

Recursive Reasoning Graph for Multi-Agent Reinforcement Learning

Xiaobai Ma¹, David Isele², Jayesh K. Gupta¹, Kikuo Fujimura², Mykel J. Kochenderfer¹

¹Stanford University

²Honda Research Institute US

maxiaoba@stanford.edu, disele@honda-ri.com, jkg@cs.stanford.edu, kfujimura@honda-ri.com, mykel@stanford.edu

Abstract

Multi-agent reinforcement learning (MARL) provides an efficient way for simultaneously learning policies for multiple agents interacting with each other. However, in scenarios requiring complex interactions, existing algorithms can suffer from an inability to accurately anticipate the influence of self-actions on other agents. Incorporating an ability to reason about other agents' potential responses can allow an agent to formulate more effective strategies. This paper adopts a recursive reasoning model in a centralized-training-decentralized-execution framework to help learning agents better cooperate with or compete against others. The proposed algorithm, referred to as the Recursive Reasoning Graph (R2G), shows state-of-the-art performance on multiple multi-agent particle and robotics games.

Introduction

Recent advances in deep reinforcement learning have shown impressive success in single-agent scenarios including games (Hessel et al. 2018) and robotics (Johannink et al. 2019). However, many real-world problems involve interactions between multiple agents with limited information exchange, where multi-agent reinforcement learning (MARL) is needed (Vinyals et al. 2019; Wen et al. 2020; Yang et al. 2019). The simplest approach to MARL is independent reinforcement learning (Tan 1993), which trains each agent independently by treating the other learning agents as part of the environment. Unfortunately, these methods often have stability issues since the environment dynamics from the perspective of each learning agent is non-stationary due to the learning of other agents (Mohseni-Kabir, Isele, and Fujimura 2019). To account for this, we can build behavior models of other agents and thus separate this unstable component out of the environment dynamics. Many decentralized MARL algorithms follow this idea (He et al. 2016; Wen et al. 2018; Shen and How 2021).

However, modeling other agents' could be difficult as they are continuously learning (Albrecht and Stone 2018). In the *centralized-training-decentralized-execution* (CTDE) framework, such problems are avoided by allowing the learning algorithm to have direct access to all agents' internal information (policy, value network, etc.) at training

time. Algorithms adopting this setting often demonstrate better interactive strategies (Sunehag et al. 2018; Rashid et al. 2018; Gupta, Egorov, and Kochenderfer 2017). However, most existing algorithms are limited due to their lack of ability at explicitly considering the reasoning process of other agents (Lowe et al. 2017; Yang et al. 2019; Wei et al. 2018). They therefore cannot develop plans that account for the influence of their behavior changes on opponents' response (Wen et al. 2018).

Reasoning about others' reasoning, referred to as *recursive reasoning*, is important for humans interacting with each other. Von Der Osten, Kirley, and Miller (2017) use recursive nested beliefs to predict the actions of other agents, but their method is limited to few state variables and discrete action spaces. The PR2 method (Wen et al. 2018) applies recursive reasoning by explicitly modeling the other agent's response to the ego agent's action. However, as PR2 adopts decentralized-training, each learning agent uses a single model to learn the response of all the opponents by assuming that the other agents share the same reward as itself, which limits its application to fully-cooperative games. Li et al. (2019) augment the multi-agent deep deterministic policy gradient (MADDPG) (Lowe et al. 2017) using adversarial training with linear approximations to model the minimax optimization, which is limited to zero-sum games.

In this paper, we propose to augment the existing CTDE framework with auxiliary central actors to learn the optimal response of each agent given opponents' actions. Based on this, we build a graph structure to model the recursive reasoning procedure between interacting agents. This graph structure allows us to model the relationships between agents and explicitly consider their responses. The recursive actions of each agent are efficiently sampled and shared through message passing in the graph. The proposed method, which we call the Recursive Reasoning Graph (R2G), works in both competitive and cooperative games. Our contributions are as follows:

- We propose R2G, a multi-agent reinforcement learning framework that explicitly models the recursive reasoning process of the agents in general-sum games.
- We augment the existing centralized-training-decentralized-execution algorithms with centralized actors and graph-like message passing to efficiently train the learning agents under R2G framework.

- We demonstrate state-of-the-art performance on multiple Particle World (Lowe et al. 2017) and RoboSumo (Al-Shedivat et al. 2018) environments with complex reward structure and non-trivial equilibrium.

Background

Markov Game

A Markov Game (MG) is commonly used to model multi-agent reinforcement learning problems (Littman 1994). A MG is specified by $(S, \{A^i\}_{i=1}^n, T, \{r^i\}_{i=1}^n, \gamma, s_0)$, where n is the number of agents; S is the state space containing the state for all agents; A^i represents the action space for agent i ; $T : S \times \prod_{i=1}^n A^i \rightarrow S$ represents the (stochastic) transition model conditioned on the current state as well as the actions of all agents; $r^i : S \times \prod_{i=1}^n A^i \times S \rightarrow \mathbb{R}$ represents the reward for agent i ; and s_0 represents the initial state distribution of all agents.

The learning objective in MGs is to get a set of policies $\{\pi^i\}_{i=1}^n$, where for each agent i , $\pi^i : S \rightarrow A^i$ maps the state to its action. However, unlike the clear optimization goal in the single-agent reinforcement learning, the concept of optimality in MARL is more complex. In fully cooperative games where all the agents share the same reward function, the optimization goal is to maximize the joint return for all agents. When agents have conflict of interests, the concept of equilibrium is introduced. A common objective is to find the Nash Equilibrium (NE), where all agents act in best response to each others' current strategy. However, a Nash Equilibrium might be difficult to compute and can diverge from human behavior (Wright and Leyton-Brown 2010).

Multi-Agent Actor-Critic

The actor-critic framework is a common training structure in single-agent reinforcement learning. In this framework, a critic, $Q_\theta(s, a)$, is trained to estimate the return value of the state-action pair (s, a) with the loss $J_{Q_\theta} = \mathbb{E}_{s, a \sim \mathcal{D}}[(Q_\theta(s, a) - \hat{Q})^2]$, where \mathcal{D} is the replay buffer storing the exploration experiences and \hat{Q} is an empirical estimate of the return value. An actor, $\pi_\phi(s)$, is trained to maximize the return value with the loss $J_{\pi_\phi} = \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\phi(s)}[-Q_\theta(s, a)]$. Additional terms like the policy entropy could also be added to J_{π_ϕ} to improve the training (Haarnoja et al. 2018). The actor-critic framework could be naturally generalized to the multi-agent setting with centralized training. A central critic, $Q_\theta^i(s, a^i, a^{-i})$, for each agent i , where a^{-i} indicates the actions of agents except agent i , is trained to estimate the return value of agent i given the state and the joint-action; i.e., $J_{Q_\theta^i} = \mathbb{E}_{s, a^i, a^{-i} \sim \mathcal{D}}[(Q_\theta^i(s, a^i, a^{-i}) - \hat{Q})^2]$. Each actor, $\pi_\phi^i(s)$, is then trained to minimize the loss $J_{\pi_\phi^i} = \mathbb{E}_{s \sim \mathcal{D}, a^i \sim \pi_\phi^i(s)}[-Q_\theta^i(s, a^i, a^{-i})]$.

