A Proofs

We present proofs of our propositions and theorem introduced in the main text. The numbers of proposition, theorem and equations are reused in restated propositions.

Proposition 1. *$\mathcal{T} : \mathcal{C} \mapsto \mathcal{C}$ is a γ-contraction.*

Proof. We consider the sup-norm contraction,

$$\left|\mathcal{T}C_{(1)}(\boldsymbol{s},\boldsymbol{u}) - \mathcal{T}C_{(2)}(\boldsymbol{s},\boldsymbol{u})\right| \leq \gamma \left\|C_{(1)}(\boldsymbol{s},\boldsymbol{u}) - C_{(2)}(\boldsymbol{s},\boldsymbol{u})\right\|_{\infty} \quad \forall \boldsymbol{s} \in \mathcal{S}, \boldsymbol{u} \in \mathcal{U}. \tag{9}$$

The sup-norm is defined as $\|C\|_{\infty} = \sup_{\boldsymbol{s}\in\mathcal{S},\boldsymbol{u}\in\mathcal{U}} |C(\boldsymbol{s},\boldsymbol{u})|$ and $C \in \mathbb{R}$.

In $\{C_i\}_{i=1}^N$, the risk level is fixed and can be considered implicit input. Given two different return distributions $Z_{(1)}$ and $Z_{(2)}$, we prove:

$$\begin{aligned}
\left|\mathcal{T}C_{(1)} - \mathcal{T}C_{(2)}\right| &\leq \max_{\boldsymbol{s},\boldsymbol{u}} \left|\left[\mathcal{T}C_{(1)}\right](\boldsymbol{s},\boldsymbol{u}) - \left[\mathcal{T}C_{(2)}\right](\boldsymbol{s},\boldsymbol{u})\right| \\
&= \max_{\boldsymbol{s},\boldsymbol{u}} \left|\gamma \sum\nolimits_{\boldsymbol{s}'} \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{u}) \left(\max_{\boldsymbol{u}'} C_{(1)}(\boldsymbol{s}',\boldsymbol{u}') - \max_{\boldsymbol{u}'} C_{(2)}(\boldsymbol{s}',\boldsymbol{u}')\right)\right| \\
&\leq \gamma \max_{\boldsymbol{s}'} \left|\max_{\boldsymbol{u}'} C_{(1)}(\boldsymbol{s}',\boldsymbol{u}') - \max_{\boldsymbol{u}'} C_{(2)}(\boldsymbol{s}',\boldsymbol{u}')\right| \\
&\leq \gamma \max_{\boldsymbol{s}',\boldsymbol{u}'} \left|C_{(1)}(\boldsymbol{s}',\boldsymbol{u}') - C_{(2)}(\boldsymbol{s}',\boldsymbol{u}')\right| \\
&= \gamma \left\|C_{(1)} - C_{(2)}\right\|_{\infty}
\end{aligned} \tag{10}$$

This further implies that

$$\left|\mathcal{T}C_{(1)} - \mathcal{T}C_{(2)}\right| \leq \gamma \left\|C_{(1)} - C_{(2)}\right\|_{\infty} \quad \forall \boldsymbol{s} \in \mathcal{S}, \boldsymbol{u} \in \mathcal{U}. \tag{11}$$

□

Proposition 2. *For any agent $i \in \{1,\dots,N\}$, $\exists \lambda(\tau_i,u_i) \in (0,1]$, such that $C_i(\tau_i,u_i) = \lambda(\tau_i,u_i)\mathbb{E}\left[Z_i(\tau_i,u_i)\right]$.*

