ROMAX: Certifiably Robust Deep Multiagent Reinforcement Learning via Convex Relaxation

Chuangchuang Sun[1] Dong-Ki Kim[1] and Jonathan P. How[1]

***Abstract*— In a multirobot system, a number of cyber-physical attacks (e.g., communication hijack, observation perturbations) can 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.**

I. 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 [5, 6]. While there are approaches that stabilize policy learning in MARL (e.g., centralized training and decentralized execution frameworks [7, 8]), agents generally overfit other agents' policies during training, resulting in a failure when playing against new strategies not interacted before. This robustness issue becomes more severe in a competitive setting, where an opponent can intentionally apply cyber-physical attacks (e.g., communication hijack, observation perturbations), fully exploit an agent's brittle policy, and thus dominate a game [9].

[1]Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139. {ccsun1,dkkim93,jhow}@mit.edu

Our contributions. To address the robustness problem, we propose a new framework, called **ROMAX**: **RO**bust **MA**RL via convex rela**X**ation. While the minimax optimization enables learning of robust multiagent policy [10], solving a general nonconvex-nonconcave minimax optimization problem is computationally intractable [11]. Assuming that each agent's policy is parameterized by deep neural networks, we develop a computationally efficient approach that can approximately solve the minimax optimization and infer the worst-case actions of other agents via the convex relaxation of neural networks. We note that this convex relaxation has an important benefit in that it can explore the approximate *globally* worst situation while achieving *certified* robustness from the guaranteed bound of the relaxation. We empirically evaluate our algorithm on multiple mixed cooperative-competitive tasks and show that ROMAX outperforms baselines by a significant margin, demonstrating the necessity to compute the worst-case scenarios to improve robustness in MARL.

II. 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 [12]. The recent studies regarding the centralized training with decentralized execution framework, for instance, account for the behaviors of others through a centralized critic [7, 8, 13–15]. 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 minimax optimization in multiagent learning settings [16, 17]. Minimax provides a game-theoretical concept that encourages an agent to learn a robust policy by maximizing its performance in a worst-case scenario [10, 18]. One of the noticeable studies in this category is [19], 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 thus can still result in unstable learning. Our approach aims to address this drawback by computing the approximate globally worst situation based on convex relaxation. The work by [20] applies the similar linear relaxation technique in a single-agent robust RL problem to certify the robustness under uncertainties from the environments. However, in our multiagent settings, the robustness is more challenging to certify due to the concurrent policy learning amongst multiple agents.

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 [7, 21, 22]. For example, the population-based training technique, which was originally proposed to find a set of hyperparameters for optimizing a neural network [23], was applied in MARL by evolving a population of agents [24]. 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 [25]. 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 aware 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 [26] which attempted to discover the best response adaptation to the anticipated future policy of other agents. Our work is also related to [27, 28] that shape the learning process of others. Another relevant idea explored by [29] is to interpolate between the frameworks of [26] and [27] in a way that guarantees convergence while influencing the opponent's future policy. Recently, [6] addresses non-stationarity by considering both an agent's own non-stationary policy dynamics and the non-stationary policy dynamics of other agents within a meta-learning objective. While these approaches alleviate non-stationarity by considering the others' learning, they do not solve the minimax objective and cannot guarantee robustness when playing against a new opponent. This weakness can be exploited by a carefully trained adversary agent [9].

Robustness verification and neural network relaxation. To verify the robustness of neural networks, it is important to compute the lower and upper bound of the output neurons under input perturbations. In supervised learning settings, for example, the margin between predicting the ground-truth class and other classes indicates the robustness of neural networks (i.e., measuring the chance of misclassification). However, due to the nonconvexity in neural networks, the work by [30] proved that finding the true range of neural network's output is nonconvex and NP-complete. To address this issue, convex relaxation methods are proposed to efficiently compute the outer approximation (a more conservative estimate) of neural network's output range. Many prior works are based on the linear relaxation of the nonlinear units in neural networks: FastLin [31], DeepZ [32], Neurify [33], DeepPoly [34], and CROWN [35]. There are also other approaches based on semidefinite relaxation [36, 37], which admit tighter bounds but are more computationally expensive. See [38] for in-depth surveys on this topic.