There are multiple choices for sampling a^{-i} during the training of actors. For example, in MADDPG (Lowe et al. 2017), a^{-i} is from the stored experiences in the replay buffer. The stored actions are sampled from the policies earlier in the training or from an exploration strategy. In this case, the actor is actually learning the best response with

respect to the action distribution stored in the replay buffer, whose performance largely depends on the exploration strategy.

An alternative is to sample a^{-i} directly from the other agents' current policies. Unfortunately, this can lead to the problem of relative overgeneralization (Wei and Luke 2016). During training and exploration, a suboptimal Nash Equilibrium (NE) could be preferred over the optimal NE, when each agent's action in the suboptimal NE is estimated with a higher return against the current action distributions from the other agents. Wei et al. (2018) address this problem by having each agent learn the optimal joint action of all agents. However, this assumes that all agents are optimizing the same reward function and thus is limited to fully cooperative games.

Sampling from the other agents' current policy could also lead to oscillatory learning as all agents learn concurrently, and the best response at this iteration might be suboptimal in the next iteration. For example, in the ROCK-PAPER-SCISSORS game, if we know the opponent's current policy is playing ROCK with probability 1, then our best response is to play PAPER. However, when the opponent finds out our new policy at the next iteration, they would change to play SCISSOR. As one player changes its policy completely at each iteration, the equilibrium where each player plays the three options uniformly randomly could never be reached.

Methods

We propose to use a recursive reasoning model to sample the opponents' response during policy training in multi-agent actor-critic.

Logit Level-K Reasoning

The recursive reasoning refers to the process of reasoning about the other agent's reasoning process during decision making. It allows the ego agent to consider the potential change in strategy of other agents instead of treating them as fixed. A classic model of recursive reasoning is the logit level- k model: At level 0, all agents choose their actions based on some base policies, $\pi^{(0)}$. At each level k , each agent chooses the best policy by assuming the others follow their level $k - 1$ policies. In multi-agent RL, a natural level-0 policy is the agent's current policy, i.e. $\pi^{i, (0)} = \pi^i$. Given the actions of other agents at level $k - 1$: $a^{-i, (k-1)}$, the best level- k action for agent i should be

$$a^{i, (k)} = \arg \max_{a^i} Q^i(s, a^i, a^{-i, (k-1)}) \quad (1)$$

where Q^i is the estimated return of agent i . This formulation holds for general-sum games.

Solving the optimization in eq. (1) is not trivial in continuous action spaces. Thus, we introduce a central actor $\pi_{c, \psi}^i(s, a^{-i})$ which learns the best response for agent i given state s and the other agents' actions a^{-i} . We train $\pi_{c, \psi}^i(s, a^{-i})$ to approximate $\arg \max_{a^i} Q_\theta^i(s, a^i, a^{-i})$ by minimizing the loss:

$$J_{\pi_{c, \psi}^i} = \mathbb{E}_{s, a^{-i} \sim \mathcal{D}, a^i \sim \pi_{c, \psi}^i(s, a^{-i})}[-Q_\theta^i(s, a^i, a^{-i})] \quad (2)$$

Recursive Reasoning Graph

With the help of π_c^i , we can calculate $a^{-i,(k)}$ using a message passing process in a recursive reasoning graph (R2G): $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V} = \{\pi_c^1, \dots, \pi_c^n\}$ contains the central actor node for each agent, and the edge set \mathcal{E} contains edges between all interacting agents. We use an undirected, fully-connected graph by assuming all agents are interacting with each other. A more sparse graph could be used if we have prior knowledge on the interacting structure between agents. The messages in the edges are the sampled actions $a^{i,(k)}$ from the central actors.

Figure 1 gives an illustration of the recursive reasoning graph in a 3-agent game. The initial level-0 actions are sampled from the individual policies:

$$a^{i,(0)} \sim \pi^i(s) \quad (3)$$

At each level $k \geq 1$, we have:

$$a^{i,(k)} \sim \pi_c^i(s, \text{AGG}_{j \in \mathcal{N}(i)} a^{j,(k-1)}) \quad (4)$$

where AGG is the aggregation function and \mathcal{N} is the node neighborhood function. In practice, we use concatenation for AGG. Thus, $\text{AGG}_{j \in \mathcal{N}(i)} a^{j,(k-1)}$ is interchangeable with a^{-i} in fully-connected graphs. The generalization from fully-connected graph to sparse graph is straightforward by limiting the message passing between central actors. An attention mechanism (Veličković et al. 2018) could also be used to dynamically learn the interaction structures.

At level k , each central actor node takes the input message of $a^{-i,(k-1)}$ and outputs its best response $a^{i,(k)}$. Thus, one complete message passing through the graph gives one-level up in the recursion. Hypothetically, the level of recursion could go to infinity. In the following discussion, we focus on the level-1 recursion. The comparison of different recursion levels are provided in the appendix.

The output level k actions are then fed to the central critics to calculate the estimated Q-values of the policy actions. The policy loss, $J_{\pi_\phi^i}$, is then formulated as the KL-divergence of the policy action distribution to the energy-based distribution represented by Q_θ^i :

$$J_{\pi_\phi^i} = \mathbb{E}_{a^{i,(0)} \sim \pi_\phi^i, a^{-i,(k)} \sim \mathcal{G}, s \sim \mathcal{D}} [\alpha^i \log(\pi_\phi^i(a^{i,(0)}|s)) - Q_\theta^i(s, a^{i,(0)}, a^{-i,(k)})] \quad (5)$$

where α^i is the temperature variable trained similarly as in SAC (Haarnoja et al. 2018).

For the training of the central critic Q_θ^i , we adopt the soft Bellman residual (Haarnoja et al. 2018):

$$J_{Q_\theta^i} = \mathbb{E}_{\mathcal{D}} [(Q_\theta^i(s, a^i, a^{-i}) - (r^i(s, a^i, a^{-i}) + \gamma \hat{V}(s'))))^2] \quad (6)$$

where the next state value $\hat{V}(s')$ is estimated by

$$\hat{V}(s') = \mathbb{E}_{a^{i,(0)} \sim \pi_\phi^i, a'^{-i,(k)} \sim \mathcal{G}} [Q_\theta^i(s', a^{i,(0)}, a'^{-i,(k)}) - \alpha^i \log \pi_\phi^i(a^{i,(0)}|s')] \quad (7)$$

where Q_θ^i is the delayed updated version of the critic network for agent i .

