\mathtt{agt}})$

Figure 2: Architecture of DRIMA. Each agent-wise utility function z_i and true action-value estimator $Z_{\mathtt{jt}}$ are built based on IQN and input a quantile of cooperative risk $w_{\mathtt{agt}}$ and environmental risk $w_{\mathtt{env}}$, respectively. Further details for each component are provided in Appendix B.

3. DRIMA: Disentangled risk-sensitive MARL

In this section, we propose a new method, called DRIMA, which aims at disentangling the two sources of risk. In Section 3.1, we present a systematic analysis of the sources of risk for multi-agent reinforcement learning and a motivation for why our architecture is needed. In Section 3.2, we propose the training objective of DRIMA for training our action-value estimators. Finally, in Section 3.3, we introduce a new architecture for the action-value estimators. The overall architectural sketch is given in Figure 2.

3.1. Motivation

Environmental risk in MARL. Instead of training a scalar joint state-action estimator $Q_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u})$, distributional multi-agent reinforcement learning represents an action-value as a random variable, denoted $Z_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u})$. The distributional Bellman operator for policy evaluation in multi-agent reinforcement learning can then be expressed as follows:

$$Z^{\pi}_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}) \overset{D}{=} R(s, \boldsymbol{\tau}, \boldsymbol{u}) + \gamma Z^{\pi}_{\mathtt{jt}}(S', \mathcal{T}', U'), \qquad (3)$$

where random variables $R, S', \mathcal{T}'$, and U' are distributed according to $P_R(\cdot|s, \boldsymbol{\tau}, \boldsymbol{u})$, $P_{S'}(\cdot|s, \boldsymbol{\tau}, \boldsymbol{u})$, $P_{\mathcal{T}'}(\cdot|s, \boldsymbol{\tau}, \boldsymbol{u})$, and $\pi(\cdot|s', \boldsymbol{\tau}')$, respectively. Furthermore, $A \overset{D}{=} B$ denotes the two random variables which follow the same probability distribution. Here, the source of risk is the stochasticity of the environment, and policy represented by $\pi(\boldsymbol{u}'|s', \boldsymbol{\tau}')$. Then, the scalar action-value function is defined as $Q^{\pi}_{\mathtt{jt}} = \mathbb{E}[Z^{\pi}_{\mathtt{jt}}]$, and can be characterized by the Bellman equation:

$$Q^{\pi}_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}) = \mathbb{E}[R(s, \boldsymbol{\tau}, \boldsymbol{u})] + \gamma \mathbb{E}[Q^{\pi}_{\mathtt{jt}}(s', \boldsymbol{\tau}', \boldsymbol{u}')].$$

Cooperative risk in VDN and QMIX. VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) are methods that attempt to decentralize $Q_{\mathtt{jt}}$ based on additive and monotonic factorization, respectively. For VDN, we define a joint action-value function as the linear sum of individual agent-wise utility functions q_i as follows:

$$Q_{\mathtt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{N} q_i(\tau_i, u_i).$$

This linear value factorization structure solves the decentralized decision making problem.

However, due to the limited expressive power of the factorized estimator, the learned values include additional randomness. For the Bellman equation target, Wang et al. (2020a) proves the closed-form solution for individual agent-wise utility function $q_i(\tau_i, u_i)$ which is calculated as:

$$\underset{(s,\tau_{-i},u_{-i})\sim P(\cdot|\tau_i)}{\mathbb{E}} [y(s, \tau_i \oplus \tau_{-i}, u_i \oplus u_{-i})] + v_i(\tau_i),$$
$$y(s, \boldsymbol{\tau}, \boldsymbol{u}) = \mathbb{E}[R(s, \boldsymbol{\tau}, \boldsymbol{u})] + \gamma \mathbb{E}[Q^{\pi}_{\mathtt{jt}}(s', \boldsymbol{\tau}', \boldsymbol{u}')]$$

where $(\tau_i \oplus \tau_{-i})$ denotes $< \tau_1, \ldots, \tau_i, \ldots, \tau_N >$ and τ_{-i} denotes the elements of all agents except for agent i. Also, $v_i(\tau_i)$ is the observation-dependent baseline that does not influence action choices. The first term is the expected value of an individual action u_i over the actions of other agents, which evaluates the expected return of executing an individual action u_i. For example, as shown in Figure 1, the value and optimal action of one agent may vary according to the actions of other agents. The source of risk in the equation is the limited expressive power of value factorization and the stochasticity of the state and actions of other agents from the point of view of one agent.

Inspired by this, rather than handling risk in an entangled manner, we disentangle them via two separate action-value estimators with different roles; (i) a true action-value estimator $Z_{\mathtt{jt}}(w_{\mathtt{env}})$ which learns the joint action-value y without structural constraints and captures environmental risk with a risk level $w_{\mathtt{env}}$, (ii) a transformed action-value estimator $Z_{\mathtt{tran}}(w_{\mathtt{agt}})$ which learns an action-value guided by the true-action value estimator while being decentralizable into the utility functions and capturing cooperative risk with a risk level $w_{\texttt{agt}}$.

3.2. Training objectives

The training objective of DRIMA is twofold. First, DRIMA trains the true action-value estimator to approximate the distributional true action-value estimator $Z_{\texttt{jt}}$ based on the distributional Bellman operator (Equation 3). Next, the transformed action-value estimator attempts to imitate the behavior of the true action-value estimator through weights. Unlike the action-value estimator of DMIX, the true action-value estimator $Z_{\texttt{jt}}$ of DRIMA does not have any structural restrictions such as monotonicity. Therefore, $Z_{\texttt{jt}}$ can accurately represent only the risk of the environment. On the other hand, the transformed action-value estimator $Z_{\texttt{tran}}$ limits its expressive power to satisfy the decentralization condition (Equation 1). We use a weighting mechanism to track the $\mathbb{E}_{w_{\texttt{env}}}[Z_{\texttt{jt}}(w_{\texttt{env}})]$ to exclude the risk of the environment.

Loss for the true action-value estimator. The loss $\mathcal{L}_{\texttt{td-IQN}}$ trains the true action-value estimator $Z_{\texttt{jt}}$ with consideration of the environmental risk. To be specific, the loss is designed based on the IQN loss derived from the Bellman operator (Equation 3) and the Huber quantile loss (Huber, 1964):

$$\mathcal{L}_{\texttt{td-IQN}} = \begin{cases} |w_{\texttt{env}} - \mathbb{1}_{\delta>0}| \times \frac{1}{2}\delta^2, & \text{if } |\delta| \le 1, \\ |w_{\texttt{env}} - \mathbb{1}_{\delta>0}| \times (|\delta| - \frac{1}{2}), & \text{otherwise}, \end{cases}$$
$$\delta = Z_{\texttt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}, w_{\texttt{env}}) - (r + \gamma Z^{\texttt{tar}}_{\texttt{jt}}(s', \boldsymbol{\tau}', \boldsymbol{u}'_{\texttt{opt}}, w'_{\texttt{env}})),$$

where $Z^{\texttt{tar}}_{\texttt{jt}}$ is the target network whose parameters are updated periodically from the original estimator $Z_{\texttt{jt}}$. Furthermore, $\boldsymbol{u}'_{\texttt{opt}}$ is the actions maximizing the utility functions $z_i(\tau'_i, u'_i)$ for $i \in \mathcal{N}$. For the $\mathcal{L}_{\texttt{td-IQN}}$, risk-level samples $w_{\texttt{env}}, w'_{\texttt{env}}$ are drawn from a uniform distribution $\mathcal{U}(0, 1)$.

Loss for the transformed action-value estimator. We propose $\mathcal{L}_{\texttt{opt}}, \mathcal{L}_{\texttt{nopt}}, \mathcal{L}_{\texttt{ub}}$ to encourage the transformed action-value estimator to track the value of the true action-value estimator while considering cooperative risk level $w_{\texttt{agt}}$. The formulation of each loss is given as follows, where we omit the common function arguments $(s, \boldsymbol{\tau})$ for presentational simplicity:

$$\mathcal{L}_{\texttt{opt}} = \left(Z_{\texttt{tran}}(\boldsymbol{u}_{\texttt{opt}}, w_{\texttt{agt}}) - Q_{\texttt{jt}}(\boldsymbol{u}_{\texttt{opt}})\right)^2,$$
$$\mathcal{L}_{\texttt{nopt}} = \alpha(Z_{\texttt{tran}}(\boldsymbol{u}, w_{\texttt{agt}}) - Q_{\texttt{jt}}(\boldsymbol{u}))^2,$$
$$\mathcal{L}_{\texttt{ub}} = \beta(Z_{\texttt{tran}}(\boldsymbol{u}, w_{\texttt{agt}}) - Q_{\texttt{ub}}(\boldsymbol{u}))^2,$$
$$\alpha = \begin{cases} 1, & \text{if } Z_{\texttt{tran}}(\boldsymbol{u}, w_{\texttt{agt}}) \le Q_{\texttt{jt}}(\boldsymbol{u}), \\ 2(1 - w_{\texttt{agt}}), & \text{otherwise}, \end{cases}$$
$$\beta = \begin{cases} 1, & \text{if } Z_{\texttt{tran}}(\boldsymbol{u}, w_{\texttt{agt}}) \ge Q_{\texttt{ub}}(\boldsymbol{u}), \\ 0, & \text{otherwise}, \end{cases}$$

where $Q_{\texttt{jt}}(\boldsymbol{u}) = \mathbb{E}_{w_{\texttt{env}}}[Z_{\texttt{jt}}(\boldsymbol{u}, w_{\texttt{env}})]$ is a expected value of true action-value, $Q_{\texttt{ub}}(\boldsymbol{u}) = \max(Q_{\texttt{jt}}(\boldsymbol{u}), Q_{\texttt{jt}}(\boldsymbol{u}_{\texttt{opt}}))$ approximately maximizes the expected true action-value, and $\boldsymbol{u}_{\texttt{opt}}$ is an "optimal" action maximizing the utility function $z_i(w_{\texttt{agt}})$ for $i \in \mathcal{N}$.