III. Background

A. Markov game

Interactions between n agents can be represented by a partially observable Markov game [10], defined as a tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, \mathcal{T}, \{\mathcal{R}_i\}_{i\in\mathcal{I}}, \gamma\rangle$; $\mathcal{I}=[1,...,n]$ is a set of n agents, $\mathcal{S}$ is the state space, $\{\mathcal{O}_i\}_{i\in\mathcal{I}}$ is the set of observation spaces, $\{\mathcal{A}_i\}_{i\in\mathcal{I}}$ is the set of action spaces, $\mathcal{T}$ is the state transition function, $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ is the set of reward functions, and γ is the discount factor. Each agent i chooses an action a_i according to its stochastic policy $\pi_{\theta_i}: \mathcal{O}_i \times \mathcal{A}_i \rightarrow [0,1]$, where θ_i denotes agent i's policy parameters. Then, the joint action $a = \{a_i, a_{-i}\}$ yields a transition to the next state according to the state transition function $\mathcal{T}: \mathcal{S} \times \{\mathcal{A}_i\}_{i\in\mathcal{I}} \rightarrow \mathcal{S}$. Note that the notation $-i$ indicates all other agents except agent i. Then, agent i obtains a reward as a function of the state and the joint action $r_i: \mathcal{S} \times \{\mathcal{A}_i\}_{i\in\mathcal{I}} \rightarrow \mathbb{R}$, and receives its private observation according to $o_i: \mathcal{S} \rightarrow \mathcal{O}_i$. Each agent aims to maximize its own total expected discounted return $R_i = \mathbb{E}_\pi[\sum_{t=1}^T \gamma^t r_i^t]$, where r_i^t denotes i's reward received at timestep t, π denotes the joint policy, and T denotes the episodic horizon.

B. Multiagent deep deterministic policy gradient

To stabilize learning in MARL, MADDPG [7] introduced 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 $Q_i^\mu(o, a_i, a_{-i}) = \mathbb{E}_\mu[R_i | o^1 = o, a^1 = \{a_i, a_{-i}\}]$, where o and μ denote the joint observation and policy, respectively. The policy gradient for agent i's deterministic policy μ_{θ_i} (abbreviated as μ_i) with respect to the expected return $J(\theta_i) = \mathbb{E}_\mu[R_i]$ is:

$$\begin{aligned}&\nabla_{\theta_i} J(\theta_i) \\ &= \mathbb{E}_{o,a\sim\mathcal{D}}[\nabla_{\theta_i}\mu_i(a_i|o_i)\nabla_{a_i} Q_i^\mu(o, a_i, a_{-i})|_{a_i=\mu_i(o_i)}],\end{aligned} \tag{1}$$

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

$$\begin{aligned}&\mathcal{L}(\theta_i) = \mathbb{E}_{o,o',a,r\sim\mathcal{D}}[Q_i^\mu(o, a_i, a_{-i}) - y]^2, \\ &\text{s.t.} \quad y = r_i + \gamma Q_i^{\mu'}(o', a_i', a_{-i}')|_{a_j'=\mu_j'(o_j'), \forall j\in\mathcal{I}},\end{aligned} \tag{2}$$

where $\mu' = \{\mu_{\theta_i'}\}_{i\in\mathcal{I}}$ denotes the set of target policies.

C. Minimax MultiAgent deep deterministic policy gradient

To learn a robust policy in MARL, an agent should account for the changes in the policy of other simultaneously learning agents. [19] proposed M3DDPG, a robust multiagent learning approach based on the minimax optimization by assuming other agents are adversarial agents. Specifically, each agent i in [19] optimizes the following learning objective:

$$\max_{\theta_i} \min_{a_{-i}\in\mathcal{B}_{-i}} Q_i^\mu(o, a_i, a_{-i})|_{a_i=\mu_i(o_i)}, \tag{3}$$

where $\mathcal{B}_{-i}$ is a compact constraint set of a_{-i} (e.g., a l_p norm ball). Because solving a general nonconvex-nonconcave minimax optimization problem is generally intractable [11], M3DDPG replaces the inner minimization with a one-step gradient descent:

$$\begin{aligned} a^*_{-i} &= \arg\min_{a_{-i}} Q^\mu_i(o, a_i, a_{-i}) \\ &\approx a_{-i} - \alpha_{-i} \nabla_{a_{-i}} Q^\mu_i(o, a_i, a_{-i}), \end{aligned} \tag{4}$$

where $\alpha_{-i} \geq 0$ denotes the learning rate.