The central actor $\pi_{c,\psi}^i$ is trained by the loss given in eq. (2). We can omit the entropy term in $J_{\pi_{c,\psi}^i}$ since the exploration does not rely on $\pi_{c,\psi}^i$, and there always exists a deterministic optimal response given the strategies of other agents.

The training process is outlined in algorithm 1. The convergence of the algorithm are discussed in the appendix.

Algorithm 1: Recursive Reasoning Graph (R2G)

Result: Policy $\pi_\phi^i, \forall i \in 1, \dots, n$

Initialize $\pi_\phi^i, Q_\theta^i, Q_\theta^i$, and $\pi_{c,\psi}^i, \forall i \in 1, \dots, n$;

$\mathcal{D} \leftarrow$ empty reply buffer;

for each epoch do

 Collect exploration experiences using $\pi_\phi^{i=1:n}$;

 Add tuples $(s, a^{i=1:n}, r^{i=1:n}, s')$ to \mathcal{D} ;

for each training iteration do

 Sample $\{(s_j, a_j^{i=1:n}, r_j^{i=1:n}, s'_j)\}_{j=1}^B$ from \mathcal{D} with batch-size B ;

 Calculate $\{a_j^{i=1:n,(k)}\}_{j=1}^B$ at $\{s_j\}_{j=1}^B$ and

$\{a'_j^{i=1:n,(k)}\}_{j=1}^B$ at $\{s'_j\}_{j=1}^B$ from eq. (4);

for each agent i do

 Update π_ϕ^i with $J_{\pi_\phi^i}$ in eq. (5);

 Update $\pi_{c,\psi}^i$ with $J_{\pi_{c,\psi}^i}$ in eq. (2);

 Update Q_θ^i with $J_{Q_\theta^i}$ in eq. (6);

 Update target Q network as

$\hat{\theta} \leftarrow \tau \theta + (1 - \tau) \hat{\theta}$ with factor τ ;

end

end

end

The relative over-generalization problem in R2G is mitigated as learning the conditional best response in $\pi_c^i(s, a^{-i})$ is much easier than learning the marginal best response in $\pi^i(s)$, which makes the recursive actions a better approximation to the opponents' optimal response. Let $\pi^{i,(0)}$ denote the policy trained with opponents' current strategies, where $J_{\pi^{i,(0)}} = \mathbb{E}_{s \sim \mathcal{D}, a^{-i} \sim \pi_c^{-i,(0)}(s), a^i \sim \pi^i(s)} [-Q_\theta(s, a^i, a^{-i})]$.

We first notice that $J_{\pi^{i,(0)}}$ has a much larger variance than $J_{\pi_c^i}$ in eq. (2) due to the additional variance introduced by $a^{-i} \sim \pi_c^{-i,(0)}(s)$, where $\pi_c^{-i,(0)}$ could be both stochastic and learning. Without loss of generality, let us consider the influence of using $a^{2,(1)}$ in the training of π^1 . In a two-player game, from the above argument, $a^{2,(1)} \sim \pi_c^2(s, a^1)$ is a better approximation to the optimal response of player 2 to a^1 than $a^{2,(0)} \sim \pi_c^{2,(0)}(s)$. In games with more than 2 players, the above argument is not as clear, since $a^{2,(1)} \sim \pi_c^2(s, a^{-i})$ is best responding not only a^1 but also other players. However, in most games, the importance of different players is mutual, i.e., if the strategy of player 2 is important to player 1, then so is player 1 to player 2. Thus, if $a^{2,(1)}$ is important for the training of π^1 , then $\pi_c^2(s, a^{-i})$ should also consider a^1 more than other players' actions. Thus,

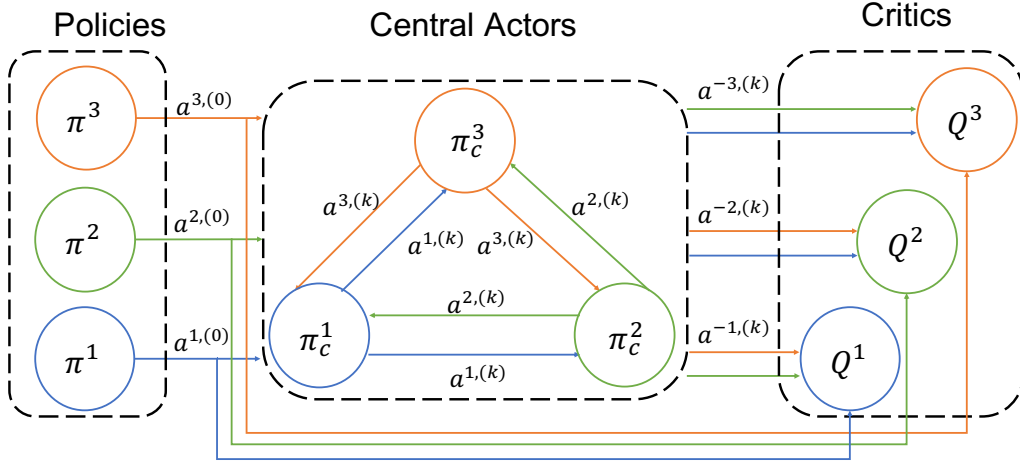


Figure 1: Recursive Reasoning Graph: The forward computation graph for Q values in the policy loss in a 3-agent game.

$Q_{\theta}^i(s, a^{i,(0)}, a^{-i,(1)})$ gives a generally closer approximation than $Q_{\theta}^i(s, a^{i,(0)}, a^{-i,(0)})$ on the return of the ego agent's action when opponents response optimally.

The oscillatory learning problem is also avoided by training π^i with $a^{-i,(1)}$. The change of other agents' actions due to the ego agent's action change is accounted in the policy loss using recursive actions.

A critical concern for an MARL algorithm is its scalability with respect to the number of agents, n . The computation complexity of most of the centralized-training-decentralized-execution approaches scales quadratically with the number of agents: each agent is trained through a centralized critic taking inputs of joint actions whose forward and backward propagation scales linearly with respect to n (for general neural network structures). While R2G adds an additional centralized component, the central actor, for each agent, the overall scalability does not degrade much as recursive actions are shared during policy training. By the message passing mechanism, the recursive action for each agent is only calculated once at each recursion level and is used repeatedly for $n - 1$ opponents. Thus, the overall training still scales quadratically with n and linearly with the recursion level, k .

