Risk Perspective Exploration in Distributional Reinforcement Learning

Jihwan Oh [1] **Joonkee Kim** [1] **Se-Young Yun** [1]

Abstract

Distributional reinforcement learning demonstrates state-of-the-art performance in continuous and discrete control settings with the features of variance and risk, which can be used to explore. However, the exploration method employing the risk property is hard to find, although numerous exploration methods in Distributional RL employ the variance of return distribution per action. In this paper, we present risk scheduling approaches that explore risk levels and optimistic behaviors from a risk perspective. We demonstrate the performance enhancement of the DMIX algorithm using risk scheduling in a multi-agent setting with comprehensive experiments.

1. Introduction

Reinforcement learning (Sutton & Barto, 2018) is a machine learning method that imitates the way humans learn to train models used in various domains such as robotics, autonomous driving, video games, economy, and industrial resource optimization. With one step further, the distributional perspective of deep reinforcement learning (Bellemare et al., 2017; Dabney et al., 2018b;a), has been highlighted with outperforming performance in Mujoco and Atari environment (Bellemare et al., 2013) compared to general reinforcement learning algorithms. In particular, one of the reasons why distributional RL shows state-of-the-art performance is that it utilizes exploration strategies with variances of return distributions which is a challenging issue in the reinforcement learning domain. Mavrin et al. (2019) and Zhou et al. (2021) propose exploration methods using the variance of a return distribution of distributional RL motivated by the existing reinforcement learning exploration strategy from Burda et al. (2018), Auer (2002). However, exploration utilizing risk level, which determines cautious or daring behavior for agents, is hard to find despite having suitable property for exploration compared to methods using the variance in distributional RL. In this paper, we propose risk perspective exploration utilizing risk levels by simply scheduling it, enabling exploration of the risk sampling domain and less explored action space. We evaluate our method with Multi-Agent Reinforcement Learning (MARL) algorithm, which has a problem of challenging exploration. We employ the distributional MARL method, DMIX, a variation of DFAC (Sun et al., 2021) that demonstrates state-of-the-art performance with variance and risk features but only implicitly employs a variance of return distribution as an exploration factor. By using risk as a factor of exploration in DMIX, we train agents based on the risk and show how the performance improves. We summarize our contributions as follows:

- We propose a novel risk perspective exploration methods by scheduling risk levels.
- We extensively evaluate the performance of scheduling risk levels in MARL environment and analyze the relation between risk levels and behaviors from it.

2. Related Works

2.1. Distributional Reinforcement Learning

Distributional RL outputs a distribution of returns per an action that can be defined by a Dirac delta function as follows.

$$Z_\theta(x,a) := \sum_{i=1}^{N} p_i \delta_{\theta i(x,a)} \tag{1}$$

where $\sum p_i = 1$. θ_i, x, and a represent return value, a state, and an action, respectively. Generally, loss functions are defined as a Wasserstein distance which is given by,

$$W_p(U,Y) = \left(\int_0^1 \left| F_Y^{-1}(\tau) - F_U^{-1}(\tau) \right|^p d\tau \right)^{1/p} \tag{2}$$

for TD-error of distribution with a Huber loss (Huber, 1992) where the inverse CDF F_Y^{-1} of a random variable Y is defined as,

$$F_Y^{-1}(\tau) := \inf\{ y \in \mathbb{R} : \tau \leq F_Y(y) \} \tag{3}$$

which is called a quantile function. Distributional RL is known to have low variance in performance, ensemble effect

---

[1]Kim Jaechul Graduate School of Artificial Intelligence, KAIST, Seoul, South Korea. Correspondence to: Se-Young Yun <yunseyoung@kaist.ac.kr>.

Presented at ICML Workshop on AI for Agent-Based Modelling (AI4ABM). Copyright 2022 by the author(s)

by making multiple outputs, and enabling risk-sensitive action selection as advantages.