Proof. We first provide that given a return distribution Z, return random variable $\mathscr{Z}$ and risk level $\alpha \in \mathcal{A}$, $\forall z$, $\Pi_\alpha Z$ can be rewritten as $\mathbb{E}\left[\mathscr{Z}|\mathscr{Z} < z\right] < \mathbb{E}\left[\mathscr{Z}\right]$. This can be easily proved by following [32]'s proof. Thus we can get $\Pi_\alpha Z < \mathbb{E}\left[Z\right]$, and there exists $\lambda_{(\tau_i,u_i)} \in (0,1]$, which is a value of agent's trajectories, such that $\Pi_\alpha Z_i(\tau_i,u_i) = \lambda_{(\tau_i,u_i)}\mathbb{E}\left[Z_i(\tau_i,u_i)\right]$. □

Proposition 2 implies that we can view the CVaR value as truncated values of Q values that are in the lower region of return distribution $Z_i(\tau_i,u_i)$. CVaR can be decomposed into two factors: $\lambda_{(\tau_i,u_i)}$ and $\mathbb{E}[Z_i(\tau_i,u_i)]$.

Theorem 1. *We summarize the following properties:*
(1) Given $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ where α values in each set are identical, $\mathbb{E}[\Psi_{\boldsymbol{\alpha}_1}] \leq \mathbb{E}[\Psi_{\boldsymbol{\alpha}_2}]$ for $0 < \alpha_1 \leq \alpha_2 \leq 1$, $\alpha_1 \in \boldsymbol{\alpha}_1$ and $\alpha_2 \in \boldsymbol{\alpha}_2$.
(2) $\exists \alpha_i \in (0,1]$ and $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^N$, $\mathbb{E}[\Psi_{\boldsymbol{\alpha}}] < 0$.

Proof. We first consider the static risk level for each agent and the linear additivity mixer [43] of individual CVaR values.

(1) Given $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ and the mixer is the additivity function, we derive $\mathbb{E}[\Psi_\alpha]$ as follows

$$\mathbb{E}[\Psi_\alpha] \triangleq \mathbb{E}\left[r + \gamma \max_{\boldsymbol{u}'} C^{\text{tot}}(\boldsymbol{s}',\boldsymbol{u}') - \left(r + \gamma \max_{\boldsymbol{u}'} Q(\boldsymbol{s}',\boldsymbol{u}')\right)\right] \tag{12}$$

$$= \mathbb{E}\left[\gamma \left(\max_{\boldsymbol{u}'} C^{\text{tot}}(\boldsymbol{s}',\boldsymbol{u}') - \max_{\boldsymbol{u}'} Q(\boldsymbol{s}',\boldsymbol{u}')\right)\right] \tag{13}$$

$$= \mathbb{E}\left[\gamma \left(\sum\nolimits_{i=1}^{N} \max_{u_i} C_i(\tau_i,u_i,\alpha) - \sum\nolimits_{i=1}^{N} \max_{u_i} Q_i(\tau_i,u_i)\right)\right] \tag{14}$$

Following Proposition 2 and Eqn. 2, given $\alpha_1 \leq \alpha_2$ and for any $i \in N$ and $u \in \mathcal{U}$, we can easily derive

$$C_i(\tau_i,u_i,\alpha_1) \leq C_i(\tau_i,u_i,\alpha_2) \tag{15}$$

Thus,

$$\max_{u_i} C_i(\tau_i, u_i, \alpha_1) \leq \max_{u_i} C_i(\tau_i, u_i, \alpha_2) \tag{16}$$

Then, we can get

$$\mathbb{E}[\Psi_{\alpha_1}] = \mathbb{E}\left[\gamma\left(\sum\nolimits_{i=1}^{N} \max_{u_i} C_i(\tau_i, u_i, \alpha_1) - \sum\nolimits_{i=1}^{N} \max_{u_i} Q_i(\tau_i, u_i)\right)\right] \tag{17}$$

$$\leq \mathbb{E}\left[\gamma\left(\sum\nolimits_{i=1}^{N} \max_{u_i} C_i(\tau_i, u_i, \alpha_2) - \sum\nolimits_{i=1}^{N} \max_{u_i} Q_i(\tau_i, u_i)\right)\right] \tag{18}$$

$$= \mathbb{E}[\Psi_{\alpha_2}] \tag{19}$$

Finally, we get $\mathbb{E}[\Psi_{\alpha_1}] \leq \mathbb{E}[\Psi_{\alpha_2}]$.

