A Environment Details

A.1 Iterated Games

In Iterated Games, agents' episode length is 10. Action of each agent is a 1-dimensional vector, $a_i = \{h_j, j \in \{0, 1\}\}$, where $h_j = 0$ denotes taking Stag action and $h_j = 1$ denotes taking Hare action. Agents can condition their actions on past history. Similar like [12], we consider the history length 1. So the observation of each agent is actions taking by itself and its opponent in the last round. We initialize the observation at the first round as $o_i = [-1, -1]$ for both agents.

A.2 Monster-Hunt

In Monster-Hunt, agents' episode length is 20. Each agent can choose to move one step in any of the four directions (Up, Down, Left, Right) or standstill, so the action of each agent is a 5-dimensional one-hot vector. The position of each agent can not exceed the border of 5-by-5 grid, where action execution is invalid. One Monster and two apples respawn in the different grids at the initialization. If an agent eats (move over in the grid world) an apple, it can gain 2 points. Sometimes, two agents try to eat the same apple, then they receive 2 points respectively. Catching the monster alone causes an agent lose 10 points, but if two agents catch the stag simultaneously, each agent can gain 5 points. At each time step, the monster and apples will respawn randomly elsewhere in the grid world if they are wiped. In addition, the monster chases the agent closest to it at each timestep. Each agent's observation o_i is a 10-dimensional vector and formed by concatenating its own position p_i , the other agent's position p_{1-i} , monster's position p_m and sorted apples' position p_{a_0} , p_{a_1} , i.e., $o_i = (p_i, p_{1-i}, p_m, p_{a_0}, p_{a_1}), i \in \{0, 1\}$ where $p = (x, y)$ denotes the 2-dimensional coordinates in the gridworld.

A.3 Escalation

In Escalation, agents' episode length is 30. Two agents appear randomly and one grid lights up at the initialization. If two agents step on the lit grid simultaneously, each agent can receive 1 point, and the lit grid will go out with an adjacent grid lighting up. Both agents can receive 1 point again if they step on the next lit grid together. But if one agent steps off the path, the other agent will lose $1.5L$ points, where L is the current length of stepping together, and the game is over. Another option is that two agents choose to step off the path simultaneously, neither agent will be punished, and the game continues. As the length L of stepping together increases, the cost of betrayal increases linearly. Each agent can choose to move one step in any of the four directions (Up, Down, Left, Right) or standstill, so the action of each agent is a 5-dimensional one-hot vector. The observation o_i of agent i is composed of its own position p_i, the other agent's position p_{1-i} and the lit grid's position p_{lit}, i.e., $o_i = (p_i, p_{1-i}, p_{lit}), i \in \{0, 1\}$, where $p = (x, y)$ denotes the 2-dimensional coordinates in the gridworld.

B Implementation Details

B.1 Architecture

We split the Q value network into two parts: feature extractor $\mathcal{E}_\phi$ and decision maker $\mathcal{D}_{\psi_a}$. The auxiliary opponent modeling task shares a common feature extractor $\mathcal{E}_\phi$ with the value network, and the supervised prediction head is $\mathcal{D}_{\psi_s}$. The feature extractor $\mathcal{E}_\phi$ consists of three linear layers: the input and output layers and one hidden layer with ReLU activation function. The output from $\mathcal{E}_\phi$ is the input to both $\mathcal{D}_{\psi_a}$ and $\mathcal{D}_{\psi_s}$. Both $\mathcal{D}_{\psi_a}$ and $\mathcal{D}_{\psi_s}$ consist of two linear layers: the input and output layers with ReLU activation function, respectively. The $\mathcal{E}_\phi$'s output layer has 50 nodes while other hidden layers consist of 128 nodes. We use layer normalization on $\mathcal{E}_\phi$'s outputs in Iterated Games and Escalation but ReLU activation in Monster-Hunt to stabilize training. We use Adam optimizer with learning rate 3×10^{-4} for GRSP in all experiments.

Our codes can be found at this link: `https://anonymous.4open.science/r/GRSP-8DEC`.

B.2 Baselines

IAC IAC is a decentralized multi-agent policy gradient method in which each agent learns policy and value networks based on his local observations and treat other agents as part of the environment. The IAC implementation is based on the following well-known repository: `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail/tree/master/a2c_ppo_acktr`.

LIAM LIAM[13] learns latent representations of the modeled agents from the local information of the controlled agent using encoder-decoder architecture. The modeled agent's observations and actions are utilized as reconstruction targets for the decoder, and the learned latent representation conditions the policy of the controlled agent in addition to its local observation. The policy and model are optimized based on A2C algorithm. The LIAM implementation is based on its official open-sourced code: `https://github.com/uoe-agents/LIAM`.

Centralized training and decentralized execution (CTDE) methods. MADDPG[30] and MAPPO[31] are two kinds of multi-agent policy gradient methods which improve upon decentralized RL by adopting an actor-critic structure and learning a centralized critic. We implement the two algorithms based on their open-sourced codes and perform a limited grid-search over certain hyper-parameters, including learning rate, entropy bonus coefficient and buffer size (MADDPG). The MAPPO implementation is based on its official open-sourced code: `https://github.com/marlbenchmark/on-policy`. The MADDPG implementation is based on its official open-sourced code: `https://github.com/openai/maddpg`. It is noteworthy that we do not use the gumbel-max trick to optimize the actor network, instead, the critic's inputs contain the actor network's outputs after soft-max activation, so we can update the actor network's parameters using gradients from the critic loss, which empirically performs better in our experiments.

B.3 Hyperparameters

We detail all GRSP hyperparameters used in four multi-agent environments in Table 2 and Table 3.