Experiments

We compare R2G with several baselines spanning centralized and decentralized learning, as well as on-policy and off-policy algorithms applicable in the continuous action space context:

- PPO: The proximal policy optimization (PPO) (Schulman et al. 2017) is one of the most successful on-policy policy optimization algorithm for single-agent continuous control. In the experiments, PPO is applied to multi-agent problems through independent learning.
- COMA: The counterfactual multi-agent policy gradients (COMA) (Foerster et al. 2018) is originally developed for multi-agent on-policy learning in discrete action spaces. To extend it to continuous control problems, we use a

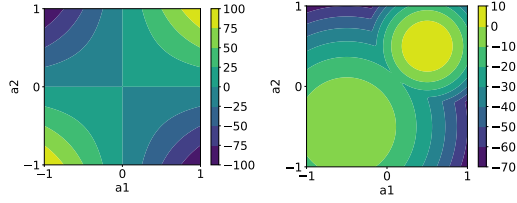
surrogate loss and likelihood ratio clipping similar as in PPO, as well as a Monte Carlo estimation for the counterfactual baseline.

- SAC: Soft actor-critic (Haarnoja et al. 2018) is a single-agent off-policy learning algorithm. Similarly as PPO, an independent learning scheme is applied for multi-agent problems.
- MADDPG: The multi-agent deep deterministic policy gradient (MADDPG) (Lowe et al. 2017) is a generalization from DDPG (Lillicrap et al. 2015) by introducing the central critic. In MADDPG, the a^{-i} is sampled from the replay buffer.
- MASAC: MASAC is a similar generalization from SAC as MADDPG. During the training, the a^{-i} is sampled from the current policies of the other agents. Thus, MASAC could be regarded as R2G without the recursive reasoning.
- PR2: Probabilistic recursive reasoning (PR2) (Wen et al. 2018) is a decentralized off-policy learning algorithm which also uses recursive reasoning in its training. The derivation of PR2 is based on the fully-cooperative assumption, while it also shows decent performance on general-sum games.

Differential Games

We first demonstrate the advantage of R2G analytically by focusing on two-player single-state Differential Games: the Zero Sum and the Max of Two (as in Wen et al. (2018)). The reward landscapes of the two games are shown in figs. 2a and 2b. More details of the environment as well as the experiments are given in the appendix.

For Zero Sum, the best action for agent 1 given a^2 is $a^{1,*} = 1 \cdot \text{sign}(a^2)$, and the best action for agent 2 given a^1 is $a^{2,*} = -1 \cdot \text{sign}(a^1)$. Thus, if both agents only consider their opponent's current strategy, they will have an oscillatory learning by alternating between 1 and -1 . Such behavior indeed appears in the training of PPO, COMA, and MASAC as shown in figs. 3a, 3b and 3e. However, if the



(a) Zero Sum ($r^1 = -r^2$) (b) Max of Two ($r^1 = r^2$)

Figure 2: Differential Game: Reward landscapes

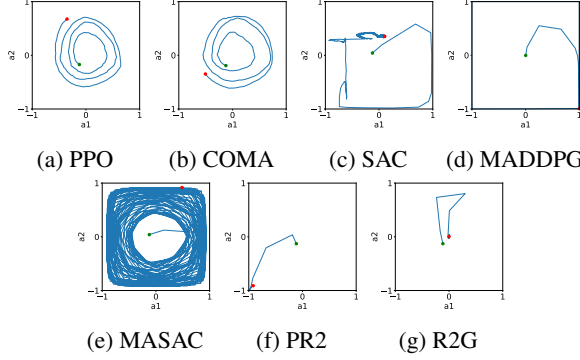


Figure 3: Zero Sum: Trajectories of most likely actions for different algorithms over 1000 iterations.

agent is aware of the potential response of the opponent to its own action change, their actions should converge to the (0,0) point, as any action deviates from 0 would get a lower reward when the opponent takes response. The central actor in R2G successfully captures this potential response as shown in fig. 5a. As a result, the R2G agents successfully converge to (0,0) after a short exploration as shown in fig. 3g. The action trajectory of SAC, MADDPG, and PR2 is quite chaotic due to changing opponent's behavior, the non-optimal opponent's action distribution in buffer, and the violation of cooperative assumption.

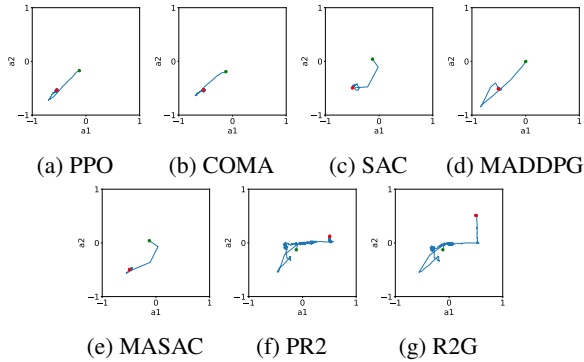


Figure 4: Max of Two: Trajectories of most likely actions for different algorithms over 1000 iterations.

For Max of Two, as a cooperative game, there is a local optimum at $(-0.5, -0.5)$ and a global optimum at $(0.5, 0.5)$. Due to the shape of the reward landscape, as the initial action distributions of agents are in the vicinity of $(0, 0)$, even with sufficient exploration, each agent's reward estimation

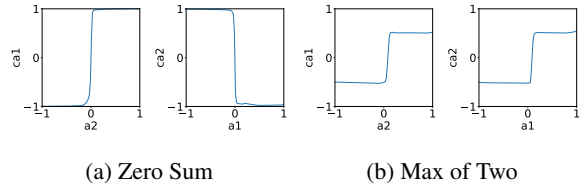


Figure 5: Differential Game: Central actors of R2G. Left: Most likely output of π_c^1 with respect to a^2 ; Right: Most likely output of π_c^2 with respect to a^1 . The central actors successfully learn the optimal response in all scenarios.

at -0.5 would be higher than that at 0.5 given the distribution of its opponent's action. Thus, the learning agents are easily attracted into the local optimum. As shown in figs. 4a to 4e, all methods except for PR2 and R2G suffers from such relative over-generalization problem. However, if the agent could predict the cooperative response of the opponent, e.g., when the agent acts at the global optimum, the opponent would also act at the global optimum, its value estimation of the global optimum action would be higher than the local optimum action. Thus, both agents would converge to the global optimum. The central actor in R2G successfully captures this potential response as shown in fig. 5b, and the R2G agents successfully converge to $(0.5, 0.5)$ as shown in fig. 4g. Similar behavior could also be observed in the training of PR2 as shown in fig. 4f.