Note that, it also applies when the mixer is the monotonic network by following the proof of Theorem 1 in [34]. Here we present the proof in RMIX for the convenience of readers.

With monotonicity network f_{m}, in RMIX, we have

$$C^{\mathrm{tot}}(\boldsymbol{s}, \boldsymbol{u}) = f_{\mathrm{m}}(C_1(\tau_1, u_1, \alpha_1), \ldots, C_n(\tau_n, u_n, \alpha_n)) \tag{20}$$

Consequently, we have

$$C^{\mathrm{tot}}(\boldsymbol{s}, \{\arg\max_{u'} C_i(\tau_i, u', \alpha_i)\}_{i=1}^{N}) = f_{\mathrm{m}}(\{\max_{u'} C_i(\tau_i, u', \alpha_i)\}_{i=1}^{N}) \tag{21}$$

By the monotonocity property of f_{m}, we can easily derive that if $j \in \{1, \ldots, N\}$, $u_j^* = \arg\max_{u_i} C_j(\tau_j, u_j, \alpha_j)$, $\alpha_j \in (0, 1]$ is the risk level given the current return distributions and historical return distributions, and actions of other agents are not the best action, then we have

$$f_{\mathrm{m}}(\{C_j(\tau_j, u_j, \alpha_j)\}_{i=1}^{N}) \leq f_{\mathrm{m}}(\{C_j(\tau_j, u_j, \alpha_j)\}_{i=1, i\neq j}^{N}, C_j(\tau_j, u_j^*, \alpha_j)). \tag{22}$$

So, for all agents, $\forall j \in \{1, \ldots, N\}$, $u_j^* = \arg\max_{u_j} C_j(\tau_j, u_j, \alpha_j)$, we have

$$f_{\mathrm{m}}(\{C_j(\tau_i, u_i, \alpha_i)\}_{i=1}^{N}) \leq f_{\mathrm{m}}(\{C_j(\tau_j, u_j, \alpha_j)\}_{i=0, i\neq j}^{n-1}, C_j(\tau_j, u_j^*)) \tag{23}$$

$$\leq f_{\mathrm{m}}(\{C_i(\tau_i, u_i^*, \alpha_i)\}_{i=1}^{N}) \tag{24}$$

$$= \max_{\{u_i\}_{i=1}^{N}} f_{\mathrm{m}}(\{C_i(\tau_i, u_i, \alpha_i)\}_{i=1}^{N}). \tag{25}$$

Therefore, we can get

$$\max_{\{u_i\}_{i=1}^{N}} f_{\mathrm{m}}(\{C_i(\tau_i, u_i, \alpha_i)\}_{i=1}^{N}) = \max_{\boldsymbol{u}} C^{\mathrm{tot}}(\boldsymbol{s}, \boldsymbol{u}), \tag{26}$$

which implies

$$\max_{\boldsymbol{u}} C^{\mathrm{tot}}(\boldsymbol{s}, \boldsymbol{u}) = C^{\mathrm{tot}}(\boldsymbol{s}, \{\arg\max_{u'} C_i(\tau_i, u')\}_{i=1}^{N}). \tag{27}$$

The Eqn. 26 can be used in Eqn. 13 to derive the above results.