Risk-sensitiveness is prevalent in economics and the stock market, where cautious or daring decisions are required. This approach was adopted in the domain of reinforcement learning known as risk-sensitive reinforcement learning for the benefit of selecting actions depending on risk. Generally, in the risk-sensitive RL area, risk-levels can be separated into 3 sections risk-averse, risk-neutral, and risk-seeking. Due to the variation in action space, a policy that is sensitive to risk can be read differently depending on the context. Nonetheless, we describe the general concept of risk-related policy in this section. A risk-averse policy can be interpreted as selecting the action with the highest state-action value among the worst-case scenarios per action. A risk-seeking policy entails selecting the same action as a risk-averse policy, but based on the best-case scenario. Risk-neutral policy positions amid risk-averse and seeking policy positions.

Exploration in Distributional RL Zhou et al. (2021) utilize Random Network Distillation (Burda et al., 2018) algorithm to make two same architectures with randomly initialized parameters. They use the Wasserstein metric between two random networks' output per action to measure how many times the action was explored. Mavrin et al. (2019) use left truncated variance of the return distribution, which means optimistic action selection using outputs' distribution.

Distributional MARL Recently, some multi-agent reinforcement learning (MARL) algorithms adopted distribution-based architecture. Sun et al. (2021) integrated distributional RL and MARL by mean-shape decomposition. Qiu et al. (2021) adopt the conditional value at risk (CVaR) metric as a surrogate of joint action-state value Q_{joint} and make a model that enables an adaptive risk level estimator with CVaR at every step.

2.2. Environments

In a multi-agent system, SMAC(Samvelyan et al., 2019) provides the most complex and dynamic environments for MARL researchers with partially observable MDP (POMDP), various scenarios, centralized training, and decentralized execution framework. Most MARL papers set SMAC as a default environment where they can prove their algorithm's performance. We experiment with our idea in this environment and show the relationship between risk levels and demanding skills.

3. Risk perspective exploration

In this paper, we only have interest on the distributional reinforcement learning's perspective on exploration and risk-sensitiveness. With this risk concept, we try exploration for various risk levels by scheduling these risks accompanying exploration for less explored actions which show a similar effect of using variance.

3.1. Selecting Risk levels

In Distributional RL for a risk-sensitive algorithm, the output distribution is represented by a quantile function (inverse CDF) with a domain of $[0, 1]$, as stated in subsection 2.1. Then, quantile fractions (which are typically expressed as τ) can be sampled from $[0, 1]$. By sampling quantile fractions, we can induce risk-averse or risk-seeking behavior in the model. If we wish to make agents risk-averse, for instance, we can select quantile fractions from $[0, 0.25]$ which samples relatively low returns. Contrarily, sampling quantile fractions between $[0.75, 1]$ indicate risk-seeking behavior which induces sampling high returns.

3.2. Exploration by risk scheduling

For decades, value-based RL algorithms have depended on ϵ-greedy(Watkins, 1989) decaying exploration or adding noise(Fortunato et al., 2017) to the model parameters for exploration. Considering the method of decaying epsilon value from high to low, the thought flashes that with distributional RL, which can handle risk-sensitiveness, will it be possible to schedule or decay risk levels like the ϵ-greedy exploration strategy?

In the ϵ-greedy exploration of distributional RL, we can select actions as follows:

$$\begin{cases} \arg\max_a \mathbb{E}_{\tau\sim\mathcal{U}[0,1]}[\mathbb{Z}(s,a)] & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \tag{4}$$

where s and a mean action and state each, and $\mathbb{Z}(s, a)$ is the distribution of return given state and action, which makes action-state value $Q(s, a)$ by taking expectation. We select the action that makes state-action value $Q(s, a)$ the best with the probability $1 - \epsilon$ and random action with the probability ϵ. We adapt this decaying idea from ϵ-greedy methods to risk scheduling which makes it to explore a variety of risk levels and also optimistic actions as follows:

$$\begin{cases} \text{argmax}_a \mathbb{E}_{\tau\sim\mathcal{U}[\alpha,\beta]}[\mathbb{Z}(s,a)] & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \tag{5}$$

where α and β adjust the risk levels and keep changing through the scheduling steps. Like decaying epsilon from 1 to 0.05, decaying risks gradually from seeking level to specific risk level is the basic format of our method. When scheduling risks within the range of risk-seeking level, β will change from 1 to 0 with α fixed to 0. β will be set to 0 when scheduling the risk levels within the range of risk-averse level. The details of how to schedule risks are explained in subsection 4.1.