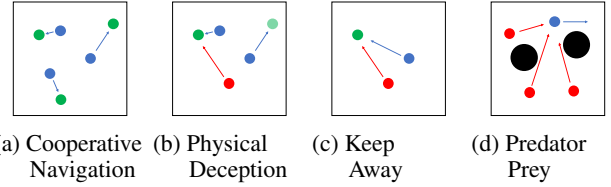


Figure 6: Particle World: Illustration of scenarios. The blue and red dots are good agents and adversaries. The green dots indicate the landmarks. In Physical Deception, a shadow green dot indicates the fake target landmark. The black dots are static obstacles. Arrows indicate the desired motion of agents.

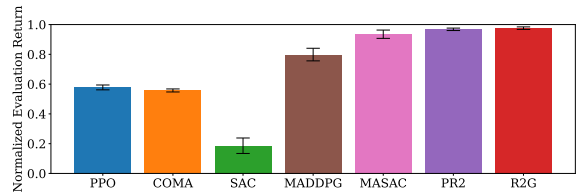


Figure 7: Cooperative Navigation: Normalized evaluation return of different algorithms.

Particle World

We further compare R2G against other baselines on several multi-player multi-state Particle World environments (Lowe et al. 2017) including:

- **Cooperative Navigation:** A cooperative game with 3 agents and 3 landmarks. Agents are rewarded to spread and cover all landmarks and penalized by collisions.



Figure 8: Particle World: Pairwise comparison of agent and adversary returns playing against the same opponents. The left bar indicates the baseline return, and the red bar indicates the R2G return.

- **Physical Deception:** A competitive game with 1 adversary, 2 good agents, and 2 landmarks. All agents could observe the positions of other agents and landmarks. There is only one true target landmark which is only known to good agents. The good agents are rewarded on how close any one of them is to the target landmark and penalized if the adversary is close to the target landmark. The adversary is rewarded on how close it is to the target landmark.
- **Keep Away:** A competitive game with 1 adversary, 1 good agent, and 1 landmark. The good agent is rewarded on how close it is to the landmark. The adversary is rewarded if it is close to the landmark, and if the agent is pushed away from the landmark.
- **Predator-Prey:** A competitive game with 3 adversaries (predators), 1 good agent (prey), and several obstacles. The good agent is faster and is rewarded to run away from the three adversaries. The adversaries are slower and are rewarded to catch the good agent.

We first compare the performance of different approaches on the Cooperative Navigation environment by evaluating the test time returns of agents trained together. The returns are averaged over 5 random seeds each with 1000 trajectories, and are reported using Min-Max normalization (Patro and Sahu 2015). As shown in fig. 7, R2G and PR2 achieve the best performance among all tested approaches. In this game, a positive return needs each agent cover one of the

landmarks. Thus, an agent is only motivated to approach one landmark when it expects the other agents also approaching other landmarks. The recursive reasoning in PR2 and R2G successfully capture this. The off-policy actor-critic methods (MADDPG, MASAC, PR2, R2G) are generally better than on-policy methods (PPO and COMA), this is likely due to the better data efficiency of actor-critic methods. The independent SAC performs much worse than the others, since the experience stored in the buffer is highly biased due to the learning of other agents.

To compare the performance of R2G on competitive games, we plot the pair-wise returns of different approaches playing against the same opponents. The results are shown in fig. 8. In each pair, we first evaluate the returns of different good agents playing against the same adversary trained from the baseline (fig. 8a) or from R2G (fig. 8b). Then we evaluate the returns of different adversaries playing against the same good agent trained from the baseline (fig. 8c) or from R2G (fig. 8d).

In Physical Deception, the R2G agents behave better in most comparisons and only slightly worse in some of the adversary comparisons. For Physical Deception, training good agents' behavior is harder than training adversaries as two good agents need to cooperate to confuse the adversary, where the best response for the adversary is to randomly approach one of the landmarks. Thus, the advantage of R2G on the strategic reasoning is more meaningful when training the good agents.

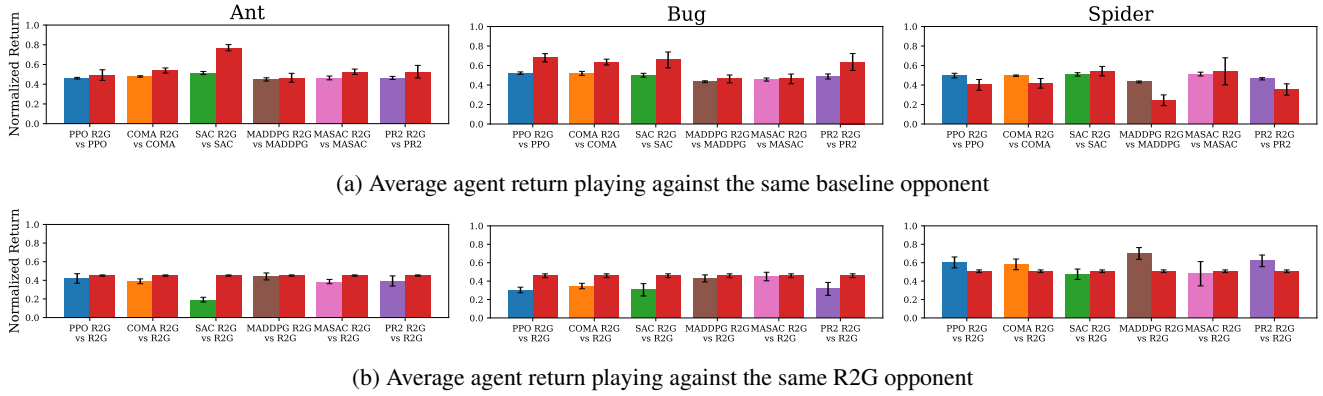


Figure 9: RoboSumo: Pairwise comparison of R2G and the baseline against the same opponent. Left bar: Baseline agents; Right bar: R2G agents.

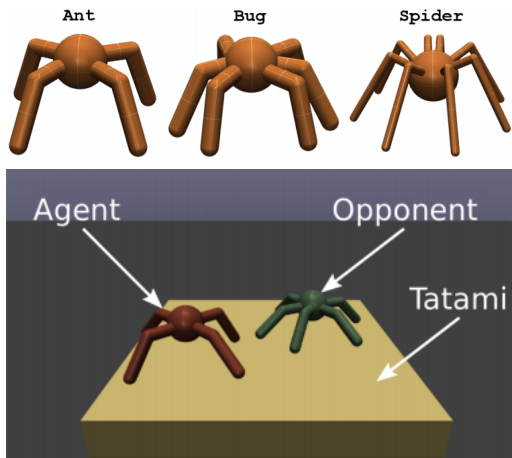


Figure 10: RoboSumo: Illustration of the different types of robots (top) and the game environment (bottom).

In Keep Away and Predator-Prey, the R2G agents also show better performance in most comparisons. As there is no clear optimal equilibrium strategies in these games, a dominate strategy is hard to achieve. While there are some comparisons where R2G is slightly worse, the overall winning rate of R2G is still higher than any other baselines.