(2) By following Proposition 2. We can get

$$\mathbb{E}[\Psi_{\boldsymbol{\alpha}}] = \mathbb{E}\left[\gamma\left(\sum\nolimits_{i=1}^{N} \max_{u_i} C_i(\tau_i, u_i, \alpha_i) - \sum\nolimits_{i=1}^{N} \max_{u_i} Q_i(\tau_i, u_i)\right)\right] \tag{28}$$

$$= \mathbb{E}\left[\gamma\left(\sum\nolimits_{i=1}^{N} \max_{u_i} C_i(\tau_i, u_i, \alpha_2) - \sum\nolimits_{i=1}^{N} \max_{u_i} Q_i(\tau_i, u_i)\right)\right] \tag{29}$$

$$= \mathbb{E}\left[\gamma\left(\sum\nolimits_{i=1}^{N} \max_{u_i} \lambda_{(\tau_i, u_i)} Q_i(\tau_i, u_i) - \sum\nolimits_{i=1}^{N} \max_{u_i} Q_i(\tau_i, u_i)\right)\right] \tag{30}$$

$$= \mathbb{E}\left[\gamma\left(\sum\nolimits_{i=1}^{N} \max_{u_i} \left((\lambda_{(\tau_i, u_i)} - 1) Q_i(\tau_i, u_i)\right)\right)\right] \tag{31}$$

We can get that $\exists \alpha_i \in (0, 1]$ and $i \in \{1, \ldots, N\}$, $\mathbb{E}[\Psi_{\alpha}] < 0$. □

B Environmental Settings

B.1 MACN

We customize the cliff walking environment [44] in single-agent domain and develop Multi-Agent Cliff Navigation (MACN) for multi-agent navigation with risk. In MACN, there are two agents whose task is to complete the navigation from the starting position to the goal. At each time step, each agent observes an observation with a dimension of 3×3. Agents can take an action at each time step and the action set is: $\{\text{UP}, \text{DOWN}, \text{LEFT}, \text{RIGHT}\}$. The two agents share the team reward. As depicted in Fig. 13, there are some regions that are dangerous and agents will be rewarded with a -100 reward when any agent steps into these region, and consequently the episode ends. Agents will receive a -1 reward at each time step when they are at the safe region. When one agent reaches the goal, the agents will be rewarded with a -0.5 reward. If the two agents arrive at the goal at the same time, agents will be rewarded with a 0 reward and the episode ends. The objective of the agents is to maximize the accumulated rewards in each episode.

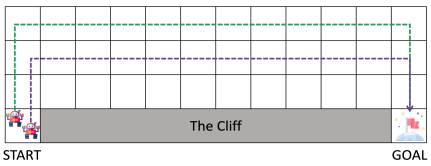

Figure 13: Multi-Agent Cliff Navigation.

B.2 SCII

SMAC benchmark is a challenging set of cooperative StarCraft II[TM4] maps for micromanagement developed by Samvelyan et al. [39] built on DeepMind's PySC2 [51]. We introduce **states and observations**, **action space** and **rewards** of SMAC, and **environmental settings of RMIX** below.

States and Observations. At each time step, agents get local observations within their field of view, which contains information (relative x, relative y, distance, health, shield, and unit type) about the map within a circular area for both allied and enemy units and makes the environment partially observable for each agent. The global state is composed of the joint observations, which could be used during training. All features, both in the global state and in individual observations of agents, are normalized by their maximum values.

Action Space. actions are in the discrete space. Agents are allowed to make move[direction], attack[enemy id], stop and no-op. The no-op action is only legal action for dead agents. Agents can only move in four directions: north, south, east, or west. The shooting range is set to for all agents. Having a larger sight range than a shooting range allows agents to make use of the move commands before starting to fire.

Rewards. At each time step, the agents receive a shared reward equal to the total damage dealt on the enemy agents. In addition, agents receive a bonus of 10 points after killing each opponent, and 200 points after killing all opponents for winning the battle. The rewards are scaled so that the maximum cumulative reward achievable in each scenario is around 20.