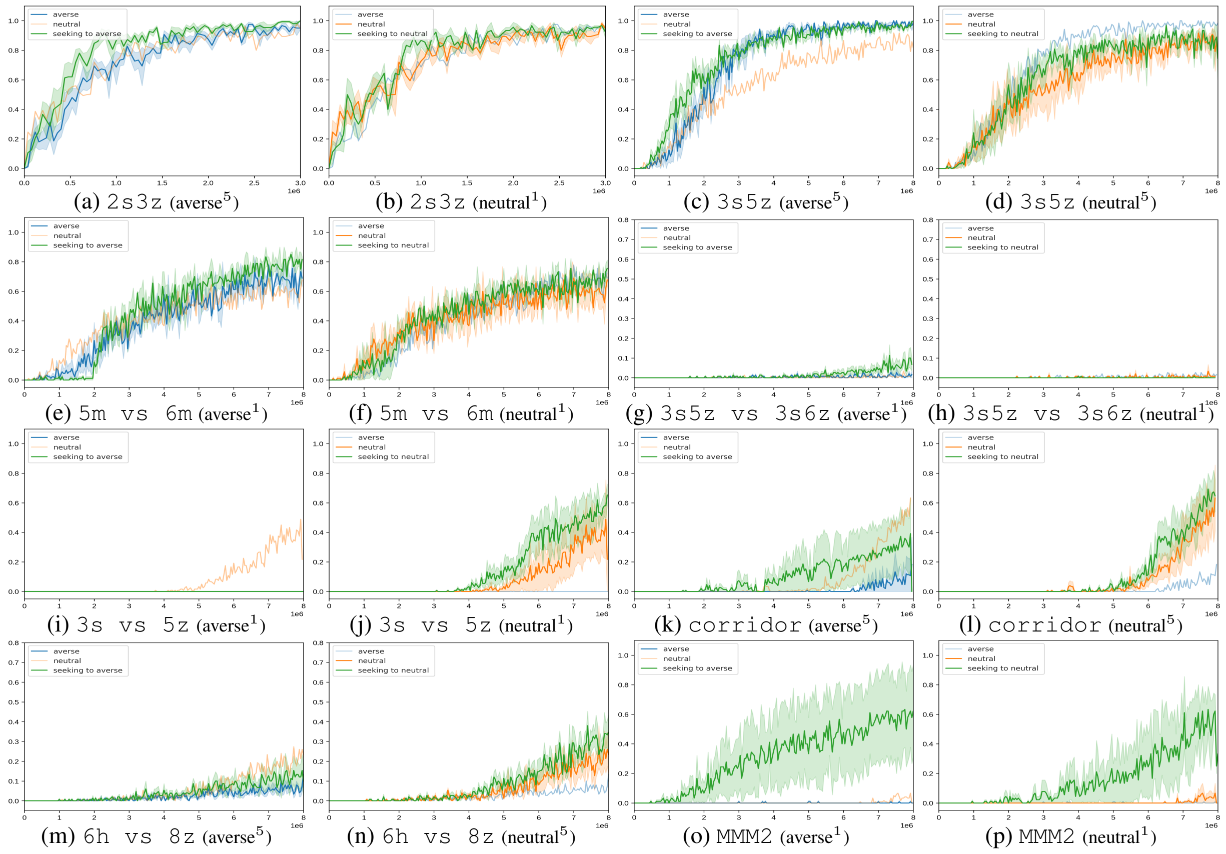

Figure 1. A Win-rate graph where the x-axis is training steps. We compare the averse and neutral policy with scheduling strategy from risk-seeking to averse or neutral. Lines are made by means of 5 runs, and shaded areas show a 75% confidence interval using five parallel training. We set the maximum win-rate to 0.8 for the scenarios that show performance lower than 0.5;otherwise, set it to 1.0. The number 1 and 5 located on the upper right of the letter 'averse' and 'neutral' means 10k and 50k scheduling timestep each.