RoboSumo

In order to test the scalability and performance of R2G on larger state and action spaces, we compare R2G and other baselines on the RoboSumo (Al-Shedivat et al. 2018) simulator, which has much larger state and action spaces with state dimensions ranging from 120 to 208, and action dimensions ranging from 8 to 16. RoboSumo contains a set of two-player, competitive environments where two robots are competing for pushing the opponent out of a square platform. In our experiments, three types of robots, Ant, Bug, and Spider, are trained and tested by playing against the same type of the opponent robot.

The tested robots as well as the competing platform is

shown in fig. 10. Since the game is symmetric, we choose the agent which performs the best among the two agents trained under the same method, and use it to compare with the best agent of other methods. The results are shown in fig. 9. The pairwise comparison results indicate that the R2G agent performs better than all other baselines on the Ant and Bug robots, and most of the baselines except MADDPG and PR2 on the Spider robot. The performance degeneration in Spider is likely due to a larger state and action space, where learning the basic robot motions is itself difficult given limited samples, and the advantage of strategic reasoning for R2G may not contribute as much.

Conclusion

In this paper, we proposed Recursive Reasoning Graph (R2G), a multi-agent reinforcement learning (MARL) framework that explicitly incorporates the recursive reasoning process into the training process. Based on existing multi-agent actor-critic algorithms, we augment each learning agent with a central actor component to model its conditional response given the strategies of its opponents. We demonstrated how to efficiently train the central actors using experiences stored in the replay buffer. Treating these central actors as nodes and their responses as messages, a recursive reasoning graph was built to efficiently calculate the optimal response of each agent given opponents actions. Thus, the outputs of the graph provided higher level recursion actions which is used for training the individual policies. In our experiments, R2G was able to converge to the ideal performance while other baselines could not in two Differential Games, which demonstrated how R2G addresses relative overgeneralization and oscillatory learning problems. The performance of R2G on multi-player multi-state Particle World and RoboSumo environments was also significantly better than baselines.

Adopting a graph structure for interactions between agents also brings the possibility of using graph neural networks (Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018) for dynamic interaction structure inferring, and efficient parameter sharing between agents.

References

- Al-Shedivat, M.; Bansal, T.; Burda, Y.; Sutskever, I.; Mordatch, I.; and Abbeel, P. 2018. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*.
- Albrecht, S. V.; and Stone, P. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258: 66–95.
- Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 66–83. Springer.
- Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.
- Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 1024–1034.
- He, H.; Boyd-Graber, J.; Kwok, K.; and Daumé III, H. 2016. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 1804–1813.
- Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 32.
- Johannink, T.; Bahl, S.; Nair, A.; Luo, J.; Kumar, A.; Loskyll, M.; Ojea, J. A.; Solowjow, E.; and Levine, S. 2019. Residual reinforcement learning for robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 6023–6029.
- Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; and Russell, S. 2019. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- Littman, M. 1994. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*.
- Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 6379–6390.
- Mohseni-Kabir, A.; Isele, D.; and Fujimura, K. 2019. Interaction-aware multi-agent reinforcement learning for mobile agents with individual goals. In *International Conference on Robotics and Automation (ICRA)*, 3370–3376. IEEE.
- Patro, S.; and Sahu, K. K. 2015. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.
- Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 4295–4304.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Shen, M.; and How, J. P. 2021. Robust Opponent Modeling via Adversarial Ensemble Reinforcement Learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, 578–587.
- Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.
- Tan, M. 1993. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *International Conference on Machine Learning (ICML)*, 330–337.
- Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.
- Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.
- Von Der Osten, F. B.; Kirley, M.; and Miller, T. 2017. The Minds of Many: Opponent Modeling in a Stochastic Game. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 3845–3851.
- Wei, E.; and Luke, S. 2016. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, 17(1): 2914–2955.
- Wei, E.; Wicke, D.; Freelan, D.; and Luke, S. 2018. Multiagent soft Q-learning. In *AAAI Spring Symposium Series*.
- Wen, Y.; Yang, Y.; Luo, R.; and Wang, J. 2020. Modelling Bounded Rationality in Multi-Agent Interactions by Generalized Recursive Reasoning. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
- Wen, Y.; Yang, Y.; Luo, R.; Wang, J.; and Pan, W. 2018. Probabilistic Recursive Reasoning for Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*.
- Wright, J.; and Leyton-Brown, K. 2010. Beyond equilibrium: Predicting human behavior in normal-form games. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- Yang, J.; Nakhaei, A.; Isele, D.; Fujimura, K.; and Zha, H. 2019. CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning. In *International Conference on Learning Representations*.

Experiment Details

Differential Games

The Differential Games contain two single-state two-player games, the Zero Sum and the Max of Two. Each game has a one-hot state space indicating the identity of the player, and a scalar action space of $[-1, 1]$. The reward function of each game is:

- Zero Sum: $r^1(a^1, a^2) = -r^2(a^1, a^2) = 10a^1 \cdot 10a^2$.
- Max of Two: $r^1(a^1, a^2) = r^2(a^1, a^2) = \max(f_1, f_2)$, where $f_1 = 0.8 \times [-(\frac{a^1+0.5}{0.3})^2 - (\frac{a^2+0.5}{0.3})^2]$ and $f_2 = 1.0 \times [-(\frac{a^1-0.5}{0.1})^2 - (\frac{a^2-0.5}{0.1})^2] + 10$.

We use a multi-layer perceptron (MLP) with 2 hidden layers each with 16 hidden units for all the Q networks, policies, as well as central actors. We use a diagonal Gaussian as the output distribution for policies. Each algorithm is trained with 1000 epochs with 100 exploration steps per epoch and a batch size of 256. We use a learning rate of 1×10^{-3} for the Q and value network, and 1×10^{-4} for the policies. To encourage exploration, for PPO and COMA, we add an additional entropy loss in the policy loss with a weight of 0.01. For MADDPG, we implement the Ornstein-Uhlenbeck process for action noise as described in Lillicrap et al. (2015). For SAC, MASAC, PR2, and R2G, we implemented the automatic entropy adjustment as introduced in Haarnoja et al. (2018). Our implementation is based on the *rlkit*¹ open source reinforcement learning package.

Particle World

We use similar network structures as well as other hyperparameters as in the Differential Games with hidden dimensions increased to 64 and number of exploration steps per epoch increased to 1×10^3 . Each training is repeated 5 times with different random seeds and the average performance is reported. The maximum trajectory length of the environment is set to 25.