In the subsequent paragraphs, detailed descriptions of each loss are provided. Firstly, for optimal actions, $\mathcal{L}_{\texttt{opt}}$ encourages the transformed action-value estimator $Z_{\texttt{tran}}$ to follow the value of the true action-value estimator $Q_{\texttt{jt}}$. We can adjust the environmental risk sensitivity according to how we calculate the expected value $Q_{\texttt{jt}}$ from the distributional true action-value estimator $Z_{\texttt{jt}}$. If the expectation is calculated by sampling environmental risk level $w_{\texttt{env}}$ from a uniform distribution $\mathcal{U}(0, 1)$, the agents learn a risk-neutral policy for the environmental risk. By changing this sampling distribution, we can learn the optimal policy for the desired environmental risk-sensitivity. For example, if we learn $Z_{\texttt{tran}}$ for $Q_{\texttt{jt}} = \mathbb{E}_{w_{\texttt{env}}}[Z_{\texttt{jt}}(w_{\texttt{env}})]$ calculated by sampling $w_{\texttt{env}}$ from a uniform distribution $\mathcal{U}(0, 0.25)$, agents take environmental risk-averse behaviors.

Secondly, for non-optimal actions, $\mathcal{L}_{\texttt{nopt}}$ aims to make $Z_{\texttt{tran}}$, which has lower representational power, follow the true action-value estimator $Q_{\texttt{jt}}$ efficiently, utilizing cooperative risk level as a weight. If the value of $Q_{\texttt{jt}}$ is greater than the value of the transformed action-value estimator $Z_{\texttt{tran}}$, then this is an optimistic sample where the corresponding action is likely to be the optimal action, so the transformed action-value estimator follows the true action-value estimator exactly. Conversely, for a relatively small true action-value, we softly ignore it and follow the true action-value estimator through a small weight whose value depends on the cooperative risk $w_{\texttt{agt}}$. The comparison of this loss function with existing MARL methods such as OW-QMIX (Rashid et al., 2020a) is described in Appendix C.3.

Finally, we add a loss function $\mathcal{L}_{\texttt{ub}}$ which makes an upper bound condition for $Z_{\texttt{tran}}$ for numerical stability because only the lower bound of $Z_{\texttt{tran}}$ exists in the $\mathcal{L}_{\texttt{nopt}}$ when $w_{\texttt{agt}} = 1$. By combining our three loss functions $\mathcal{L}_{\texttt{opt}}, \mathcal{L}_{\texttt{nopt}}, \mathcal{L}_{\texttt{ub}}$, we obtain the following objective which is minimized in an end-to-end manner to train the true and the transformed action-value estimators:

$$\mathcal{L} = \mathcal{L}_{\texttt{td-IQN}} + \lambda_{\texttt{opt}}\mathcal{L}_{\texttt{opt}} + \lambda_{\texttt{nopt}}\mathcal{L}_{\texttt{nopt}} + \lambda_{\texttt{ub}}\mathcal{L}_{\texttt{ub}}$$

where $\lambda_{\texttt{opt}}, \lambda_{\texttt{nopt}}, \lambda_{\texttt{ub}} > 0$ are hyperparameters controlling the importance of each loss function. The training algorithm of DRIMA is provided in Appendix A.

3.3. Network architectures

Here, we introduce our architectures for the action-value estimators. First, we construct the estimators using the utility functions $z_1(w_{\texttt{agt}}), \ldots, z_N(w_{\texttt{agt}})$ with cooperative risk level $w_{\texttt{agt}}$. Our main contribution in designing the estimators is twofold: (i) using IQN (Dabney et al., 2018a) for the true action-value estimator $Z_{\mathtt{jt}}(w_{\mathtt{env}})$ with environmental risk level $w_{\mathtt{env}}$ and (ii) using a monotonic mixing network for the transformed action-value estimators $Z_{\mathtt{tran}}$ with cooperative risk level $w_{\mathtt{agt}}$.

Agent-wise utility function z_i. In a partially observable setting, agents can learn better policies by using their action-observation history τ_i instead of their current observation. We represent each agent-wise utility function as a DRQN (Hausknecht & Stone, 2015) that receives action-observation history as input and outputs z_i for each action and $w_{\mathtt{agt}}$. The utility functions aim to accurately extract the optimal action from the joint action-value.

True action-value estimator $Z_{\mathtt{jt}}$. The estimator $Z_{\mathtt{jt}}$ aims to express distributions with additional representation power; note that the transformed action-value estimator has limited power due to the decentralization condition (Equation 1) (Rashid et al., 2018). For the true action-value estimator, we employ a feed-forward network that takes the state s and set of utility functions z_i for $i \in \mathcal{N}$. Since we use z averaged over multiple $w_{\mathtt{agt}}$ samples, the feed-forward network is not conditioned on $w_{\mathtt{agt}}$. To apply IQN for the true action-value estimator, we use an additional network ϕ that computes an embedding $\phi(w_{\mathtt{env}})$ for the sample point $w_{\mathtt{env}}$. We calculate the embedding of $w_{\mathtt{env}}$ with cosine basis functions and utilize element-wise (Hadamard) product, as done in the IQN (Dabney et al., 2018a) paper.

Transformed action-value estimator $Z_{\mathtt{tran}}$. The architecture of the transformed action-value estimator is largely the same as the mixing network of Rashid et al. (2018), and it is expressed as follows:

$$Z_{\mathtt{tran}}(s, \boldsymbol{\tau}, \boldsymbol{u}, w_{\mathtt{agt}}) = \\ f_{\mathtt{mix}}\big(z_1(\tau_1, u_1, w_{\mathtt{agt}}), \ldots, z_N(\tau_N, u_N, w_{\mathtt{agt}}); \theta_{\mathtt{tran}}(s)\big),$$

where $\theta_{\mathtt{tran}}(s, w_{\mathtt{agt}})$ is a non-negative parameter obtained from state-dependent hypernetwork (Ha et al., 2017). A more detailed discussion is available in Appendix D.

4. Experiments

We design our experiments to answer the following questions:

- **Q1:** Are there MARL scenarios that existing distributional MARL methods cannot solve? Can DRIMA solve them through disentangling sources of risk? (see Figure 3)
- **Q2:** Can DRIMA achieve robust performance through risk sensitivity control in the presence of noisy (random or adversarial) agents? (see Figure 5)
- **Q3:** Can DRIMA improve sample efficiency and final performance over baseline methods, even under traditional MARL scenarios? (see Figure 6)
- **Q4:** How does risk sensitivity affect the performance in multi-agent distributional RL methods? (see Figure 7-9)

4.1. Experimental setup

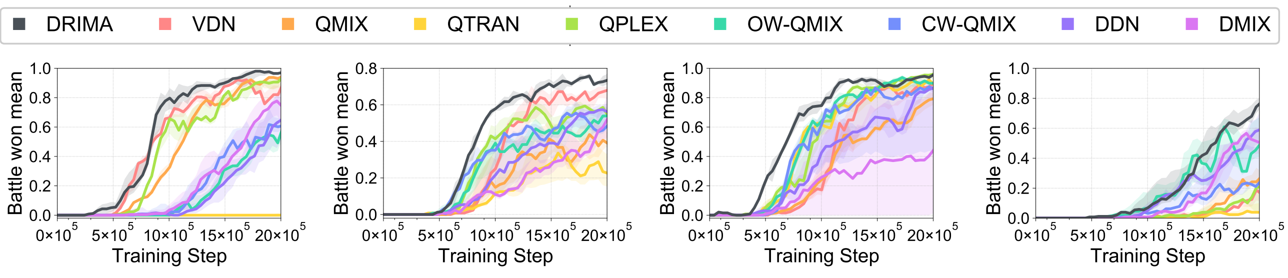
Environments. We mainly evaluate our method on the Starcraft Multi-Agent Challenge (SMAC) environment (Samvelyan et al., 2019). In this environment, each agent is a unit participating in combat against enemy units controlled by handcrafted policies. The agent is required to collaborate with ally units Each unit is required to collaborate with ally units (without communication) to achieve high win rate against another team of enemy units following handcrafted policy. In the environment, agents receive individual local observation containing distance, relative location, health, shield, and unit type of other allied and enemy units within their sight range.

Scenarios. To verify the effectiveness of disentangling risks, we consider the following scenarios:

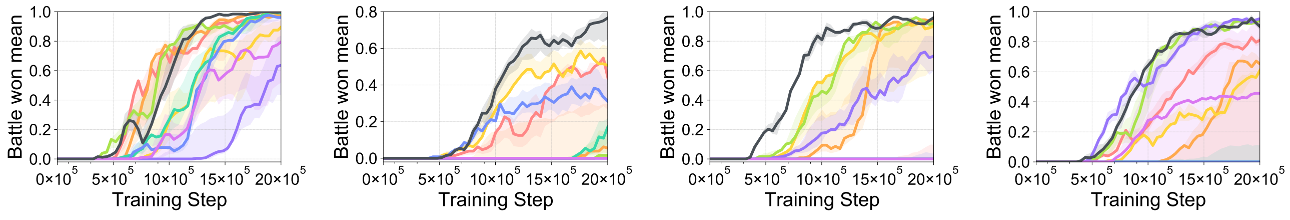
- **Explorative** (for **Q1):** Agents behave heavily exploratory in the training phase, so it is difficult to learn cooperation unless they consider the cooperative risk.
- **Dilemmatic** (for **Q1**): A social dilemma exists in which agents can learn local optimum policies, and agents should consider cooperative risk to learn optimal policies.
- **Noisy** (for **Q2**): During the test phase, some agents may behave incorrectly. Hence, learning robust policies with risk sensitivity control against the failure of other agents is required.
- **Basic** (for **Q3** and **Q4**): This is a traditional MARL environment which was used in prior works (Rashid et al., 2018; Samvelyan et al., 2019; Wang et al., 2020b), i.e., neither social dilemma nor noisy agent.