4. Experiments

We evaluate our method with the MARL algorithm, which suffers from intricate exploration caused by a cooperative goal, multiple agents, and POMDP setting. In this experiment, the DMIX algorithm, a DFAC variant, is used. We perform experiments in SMAC (Samvelyan et al., 2019) environment, with 2s3z, 3s5z (*easy*), 5m vs 6m, 3s vs 5z (*hard*), and corridor, 3s5z vs 3s6z, 6h vs 8z, MMM2 (*super hard*) scenarios each. Except for running SMAC environment in a parallel manner with five runners, we follow the default setting of DFAC with 8 million training time step.

4.1. Risk Scheduling

We set **risk-averse**, **risk-neutral**, and **risk-seeking** as a way of sampling quantile fractions (τ) from a uniform distribution $\mathcal{U}[0, 0.25]$, $\mathcal{U}[0, 1]$, and $\mathcal{U}[0.75, 1]$ each. Although there are many decaying skills in scheduling, we select the most basic decaying method, *Linear* decaying. For example, if we want to schedule risk from seeking to averse, we set α and β in Equation 5 to 0.75 and 1.0 each. Then sample quantile fractions from $[0.75, 1]$ and α start decaying to 0 linearly. When α becomes 0, β begins to move toward 0.25 linearly. The sampling range reaches $[0, 0.25]$ at the final scheduling step and is fixed until the training is finished. Seeking to averse and seeking to neutral risk policies are adopted in the experiments to compare with static risk level policies. Because the risk-seeking policy does not work in any scenarios, we did not insert the results of risk-seeking policy in this paper. We search decaying steps in a set of $\{10\text{k}, 25\text{k}, 50\text{k}\}$.

4.2. Results

Figure 1 demonstrates the learning curves in SMAC scenarios. Figures from 1a to 1h show better performance when taking risk-averse policy, and Figures from 1j to 1p show better performance when taking risk-neutral policy. Also, it can be observed that the learning curves with risk scheduling grow faster and have higher final win-rates than static policies, as shown in Table 1. In addition, we find that there is a tendency that the more complex the scenario is, (the less

Table 1. Final win-rate performance

Maps	neutral	neutral*	averse	averse*
2s3z	0.951	0.948	0.951	**0.988**
3s5z	0.822	0.849	0.977	**0.982**
5m vs 6m	0.607	0.718	0.696	**0.792**
3s vs 5z	0.422	**0.604**	0.0	0.0
corridor	0.533	**0.662**	0.122	0.320
6h vs 8z	0.235	**0.340**	0.084	0.124
MMM2	0.044	0.518	0.002	**0.607**
3s5z vs 3s6z	0.006	0.0	0.013	**0.086**

$*$: scheduling method

Table 2. Challenging skills & risk level

Maps	challenging skills	risk levels
2s3z	focusing fire	averse = neutral
3s5z	focusing fire	averse
5m vs 6m	focusing fire	averse
3s vs 5z	kiting	neutral
corridor	breaking each	neutral
6h vs 8z	kiting	neutral
MMM2	focusing fire	neutral
3s5z vs 3s6z	focusing fire	averse

likely the winning rate is), the performance improvements increase.

4.3. Analysis

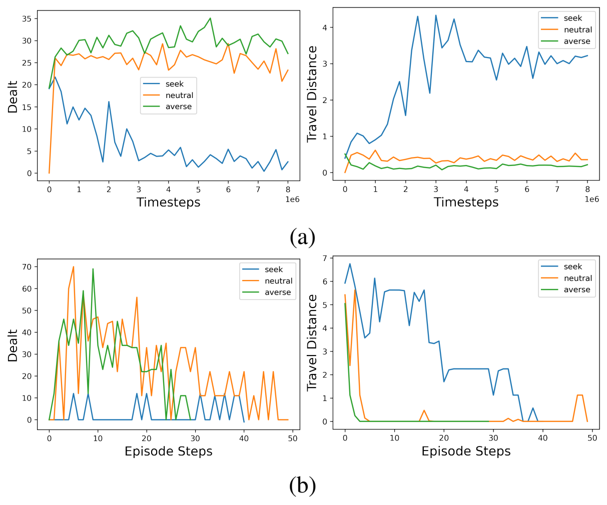

Figure 2. Results on 6h vs 8z scenario. (a) left shows average dealt from agents to enemies and right shows travel distance of agents per timestep in a single episode each. (b) left shows total dealt from agents to enemies and right shows travel distance of agents at each timestep in a single episode.

