RL Final

Team 23

PH Lai YH Wu CC Hsu

December 1, 2025

Main Goal

Design a Multi-Agent Reinforcement Learning algorithm that converges to a **Risk-Averse Quantal Response Equilibrium (RQE)**.

Formal Definition of RQE

Definition 5 (Mazumdar et al., 2025)

A **Risk-Averse Quantal Response Equilibrium (RQE)** is a joint strategy $\pi^* \in \mathcal{P}$ such that for each player i:

$$\tilde{f}_i(\pi_i^*, \pi_{-i}^*) \leq \tilde{f}_i(\pi_i, \pi_{-i}^*) \quad \forall \pi_i \in \Delta_{A_i}$$

This equilibrium exists in a modified game where each player's cost $\tilde{f}_i$ is their risk-adjusted cost f_i plus a penalty for deviating from bounded rationality:

$$\tilde{f}_i(\pi_i, \pi_{-i}) = \underbrace{f_i(\pi_i, \pi_{-i})}_{\text{Risk-Averse Cost}} + \underbrace{\varepsilon_i \nu_i(\pi_i)}_{\text{Bounded Rationality}}$$

where ν_i is a strictly convex regularizer.

Motivation: Why RQE?

- **Tractability:** Standard solutions like Nash Equilibrium are often computationally intractable. RQE provides a tractable alternative.
- **Realism:** RQE is inspired by behavioral economics, creating agents that better model human-like decision-making by accounting for risk aversion and imperfect optimization.
- **Applications:** These properties are valuable in real-world strategic interactions, including **autonomous driving**, **finance**, and decentralized control of the **power grid**.

Explaining RQE

RQE = Risk Aversion + Bounded Rationality

RQE combines two key concepts from behavioral economics:

- **Risk Aversion:** Agents prefer "safer" outcomes and may avoid high-risk, high-reward paths. We model this using a **convex risk measure**.

Explaining RQE

RQE = Risk Aversion + Bounded Rationality

RQE combines two key concepts from behavioral economics:

- **Risk Aversion:** Agents prefer "safer" outcomes and may avoid high-risk, high-reward paths. We model this using a **convex risk measure**.

- **Bounded Rationality:** Agents are not perfect optimizers; they make mistakes and explore. We model this using an **entropy regularizer**.

Measuring Convergence: Risk-Averse NashConv

Standard NashConv is not the right metric for RQE. Instead, we use a generalized version called **Risk-Averse NashConv (RA-NashConv)**.

- It still measures the collective "regret," but it's a special kind of regret.
- **Risk-Averse Regret**: How much more *risk-adjusted utility* an agent could get by unilaterally changing its policy.
- This correctly measures the incentive to deviate for risk-averse, boundedly-rational agents. Our goal is to minimize this value.

Formal Definition of RA-NashConv

First, we define the **Risk-Averse Regret** for player i as the gain they get from switching to a risk-averse best-response policy π_i':

$$\widehat{\delta}_i^{\mathrm{RA}}(\pi) = \widehat{J}_i^{\mathrm{RA}}(\pi_i', \pi_{-i}) - \widehat{J}_i^{\mathrm{RA}}(\pi_i, \pi_{-i})$$

where $\widehat{J}_i^{\mathrm{RA}}$ is the estimated risk-averse objective function.

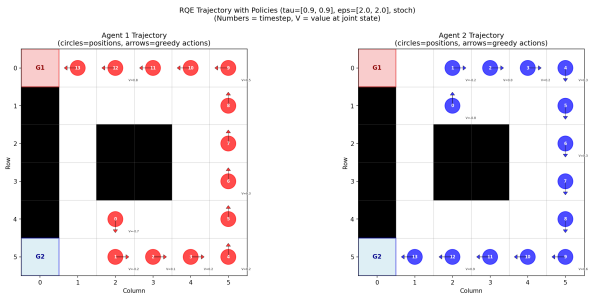
The **Risk-Averse NashConv** is the sum of these regrets over all players:

$$\mathrm{RA-NashConv}(\pi) = \sum_{i=1}^{n} \widehat{\delta}_i^{\mathrm{RA}}(\pi)$$