In particular, to design explorative and dilemmatic scenarios, we consider modified exploration schedules and reward functions, given the basic scenario. More details of the above setups are given in Appendix D.

In explorative and dilemmatic scenarios, it is likely to obtain more non-cooperative behaviors from teammates. This represents the importance of considering cooperative risk explicitly. As shown in Section 3.1, the choice of training data distribution affects the action-value estimator in value factorization. Rashid et al. (2020a) also demonstrates the importance of change in exploration scheduling in the

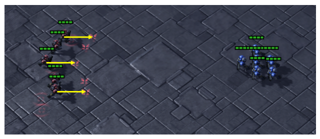


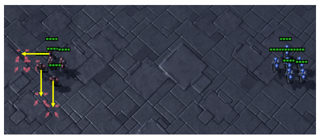
(a) 3s_vs_5z (explorative) (b) 5m_vs_6m (explorative) (c) 8m_vs_9m (explorative) (d) MMM2 (explorative)

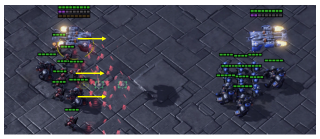


(e) 3s_vs_5z (dilemmatic) (f) 5m_vs_6m (dilemmatic) (g) 8m_vs_9m (dilemmatic) (h) MMM2 (dilemmatic)

Figure 3: Median test win rate in **explorative** and **dilemmatic** scenarios with 25%-75% percentile over four random seeds, comparing DRIMA with eight baselines.


(a) 5m_vs_6m (DRIMA)


(b) 5m_vs_6m (DMIX)


(c) MMM2 (DRIMA)

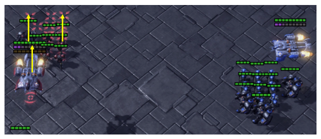

(d) MMM2 (DMIX)

Figure 4: Illustration of the policies learned using DRIMA and DMIX in dilemmatic scenarios. Agents trained by DMIX run away from enemies without fighting (d, b). In contrast, DRIMA trains the agents to fight the enemies, obtaining a higher win rate in overall (c, a).

valued-based CTDE methods. In dilemmatic scenarios, if two sources of uncertainty are not disentangled well, units are induced to have a "selfish" behavior (e.g., running away) in order to avoid being damaged.

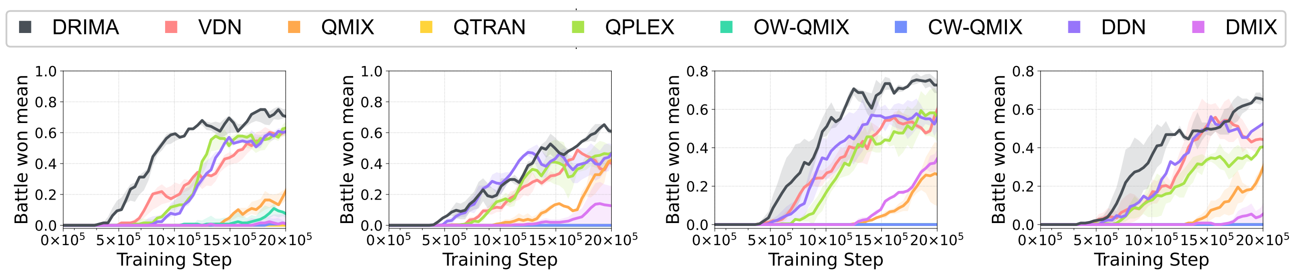
In noisy scenarios, we deal with the unwanted behavior of agents that can occur in real-world applications. Early formulations of MARL methods focused on decentralized decision-making. However, if agents are to interact with each other in the real world, they are necessary to mitigate some potential dangers. We use two different noisy agents in the noisy scenarios. The first is a *random* agent that moves randomly, and the second is an *adversarial* agent that selects the malicious action that minimizes the action-value.

Evaluation. We run 32 test episodes without exploration for every $4 * 10^4$-th time step. The percentage of episodes where the agents defeat all enemy units, i.e., *test win rate*, is reported as the performance metric of the algorithms. We report the median performance with shaded 25-75% confidence intervals with four random seeds. For visual clarity, we smooth all the curves by moving average filter with a window size of 4.

4.2. Main results

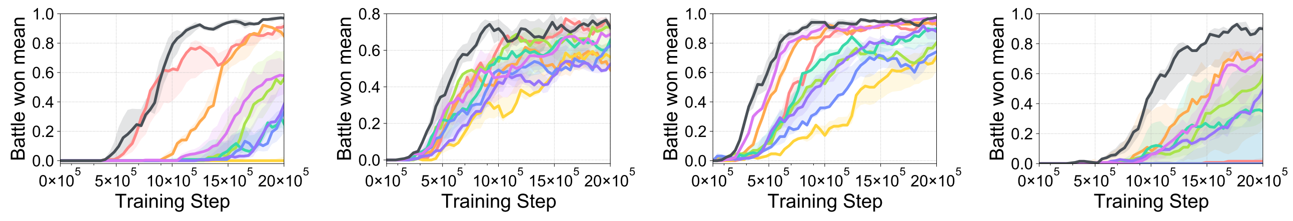
We evaluate DRIMA compared to the eight baselines, including non-distributional – VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), QTRAN (Son et al., 2019), QPLEX (Wang et al., 2020b), OW-QMIX (Rashid et al., 2020a) and CW-QMIX (Rashid et al., 2020a), and distributional value-based CTDE methods – DDN (Sun et al., 2021) and DMIX (Sun et al., 2021).

Comparisons in explorative and dilemmatic scenarios. Notably, as shown in Figure 3, DRIMA obtains significant gains in explorative and dilemmatic scenarios, where separating cooperative risk and environmental risk is critical; it is likely to reach a suboptimal equilibrium (i.e., selecting actions of running away) if risk sources are not disentangled in such environments. This result is significant since there is no baseline that consistently achieves the second-best result. For relatively easy tasks such as 5m_vs_6m, 3s_vs_5z, and 8m_vs_9m, we set (cooperative risk-sensitivity, environmental risk-sensitivity) to (seeking, neutral). For a super hard task like MMM2, we set to (seeking, seeking). The rationale of this setting is that for easy tasks, considering en-



(a) 3s_vs_5z (random) (b) 3s_vs_5z (adversarial) (c) 5m_vs_6m (random) (d) 5m_vs_6m (adversarial)

Figure 5: Median test win rate in **noisy** scenarios with 25%-75% percentile over four random seeds, comparing DRIMA with eight baselines.



(a) 3s_vs_5z (basic) (b) 5m_vs_6m (basic) (c) 8m_vs_9m (basic) (d) MMM2 (basic)

Figure 6: Median test win rate in **basic** scenarios with 25%-75% percentile over four random seeds, comparing DRIMA with eight baselines.

vironmental risks does not crucially affect the search for the optimal strategy, but for hard tasks which require extensive exploration, it is much beneficial to consider environmental risks; environmental risk-averse behavior is likely to avoid dangerous actions, which makes the agents less explorative.

We also illustrate in Figure 4 how DRIMA solves the most difficult scenarios. As demonstrated in Figures 4b and 4d, existing algorithms such as DMIX may end up learning a "locally optimal" policy where the agents run away from the enemy units to avoid receiving negative rewards. However, in Figure 4a and 4c, we observe that DRIMA successfully escapes this local optimum and chooses to fight the enemies to achieve a higher win rate.

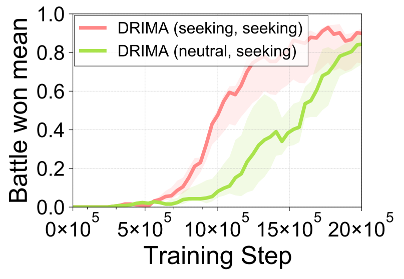
Comparisons in noisy scenarios. Figure 5 demonstrates DRIMA achieves robust and high performance through risk sensitivity control even in the presence of noisy agents. In these scenarios, we set (cooperative risk-sensitivity, environmental risk-sensitivity) to (averse, neutral). Cooperative risk-averse agents learn a policy that considers other non-cooperative agents. Since the agents can behave incorrectly in the test phase, all methods show relatively low performance compared to basic scenarios in Figure 6. In particular, as shown in Figures 5a and 5b, some algorithms like QMIX and DMIX achieve very low performance compared to basic scenarios in Figure 6a.

Comparisons in basic scenarios. In Figure 6, one can find that DRIMA generally achieves the state-of-the-art performance both sample-efficiently and asymptotically, while the second-best method varies for each scenario. Moreover, DRIMA shows consistent superiority regardless of difficulty and desirable tactics. We use the same risk-sensitivity as in explorative and dilemmatic scenarios. Intriguingly, in relatively easier tasks (i.e., 5m_vs_6m), simple methods such as VDN show better results than more complex methods (i.e., DDN and DMIX[2]). We understand that a simple structure is more efficient for learning plausible strategies in such easy tasks. However, the reason why DRIMA is able to learn easy tasks despite its sophisticated structure comes from that disentanglement of risk sources which makes useful learning signals for conquering easy tasks efficiently. We provide additional experiments on more maps in Appendix F.