IV. Approach

While the single-step gradient approximation in M3DDPG [19] (see Section III-C) improves robustness, we note that the framework has several limitations:

- The single-step gradient approximation can explore the locally worst situation and thus still lead to unsatisfying behavior when testing with new opponents that have drastically different strategies. Applying Equation (4) multiple times for the inner minimization can potentially alleviate this issue, but this results in a double-loop approach in solving Equation (3), which is computationally prohibitive [39].
- 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 Equation (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 issues by employing convex relaxation of neural networks and solving the inner minimization to explore the approximate *globally* worst situation while achieving *certified* robustness from the guaranteed bound of the convex relaxation.

A. Convex relaxation of neural networks

We propose to convexify the centralized action-value function in MARL and efficiently solve the inner minimization problem in Equation (3). Specifically, 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:

$$\begin{aligned} z^l &= W^l_c x^{l-1}, x^l = \sigma(z^l), \forall l = 1, ..., L-1 \\ Q^\mu_i &= W^L_c z^L, \end{aligned} \tag{5}$$

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 [31, 40]. To address this, we employ a convex relaxation technique [40] to verify robustness verification of neural networks and apply the

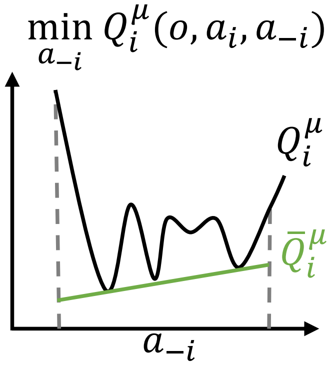

Fig. 1: Illustration of the convex relaxation approach for solving the inner minimization in Equation (7).

following linear convexification to the centralized Q-function by assuming the ReLU activation function:

$$\begin{aligned} z^l &= W^l_c x^{l-1}, x^l \leq u^l \odot (z^l - l^l) \oslash (u^l - l^l) \\ x^l &\geq 0, x^l \geq z^l, \forall l = 1, \ldots, L-1 \\ \bar{Q}^\mu_i &= W^L_c z^L, \end{aligned} \tag{6}$$

where l^l and u^l are lower and upper bounds for z^l, respectively, and $\odot$ and $\oslash$ are the element-wise multiplication and division, respectively. Note that $\bar{Q}^\mu_i$ indicates the relaxed version of Q^μ_i (i.e., $l^l \leq z^l \leq u^l$). Thanks to this relaxation, all equations in (6) are convex, so $\min_{a_{-i}} \bar{Q}^\mu_i(o, a_i, a_{-i})$ is a linear programming and can be solved efficiently. In the evaluation, we empirically show that this new certification module is computationally efficient and does not add much burden on top of a base MARL algorithm.

B. Solving minimiax optimization via convex relaxation

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

$$\begin{aligned} \bar{Q}^\mu_i(o, a_i, a^*_{-i}) &= \min_{a_{-i} \in \mathcal{B}_{-i}} \bar{Q}^\mu_i(o, a_i, a_{-i}) \\ &\leq \min_{a_{-i} \in \mathcal{B}_{-i}} Q^\mu_i(o, a_i, a_{-i}), \end{aligned} \tag{7}$$

where $\bar{Q}^\mu_i(o, a_i, a_{-i})$ is the lower bound of $Q^\mu_i(o, a_i, a_{-i})$ from the relaxation in Equation (6) (see Figure 1). $\bar{Q}^\mu_i(o, a_i, a_{-i})$ 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 Equation (7) over (4) is that the former does not need the step size hyperparameter α_{-i}, which can be difficult to tune. The performance of [19] 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:

$$\begin{aligned} &\max_{\theta_i} \left[(1-\kappa_i) Q^\mu_i(o, a_i, a_{-i}) + \kappa_i \bar{Q}^\mu_i(o, a_i, a^*_{-i})\right] \\ &\text{with} \quad a_i = \mu_i(o_i), \end{aligned} \tag{8}$$

