ROMAX: Certifiably Robust Deep Multiagent Reinforcement Learning via Convex Relaxation

Anonymous Author(s)

Abstract: In a multirobot system, there are a number of cyber-physical attacks (e.g., communication hijack, observation perturbations) that challenge the robustness of agents. This robustness issue worsens in multiagent reinforcement learning because there exists the non-stationarity of the environment caused by simultaneously learning agents whose changing policies affect the transition and reward functions. In this paper, we propose a minimax MARL approach to infer the worst-case policy update of other agents. As the minimax formulation is computationally intractable to solve, we apply the convex relaxation of neural networks to solve the inner minimization problem. Such convex relaxation enables robustness in interacting with peer agents that may have significantly different behaviors, and also achieves a certified bound of the original optimization problem. We evaluate our approach on multiple mixed cooperative-competitive tasks and show that our method outperforms the previous state of the art approaches on this topic.

Keywords: Multirobot System, Deep Multiagent Reinforcement Learning, Robust Learning, Convex Relaxation

1 Introduction

Multirobot systems have recently attracted much attention in robotics. Compared to a single robot approach, a multirobot system provides several unique benefits, including 1) improved efficiency since a sophisticated problem can be decomposed into simpler sub-problems, distributed across robots, and then solved simultaneously and 2) improved mission success rate because a single robot failure can be addressed by another teammate [1]. These advantages have resulted in emerging multirobot applications, such as formation control [2], cooperative manipulation [3], and human-swarm interaction [4].

Multiagent reinforcement learning (MARL) provides a principled framework for solving problems in which multiple robots interact with one another in a shared environment. However, there remain difficulties in learning intelligent multiagent policies. Amongst these, instability in policy learning is particularly problematic in that agents generally show poor performance when interacting with unseen agents. While there are approaches that stabilize learning in MARL (e.g., centralized training and decentralized execution frameworks [5, 6]), agents generally overfit to other agents’ policies during training, resulting in a failure when playing against policies not interacted before. This issue becomes more catastrophic in a competitive scenario, where an opponent may intentionally apply cyber-physical attacks (e.g., communication hijack, observation perturbations), fully exploit an agent’s brittle policy, and dominate a game [7]

Our contributions. To address this robustness issue, we propose ROMAX, a new framework for RObust MARL via convex relaxAtion, which can compute the approximate globally worst-case scenario. In general, it is computationally intractable to solve the nonconvex-nonconcave minimax optimization [8]. Assuming that each agent’s policy is parameterized by deep neural networks, we develop an approach that can approximately solve the minimax optimization efficiently via the convex relaxation of the neural networks. In summary, this paper provides the following main contributions:
• **Convex relaxation:** We solve the minimax optimization approximately via the convex relaxation of a neural network in the inner optimization, which infers the worst actions of the other agents. This convex relaxation has the important benefit that it can explore the approximate globally worst situation while achieving certified robustness from the guaranteed bound of the relaxation.

• **Empirical evaluation:** We empirically evaluate our algorithm on multiple mixed cooperative-competitive tasks and show that our algorithm outperforms baselines by a significant margin. This demonstrates the necessity to compute the worst-case scenarios to improve robustness in MARL.

2 Related Works

Centralized training with decentralized execution. The standard approach for addressing non-stationarity in MARL is to consider information about other agents and reason about the effects of joint actions [9]. The recent studies regarding the centralized training with decentralized execution framework, for instance, account for the behaviors of others through a centralized critic [5, 6, 10, 11, 12]. While this body of work partially alleviates non-stationarity, converged policies generally overfit the current behaviors of other agents and thus show poor performance when interacting with agents with new behaviors. In contrast, our agents learn robust policies based on minimax optimization by applying convex relaxation.