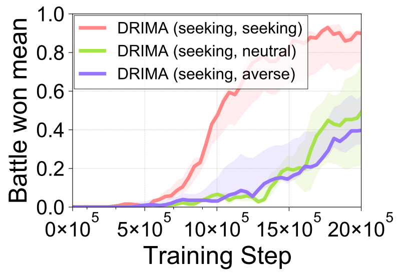
4.3. Ablation studies

DRIMA with varying risk-levels. To investigate the effect of different risk levels in DRIMA, we conduct experiments that adjust cooperative and environmental risk levels. As shown in Figure 7, we find that leveraging risk levels with disentanglement indeed affects the performance in a hard task (i.e., MMM2). Interestingly, one can note that cooperative risk-seeking and environmental risk-seeking shows the strongest performance. We believe that it is critical

[2]In the original paper of DDN (Sun et al., 2021) and DMIX (Sun et al., 2021), the results of 5m_vs_6m, 3s_vs_5z, and 8m_vs_9m were not reported.



(a) cooperative risk



(b) environmental risk

Figure 7: Ablation studies on varying risk-levels (cooperative risk $w_{\mathtt{agt}}$, environmental risk $w_{\mathtt{env}}$) of DRIMA in MMM2-v1, one of *SUPER HARD* tasks.

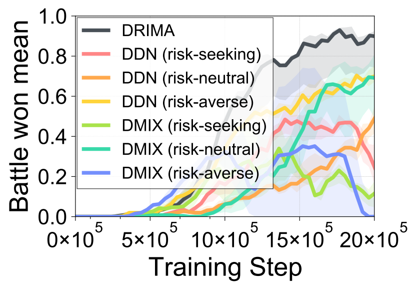
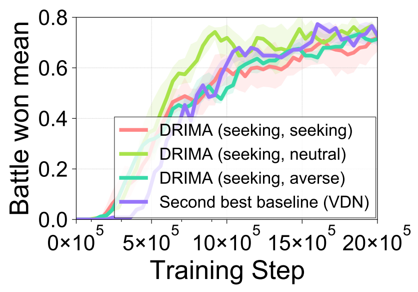
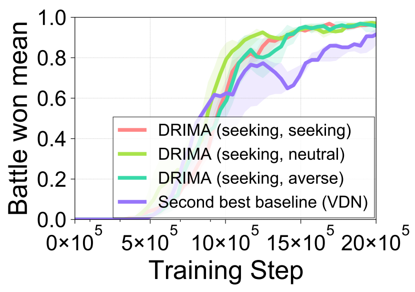


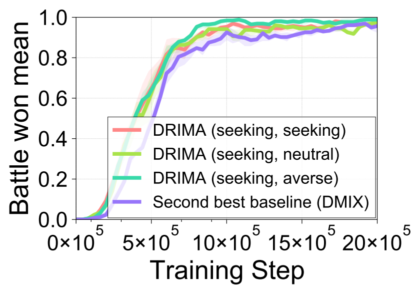
Figure 8: Comparison of DRIMA to risk-sensitive variants of DDN and DMIX in MMM2-v1.



(a) 5m_vs_6m-v1



(b) 3s_vs_5z-v1



(c) 8m_vs_9m-v1

Figure 9: Robustness of DRIMA in relatively easy tasks. Differing risk-levels (cooperative risk level $w_{\mathtt{agt}}$, environmental risk level $w_{\mathtt{env}}$) in DRIMA do not degrade below the second-best baseline.

for teammates to act cooperatively (i.e., cooperative risk-seeking) and enhance exploration through an environmental risk-seeking objective in order to find the optimal tactic in such a hard task.

Comparisons to risk-sensitive variants of DDN and DMIX. To further understand the effectiveness of disentangling risk sources, we compare DRIMA and risk-sensitive variants of DDN and DMIX in Figure 8. Although DDN and DMIX propose only risk-neutral policies in their original paper, we additionally implement their risk-sensitive variants by adjusting the quantile-sampling range, i.e., sampling quantiles in $[0, 0.25]$ for risk-averse policy and $[0.75, 1]$ for risk-seeking policy. As shown in Figure 8, DRIMA also achieves superior performance over risk-sensitive variants of DDN and DMIX. The gain mainly comes from the ability of DRIMA to explicitly separate the two risk sources.

Robustness of DRIMA. In Figure 9, in order to show the robustness of DRIMA, we report the performance over the second-best baseline across relatively easy tasks: 5m_vs_6m-v1, 3s_vs_5z-v1, and 8m_vs_9m-v1. We note that changing environmental-wise risks still result to DRIMA performing the best. It shows not only the empirical strength of DRIMA but also one interesting finding; environmental-risk does not matter in finding a desirable tactic in such easy tasks. However, note that in 8m_vs_9m-v1, only environmental risk-averse agents reached a 100% win rate, although other agents also achieve high win rates. This means that a safe policy can be beneficial for easy tasks.

5. Conclusion

In this paper, we present DRIMA, a new distributional MARL framework that disentangles two sources of risk in order to obtain optimal policies efficiently. Our main idea is to use disentangled sources of risk along with the proposed hierarchical quantile structure and quantile regression. Through extensive experiments, we show that DRIMA achieves state-of-the-art performance thanks to the disentanglement of risk sources. We believe that DRIMA is an important step toward training robust MARL agents in the real-world scenarios.

Acknowledgements

This research was supported by the ICT Research and Development Program of MSIP/IITP under projects [2022-0-00531, Development of in-network computing techniques for efficient execution of AI applications], the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921)

References

Baker, B. Emergent reciprocity and team formation from randomized uncertain social preferences. *arXiv preprint arXiv:2011.05373*, 2020.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *Proceedings of ICML*, pp. 449–458. PMLR, 2017.

Chow, Y. and Ghavamzadeh, M. Algorithms for cvar optimization in mdps. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pp. 3509–3517, 2014.

Da Silva, F. L., Costa, A. H. R., and Stone, P. Distributional reinforcement learning applied to robot soccer simulation. In *Adapative and Learning Agents Workshop, AAMAS*, 2019.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of ICML*, pp. 1096–1105. PMLR, 2018a.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018b.

Du, Y., Han, L., Fang, M., Liu, J., Dai, T., and Tao, D. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. In *Proceedings of NeurIPS*, 2019.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of AAAI*, 2018.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. In *Proceedings of ICLR*, 2017.

Hausknecht, M. and Stone, P. Deep recurrent q-learning for partially observable mdps. In *Proceedings of AAAI Fall Symposium Series*, 2015.

Huber, P. J. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, pp. 73–101, 1964.

Iqbal, S. and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In *Proceedings of ICML*, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arxiv preprint arXiv:1509.02971*, 2015.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Pieter, A., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of NeurIPS*, pp. 6379–6390, 2017.

Lyu, X. and Amato, C. Likelihood quantile networks for coordinating multi-agent reinforcement learning. *arXiv preprint arXiv:1812.06319*, 2018.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. In *Proceedings of NeurIPS*, 2019.

Matignon, L., Laurent, G. J., and Le Fort-Piat, N. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 64–69. IEEE, 2007.

Oliehoek, F. A., Amato, C., et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *arxiv preprint arXiv:1703.06182*, 2017.

Panait, L., Sullivan, K., and Luke, S. Lenient learners in cooperative multiagent systems. In *Proceedings of the 5th AAMAS*, pp. 801–803, 2006.

Panait, L., Tuyls, K., and Luke, S. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *The Journal of Machine Learning Research*, 9: 423–457, 2008.

Qiu, W., Wang, X., Yu, R., He, X., Wang, R., An, B., Obraztsova, S., and Rabinovich, Z. Rmix: Learning risk-sensitive policies for cooperative reinforcement learning agents. *arXiv preprint arXiv:2102.08159*, 2021.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of ICML*, pp. 4295–4304. PMLR, 2018.

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation. *arXiv preprint arXiv:2006.10800*, 2020a.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2003.08839*, 2020b.

Rowland, M., Omidshafiei, S., Hennes, D., Dabney, W., Jaegle, A., Muller, P., Pérolat, J., and Tuyls, K. Temporal difference and return optimism in cooperative multi-agent reinforcement learning. In *AAMAS ALA Workshop*, 2021.

Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. In *Proceedings of AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of ICML*, 2019.

Sun, W.-F., Lee, C.-K., and Lee, C.-Y. Dfac framework: Factorizing the value function via quantile mixture for multi-agent distributional q-learning. In *Proceedings of ICML*, 2021.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

Wang, J., Ren, Z., Han, B., and Zhang, C. Towards understanding linear value decomposition in cooperative multi-agent q-learning. *arXiv preprint arXiv:2006.00587*, 2020a.

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

Wang, T., Dong, H., Lesser, V., and Zhang, C. Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020c.

Wang, Y., Han, B., Wang, T., Dong, H., and Zhang, C. {DOP}: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*, 2021.

Wei, E. and Luke, S. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research*, 17(1):2914–2955, 2016.

Yang, Y., Hao, J., Liao, B., Shao, K., Chen, G., Liu, W., and Tang, H. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.