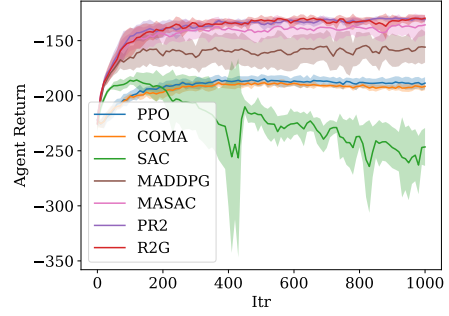
Figure 11 illustrates the learning curves of different methods. For Cooperative Navigation, all agents share the same reward function. The learning curves show a similar comparison results as in fig. 7. For Physical Deception, Keep Away, and Predator-Prey, the adversaries rewards are omitted since these games are nearly zero-sum. Notice that since the evaluation return is computed by playing the agent against the opponents trained together under the same method, this could NOT provide a meaningful comparison of different methods on competitive games.

RoboSumo

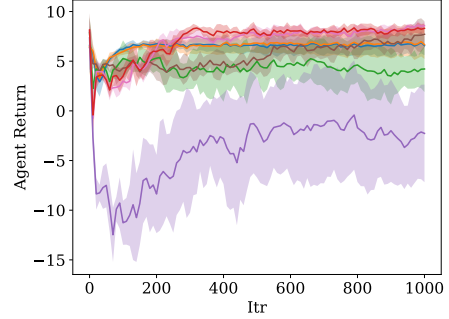
As the environment becomes more complex, we use larger networks with 3 hidden layers and 64 hidden units. We increase the number of exploration steps per epoch as well as the total epoch number to 2×10^3 . The maximum trajectory length of the environment is set to 100. We use a scaling factor of 0.01 on the reward to stabilize the training.

Figure 12 illustrates the learning curves of different methods. Similarly, since the evaluation return is computed by

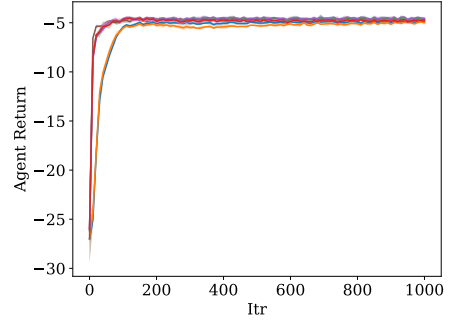
¹<https://github.com/vitchyr/rlkit>



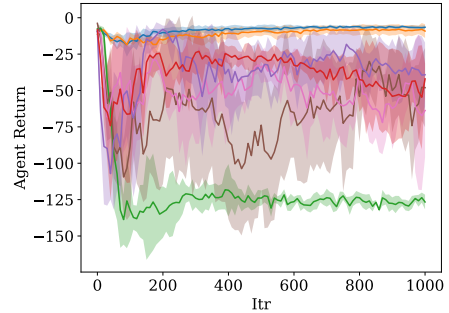
(a) Cooperative Navigation



(b) Physical Deception



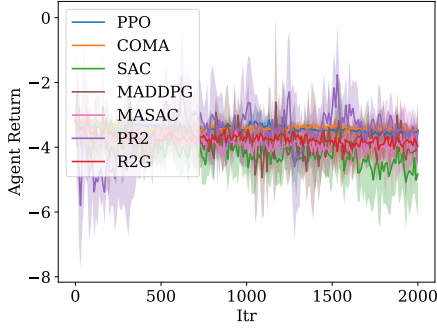
(c) Keep Away



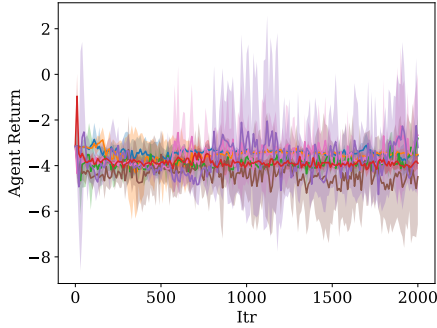
(d) Predator-Prey

Figure 11: Particle World: Learning curves of different methods. The agent return is calculated by playing the agent against the opponents that are trained together under the same method, and is averaged by 40 trajectories. This could NOT provide meaningful comparisons for performance of different methods on competitive games.

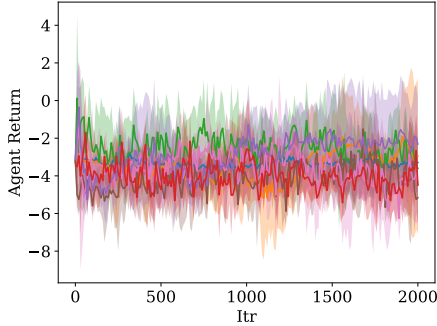
playing the agent against the opponents trained together under the same method, this could NOT provide a meaningful comparison of different methods.



(a) Spider



(b) Bug



(c) Spider

Figure 12: RoboSumo: Learning curves of different methods. The agent return is calculated by playing the agent against the opponents that are trained together under the same method, and is averaged by 20 trajectories. This could NOT provide meaningful comparisons for performance of different methods.

Ablation Study: Different Recursion Levels

The R2G performance shown in previous results are trained with a recursion level of 1. In this section, we provide the comparison results of using different recursion levels. With the help of the central actors and recursive reasoning graphs, the computation complexity scales linearly with the level of

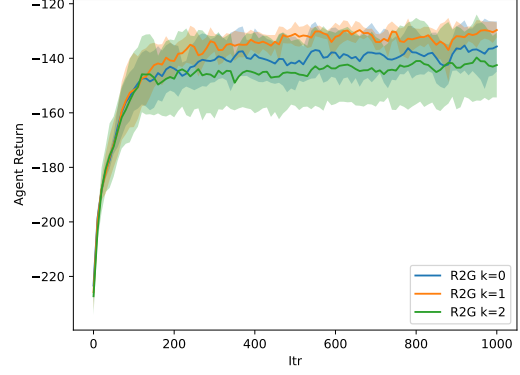


Figure 13: Cooperative Navigation: Comparison of different recursion levels of R2G.

recursions, k . Notice that, the R2G with level-0 recursion is the same as MASAC.

Figure 13 shows the learning curve of R2G with $k = 0, 1, 2$ in the Cooperative Navigation environment. In this environment, the level-1 recursions behaves slightly better than level-0, which is again slightly better than level-2.

Figure 14 shows the return matrices between R2G with different levels on the three competitive particle environments. The row of a matrix indicates the same adversary playing against different agents, and the column of a matrix indicates the same agent playing against different adversaries. In general, the level-0 and level-1 recursions behave better than level-2 recursions, while none of them is dominating.