Robust MARL. Our framework is closely related to prior works that apply the minimax optimization in multiagent learning settings [13, 14, 15]. Minimax provides a game-theoretical concept that encourages an agent to learn a robust policy by maximizing its performance in a worst-case scenario [16]. One of the noticeable studies in this category is [17], which computes the worst-case perturbation by taking a single gradient descent step assuming that other agents act adversarial. However, the single-step gradient approximation can only explore the locally worst situation and can still result in unstable learning. Our approach addresses this drawback by computing the approximate globally worst situation based on the proposed convex relaxation.

Ensemble training in MARL. Another relevant approach to learning a robust policy is ensemble training, where each agent interacts with a group of agents instead of a particular agent only [5, 18, 19]. For example, the population-based training technique, which was originally proposed to find a set of hyperparameters for optimizing a neural network [20], was applied in MARL by evolving a population of agents [21]. This approach showed robust and superhuman level performance in a competitive game. The literature on self-play, which plays against random old versions of itself to improve training stability and robustness, can also be classified into this category [22]. However, maintaining and/or evolving a population is often computationally heavy. Additionally, these methods do not employ minimax optimization, so agents may not be able to cope well with the worst scenario.

Learning awareness MARL. Our framework is also related to prior works that consider the learning of other agents in the environment to address non-stationarity. These works include [23] which attempted to discover the best response adaptation to the anticipated future policy of other agents. Our work is also related to [24, 25] that shape the learning process of others. Another relevant idea explored by [26] is to interpolate between the frameworks of [23] and [24] in a way that guarantees convergence while influencing the opponent’s future policy. Recently, [27] addresses non-stationarity by considering both an agent’s own non-stationary policy dynamics and the non-stationary policy dynamics of other agents in a meta-learning objective. While these approaches alleviate non-stationarity by considering the others’ learning, they do not solve the minimax objective and thus cannot guarantee robustness when playing against a new opponent.

Robustness verification and neural network relaxation. To verify the robustness of neural networks, it is desirable to compute the lower and upper bound of the output neurons under input perturbations. In supervised learning settings, for example, the margin between the ground-truth class and any other classes indicates the robustness of neural networks (i.e., measuring the chance of misclassification). However, due to the nonconvexity in neural networks, [28] proved that finding the true range of the neural network output is nonconvex and NP-complete. As a result, methods based on convex relaxation are proposed to efficiently compute the outer approximation (a more
conservative estimate) of the neural network output range. Many prior works are based on the linear relaxation of the nonlinear units in neural networks: FastLin [29], DeepZ [30], Neurify [31], DeepPoly [32], and CROWN [33]. There are also other approaches based on semidefinite relaxation [34, 35], which admit tighter bounds but are more computationally expensive. See [36] for in-depth surveys on this topic.

3 Background

3.1 Markov games

Interactions between multiple agents can be represented by a partially observable Markov game [13], defined as a tuple $⟨I, S, \{O_i\}_{i \in I}, \{A_i\}_{i \in I}, T, \{R_i\}_{i \in I}, \gamma⟩$: $I = \{1, \ldots, n\}$ is a set of $n$ agents, $S$ denotes the state space, $\{O_i\}_{i \in I}$ is the set of observation spaces, $\{A_i\}_{i \in I}$ is the set of action spaces, $T$ is the state transition function, $\{R_i\}_{i \in I}$ is the set of reward functions, and $\gamma$ is the discount factor. Each agent $i$ chooses an action $a_i$ with its stochastic policy $\pi_{\theta_i} : O_i \times A_i \rightarrow [0, 1]$, where $\theta_i$ denotes the policy parameters. Then, the joint action $a = \{a_i, a_{-i}\}$ yields a transition to the next state according to the state transition function $T : S \times \{A_i\}_{i \in I} \rightarrow S$. Note that the notation $-i$ indicates all other agents except agent $i$. Agent $i$ obtains a reward as a function of the state and the joint action $r_i : S \times \{A_i\}_{i \in I} \rightarrow \mathbb{R}$, and receives a private observation correlated with the state $o_i : S \rightarrow O_i$. Agent $i$ aims to maximize its own total expected discounted return $R_i = \sum_{t=1}^{T} \gamma^t r_i^t$, where $r_i^t$ denotes agent $i$’s reward received at timestep $t$ and $T$ denotes the episodic horizon.