A. DRIMA training algorithm

The training algorithm for DRIMA is provided in Algorithm 1

Algorithm 1 DRIMA algorithm

```
1:  Initialize replay memory $\mathcal{B} \leftarrow \emptyset$ and target parameters $\theta^- \leftarrow \theta$
2:  for episode = 1 to $M$ do
3:      Observe initial state $\boldsymbol{s}^0$ and observation $\boldsymbol{o}^0 = [O(\boldsymbol{s}^0, i)]_{i=1}^N$ for each agent $i$
4:      for $t = 1$ to $T$ do
5:          With probability $\epsilon$, each agent $i$ select an random action $u_i^t$
6:          Otherwise, set $u_i^t = \arg\max_{u_i^t} z_i(\tau_i^t, u_i^t, w_{\mathtt{agt}})$ for each agent $i$ based on our cooperative risk-sensitivity $w_{\mathtt{agt}}$.
7:          Take action $\boldsymbol{u}^t$, and retrieve next state, observation and reward $(s^{t+1}, \boldsymbol{o}^{t+1}, r^t)$
8:          Store transition $(s^t, \boldsymbol{\tau}^{t+1}, \boldsymbol{u}^t, r^t, s^{t+1}, \boldsymbol{\tau}^{t+1})$ in $\mathcal{B}$
9:          Sample a transition $(s, \boldsymbol{\tau}, \boldsymbol{u}, r, s', \boldsymbol{\tau}')$ from $\mathcal{B}$
10:         Sample environmental risk levels $w_{\mathtt{env}}, w'_{\mathtt{env}}$ uniformly from $\mathcal{U}(0, 1)$
11:         Compute loss $\mathcal{L}_{\mathtt{td-IQN}}$ for true action-value estimator from Section 3.2
12:         Sample environmental risk levels $w_{\mathtt{env}}$ based on our environmental risk-sensitivity
13:         Sample cooperative risk levels $w_{\mathtt{agt}}$ uniformly
14:         Compute loss $\mathcal{L}_{\mathtt{opt}}, \mathcal{L}_{\mathtt{nopt}}$, and $\mathcal{L}_{\mathtt{ub}}$ for transformed action-value estimator from Section 3.2
15:         Update $\theta$ by minimizing the losses
16:         Update target network parameters $\theta^- \leftarrow \theta$ with period $I$
17:     end for
18: end for
```

B. Detailed architectural illustration

In this section, we provide detailed illustration for our architectural components: (i) agent-wise utility function, (ii) true action-value estimator, and (iii) transformed action-value estimator.

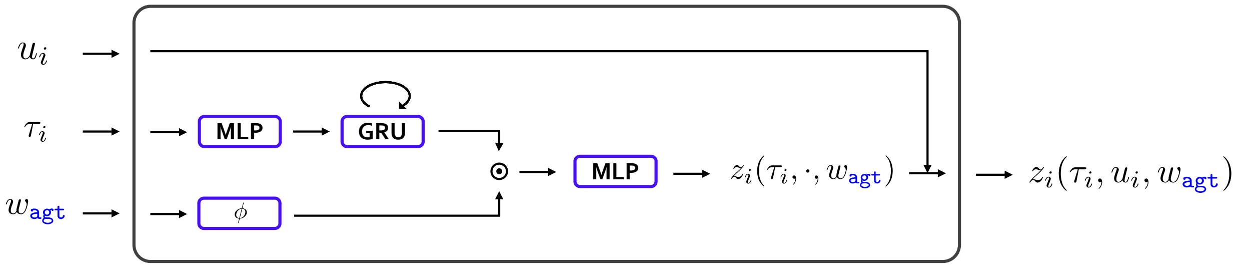

u_i
τ_i
$w_{\texttt{agt}}$
MLP
GRU
ϕ
MLP
$z_i(\tau_i, \cdot, w_{\texttt{agt}})$
$z_i(\tau_i, u_i, w_{\texttt{agt}})$

Figure 10: Agent-wise utility function. $\phi : [0, 1] \to \mathbb{R}^d$ denotes a quantile embedding function (Dabney et al., 2018a)

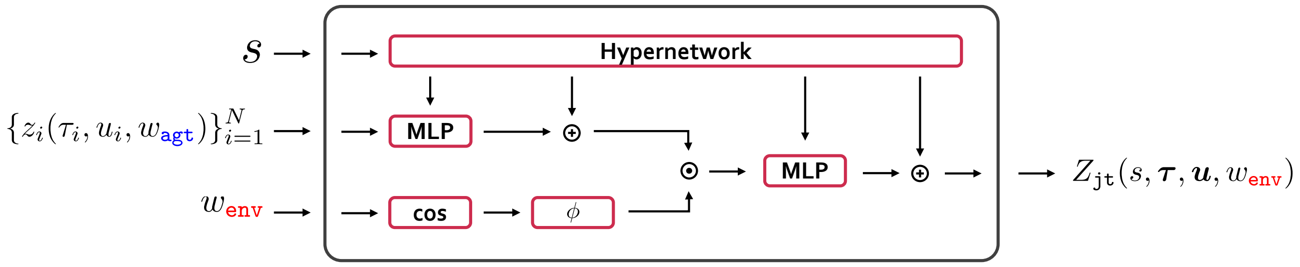

s
$\{z_i(\tau_i, u_i, w_{\texttt{agt}})\}_{i=1}^N$
$w_{\texttt{env}}$
Hypernetwork
MLP
cos
ϕ
MLP
$Z_{\texttt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}, w_{\texttt{env}})$

Figure 11: True action-value estimator. $\phi : [0, 1] \to \mathbb{R}^d$ denotes a quantile embedding function (Dabney et al., 2018a).

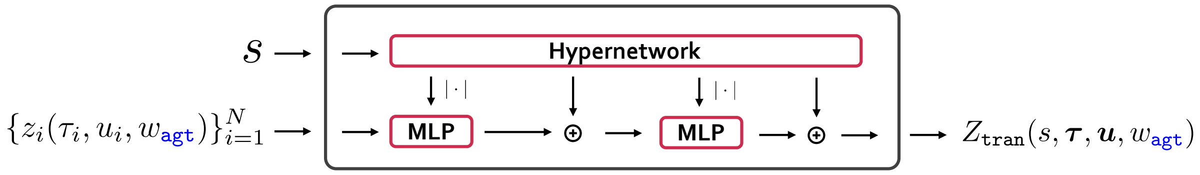

s
$\{z_i(\tau_i, u_i, w_{\texttt{agt}})\}_{i=1}^N$
Hypernetwork
$|\cdot|$
$|\cdot|$
MLP
MLP
$Z_{\texttt{tran}}(s, \boldsymbol{\tau}, \boldsymbol{u}, w_{\texttt{agt}})$

Figure 12: Transformed action-value estimator. $|\cdot|$ is employed to enforce monotonicity constraint (Rashid et al., 2018).

C. Related work

C.1. Centralized training with decentralized execution

Centralized training with decentralized execution (CTDE) has emerged as a popular paradigm under the multi-agent reinforcement learning framework. It assumes the complete state information to be fully accessible during training, while individual policies allow decentralization during execution. To train agents under the CTDE paradigm, both policy-based (Foerster et al., 2018; Lowe et al., 2017; Du et al., 2019; Iqbal & Sha, 2019; Wang et al., 2020c; 2021) and value-based methods (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019; Yang et al., 2020; Sun et al., 2021) have been proposed. At a high level, the policy-based methods rely on the actor-critic framework with independent actors to achieve decentralized execution. On the other hand, the value-based methods attempt to learn a joint action-value estimator, which can be cleverly decomposed into individual agent-wise utility functions.

Among the policy-based methods, COMA (Foerster et al., 2018) trains individual policies with a joint critic and solves the credit assignment problem by estimating a counterfactual baseline. MADDPG (Lowe et al., 2017) extends the DDPG (Lillicrap et al., 2015) algorithm to learn individual policies in a centralized manner on both cooperative and competitive games. MAAC (Iqbal & Sha, 2019) includes an attention mechanism in critics to improve scalability. LIIR (Du et al., 2019) introduces a meta-gradient algorithm to learn individual intrinsic rewards to solve the credit assignment problem. ROMA (Wang et al., 2020c) proposes a role-oriented framework to learn roles via deep RL with regularizers and role-conditioned policies. Finally, DOP (Wang et al., 2021) proposes factorized critic for multi-agent policy gradients.

Among the value-based methods, value-decomposition networks (VDN, Sunehag et al. 2018) learns a centralized yet factored joint action-value estimator by representing the joint action-value estimator as a sum of individual agent-wise utility functions. QMIX (Rashid et al., 2018) extends VDN by employing a *mixing network* to express a non-linear monotonic relationship among individual agent-wise utility functions in the joint action-value estimator. Qatten (Yang et al., 2020) introduces a multi-head attention mechanism for approximating the decomposition of the joint action-value estimator, which is based on theoretical findings. However, our method satisfies both theoretical guarantees and practical performance.

QTRAN (Son et al., 2019) has been proposed to eliminate the monotonic assumption on the joint action-value estimator in QMIX (Rashid et al., 2018). Instead of directly decomposing the joint action-value estimator into utility functions, QTRAN proposes a training objective that enforces the decentralization of the joint action-value estimator into the summation of individual utility functions. However, recent studies have found that despite its promise, QTRAN performs empirically worse than QMIX in complex MARL environments (Samvelyan et al., 2019; Rashid et al., 2020b).

One can find connections between DRIMA and QTRAN (Son et al., 2019), but DRIMA has a larger capacity to represent risk-sensitive policies. As for QTRAN, let's say $w_{\texttt{agt}} = 1$ in DRIMA, then our loss functions $\mathcal{L}_{\texttt{opt}}$ and $\mathcal{L}_{\texttt{nopt}}$ become similar to QTRAN. However, DRIMA is capable of representing a wider range of risk-sensitive policies. For example, if the cooperative risk level $w_{\texttt{agt}}$ is 0.5, both $\mathcal{L}_{\texttt{nopt}}$ and $\mathcal{L}_{\texttt{opt}}$ follow the true action-value without weighting, which is cooperative risk-neutral. This cannot be represented by QTRAN.