Algorithm 1 Robust MARL via convex relaxation (ROMAX)

```
 1: Require: batch size $S$, actor learning rate $\alpha_a$, critic learning
    rate $\alpha_c$, target update $\tau$, random process $\mathcal{N}$, episode length $T$
 2: Initialize replay buffer $\mathcal{D}$
 3: for Episode=1 ... do
 4:     Initialize environment and get initial observations $o$
 5:     for $t = 1...T$ do
 6:         For each agent, select action $a_i = \mu_{\theta_i}(o_i) + \mathcal{N}_t$
 7:         Execute joint action $a$ and receive $r$ and $o'$
 8:         Store $(o, a, r, o')$ into $\mathcal{D}$, set $o \leftarrow o'$
 9:         for Each agent $i \in \mathcal{I}$ do
10:             Get $S$ samples $(o, a, r, o')$ from $\mathcal{D}$
11:             Solve inner optimization via relaxation in (10)
12:             Update critic via loss function in (10) with $\alpha_c$
13:             Solve inner optimziation via relaxation in (9)
14:             Update actor via policy gradient in (9) with $\alpha_a$
15:         end for
16:         Update target network $\theta_i = \tau\theta_i + (1-\tau)\theta'_i$
17:     end for
18: end for
```

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., $\bar{Q}_i^\mu(o, a_i, a_{-i}^*)$) 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 Equation (8). Because this inner minimization needs to be solved whenever the policy is updated, the convex relaxation problem in Equation (7) should be efficient enough with a tight bound. Therefore, there is a trade-off to choose a certain convex relaxation method among many candidates, in which we refer to the appendix for details.

C. Integrating with MARL algorithm

Our framework based on convex relaxation in Section IV-B can be readily integrated into general MARL frameworks. We implement our method based on MADDPG (see Section III-B). Integrating the minimax formulation and the convex relaxation in Equation (8) together with the actor update in Equation (1) yields:

$$\begin{aligned}\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{o,a\sim\mathcal{D}}\Big[\nabla_{\theta_i}\mu_i(o_i)\nabla_{a_i}\Big(\kappa_i \bar{Q}_i^\mu(o, a_i, a_{-i}^*) \\ + (1-\kappa_i)Q_i^\mu(o, a_i, a_{-i})\Big)\Big], \\ a_i = \mu_i(o_i), \quad a_{-i}^* = \arg\min_{a_{-i}\in\mathcal{B}_{-i}} \bar{Q}_i^\mu(o, a_i, a_{-i}),\end{aligned} \tag{9}$$

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

$$\begin{aligned}\mathcal{L}(\theta_i) = \mathbb{E}_{o,o',a,r\sim\mathcal{D}}[Q_i^\mu(o, a_i, a_{-i}) - y]^2, \\ y = r_i + \gamma\Big((1-\kappa_i)Q_i^{\mu'}(o', a'_i, a'_{-i}) \\ + \kappa_i \bar{Q}_i^{\mu'}(o', a'_i, a'^*_{-i})\Big) \\ a'_i = \mu'_i(o'_i), \quad a'^*_{-i} = \arg\min_{a'_{-i}\in\mathcal{B}'_{-i}} \bar{Q}_i^{\mu'}(o', a'_i, a'_{-i}),\end{aligned} \tag{10}$$

where $\mathcal{B}'_{-i} = \mathcal{B}'_{-i}((a'_j = \mu'_j(o'_j), \forall j \neq i), \epsilon')$ is a l_p ball centered at $(a'_j = \mu'_j(o'_j), \forall j \neq i)$ with a radius ϵ'. We summarize our algorithm in Algorithm 1.

V. Experiments

A. Evaluation domains and baselines

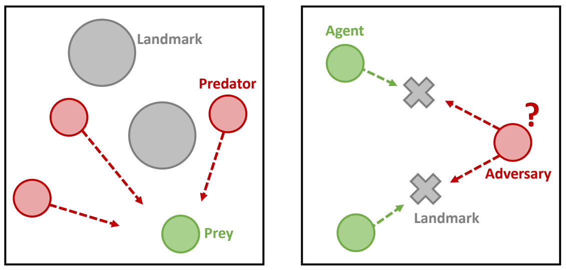

Fig. 2: Illustration of the considered tasks: Predator-prey (Left) and physical deception (Right); Reproduced from [19].

We evaluate our approach in mixed cooperative-competitive tasks from the multiagent particle benchmark [7]. 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 an important factor. Below are some descriptions of considered tasks with illustration in Figure 2.