3.2 Multiagent deep deterministic policy gradient (MADDPG)

To stabilize learning in MARL, [5] proposed the centralized training and decentralized execution paradigm, in which the centralized critic conditions on the global information and the decentralized actor only depends on the agent’s local observation. Specifically, a centralized critic for agent $i$ is defined as $Q_{\theta_i}^\mu(o, a_i, a_{-i}) = \mathbb{E}_\mu[Q_{\theta_i}^\mu(o', a_i, a_{-i}) | o, a_i, a_{-i}]$, where $o$ and $\mu$ denotes the joint observation and policy, respectively. The policy gradient for agent $i$’s deterministic policy $\mu_{\theta_i}$ (abbreviated as $\mu_i$) with respect to the expected return $J(\theta_i) = \mathbb{E}[R_i]$ is:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{o,a \sim D} [\nabla_{\theta_i} \mu_i(a_i | o_i) \nabla_{o} Q_{\theta_i}^\mu(o, a_i, a_{-i}) | a_i = \mu_i(o_i)],$$

where $D$ denotes the replay buffer. The buffer $D$ stores the tuples $(o, o', a, r)$ where $o'$ is the next joint observation and $r$ is the joint reward. The centralized critic $Q_{\theta_i}^\mu$ is updated by minimizing the following loss function:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{o,o',a,r \sim D} [Q_{\theta_i}^\mu(o, a_i, a_{-i}) - y]^2,$$

s.t. $y = r_i + \gamma Q_{\theta_i}^\mu(o', a_i', a_{-i}) | a_i' = \mu_i(o_i'), \forall j \in I$, where $\mu_i = \{\mu_{\theta_i}\}_{i \in I}$ denotes the set of target policies.

3.3 Minimax MultiAgent deep deterministic policy gradient (M3DDPG)

To learn a robust policy in MARL, an agent should account for the changes in the policy of other simultaneously learning agents. The framework by [17] proposed a robust multiagent learning approach based on the minimax optimization by assuming other agents are adversarial agents. Specifically, each agent $i$ in [17] optimizes the following learning objective:

$$\max_{\theta_i} \min_{a_i \in B_{-i}} Q_{\theta_i}^\mu(o, a_i, a_{-i}) | a_i = \mu_i(o_i)$$

where $B_{-i}$ is a compact constraint set of $a_{-i}$ (e.g., a $l_p$ norm ball). Solving a general nonconvex-nonconcave minimax optimization problem is intractable [8]. Consequently, [17] replaces the inner minimization with a one-step gradient descent:

$$a_{-i}^* = \arg \min_{a_{-i}} Q_{\theta_i}^\mu(o, a_i, a_{-i}) \approx a_{-i} - \alpha_{-i} \nabla_{a_{-i}} Q_{\theta_i}^\mu(o, a_i, a_{-i}),$$

where $\alpha_{-i} \geq 0$ is the learning rate.
4 Approach

While the single-step gradient approximation discussed in Section 3.3 [17] improves robustness, we note that the framework has several drawbacks:

- The single-step gradient approximation can only explore the locally worst situation and can still lead to unsatisfying behavior when testing with opponents that have drastically different strategies.
- Implementing (4) multiple times for the inner minimization can potentially alleviate this issue, but it can lead to a double-loop approach in solving (3), which is computationally prohibitive.
- Moreover, the one-step gradient descent approximation can only compute the upper bound of the inner minimization problem because the original problem cannot be solved to a global optimum. Hence, for the outer level, maximizing an upper bound of the inner objective cannot guarantee the maximum of the original objective in (3). In other words, even though one-step gradient descent approximation cannot find a perturbation that results in the smallest $Q^μ_i$, such perturbation can exist.