Environmental Settings of RMIX. The difficulty level of the built-in game AI we use in our experiments is level 7 (very difficult) by default as many previous works did [35, 25]. The scenarios used in Section 5 are shown in Table 2. We present the table of all scenarios in SMAC in Table 2 and the corresponding memory usage for training each scenario in Table 3. The *Ally Units* are agents trained by MARL methods and *Enemy Units* are built-in game bots. For example, 5m_vs_6m indicates that the number of MARL agent is 5 while the number of the opponent is 6. The agent (unit) type is *marine*[5]. This asymmetric setting is hard for MARL methods. Asymmetric scenario indicates the number of the ally units and that of the opponent are not equal. Heterogeneous scenario means that ally units are from different categories whose action and observation spaces are different while heterogeneous agents belong to the same category. The version of SCII simulator is 4.10. We evaluate our method for every 10,000 training steps during training by running 32 episodes in which agents trained with our method battle with built-in game bots. We report the mean test win rate (percentage of episodes win of MARL agents) along with one standard deviation of test win rate (shaded in figures).

[4]StarCraft II is a trademark of Blizzard Entertainment, Inc.

[5]A type of unit (agent) in StarCraft II. Readers can refer to `https://liquipedia.net/starcraft2/Marine_(Legacy_of_the_Void)` for more information

Table 2: SMAC Environments

Name	Ally Units	Enemy Units	Type
5m_vs_6m	5 Marines	6 Marines	homogeneous & asymmetric
8m_vs_9m	8 Marines	9 Marines	homogeneous & asymmetric
10m_vs_11m	10 Marines	11 Marines	homogeneous & asymmetric
MMM2	1 Medivac, 2 Marauders & 7 Marines	1 Medivac, 3 Marauders & 8 Marines	heterogeneous & asymmetric
1c3s5z	1 Colossi & 3 Stalkers & 5 Zealots	1 Colossi & 3 Stalkers & 5 Zealots	heterogeneous & symmetric
corridor	6 Zealots	24 Zerglings	micro-trick: wall off

C Training Details

We use the same neural network architecture used by QMIX [35]. The trajectory embedding network ϕ_i is similar to the network of the agent. The last layer of the risk level predictor generates the local return distribution and shares the same weights with the last layer of the agent network. The QR loss is minimized to periodically (empirically every 50 time steps) update the weights of the agent as simultaneously updating the local distribution and the whole network can impede training. The QR loss is used for updating the local distribution while the TD loss of centralized training is for the agent network weights learning and credit assignment. In the QR loss, the $C_i(\cdot,\cdot,\cdot)$ is considered as scalar value and gradients from $C_i(\cdot,\cdot,\cdot)$ are blocked. As the QR update relies on accurately estimation of the dummy reward C_i, we start to update C_i when the test wining rate is over 35%, which means the agents have grasped some strategies to win the game.

SCII. To make a fair comparison, we use `episode` (single-process environment for training, compared with `parallel`) runner defined in PyMARL to run all methods. We use `num_circle=1` to train QPLEX for fair training at the sample level for each update. We use Adam to optimize the CVaR and QR loss as suggeted by QR-DQN [9] with a learning rate of 5×10^{-4}. The batch size is 32 and the replay buffer size is 5000. We use $K = 10$. We set $M = 100$ for 5m_vs_6m and 8m_vs_9m. To trade off the experimental performance and the computation overhead, we use $M = 55$ for corridor, and $M = 65$ for 10m_vs_11m, 1c3s5z and MMM2. In order to explore, we use ϵ-greedy with ϵ annealed linearly from 1.0 to 0.05 over 50K time steps from the start of training and keep it constant for the rest of the training for all methods. The discount factor $\gamma = 0.99$ and we follow the default hyper-parameters used in the original papers of all baselines. The evaluation interval is $10,000$ for all methods. We use uniform probability to estimate $Z_i(\cdot,\cdot)$ for each agent. Experiments are carried out on NVIDIA Tesla V100 GPU 16G and NVIDIA GeForce RTX 3090 24G. We also provide memory usage of baselines (given the current size of the replay buffer) for training each scenario of SCII domain in SMAC as shown in Table 3.

MACN. We follow the same settings used in SCII to train RMIX and baselines in MACN scenarios. $M = 100$ and $K = 10$.

Table 3: Memory usage (given the current size of the replay buffer) for the training of each method (exclude COMA and DOP, which are on-policy methods without using replay buffers) on scenarios of SCII domain in SMAC.