- **Predator-prey.** $n_a = 3$ slower cooperative predators aim 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 achieved in the real world via sensors such as cameras, velocity meters, and LiDAR. Sim-to-real is known to be difficult, because the behaviors of other agents deployed in

Agent / Adv	MADDPG	M3DDPG	ROMAX	R_{Adv}
MADDPG	$\begin{pmatrix}-0.017_{\pm 0.012}, \\ -0.550_{\pm 0.017}\end{pmatrix}$	$\begin{pmatrix}0.160_{\pm 0.045}, \\ -0.502_{\pm 0.053}\end{pmatrix}$	$\begin{pmatrix}0.031_{\pm 0.020}, \\ -0.406_{\pm 0.025}\end{pmatrix}$	$0.174_{\pm 0.080}$
M3DDPG	$\begin{pmatrix}0.307_{\pm 0.043}, \\ -0.718_{\pm 0.051}\end{pmatrix}$	$\begin{pmatrix}0.250_{\pm 0.048}, \\ -0.609_{\pm 0.060}\end{pmatrix}$	$\begin{pmatrix}-0.043_{\pm 0.031}, \\ -0.290_{\pm 0.042}\end{pmatrix}$	$0.514_{\pm 0.158}$
ROMAX	$\begin{pmatrix}0.560_{\pm 0.032}, \\ -1.093_{\pm 0.037}\end{pmatrix}$	$\begin{pmatrix}0.428_{\pm 0.055}, \\ -0.936_{\pm 0.057}\end{pmatrix}$	$\begin{pmatrix}0.132_{\pm 0.020}, \\ -0.477_{\pm 0.026}\end{pmatrix}$	$1.12_{\pm 0.183}$
R_{Agent}	$-2.361_{\pm 0.230}$	$-2.047_{\pm 0.193}$	$-1.173_{\pm 0.083}$	
	MADDPG	M3DDPG	ROMAX	
$R_{\text{overall}} = R_{\text{Adv}} + R_{\text{Agent}}$	-2.187	-1.533	$\mathbf{-0.053}$	

TABLE I: 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 $5\times5=25$ pairs of random seeds. (●, ●) 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_{overall}.

Agent / Adv	MADDPG	M3DDPG	ROMAX	R_{Adv}
MADDPG	$\begin{pmatrix}-0.795_{\pm 0.017}, \\ 0.482_{\pm 0.005}\end{pmatrix}$	$\begin{pmatrix}-0.689_{\pm 0.031}, \\ 0.248_{\pm 0.020}\end{pmatrix}$	$\begin{pmatrix}-0.814_{\pm 0.032}, \\ 0.338_{\pm 0.0199}\end{pmatrix}$	$-2.298_{\pm 0.061}$
M3DDPG	$\begin{pmatrix}-0.742_{\pm 0.029}, \\ 0.225_{\pm 0.021}\end{pmatrix}$	$\begin{pmatrix}-0.819_{\pm 0.018}, \\ 0.467_{\pm 0.004}\end{pmatrix}$	$\begin{pmatrix}-0.839_{\pm 0.037}, \\ 0.271_{\pm 0.020}\end{pmatrix}$	$-2.4_{\pm 0.050}$
ROMAX	$\begin{pmatrix}-0.572_{\pm 0.0282}, \\ 0.128_{\pm 0.019}\end{pmatrix}$	$\begin{pmatrix}-0.613_{\pm 0.033}, \\ 0.133_{\pm 0.0193}\end{pmatrix}$	$\begin{pmatrix}-0.512_{\pm 0.010}, \\ 0.283_{\pm 0.003}\end{pmatrix}$	$-1.697_{\pm 0.048}$
R_{Agent}	$0.835_{\pm 0.150}$	$0.848_{\pm 0.139}$	$0.892_{\pm 0.033}$	
	MADDPG	M3DDPG	ROMAX	
$R_{\text{overall}} = R_{\text{Adv}} + R_{\text{Agent}}$	-1.463	-1.552	$\mathbf{-0.805}$	

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

the environment in the real-world may differ significantly from the simulation (e.g., due to varying transition dynamics). This is exactly what this work aims to address: the certified and improved robustness will enhance the resilience and applicability of multiagent algorithms from sim-to-real. Lastly, the learned policy can be easily transferred on-board, and generated actions can be further executed by a lower-level controller if necessary.

Baselines. We compare ROMAX to M3DDPG [19], 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 [7], which uses the centralized critic but does not solve minimax. Implementation details and hyperparameters are specified in the appendix.