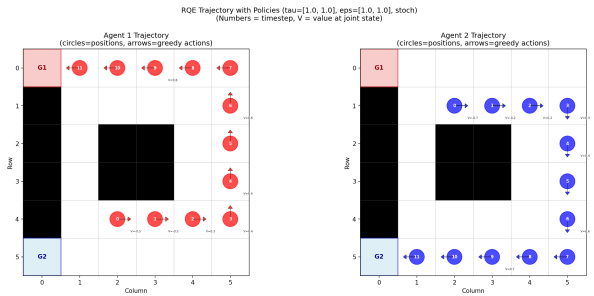
The Environment: Multi-Agent Cliff Walk

- A grid world where agents must navigate to a goal while avoiding a "cliff".
- **Multi-Agent Risk**: When agents are close, the environment becomes more stochastic, increasing the risk of falling off the cliff.

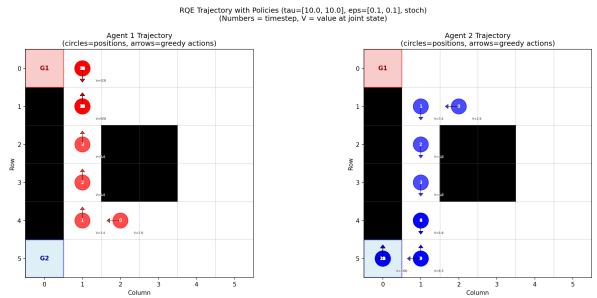
Effect of Risk Aversion (τ) on Solver Trajectories



Low Risk ($\tau = 0.9$)



Medium Risk ($\tau = 1.0$)

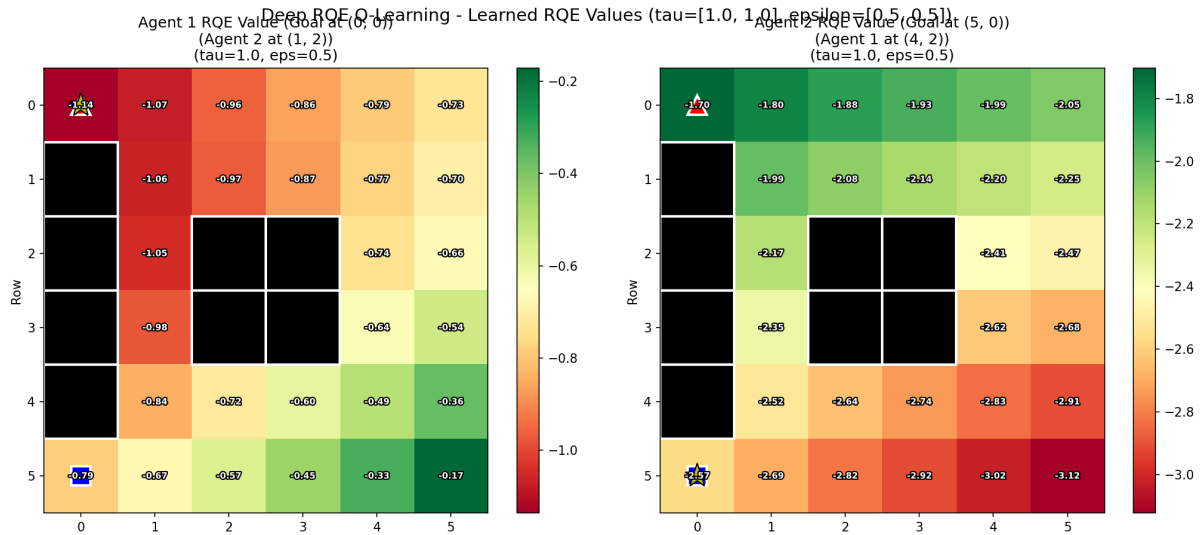


High Risk ($\tau = 10.0$)