Scenario	Memory Usage (GB)
5m_vs_6m	3
8m_vs_9m	4.9
10m_vs_11m	7.1
1c3s5z	8.6
MMM2	10.8
corridor	14.4

D Baseline Methods

We briefly introduce baselines evaluated in our experimental section and summarize them in Table 4.

IQL [46]. IQL is an independent Q-learning method for multi-agent RL. Each agent learns its independent Q values via DQN [28].

VDN [43]. To address the credit assignment issue in Dec-POMDP MARL methods, VDN utilizes a linear combination of individual Q values to represent the $Q^{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{u})$ as $Q^{\text{tot}} = \sum_{i=1}^{N} Q_i(\tau_i, u_i)$. Then, TD-learning can be used to train Q^{tot}.

QMIX [35]. Unlike the linear combination used in VDN, QMIX considers the monotonic constraint on the relationship between Q^{tot} and Q_i, that is

$$\frac{\partial Q^{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{u})}{\partial Q_i(\tau_i, u_i)} \geq 0, \forall i \in \{1, 2, \ldots, N\}$$

where $Q^{\text{tot}(\boldsymbol{\tau}, \boldsymbol{u})} = f_m(Q_1(\tau_1, u_1), \ldots, Q_N(\tau_N, u_N))$ and f_m is a mixing network built on top of a hyperntework [13] with $(Q_1(\tau_1, u_1), \ldots, Q_N(\tau_N, u_N))$ as inputs. QMIX outperforms VDN on various hard SCII challenges.

QTRAN [42]. QTRAN relaxes the constraints of VDN and QMIX and factorize the $Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{u})$ with transformation:

$$\sum_{i=1}^{N} Q_i\left(\tau_i, u_i\right) - Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{u}) + V_{\text{jt}}(\boldsymbol{\tau}) = \begin{cases} 0 & \boldsymbol{u} = \overline{\boldsymbol{u}} \\ \geq 0 & \boldsymbol{u} \neq \overline{\boldsymbol{u}} \end{cases}$$

where $V_{\text{jt}}(\boldsymbol{\tau}) = \max_{\boldsymbol{u}} Q_{\text{jt}}(\boldsymbol{\tau}, \boldsymbol{u}) - \sum_{i=1}^{N} Q_i\left(\tau_i, u_i\right)$. QTRAN outperforms QMIX and VDN on matrix game and struggles on high-dimension SCII tasks.

QPLEX [53]. Wang et al. utilize the dueling structure $Q = V + A$ [55] and propose the following factorization:

Joint Dueling: $Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}) = V_{tot}(\boldsymbol{\tau}) + A_{tot}(\boldsymbol{\tau}, \boldsymbol{u})$ and $V_{tot}(\boldsymbol{\tau}) = \max_{\boldsymbol{u}'} Q_{tot}\left(\boldsymbol{\tau}, \boldsymbol{u}'\right)$
Individual Dueling: $Q_i\left(\tau_i, a_i\right) = V_i\left(\tau_i\right) + A_i\left(\tau_i, a_i\right)$ and $V_i\left(\tau_i\right) = \max_{a_i'} Q_i\left(\tau_i, a_i'\right)$

WQMIX [33]. QMIX restricts the joint action Q-values to be a monotonic mixing of each agent's Q values. However, such restriction prevents it from representing value functions in which nonmonotonicity occurs. To address this issue, Rashid et al. [33] propose the following QMIX operator Π_w by imposing weights $w(\boldsymbol{\tau}, \boldsymbol{u})$ on the TD-error:

$$\Pi_w Q := \underset{q \in \mathcal{Q}^{mix}}{\operatorname{argmin}} \sum_{\mathbf{u} \in \mathbf{U}} w(\boldsymbol{\tau}, \boldsymbol{u})(Q(\boldsymbol{\tau}, \boldsymbol{u}) - q(\boldsymbol{\tau}, \boldsymbol{u}))^2$$

LH-IQN [23]. LH-IQN was built on top of Hysteretic Q-Learning [27] and Lenient Q-Learning [31] with IQN to trains independent learners. However, it performs poorly in complex scenarios, the reason maybe LH-IQN is an independent learning method similar to IQL.