B.3.1 Iterated Games

Table 2: GRSP hyperparameters used in Iterated Games.

Hyperparameters	Value in ISH	Value in IPD
quantile num	64	64
learning rate	3e-4	3e-4
learning rate (test time adaptation)	7e-3	7e-3
batch size	120	120
buffer size	40000	40000
discount rate (γ)	0.99	0.99
c_1	20	10
c_1 (test time adaptation)	5	5
c_2	50	100
c_2 (test time adaptation)	0	0
risk-level (λ)	-0.9	-1.0
risk-level (λ) (test time adaptation)	0.5	0.2
n_step	1	1
update freq	1	1

Table 3: GRSP hyper-parameters used in Grid-World.

Hyper-parameters	Value in M-H	Value in Esalation
quantile num	64	64
learning rate	3e-4	3e-4
learning rate (test time adaptation)	7e-3	7e-3
batch size	120	120
buffer size	10000	36000
discount rate (γ)	0.99	0.99
c_1	80	10
c_1 (test time adaptation)	5	5
c_2	80	100
c_2 (test time adaptation)	0	0
risk-level (λ)	-0.9	-1.75
risk-level (λ) (test time adaptation)	-0.5	-0.5
n_step	3	3
update freq	1	1

B.3.2 Grid-World Games

C Additional Experiment Results

C.1 Reward Shaping

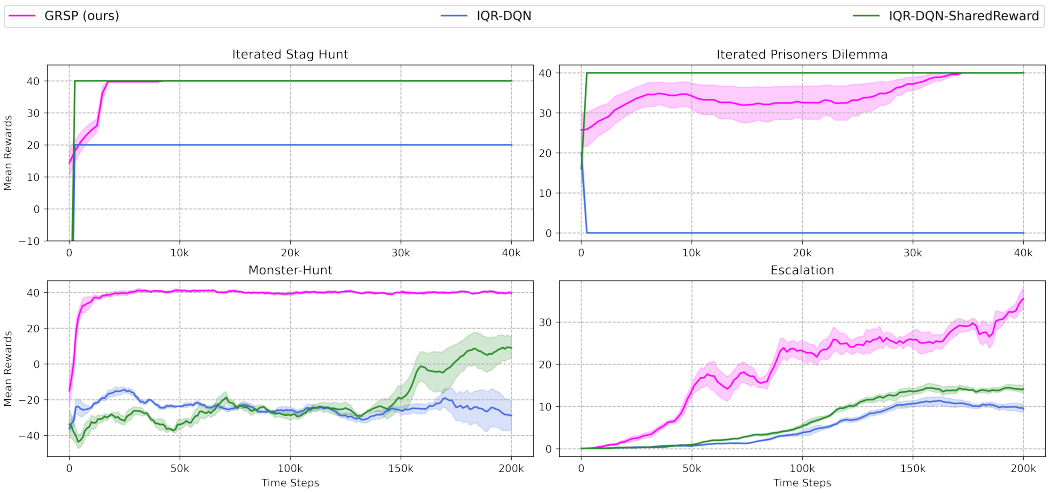

Figure 8: Mean evaluation returns for GRSP, IQR-DQN and IQR-DQN-SharedReward on four multi-agent environments.

We further compare GRSP with IQR-DQN and its reward shaping version, as shown in Fig. 8. Specifically, IQR-DQN-SharedReward means that we utilize QR-DQN and global rewards, i.e., $\frac{1}{N}\sum_{i=1}^{N} r_i$, to train each agent. Global rewards can force the agent to consider its own and other agents' payoffs equally at each time step, thus non-cooperative agents will be punished and prosocial agents have much higher probabilities to achieve mutual coordination [16]. Fig. 8 shows that agents with shared rewards converge to coordination strategy faster than GRSP agents in iterated games. However, in more complex grid-world games, GRSP outperforms other methods significantly in sample efficiency and asymptotic performance, indicating that compared with reward shaping, our risk-seeking bonus is a more stable and efficient way to encourage agents to achieve mutual coordination without restrictive assumptions.

C.2 Different Risk

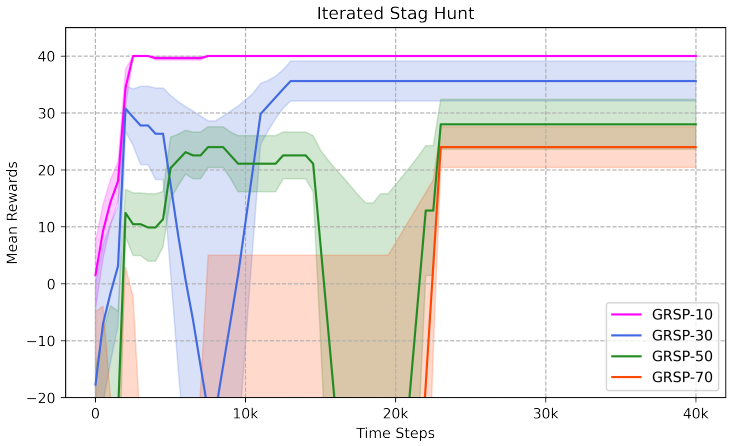

Figure 9: Mean evaluation returns for GRSP on four Iterated Stag Hunt environments which have different punishments.

Fig. 9 shows mean evaluation returns for GRSP on four Iterated Stag Hunt environments which have different punishments, e.g., GRSP-10 means that if the agent chooses to hunt stag while the other agent hunts hare, he will receive a punishment reward of -10. Higher punishment means higher risk and smaller probabilities to converge to coordination strategies. As shown in Fig. 9, our method is robust to different risks in the environment, and it is noteworthy that we do not adjust any other hyperparameters.