In general, we expect R2G with level-1 recursion performs well enough for most games for several reasons: First, level-1 recursion resembles the solution concept of evaluating the ego action by assuming others are optimally responding to it, which is similar as the minimax optimization in two-player zero-sum games and joint maximization in cooperative games; Second, while at each training loop we only do 1-level recursion, over multiple training loops, agents' base policies could be regarded as higher level strategies from previous loops. The same reasoning could be applied to higher levels of recursion, where the level of recursions could be regarded as how many steps we would like to "look ahead" in the level-0 learning dynamics. However, using higher levels of recursion would increase the variance of this "looking ahead" due to sampling from stochastic policies, and it does not give guarantees on improving the learning dynamics. In fact, there does not exist a dominating recursion level that is better than others in all games, and we focus on the level-1 recursion due to the above reasons.

Convergence of R2G Value Iteration

Here we provide the convergence proof of value iteration for level-1 R2G in cooperative games, following a similar approach as in single agent SAC (Haarnoja et al. 2018) and PR2 (Wen et al. 2018). Assuming that at each iteration, π_c^i

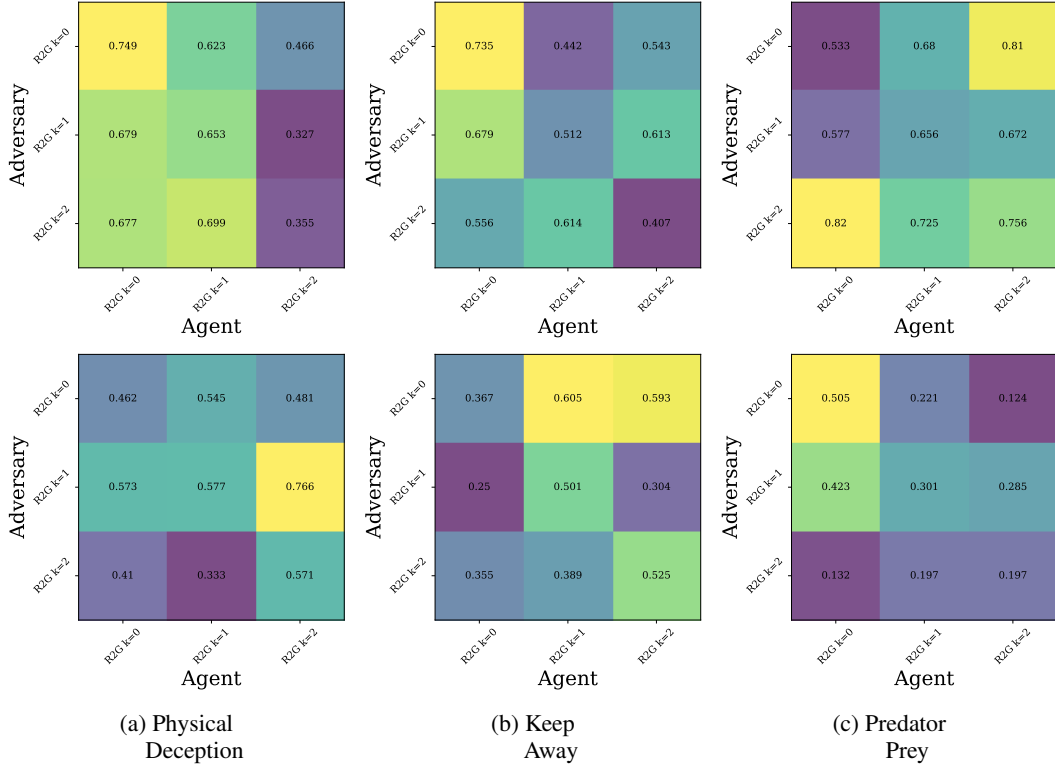


Figure 14: Particle World: Comparison of different recursion levels of R2G in competitive games. Top row: agent returns. Bottom row: adversary returns.

is trained to optimize eq. (2), which gives us $\pi_c^i(s, a^{-i}) = \arg \max_{a^i} Q^i(s, a^i, a^{-i})$. Using the fact that $r^i = r^j$ and thus $Q^i = Q^j$ in cooperative games, the value iteration operator, Γ^{π_i} , for Q^i derived from eq. (6) with $k = 1$ could be written as

$$\begin{aligned} \Gamma^{\pi_i} Q^i(s, a^i, a^{-i}) = & r^i(s, a^i, a^{-i}) + \\ & \gamma \mathbb{E}_{s', a'^i \sim T, \pi^i} \max_{a'^{-i}} Q^i(s', a'^i, a'^{-i}) \\ & - \alpha^i \log \pi^i(a'^i | s') \end{aligned} \quad (8)$$

Define the norm on Q-values as $\|Q_1^i - Q_2^i\| \triangleq \max_{s, a^i, a^{-i}} |Q_1^i(s, a^i, a^{-i}) - Q_2^i(s, a^i, a^{-i})|$, and let $\epsilon = \|Q_1^i - Q_2^i\|$. Then we have $|\max_{a'^{-i}} Q_1^i(s', a'^i, a'^{-i}) - \max_{a'^{-i}} Q_2^i(s', a'^i, a'^{-i})| \leq \epsilon$, and thus

$$\begin{aligned} \|\Gamma^{\pi_i} Q_1^i - \Gamma^{\pi_i} Q_2^i\| = & \|\gamma \mathbb{E} \max_{a'^{-i}} Q_1^i(s', a'^i, a'^{-i}) \\ & - \max_{a'^{-i}} Q_2^i(s', a'^i, a'^{-i})\| \quad (9) \\ \leq & \|\gamma \mathbb{E} \epsilon\| = \gamma \|Q_1^i - Q_2^i\| \end{aligned}$$

Thus Γ^{π_i} is a contraction mapping.

Limitations

While R2G has shown great performance on most of the environments tested in the paper, there are certain game

structures where R2G does not help. For example, in a two-player cooperative game with two optimal points $(a^{1,*}, a^{2,*})$ and $(a^{1,**}, a^{2,**})$, where $r(a^{1,*}, a^{2,*}) = r(a^{1,**}, a^{2,**}) \geq r(a^1, a^2), \forall a^1, a^2 \in A$, the two R2G agents may converge to different optimal points and the actual performance is then suboptimal. E.g., the player 1 may converge to play $a^{1,*}$ where $a^{2,(1)} = a^{2,*} = \pi_c^2(a^{1,*})$. However, player 2 may converge to $a^{2,**}$ where $a^{1,(1)} = a^{1,**} = \pi_c^1(a^{2,**})$. Games with multiple optimal points with the exact same return are very rare in practice, and R2G could be easily adapted to it if the reward structure is known prior to the training. However, we do realize that R2G is not suitable for all game structures but a promising additional candidate in the current deep MARL algorithm landscape.

Computation Platform and Cost

All experiments were performed on a 2.6GHz, 28 core Intel(R) Xeon(R) E5-2690 v4 CPU. Experiments on Differential Games for each method take approximately 1 hour. Experiments on Particle World take approximately 1 day. Experiments on RoboSumo take approximately 3 days.