Our Implementation: Deep RQE Q-Learning

How It Works:

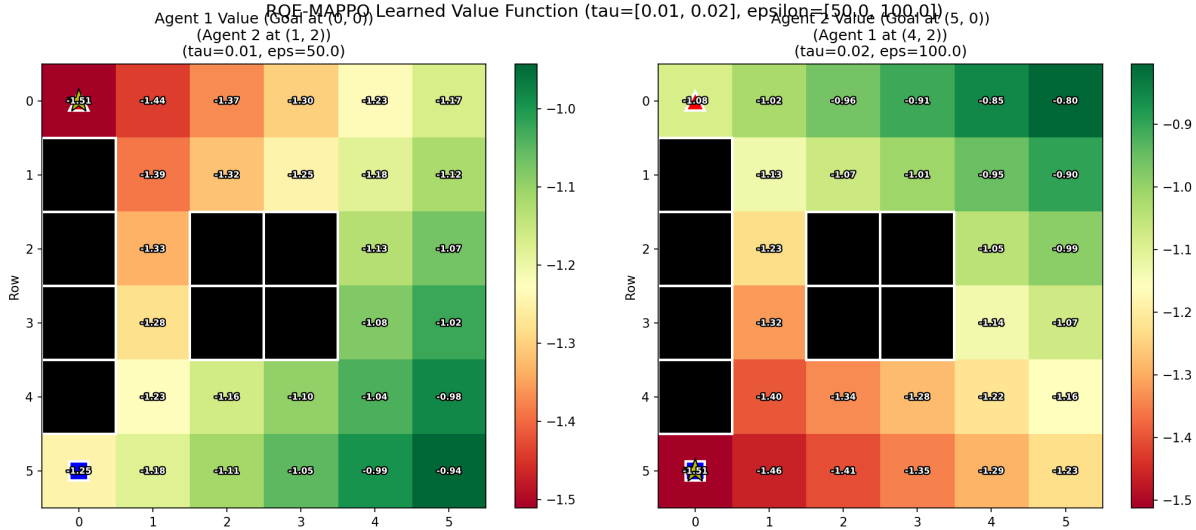
1. Each agent learns a deep **Q-Network** which outputs a full payoff matrix for the current state.
2. At each decision step, the agents feed their learned payoff matrices into the **RQE Solver**.
3. The solver computes the risk-averse equilibrium policy for the current state.
4. Agents then select actions based on this computed equilibrium policy.



Our Implementation: RQE-MAPPO

How It Works:

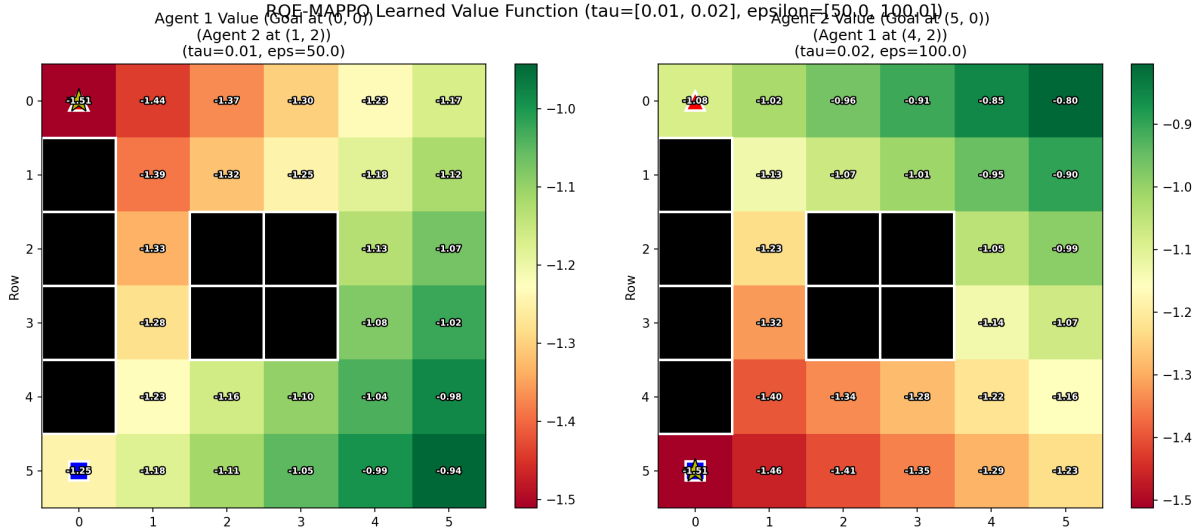
1. Learn return distribution with a **Distributional Critic**.