As we detail in this section, we address these drawbacks by employing convex relaxation of the neural network and solving the inner minimization to explore the approximate globally worst situation while achieving certified robustness from the guaranteed bound of the convex relaxation.

4.1 Convex relaxation of neural networks

We propose to convexify the centralized action-value function in MARL and efficiently solve the inner minimization problem in (3). We assume that $Q^μ_i$ is parameterized by fully connected networks with $L$ layers with an input $x_0 = (o, a_i, a_{−i})$. Then, $Q^μ_i$ can be expressed by the following form:

$$z^l = W^l_c x^{l−1}, x^l = \sigma(z^l), \forall l = 1, \ldots, L − 1$$

$$Q^μ_i = W^L_c z^L,$$

(5)

where $\sigma(\cdot)$ denotes the nonlinear activation function, and $W^l_c$ is the weight at layer $l$. For clarity, we drop the bias terms without loss of generality. Due to the nonconvexity of the activation function, verifying the robustness property of $Q^μ_i$ over a compact set of $x_0$ is difficult [37, 29]. To address this, we employ a convex relaxation technique [37] to verify robustness verification of neural networks and apply the following linear convexification to the centralized Q-function by assuming the ReLU activation function:

$$z^l = W^l_c x^{l−1}, x^l = u^l \odot (z^l − l^l) \odot (u^l − l^l)$$

$$x^l \geq 0, x^l \geq z^l, \forall l = 1, \ldots, L − 1$$

$$Q^μ_i = W^L_c z^L,$$

(6)

where $l^l$ and $u^l$ are lower and upper bounds for $z^l$, respectively, and $\odot$ and $\odot$ are the element-wise multiplication and division, respectively. Note that $Q^μ_i$ indicates the relaxed version of $Q^μ_i$ (i.e., $l^l \leq z^l \leq u^l$). Due to this relaxation, all equations in (6) are convex, so $\min_{a_{−i}} Q^μ_i(o, a_i, a_{−i})$ is a constrained linear programming and can be solved efficiently.

4.2 Solving minimiax optimization via convex relaxation

Here we employ the convex relaxation technique in Section 4.1 to solve the inner minimization problem approximately. Specifically, we propose to replace the inner minimization problem in (3) with the following relaxed objective:

$$\bar{Q}^μ_i(o, a_i, a_{−i}) = \min_{a_{−i} \in B_{−i}} Q^μ_i(o, a_i, a_{−i}) \leq \min_{a_{−i} \in B_{−i}} Q^μ_i(o, a_i, a_{−i}),$$

(7)

where $\bar{Q}^μ_i(o, a_i, a_{−i})$ is the lower bound of $Q^μ_i(o, a_i, a_{−i})$ from the relaxation (6) (see Figure 1). $Q^μ_i(o, a_i, a_{−i})$ is also a function of $μ$ as well and is a surrogate of the original non-convex nonlinear objective. The main advantage of the convex relaxation in (7) over (4) is that the former does not need the step size $α_{−i}$. The performance of [17] is highly sensitive to the step size, and it is difficult to tune. By contrast, our convex relaxation problem can be efficiently solved without needing the step size. With this lower bound, we can reformulate the outer maximization problem as:

$$\max_{\theta_1} \left[ (1 − κ_i)Q^μ_i(o, a_i, a_{−i}) + κ_i \bar{Q}^μ_i(o, a_i, a_{−i}) \right]_{a_i = μ_i(a)}.$$  

(8)
where $0 \leq \kappa_i \leq 1$ is a weight coefficient for the term that accounts for the policy change of the other agents. Because we maximize the lower bound of the inner minimization problem, the original inner objective is guaranteed to be maximized. Such a guarantee provides robustness certificates for agent $i$ as it considers the worst-case scenarios caused by other learning agents. By setting $\kappa_i \neq 1$, we do not entirely use the relaxed inner objective (i.e., $\bar{Q}_i^\kappa(o, a, a_i^*)$) as the objective of the outer maximization problem for the sake of training stability, as a relaxation gap might be big especially in the early training process. Instead, a combination of the original objective and its relaxed one is used as the objective for the outer maximization problem, as shown in (8). Because this inner minimization needs to be solved whenever the policy is updated, the convex relaxation problem in (7) has to be efficient enough with a tight bound. Therefore, there is a trade-off to choose a certain convex relaxation method among many candidates; details are deferred to Appendix A.1.