Relation between risk level & behavior We have evidence that risk levels are related to agents' behavior tendencies in the SMAC scenario. Risk-averse behavior tends to attack enemies more than other policies, as shown in Figure 2a deploy intensive fire per timestep. This entirely corresponds to the fact that risk-averse policy shows the behavior that chooses the best action among the worst case, which can be just attacking the enemies. Seeking behavior tends to do other than attacking (e.g., Move around) relative to other policies seeing the Figure 2a, which can be interpreted as a case that agents select action among the best case. Because if agents survive longer by the moving behavior, the possibility of receiving a good return could be higher than other actions. However, moving continuously by risk-seeking policy can result in a bad performance, which makes no immediate reward. Risk-neutral behavior shows intermediate temper that demonstrates adequate firepower per timestep and more moving distance than averse policy, as shown in Figure 2a. Figure 2b, conducted in a single episode, supports the above argument in detail. So Figure 2 demonstrates why the tendency of Figure 1 comes out. With this evidence, we conclude that risk-averse and neutral policy is related to *focusing fire* and *kiting* skills each. Scenarios and challenging skills are shown in Table 2.

Scheduling risk levels Training with static risk level makes the model experience only the particular risk trends even though the risk level is neutral. However, agents can efficiently explore behavior trends by scheduling risks to explore risk levels and optimistic behaviors at the beginning of learning. Especially, some *super hard* scenarios show substantial performance improvements when scheduling risks as shown in Figure 1, confirming that the effect of risk scheduling is significant where more exploration is required. In addition, we found that the longer the scheduling period is, the lower the performance, as shown in Figure 3. Extending decaying steps result in the continued exploration rather than stable learning, similar to ϵ-greedy scheduling. Therefore, appropriate scheduling steps are required.

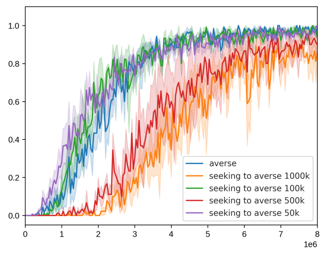

Figure 3. Control scheduling steps in scenario 3s5z

5. Conclusion & Future work

In distributional reinforcement learning, we propose risk perspective exploration via risk scheduling. Scheduling risk level from a risk-seeking level to a specified risk level significantly accelerates learning speed and improves final performance in comparison to not scheduling it. In addition, the risk level is highly correlated with the challenging skills shown in SMAC scenarios, making the training feasible and stable. In the near future, we hope to exhibit this intriguing demonstration using various distributional algorithms that can handle single or multiple agents, as well as in various simulators.

Acknowledgements

This work was conducted by Center for Applied Research in Artificial Intelligence(CARAI) grant funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD) (UD190031RD).

References

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, 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.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Huber, P. J. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pp. 492–518. Springer, 1992.

Mavrin, B., Yao, H., Kong, L., Wu, K., and Yu, Y. Distributional reinforcement learning for efficient exploration. In *International conference on machine learning*, pp. 4424–4434. PMLR, 2019.

Qiu, W., Wang, X., Yu, R., Wang, R., He, X., An, B., Obraztsova, S., and Rabinovich, Z. Rmix: Learning risk-sensitive policies forcooperative reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34, 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 *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 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 *International Conference on Machine Learning*, pp. 9945–9954. PMLR, 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. *arXiv preprint arXiv:1706.05296*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Watkins, C. J. C. H. Learning from delayed rewards. 1989.

Zhou, F., Zhu, Z., Kuang, Q., and Zhang, L. Non-decreasing quantile function network with efficient exploration for distributional reinforcement learning. *arXiv preprint arXiv:2105.06696*, 2021.