COMA [10]. Foerster et al. consider the credit assignment problem in policy graident methods and propose the following counterfactual advantage function:

$$A^i(\boldsymbol{\tau}, \mathbf{u}) = Q^{\text{tot}}(\boldsymbol{\tau}, \mathbf{u}) - \sum_{u'^i} \pi^i\left(u'^i \mid \tau^i\right) Q^{\text{tot}}\left(\boldsymbol{\tau}, \left(\mathbf{u}^{-i}, u'^i\right)\right)$$

where $A^i\left(\boldsymbol{\tau}, u^i\right)$ is a separate baseline for each agent that uses the centralised critic to reason about counterfactuals where only agent i's action changes.

DOP [54]. Wang et al. find the centralized-decentralized mismatch (CDM) in MADDPG [22]. Inspired by the successes of value decomposition in VDN and QMIX, they propose Decomposed Off-Policy policy gradient (DOP) method which decomposes the centralized Q value with a linear combination of individual Q values:

$$Q^{\text{tot}}(\boldsymbol{\tau}, \mathbf{u}) = \sum_i k_i(\boldsymbol{\tau}) Q_i\left(\boldsymbol{\tau}, u_i\right) + b(\boldsymbol{\tau})$$

where k_i is the weight and b is the bias which takes the global observation-action histories as inputs.

Table 4: Baseline algorithms

Algorithms	Brief Description
IQL [46]	Independent Q-learning
VDN [43]	Value decomposition network
COMA [10]	Counterfactual Actor-critic
QMIX [35]	Monotonicity Value decomposition
QTRAN [42]	Value decomposition with linear affine transform
LH-IQN [23]	Likelihood Hysteretic with IQN (independent learning)
DOP [54]	Policy Gradient method with a linear combination of individual Q values
WQMIX [33]	Weighted TD-error of QMIX in centralized training.
QPLEX [53]	Using a dueling network to represent the Q^{tot}

E Results

E.1 Addtional Results.

We show the test return value in Figure 14 and RMIX outperforms baseline methods as well. As shown in Figure 15, RDN shows convincing test return value over VDN on three scenarios.

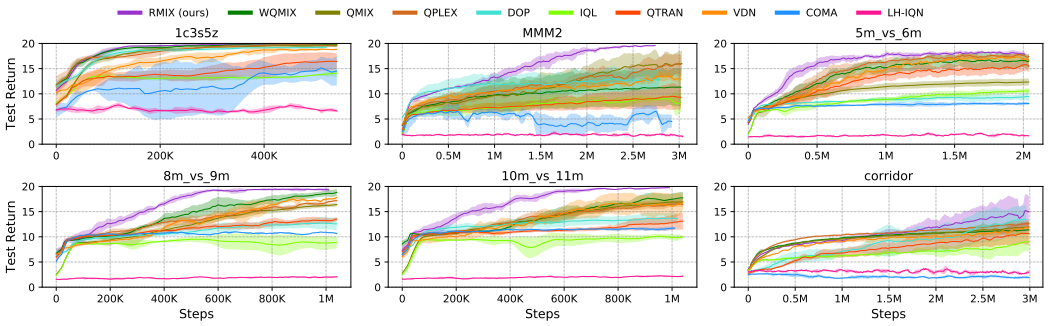

Figure 14: Test return for six scenarios.