Recently, several other methods have been proposed to solve the limitations of QMIX. QPLEX (Wang et al., 2020b) takes a duplex dueling network architecture to factorize the joint value function. Unlike QTRAN, QPLEX learns using only one joint action-value network and a single loss function. Since they learn only a single estimator, it is impossible to learn cooperative randomness and environmental randomness separately by applying distributional reinforcement learning as in our method. Also, Rashid et al. (2020a) proposed CW-QMIX and OW-QMIX, which use a weighted projection that allows more emphasis to be placed on better joint actions. Their methods apply weights to loss according to the sign of the TD-error for each sample, and it is theoretically guaranteed that the optimal policy can be learned for decentralizable tasks. However, whereas DRIMA learns the randomness of the environment using the weights from the TD-error, CW-QMIX and OW-QMIX only aim to consider the randomness of chosen actions. For this reason, they do not consider the distribution of the TD-error caused by the transition probability or the randomness of the reward in training, and it is not guaranteed that they can learn the optimal policy in stochastic environments. On the other hand, the true action-value estimator of DRIMA learns only the randomness of the environment, and the transformed action-value estimator learns through the estimated true action-value, not the TD-target. Therefore, DRIMA completely separates the randomness of the environment and the randomness of the action in training.

C.2. Distributional reinforcement learning

Instead of training a scalar state-action estimator, distributional reinforcement learning focuses on representing a distribution over returns. Empirically, distributional reinforcement learning improves sample efficiency and performance.

A number of distributional reinforcement learning methods are proposed in a single-agent domain. Bellemare et al. (2017) proposed C51, which parameterizes the return distribution as a categorical distribution over a fixed set of equidistant points. In (Dabney et al., 2018b), QR-DQN is proposed to estimate the distributions with a uniform mixture of N diracs and quantile regression. By estimating the quantile function with quantile regression, which has been shown to converge to the true quantile function, QR-DQN minimizes the Wasserstein distance to the distributional target. This estimation has been shown to converge to the true quantile function.

Implicit quantile Networks (IQN) (Dabney et al., 2018b) provides a way to learn an implicit representation of distributions by reparameterizing samples from base samples, typically $\omega \sim \mathcal{U}(0, 1)$. It learns a quantile function that maps from embeddings of sample probabilities to the corresponding quantiles, called implicit quantile networks. The quantiles are trained by Huber quantile regression loss (Huber, 1964).

Recently, distributional RL has also been applied to CTDE regime (Qiu et al., 2021; Sun et al., 2021), but they lack disentanglement of risk sources. RMIX (Qiu et al., 2021) and DFAC (Sun et al., 2021) showed promising results by extending agent-wise utility functions from deterministic variables to random variables. RMIX demonstrates the effectiveness of risk-sensitive MARL via distributional RL, but it is limited since they cannot represent policies, which have different risk levels relying on sources (i.e., cooperative risk-seeking yet environmental risk-averse policies). DFAC is another distributional MARL framework with proposed mean-shape decomposition while employing IQN network for agent-wise utility function, but they only showcase risk-neutral policies. Da Silva et al. (2019) and Lyu & Amato (2018) have applied distributional learning in fully decentralized cases, not CTDE methods. Finally, Rowland et al. (2021) and Baker (2020) propose a method to learn cooperation using optimism in a fully distributed setting instead of CTDE.

C.3. A further comparison between DRIMA and OW-QMIX in risk-sensitivity

In this section, we provide further backgrounds to understand why DRIMA is capable of disentangling cooperative and environmental risk while OW-QMIX (Rashid et al., 2020a) cannot.

Before that, note that the risk-sensitivity is not mentioned explicitly in the original OW-QMIX paper, but we find that OW-QMIX can be viewed as an optimistic (cooperative risk-seeking) algorithm due to its tight connection to hysterical Q-learning (Matignon et al., 2007; Omidshafiei et al., 2017), which study cooperative optimism in the fully distributed MARL. In detail, one can find the connection from the loss function of hysterical Q-learning as follows:

$$\begin{aligned} \mathcal{L}_{\texttt{hys-Q},i}(\tau,u) &= w_i(\tau,u)(q_i(\tau_i,u_i)-y_i)^2, \quad i\in\mathcal{N} \\ y_i &= r+\gamma q_i^{\texttt{target}}(\tau_i{}', \arg\max_{u_i'} q_i(\tau_i',u_i')), \\ w(s,\boldsymbol{u}) &= \begin{cases} \alpha_{\texttt{hys-Q}}, & \text{if } q_i(\tau_i,u_i)\le y_i, \\ \beta_{\texttt{hys-Q}}, & \text{otherwise,} \end{cases}, \end{aligned} \tag{4}$$

where the parameters $\alpha_{\texttt{hys-Q}} > \beta_{\texttt{hys-Q}} > 0$ are typically viewed as learning rate parameters, but here we equivalently view them as part of the loss. $q_i^{\texttt{target}}$ is the fixed target network whose parameters are updated periodically from q_i. Unlike single-agent RL in a stationary environment, learned policies change through adjustment of the weights (cooperative risk level) in MARL (Panait et al., 2006; 2008).

As with fully distributed settings, the idea of optimism is also applicable to value factorization methods such as QMIX (Rashid et al., 2018). QMIX is unable to represent joint action-value functions that are characterized as non-monotonic (Mahajan et al., 2019; Son et al., 2019), i.e., an agent's action-value ordering over its own actions depends on other agents' actions. To solve this problem, Rashid et al. (2020a) proposes Optimistically-Weighted QMIX (OW-QMIX) which assigns a higher weighting to those joint actions that are underestimated, and hence could be the true optimal actions in an optimistic (cooperative risk-seeking) outlook, as follows:

$$\begin{aligned} \mathcal{L}_{\texttt{OW-QMIX}}(s,\boldsymbol{\tau},\boldsymbol{u}) &= w(s,\boldsymbol{u})(Q_{\texttt{tran}}(s,\boldsymbol{\tau},\boldsymbol{u})-y)^2, \\ y &= r+\gamma Q_{\texttt{jt}}^{\texttt{target}}(s',\boldsymbol{\tau}',\boldsymbol{u}'_{\texttt{opt}}), \\ w(s,\boldsymbol{u}) &= \begin{cases} 1, & \text{if } Q_{\texttt{tran}}(s,\boldsymbol{\tau},\boldsymbol{u})\le y, \\ \alpha, & \text{otherwise,} \end{cases}, \end{aligned} \tag{5}$$

where $\alpha < 1$ is a hyperparameter and $Q_{\texttt{jt}}^{\texttt{tar}}$ is the fixed target network whose parameters are updated periodically from the original unconstrained true action-value estimator $Q_{\texttt{jt}}$. They sample (s,u,r,s'), a tuple of experience transitions from a replay buffer and $\boldsymbol{u}'_{\texttt{opt}}$ is the "optimal" action maximizing the utility functions $q_i(\tau_i',u_i')$ for $i\in\mathcal{N}$. One can find the loss function of hysterical Q-learning (Equation 4) and OW-QMIX (Equation 5) are similar, so that OW-QMIX can be understood as cooperatively optimistic algorithm like hysterical Q-learning.

To investigate similarities and differences between DRIMA and OW-QMIX to handle risk-sensitivity, we employ an one-step single-state matrix game. In this game, there is only one state, and all episodes finish after a single time-step. We find a connection between the DRIMA and OW-QMIX via following proposition:

Proposition C.1. *Consider a deterministic-reward environment with a single state that terminated immediately. Then the loss $L_{\texttt{nopt}}$ of DRIMA with parameters $w_{\texttt{agt}}$ is equivalent to the Weighted QMIX with $\alpha = 2*(1-w_{\texttt{agt}})$.*

Proof. In the simple one-step matrix game, the loss for Weighted QMIX reduces to

$$\begin{aligned} \mathcal{L}_{\texttt{OW-QMIX}}(s,\boldsymbol{\tau},\boldsymbol{u}) &= w(s,\boldsymbol{u})(Q_{\texttt{tran}}(s,\boldsymbol{\tau},\boldsymbol{u})-r)^2, \\ w(s,\boldsymbol{u}) &= \begin{cases} 1, & \text{if } Q_{\texttt{tran}}(s,\boldsymbol{\tau},\boldsymbol{u})\le r, \\ \alpha, & \text{otherwise,} \end{cases}, \end{aligned} \tag{6}$$

since there is no next state to bootstrap from. In DRIMA, $Z_{\texttt{jt}}$ estimates a deterministic value independent of $w_{\texttt{env}}$ for the deterministic-reward environment. Then the loss $L_{\texttt{nopt}}$ of DRIMA reduces to

$$\begin{aligned}
\mathcal{L}_{\texttt{nopt}}(s,\boldsymbol{\tau},\boldsymbol{u}) &= w(s,\boldsymbol{u})(Z_{\texttt{tran}}(s,\boldsymbol{\tau},\boldsymbol{u},w_{\texttt{agt}}) - Q_{\texttt{jt}}^{\texttt{DRIMA}}(s,\boldsymbol{\tau},\boldsymbol{u}))^2,\\
Q_{\texttt{jt}}^{\texttt{DRIMA}}(s,\boldsymbol{\tau},\boldsymbol{u}) &= \mathbb{E}_{w_{\texttt{env}}}[Z_{\texttt{jt}}(s,\boldsymbol{\tau},\boldsymbol{u},w_{\texttt{env}})] \approx \mathbb{E}_{w_{\texttt{env}}}[r] = r,\\
w(s,\boldsymbol{u}) &= \begin{cases} 1, & \text{if } Z_{\texttt{tran}}(s,\boldsymbol{\tau},\boldsymbol{u},w_{\texttt{agt}}) \leq Q_{\texttt{jt}}^{\texttt{DRIMA}}(s,\boldsymbol{\tau},\boldsymbol{u}),\\ 2*(1-w_{\texttt{agt}}), & \text{otherwise.}\end{cases},
\end{aligned} \tag{7}$$

Now observe that if $\alpha = 2*(1-w_{\texttt{agt}})$ and $Z_{\texttt{jt}}$ is sufficiently trained, then two equations are equal. □

This proposition says that DRIMA and OW-QMIX become equivalent in an deterministic-reward environment. However, if the environment is highly stochastic, then optimistic algorithms can induce misplaced optimism towards uncontrollable environment dynamics, leading to sub-optimal behavior. The optimistic MARL approaches ignore low returns, which are assumed to be caused by teammates' exploratory actions. This causes severe overestimation of action-values in stochastic domains (Wei & Luke, 2016; Rowland et al., 2021). Since reward and next state in the TD-target y in Weighted QMIX include randomness of the environment, cooperative risk cannot be extracted separately enough from the sign of the TD-error. Therefore, in DRIMA, we do not use the target y as done in Weighted QMIX. Instead, we use $\mathbb{E}_{w_{\texttt{env}}}[Z_{\texttt{jt}}(w_{\texttt{env}})]$ to exclude the randomness of the environment in the target. The true action-value estimator $Z_{\texttt{jt}}$ of DRIMA does not have any structural restrictions such as monotonicity in value factorization methods. Therefore, $Z_{\texttt{jt}}$ can accurately represent only the randomness of the environment.