### 4.3 Integrating with MARL algorithm

Our framework based on convex relaxation in Section 4.2 can be readily integrated into general MARL frameworks. We implement our method based on MADDPG (see Section 3.2). Integrating the minimax formulation and the convex relaxation (8) together with the updating rules of the actor (1) yields:

$$
\nabla_{\theta_i} J(\theta_i) = E_{o, a \sim D} \left[ \nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i} \left( (1 - \kappa_i) Q_i^\mu(o, a, a_i, a_{-i}) + \kappa_i \bar{Q}_i^\mu(o, a_i, a_{-i}^*) \right) \right],
$$

$$
a_i = \mu_i(o_i),
$$

$$
a_{-i}^* = \arg \min_{a_{-i} \in B_{-i}} \bar{Q}_i^\mu(o, a_i, a_{-i}),
$$

where $B_{-i} = B_{-i}((a_j = \mu_j(o_j), \forall j \neq i), \epsilon)$ is a $l_p$ ball centered at $(a_j = \mu_j(o_j), \forall j \neq i)$ with a radius $\epsilon$. Then, the critic is updated by the following: (2)

$$
L(\theta_i) = E_{o, o', a, r, \tau \sim D} [Q_i^\mu(o, a, a_{-i}) - y]^2,
$$

$$
y = r_i + \gamma \left( (1 - \kappa_i) Q_i^\mu(o', a_i', a_{-i}^*) + \kappa_i \bar{Q}_i^\mu(o', a_i', a_{-i}^*) \right),
$$

$$
a_i' = \mu_i'(o_i')
$$

$$
a_{-i}^* = \arg \min_{a_{-i}^* \in B_{-i}'} \bar{Q}_i^\mu(o', a_i', a_{-i}'),
$$

where $B_{-i}' = B_{-i}'((a_j = \mu_j(o_j), \forall j \neq i), \epsilon)$ is a $l_p$ ball centered at $(a_j = \mu_j(o_j), \forall j \neq i)$ with a radius $\epsilon$.

Figure 1: Illustration of the convex relaxation approach for solving the inner minimization in (7).
5 Experiments

5.1 Evaluation domains and baselines

We evaluate our approach in mixed cooperative-competitive tasks from the multiagent particle benchmark [5]. In these tasks, there are $n_c$ cooperative agents, $n_a$ adversary agents, and $n_L$ landmarks in a 2D continuous space. We focus on tasks that include adversarial agents such that agents need to have diverse strategies to outperform opponents and thus robustness is important. Below are some descriptions of the tasks with illustration in Figure 2.

- **Predator-prey.** $n_a = 3$ slower cooperative predators try to catch the $n_c = 1$ faster prey. $n_L = 2$ landmarks are unmovable and can impede the way of all of the agents. Once there is a collision between predators and the prey, the former get rewarded while the latter gets penalized.

- **Physical deception.** There are $n_a = 1$ adversary and $n_c = 2$ agents together with $n_L = 2$ landmarks in the environments. The adversary aims to occupy a target landmark without knowing which one of the two landmarks is the target. As a result, agents must split and cover all landmarks to deceive the adversary.

**Transfer to real robot learning.** We note that these tasks closely coincide with real-world robotic missions. For the predator-prey, multiple robots can be deployed to chase an intelligent moving target (e.g., an intruder in a market). For physical deception, we can deploy robots to protect assets of interest with intelligent behaviors to deceive opponents. The fidelity of the models and perception required in simulation can be easily achieved in the real world via sensors such as cameras, velocity meters, and LiDAR. Hence, the sample complexity won’t differ much between simulation and the real world. Lastly, the policy learned can be easily implemented on-board and the generated actions can be further executed by a lower-level controller, if necessary.