A. Distributional Reinforcement Learning

Distributioanl deep RL got popular from (Bellemare et al., 2017). The output of distributional RL is a distribution of returns for a given state and action. They use the Wasserstein metric to calculate the TD-error between the distributional bellman updated distribution and the current distribution of returns in this approach. Although it differs across methods, most of them employ the quantile function to estimate the distribution of returns in order to calculate the Wasserstein distance between the current and objective distributions. Assume we have a model that approximates a quantile function with a domain of [0, 1] and a return range of $(-\infty, +\infty)$. The output of a model may approximate the inverse of a cumulative density function like Figure 4.

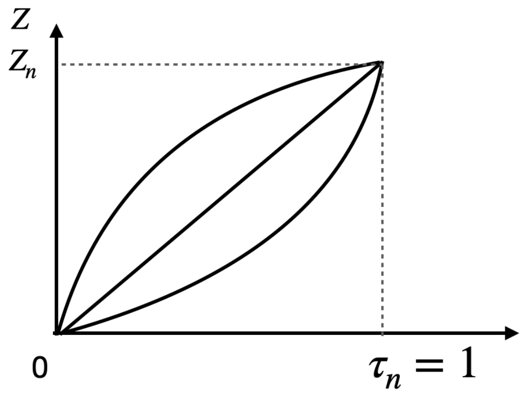

Figure 4. Quantile regression

A.1. Algorithm

DFAC DFAC(Sun et al., 2021) method, which is based on the IQN(Dabney et al., 2018a) algorithm, is the first to integrate distributional RL and multi-agent RL. IQN was utilized to sample quantile fractions from $\mathcal{U}[0, 1]$ for distributional output. The authors incorporated a distributional viewpoint in a multi-agent context using mean-shape decomposition without breaking the IGM condition, which can be stated as:

$$\arg\max_{\mathbf{u}} \mathbb{E}[Z_{joint}(\mathbf{h}, \mathbf{u})] = \begin{pmatrix} \arg\max_{\mathrm{u}_1} \mathbb{E}[Z_1(\mathrm{h}_1, \mathrm{u}_1)] \\ \vdots \\ \arg\max_{\mathrm{u}_\mathrm{N}} \mathbb{E}[Z_N(\mathrm{h}_\mathrm{N}, \mathrm{u}_\mathrm{N})] \end{pmatrix} \tag{6}$$

that can be proved by the following DFAC Theorem (Sun et al., 2021):

$$\begin{aligned} Z_{joint}(\mathbf{h}, \mathbf{u}) &= \mathbb{E}[Z_{joint}(\mathbf{h}, \mathbf{u})] + Z_{joint}(\mathbf{h}, \mathbf{u}) - \mathbb{E}[Z_{joint}(\mathbf{h}, \mathbf{u})] \\ &= Z_{mean}(\mathbf{h}, \mathbf{u}) + Z_{shape}(\mathbf{h}, \mathbf{u}) \\ &= \psi(Q_1(\mathrm{h}_1, \mathrm{u}_1), \ldots, Q_N(\mathrm{h}_N, \mathrm{u}_N)) \\ &+ \Phi(Z_1(\mathrm{h}_1, \mathrm{u}_1), \ldots, Z_N(\mathrm{h}_N, \mathrm{u}_N)) \end{aligned} \tag{7}$$

which is proven to meet the IGM condition. DFAC outperforms all other algorithms, especially in difficult scenarios. This approach may also be modified to work with IQL, VDN(Sunehag et al., 2017), and QMIX(Rashid et al., 2018). The DMIX variations of the DFAC algorithm, which combines with QMIX, are employed as our baseline.

Exploration of DMIX DMIX (Sun et al., 2021) is another value-based RL version incorporating ϵ-greedy exploration. However, this method leverages variance information from the return distribution while choosing an action. DMIX generates a return distribution using uniformly sampled quantile fractions from $\mathcal{U}[0, 1]$. If we set the number of quantile fractions to a tiny number, such as a digit of one, uniformly sampling quantile fractions stochastically results in exploiting tail information of a return distribution. For default, DMIX sets the number of quantile fractions to 8. As a result, this approach indirectly employs the variance of the return distribution per action for exploration.