D. Experimental details

We mainly evaluate our method on the Starcraft Multi-Agent Challenge (SMAC) environment (Samvelyan et al., 2019). In this environment, each agent is a unit participating in combat against enemy units controlled by handcrafted policies. The agent is required to collaborate with ally units Each unit is required to collaborate with ally units (without communication) to achieve high win rate against another team of enemy units following handcrafted policy. In the environment, agents receive individual local observation containing distance, relative location, health, shield, and unit type of other allied and enemy units within their sight range. The SMAC environment additionally assumes that a global state is available during the training of the agents. To be specific, the global state contains information on all agents participating in the scenario.

Evaluation. We run 32 test episodes without exploration for every $4 * 10^4$-th time step. The percentage of episodes where the agents defeat all enemy units, i.e., *test win rate*, is reported as the performance metric of the algorithms. We report the median performance with shaded 25-75% confidence intervals with four random seeds. For visual clarity, we smooth all the curves by moving average filter with a window size of 4.

Scenarios. To further verify the effectiveness of disentangling risks, we additionally consider harder scenarios with modified exploration schedules and reward functions. We refer to these four different types of scenarios as explorative (increased exploration), dilemmatic (increased exploration and negative rewards), noisy (incorrect action during test phase), and basic, respectively. Details are as follows:

- **Explorative**: This is a modified version of basic scenarios, where the exploration annealing schedule is changed from "$50k$" to "$500k$" time steps following setups in Rashid et al. (2020a).
- **Dilemmatic**: This has increased exploration as explorative scenarios and a different reward shaping; the reward is determined not only by the enemy units' health (positive) but also by damages dealt by our agents (negative).
- **Noisy**: The training phase is the same as the basic scenarios. During the test phase, one agents may behave incorrectly. In 3s_vs_5z environment, one agent takes a noisy action with a 100% probability, and in 5m_vs_6m, it takes a noisy action with a 20% probability.
- **Basic**: This is a traditional MARL environment which was used in prior works (Rashid et al., 2018; Samvelyan et al., 2019; Wang et al., 2020b).

In explorative and dilemmatic scenarios, it is likely to obtain more non-cooperative behaviors from teammates. This represents the importance of considering cooperative risk explicitly. As shown in Section 3.1, the choice of training data distribution affects the action-value estimator in value factorization. Rashid et al. (2020a) also demonstrates the importance of change in exploration scheduling in the valued-based CTDE methods. In dilemmatic scenarios, where negative reward exists, if two sources of uncertainty are not disentangled well, units are induced to have a "selfish" behavior (e.g., running away) in order to avoid being damaged. In noisy scenarios, we deal with the unwanted behavior of agents that can occur in real-world applications. Early formulations of MARL methods focused on decentralized decision-making. However, if agents are to interact with each other in the real world, they are necessary to mitigate some potential dangers. We use two different noisy agents in the noisy scenarios. The first is a *random* agent that moves randomly, and the second is an *adversarial* agent that selects the malicious action that minimizes the action-value.

hyperparameters The hyperparameters of training and testing configurations for VDN, QMIX, and QTRAN are the same as in the recent GitHub code of SMAC [3] (Samvelyan et al., 2019) and PyMARL [4] with StarCraft version SC2.4.6.2.69232 The architecture of all agents' policy networks is a deep recurrent Q-network (Hausknecht & Stone, 2015) consisting of two 64-dimensional fully connected layers and one 64-dimensional GRU. The mixing networks consist of a single hidden layer with 32 hidden widths and ELU activation functions. Hypernetworks consists of two layers with 64 hidden widths and ReLU activation functions. Also, the hyperparameters of WQMIX [5] (Rashid et al., 2020a) and QPLEX [6] (Wang et al., 2020b) are the same as in their GitHub codes. However, unlike they used a specific configuration for some environments, such as adding non-linearity for MMM2, we used the same setting for all environments. Finally, the hyperparameters of DFAC [7]

[3]https://github.com/oxwhirl/smac
[4]https://github.com/oxwhirl/pymarl
[5]https://github.com/oxwhirl/wqmix
[6]https://github.com/wjh720/QPLEX
[7]https://github.com/j3soon/dfac

(Sun et al., 2021) are the same as in their GitHub code. As in their paper, we use 256 hidden layer sizes for DMIX (Sun et al., 2021) in the MMM2 environment and 512 hidden layer sizes for DDN (Sun et al., 2021) in the MMM2 environment. For other scenarios, we fix all the hidden layer sizes as 64 for a fair comparison. There may be slight differences compared to their paper due to differences in StarCraft version.

Like the DFAC paper (Sun et al., 2021), following the optimizer of the IQN (Dabney et al., 2018a), we used the Adam optimizer. For other methods except for DRIMA and DFAC, according to their papers, all neural networks are trained using the RMSProp optimizer with a 0.0005 learning rate. We use ϵ-greedy action selection with decreasing ϵ from 1 to 0.05 for exploration, following Samvelyan et al. (2019). For the discount factor, we set $\gamma = 0.99$. The replay buffer stores 5000 episodes at most, and the mini-batch is 32. Using a Nvidia Titan Xp graphic card, the training time varies from 8 hours to 24 hours for different scenarios.

To apply IQN for the true action-value estimator, we use an additional network, which computes an embedding $\phi(w_{\texttt{env}})$ for the sample point $w_{\texttt{env}}$. We calculate the embedding of $w_{\texttt{env}}$ with cosine basis functions and utilize element-wise (Hadamard) product, instead of a simple vector concatenation, for merging function as in IQN paper. The element-wise product forces a sufficiently early interaction between the two representations. The only difference is that because we use mixing networks rather than DQN, the hidden layer of the mixing network, not the convolution features, is multiplied by the embedded sample point to force interaction between the features and the embedded sample point.

For the loss function $\mathcal{L}_{\texttt{ub}}$, we fixed the value only for the threshold $\widehat{Q}_{\texttt{jt}}(s, \boldsymbol{\tau}, \boldsymbol{u}_{\texttt{opt}})$ of the clipping function that receives optimal actions as input. To combine our loss functions, we obtain the following objective, which is minimized in an end-to-end manner to train the true action-value estimator and the transformed action-value estimator:

$$\mathcal{L} = \mathcal{L}_{\texttt{td}} + \lambda_{\texttt{opt}}\mathcal{L}_{\texttt{opt}} + \lambda_{\texttt{nopt}}\mathcal{L}_{\texttt{nopt}} + \lambda_{\texttt{ub}}\mathcal{L}_{\texttt{ub}}$$

where $\lambda_{\texttt{opt}}, \lambda_{\texttt{nopt}} > 0$ are hyperparameters controlling the importance of each loss function. We set $\lambda_{\texttt{opt}} = 3$ and $\lambda_{\texttt{nopt}}, \lambda_{\texttt{ub}} = 1$.

As for the loss, the calculation of $\mathcal{L}_{\texttt{td}}$ was performed for multiple samples at the same time as in IQN. We simply set the number of samples $N_{\texttt{env}}$ for $w_{\texttt{env}}$ and $N'_{\texttt{env}} = 8$ for $w'_{\texttt{env}}$. Also, for the losses for the transformed action-value estimator, the expected value of true action-value estimator $Q_{\texttt{jt}}(\boldsymbol{u}) = \mathbb{E}_{w_{\texttt{env}}}[Z_{\texttt{jt}}(\boldsymbol{u}, w_{\texttt{env}})]$ was obtained with $K_{\texttt{env}} = 8$ samples. For the IQN (Dabney et al., 2018a) architecture, we use cosine basis functions with an embedding size of 64, ReLU nonlinearity, and multiplicative interaction according to their paper. For environmental risk-sensitivity, we used sampled $w_{\texttt{env}}$ from $\mathcal{U}(0, 0.25)$ as risk-averse, and $\mathcal{U}(0.75, 1.0)$ as risk-seeking.

For cooperative risk, we don't need to learn $Z_{\texttt{tran}}$ values for all possible $w_{\texttt{agt}}$ because, unlike $Z^{\texttt{tar}}_{\texttt{jt}}$, the value distribution of $Z_{\texttt{tran}}$ is not used for the target in loss functions. Therefore, for simplicity of implementation, we sample a fixed set of $w_{\texttt{agt}}$ with intervals of 0.1 from 0.1 to 1. In the execution phase, we select optimal action based on our cooperative risk-sensitivity $w_{\texttt{agt}}$. For the cooperative risk-sensitivity, we used $w_{\texttt{agt}} = 0.5$ as risk-neutral, $w_{\texttt{agt}} = 1.0$ as risk-seeking, and $w_{\texttt{agt}} = 0.1$ as risk-averse.