**Baselines.** We compare ROMAX to M3DDPG [17], a robust MARL algorithm that also applies the minimax formulation but solves the inner optimization approximately via the one-step gradient descent. We also compare our algorithm to MADDPG [5], which uses the centralized critic but does not solve minimax. Implementation details and hyperparameters are specified in Appendix A.1.

5.2 Results

**Question 1 How much does ROMAX improve the robustness of trained policies?**

To answer this question and test the robustness, each policy from one team is evaluated against a diverse set of policies from the other team. Then the adversaries’ policies trained by one algorithm under each random seed will be evaluated against the agents’ policy trained by all of the other algorithms under all of the random seeds; vice-versa for the agent’s policy.
Table 1: Evaluation in the predator-prey task. Predator and prey correspond to adversary (Adv for short) and agent in the table, respectively. Each pair is evaluated for 250 episodes, i.e., 10 episodes for each of the $5 \times 5 = 25$ pairs of random seeds. ($\bullet, \bullet$) in each cell denotes the mean/standard error of the reward per step in the episode of the adversaries and agents, respectively. The higher the return is, the better the policy is. For each column, different adversaries compete against the same agent, and a higher adversary reward indicates better performance against the same agent; row-wise for the agents. In the last row, we summarize the overall robustness results for playing both teams via the metric $R_{\text{overall}}$.

<table>
<thead>
<tr>
<th>Adv</th>
<th>Agent</th>
<th>MADDPG</th>
<th>M3DDPG</th>
<th>ROMAX</th>
<th>$R_{\text{Adv}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MADDPG</td>
<td></td>
<td>$-0.017 \pm 0.012$, $-0.550 \pm 0.017$</td>
<td>$0.160 \pm 0.045$, $-0.502 \pm 0.053$</td>
<td>$0.033 \pm 0.029$, $-0.406 \pm 0.025$</td>
<td>$0.174 \pm 0.080$</td>
</tr>
<tr>
<td>M3DDPG</td>
<td></td>
<td>$-0.718 \pm 0.051$</td>
<td>$0.307 \pm 0.043$, $-0.609 \pm 0.060$</td>
<td>$-0.043 \pm 0.031$, $-0.290 \pm 0.042$</td>
<td>$0.514 \pm 0.158$</td>
</tr>
<tr>
<td>ROMAX</td>
<td></td>
<td>$-1.093 \pm 0.037$</td>
<td>$0.560 \pm 0.032$, $-0.428 \pm 0.055$, $-0.036 \pm 0.097$</td>
<td>$0.132 \pm 0.020$, $-0.477 \pm 0.029$</td>
<td>$1.12 \pm 0.183$</td>
</tr>
<tr>
<td>$R_{\text{Agent}}$</td>
<td></td>
<td>$-2.361 \pm 0.230$</td>
<td>$-2.047 \pm 0.198$</td>
<td>$-1.175 \pm 0.083$</td>
<td></td>
</tr>
<tr>
<td>$R_{\text{overall}}$ = $R_{\text{Adv}} + R_{\text{Agent}}$</td>
<td>MADDPG</td>
<td>M3DDPG</td>
<td>ROMAX</td>
<td>-2.187</td>
<td>-1.533</td>
</tr>
</tbody>
</table>

Table 2: Evaluation in the physical deception task. The evaluation settings and metrics shown in this table are the same as those in Table 1.