Our Implementation: RQE-MAPPO

How It Works:

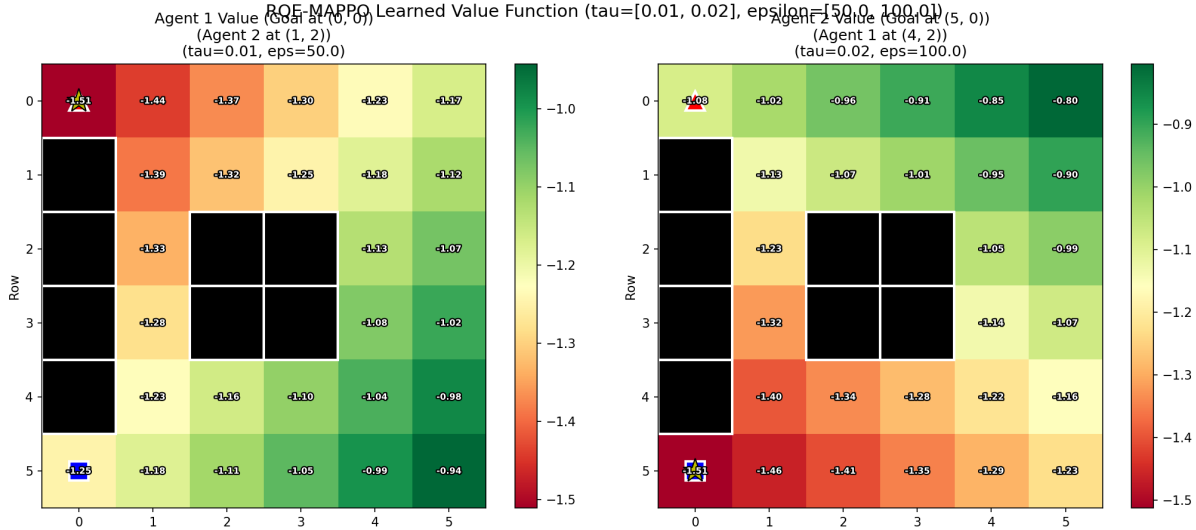
1. Learn return distribution with a **Distributional Critic**.
2. Calculate **Risk-Averse Value** (V_{RA}) from the distribution.



Our Implementation: RQE-MAPPO

How It Works:

1. Learn return distribution with a **Distributional Critic**.
2. Calculate **Risk-Averse Value** (V_{RA}) from the distribution.
3. Compute **Risk-Aware Advantage** ($\hat{A}_{RA}$) using V_{RA}.



Our Implementation: RQE-MAPPO

How It Works:

1. Learn return distribution with a **Distributional Critic**.
2. Calculate **Risk-Averse Value** (V_{RA}) from the distribution.
3. Compute **Risk-Aware Advantage** ($\hat{A}_{RA}$) using V_{RA}.
4. Update actor with PPO using the risk-aware advantage: $\mathcal{L}_{\text{PPO}}(\hat{A}_{RA})$.