E. Experiments in simple matrix games

E.1. Comparing DRIMA with DMIX in stochastic two-step matrix game

In this section, to showcase that DRIMA is indeed able to disentangle risks, we conduct experiments in a stochastic two-step matrix game which is a diagnostic illustrative environment widely used in the MARL community (Rashid et al., 2018; Son et al., 2019; Sun et al., 2021). In this game, two agents play a two-step matrix game where they select one action in $\{A, B, C\}$ and receive the shared global reward by a payoff matrix which should be maximized. As shown in Table 3, our matrix game is modified from the matrix game employed by Son et al. (2019) in order to compare the capability of different methods in handling risks. In particular, the first and the second steps are designed to assess whether an algorithm is able to handle cooperative and environmental risks, respectively. Depending on the preference of a practitioner (i.e., environmental risk-seeking or risk-neutral), the proper policy must be learned. We compare DRIMA, and DMIX (Sun et al., 2021) with varying risk-sensitivity, conducted over 50k steps. We employ a full exploration scheme (*i.e.*, $\epsilon = 1$ in ϵ-greedy) so that all available states will be visited.

Analysis. As shown in Table 4, we observe that DRIMA is able to represent diverse type of risk-sensitive policies. To be specific, cooperative risk-seeking DRIMA selects action A which requires strong cooperation of a teammate in the first step. Whereas, DMIX have limited capability in adjusting risk-sensitivity. We observe that changing risk-sensitivity in DMIX affect cooperative and environmental risks simultaneously. Namely, risk-seeking adjustment makes both cooperative and environmental risk sensitivity become seeking. Therefore, DMIX may produce a suboptimal policy in environments which require different sensitivity for each risk source; note that cooperative risk-seeking and environmental risk-neutral policies have the highest mean reward.

Table 3: Payoff matrix of the two-step game. $\{e_1, e_2\}$ denotes a reward set in which one element of this set is uniformly sampled as a reward. The payoffs of the first and the second step are designed to assess whether an algorithm is able to handle cooperative and environmental uncertainty, respectively.

	A	B	C
A	7	-12	-12
B	-12	0	0
C	-12	0	0

(a) Payoff in the first step.

	A	B	C
A	$\{-10, 10\}$	$\{-11, 9\}$	$\{-11, 9\}$
B	$\{-11, 9\}$	$\{-8, -6\}$	$\{-11, 9\}$
C	$\{-11, 9\}$	$\{-11, 9\}$	$\{-26, 12\}$

(b) Payoff in the second step.

Table 4: Test rewards in the stochastic two-step matrix game for DRIMA, DMIX with varying risk-sensitivity across twelve random seeds. DRIMA (cooperative risk-seeking, environmental risk-neutral) agents select action action A in the first step, and select action A in the second step. DRIMA (risk-neutral) agents select action B or C in the first step, and select action A in the second step. DRIMA (risk-seeking) agents select action A in the first step, and select action C in the second step.

Algorithm	Risk sensitivity		Reward
	Cooperative	Environmental	
DRIMA (Ours)	Seeking	Neutral	**6.98**
DMIX (Sun et al., 2021)	Neutral		-0.02
DMIX (Sun et al., 2021)	Seeking		0.03

E.2. Simple one-step matrix game with noisy agents

In previous environments, cooperative risk-seeking always performs well because assuming "cooperative" teammates in training helps to learn tactics to beat enemies. However, in real world, randomly behaving agents or adversarial teammates are likely to exist. To demonstrate the importance of an cooperative risk-neutral policy, we increased the randomness of the agents' behavior in this experiment. In this setting, the agents act randomly with a 50% probability even during the test phase. As shown in Table 5, we eliminate the second step for environmental risk in the previous two-step simple matrix game, and increased the penalty for choosing the wrong action in the first step. We compare DRIMA, and DMIX (Sun et al., 2021) with varying risk-sensitivity, conducted over 50k steps. We employ a full exploration scheme (*i.e.*, $\epsilon = 1$ in ϵ-greedy) so that all available states will be visited.

In this experiment, the training settings are the same as in the previous simple two-step matrix game experiment in section 1, so the agents learn the same policy as before. As shown in Table 6, we observe that risk-neutral agents outperform risk-seeking agents. This is because cooperative risk-seeking agents always assume that other agents will be cooperative with them. These agents are easily degraded in the presence of other non-cooperating agents. In the real world applications, when a malicious external agent is added, there is a risk that multi-agent system is easily broken. Therefore, learning robust multi-agent systems against adversarial agents is an interesting future direction of research. We hope that our idea will help this new future research directions such as adversarial training in multi-agent reinforcement learning.

Table 5: Payoff matrix of the one-step game.

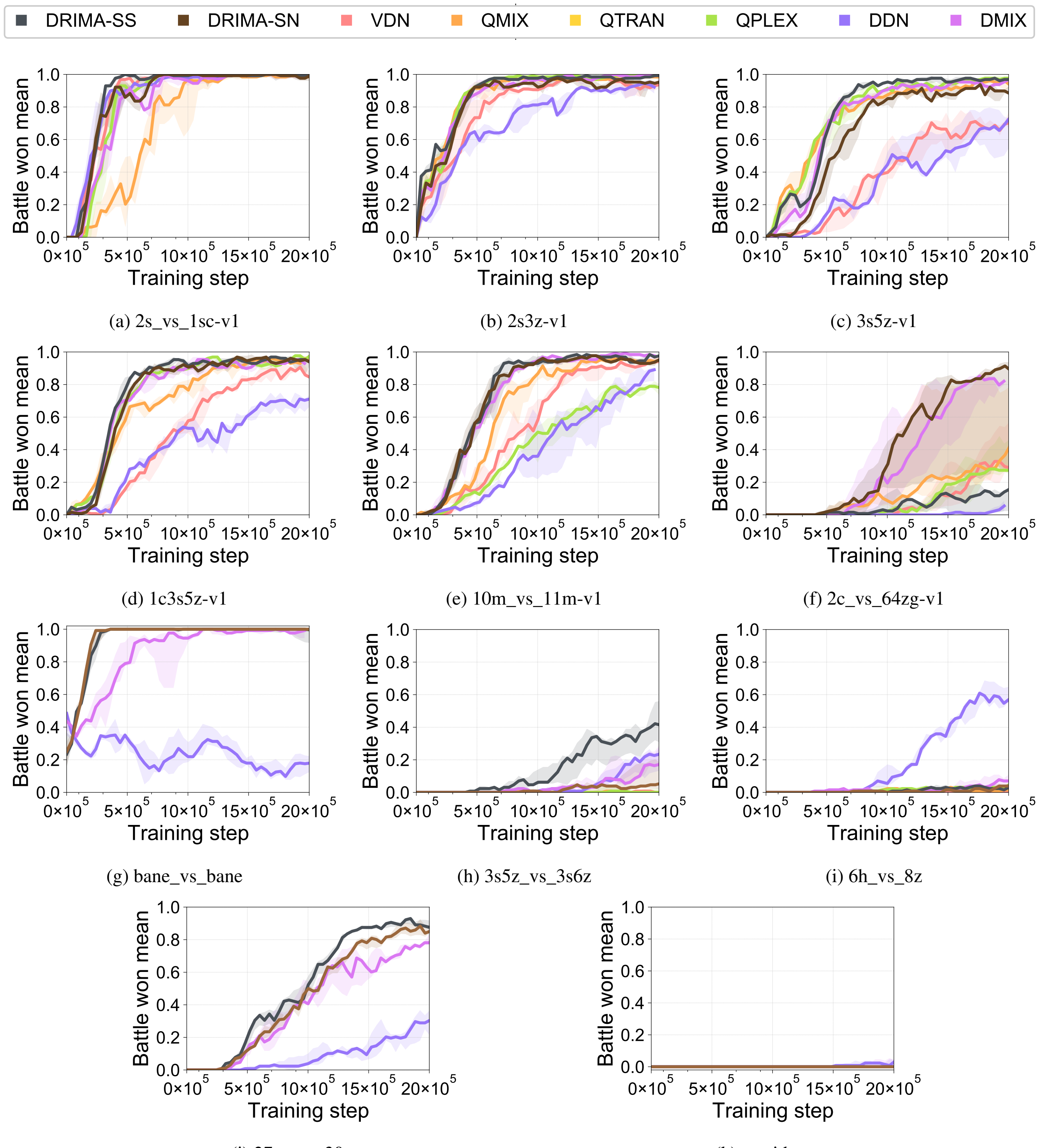
$u_1 \backslash u_2$	A	B	C
A	8	-100	-100
B	-100	0	0
C	-100	0	0

Table 6: Test rewards and trained policy in the one-step matrix game for DRIMA, DMIX, and OW-QMIX with varying risk-sensitivity across twelve random seeds.

Algorithm	Risk sensitivity		Test reward with noisy agents
	Cooperative	Environmental	Mean
DRIMA (Ours)	Neutral	Neutral	-26.92
	Seeking	Neutral	-45.01
DMIX (Sun et al., 2021)	Neutral		-26.65
	Seeking		-45.64

F. Additional experiments on SMAC

We provide experimental results across additional maps on SMAC. We report median test win rate with $25\% - 75\%$ percentile over five random seeds, comparing DRIMA with five baselines. One can observe DRIMA demonstrates competitive performance in overall.



(a) 2s_vs_1sc-v1 (b) 2s3z-v1 (c) 3s5z-v1

(d) 1c3s5z-v1 (e) 10m_vs_11m-v1 (f) 2c_vs_64zg-v1

(g) bane_vs_bane (h) 3s5z_vs_3s6z (i) 6h_vs_8z

(j) 27m_vs_30m (k) corridor

Figure 13: Median test win rate with 25%-75% percentile over four random seeds, comparing DRIMA with five baselines.