<table>
<thead>
<tr>
<th>Adv</th>
<th>Agent</th>
<th>MADDPG</th>
<th>M3DDPG</th>
<th>ROMAX</th>
<th>$R_{\text{Adv}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MADDPG</td>
<td></td>
<td>$-0.795 \pm 0.017$, $0.482 \pm 0.005$</td>
<td>$-0.689 \pm 0.031$, $0.248 \pm 0.020$</td>
<td>$-0.814 \pm 0.032$, $0.338 \pm 0.019$</td>
<td>$-2.298 \pm 0.061$</td>
</tr>
<tr>
<td>M3DDPG</td>
<td></td>
<td>$-0.712 \pm 0.029$, $0.225 \pm 0.021$</td>
<td>$-0.819 \pm 0.018$, $0.467 \pm 0.004$</td>
<td>$-0.839 \pm 0.037$, $0.271 \pm 0.020$</td>
<td>$-2.4 \pm 0.050$</td>
</tr>
<tr>
<td>ROMAX</td>
<td></td>
<td>$-0.572 \pm 0.028$, $0.128 \pm 0.019$</td>
<td>$-0.613 \pm 0.033$, $0.133 \pm 0.019$</td>
<td>$-0.512 \pm 0.018$, $0.283 \pm 0.003$</td>
<td>$-1.697 \pm 0.048$</td>
</tr>
<tr>
<td>$R_{\text{Agent}}$</td>
<td></td>
<td>$0.835 \pm 0.156$</td>
<td>$0.848 \pm 0.139$</td>
<td>$0.892 \pm 0.033$</td>
<td></td>
</tr>
<tr>
<td>$R_{\text{overall}}$ = $R_{\text{Adv}} + R_{\text{Agent}}$</td>
<td>MADDPG</td>
<td>M3DDPG</td>
<td>ROMAX</td>
<td>-1.463</td>
<td>-1.552</td>
</tr>
</tbody>
</table>

As Table 1 and Table 2 demonstrate, for both tasks, our algorithm can train more robust policies for both teams in a competitive game. For each adversary, when competing against the same set of diverse agents, our adversary get the highest return; see the $R_{\text{Adv}}$ columns in the tables. Similar conclusion can be made for the agents given the $R_{\text{Agent}}$ rows in the tables. These results demonstrate that, via computing the approximate global worst-case situation, policies can generalize and perform well when tested against unseen peer agents’ policies. We also note that M3DDPG is outperformed by MADDPG in Table 2 (see the overall robustness results). This might be due to the sensitive step-size parameter of M3DDPG in (4). This observation implies that a tuned step size for one task cannot generalize to another one, and also shows the advantage of ROMAX.

**Question 2** How much can disruptive policies exploit a fixed robust policy?

To answer the second question, we construct a disruptive policy in the predator-prey task by: 1) training both teams with each algorithm until convergence, 2) fixing the prey policy, and 3) training new adversary predators policies based on MADDPG that attempt to exploit this fixed prey trained by each method. In Figure 5.2, the robustness results of the fixed prey trained with different algorithms are shown. As the disruptive adversaries’ training proceeds, the fixed prey’s return decreases as expected. However, ROMAX achieves the highest return compared to other methods, validat-
Figure 3: The performance of the fixed agent (prey) during the training of disruptive adversaries (predators). Higher return implies a more robust policy.

We observe that M3DDPG and MADDPG perform similarly in this analysis, possibly due to the sensitive tuning of the step size.

6 Conclusion

In this paper, we propose a robust reinforcement learning algorithm for a multirobot system. To robustify learning, we consider the learning of other agents based on the worst-case scenario criterion, which inherently leads to a minimax formulation. As minimax formulation is computationally expensive to solve, convex relaxation of neural networks is applied to solve the inner minimization problem. By convex relaxation, agents can account for peer agents that possibly have drastically different behaviors, and a certified bound of the original optimization problem can be gained. We believe this is the first work that integrates robustness verification in MARL. Our algorithm outperforms existing robust MARL algorithms in mixed cooperative-competitive tasks.

There are a few important directions for future works. First, we would like to develop tight but efficient convex relaxation-based methods for neural network robustness verification. Moreover, there are several real-world robustness applications, including observation perturbation, actuation fault, malicious/stealthy attack, communication delay, that we would like to test our approach on. Lastly, developing principled and general learning methods with theoretical guarantees (e.g., convergence analysis) will be a meaningful direction.
References