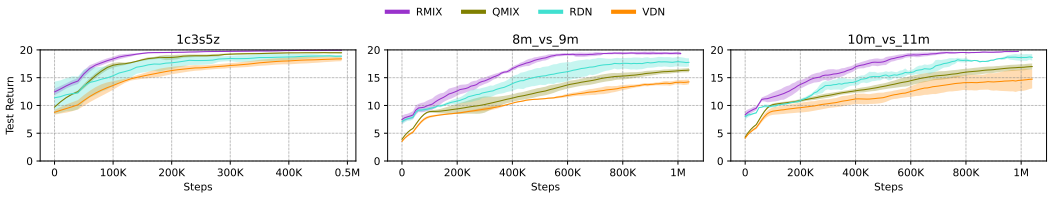

Figure 15: Test return of RMIX, RDN, VDN, QMIX on three scenarios.

E.2 Results Analysis

We use the trained model of RMIX and run the model to collect one episode data including game replay, states, actions, rewards and α values. As shown in Fig. 16, the first row shows the rewards of one episode and the second row shows the α value each agent predicts per time step. There are eight scenes in the third row to show how agents learn time-consistency α values. Scenes in Fig. 16 are screenshots from the game replay.

We use trajectories of agent 0, 1 and 3 as examples in Fig. 16 for better visualization. Number in the circle indicates the index of the agent. Scene (1): one episode starts, 6 Zealots consist of RMIX agents and 24 Zerglings compose enemies; Scene (2): in order to win the game, agent 3 draws the attention of enemies and goes to the other side of the arena. The α value is 0.3 at step 2. Many enemies are chasing agent 3. The rest agents are combating with fewer number enemies; Scene (3-4): at step 14, agent 3 is at the corner of the battlefield the α is decreasing. As being outnumbered, agent 3 quickly dies and the α is zero; Meanwhile, agent 0 and 1 show similar α

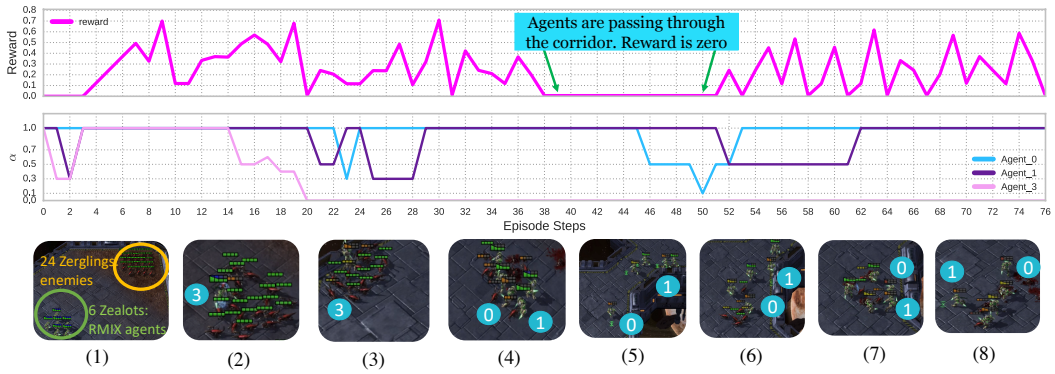

Figure 16: Results analysis of RMIX on corridor.

values as they are walking around and fighting with enemies from time step 22 to 30. Scene (5): agent 0 (low health value) is walking through the corridor alone to draw enemies to come over. To avoid being killed, α values are low (step 46-50) which means the policy is risk-averse. From step 38 and step 51, reward is zero; Scene (6): agent 0 is facing many enemies and luckily its teammates are coming to help. So, the α is increasing (step 50); Scene (7): as there are many agents around and the number of enemies is small, agents are going to win. Agent 0 and agent 1 walk outside the range of the observations of enemies in order to survive. The α values of agent 0 and agent 1 are 1 (risk-neutral); Scene (8): agents win the game. The video is available in the folder of supplementary materials. Interestingly, the result shows emergent cooperation strategies between agents at different step during the episode, which demonstrates the superiority of RMIX. We provide a link of the video for the result analysis discussed above. Readers can click this link: `https://www.dropbox.com/s/lrtdpkb45fs81ah/video.mp4?dl=0` to watch the video.