B. Results

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

To answer this question and test 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* random seeds; vice-versa for the agent's policy.

As Table 1 and 2 demonstrate, for both tasks, ROMAX 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_{Adv} columns in the tables. Similar conclusion can be made for the agents given the R_{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 Equation (4). This observation implies that a tuned step size for one task cannot generalize to another one and also shows the advantage of ROMAX. Regarding the computation efficiency, we empirically observe that the factor between wall-clock time per iteration of ROMAX (with certification) and that of MADDPG (without certification), is close to 1 (i.e., 1.08, averaged among multiple seeds). This validates that our certification module is computationally efficient.

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

To answer this question, we construct a disruptive policy in the predator-prey task by 1) training both teams with

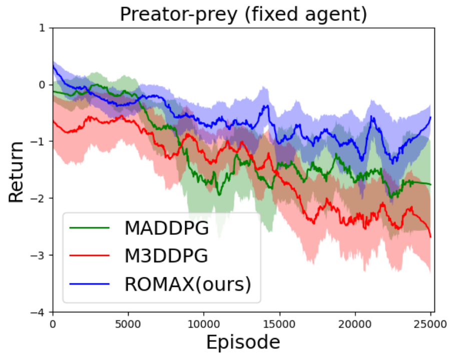

Fig. 3: The performance of the fixed agent (prey) during the training of disruptive adversaries (predators) with 3 seeds. Higher return implies a more robust policy.

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 3, 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, validating the robustness advantage of our approach. We observe that M3DDPG and MADDPG perform similarly in this analysis, possibly due to the sensitive tuning of the step size.

VI. 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.

Appendix

A. Repositories

The multiagent particle environments we used in simulation is from `https://github.com/openai/multiagent-particle-envs`. We use the implementation of the base algorithm MADDPG from `https://github.com/shariqiqbal2810/maddpg-pytorch`. Note that with relaxations in Equation (6), $\bar{Q}_i^{\mu}(o, a_i, a_{-i}^*)$ is no longer an explicit function with respect to its input without constraints. Then the framework auto_LiRPA (`https://github.com/KaidiXu/auto_LiRPA`, [41]) is used to get $\bar{Q}_i^{\mu}(o, a_i, a_{-i}^*)$ efficiently and automatically.

B. Choice of convex relaxation methods

For robustness verification of neural networks there are many convex relaxation based methods, from which we need to choose one for Equation (7). When there is a trade-off to choose a certain convex relaxation method among many candidates, we can get $\bar{Q}_i^{\mu}(o, a_i, a_{-i}^*)$ as a convex combination of the bounds from different methods [42]. For example, Interval Bound Propagation (IBP, [43]) and CROWN-IBP [42] have their respective strengths and shortcomings in terms of bound tightness, sensitivity to hyper-parameters, computational cost with the training going on. As a result, we can have:

$$\begin{aligned}\bar{Q}_i^{\mu}(o, a_i, a_{-i}^*) = \beta \bar{Q}_{i,\text{IBP}}^{\mu}(o, a_i, a_{-i}^*)\\ + (1-\beta)\bar{Q}_{i,\text{CROWN-IBP}}^{\mu}(o, a_i, a_{-i}^*),\end{aligned} \tag{11}$$

with $\beta \in [0, 1]$ a tunable parameter which can change with the training iteration index increasing. As both $\bar{Q}_{i,\text{IBP}}^{\mu}(o, a_i, a_{-i}^*)$ and $\bar{Q}_{i,\text{CROWN-IBP}}^{\mu}(o, a_i, a_{-i}^*)$ are the lower bounds of $Q_i^{\mu}(o, a_i, a_{-i})$, so are their convex combination $\bar{Q}_i^{\mu}(o, a_i, a_{-i}^*)$. Hence, the property of certified robustness is kept.

C. Hyperparameter

Some key hyperparameters are shown in Table III.

Episode length	25	batch size	1024
NN hidden dim	64	τ	0.01
learning rate	0.01	$\epsilon_{\max}$	0.1
$\beta_{\min}$	0.9	γ	0.99

TABLE III: Hyperparameters choices in the implementation.

Acknowledgements

Research supported by Scientific Systems Company, Inc. under research agreement # SC-1661-04 and ARL DCIST under Cooperative Agreement Number W911NF-17-2-0181. Dong-Ki Kim was supported by IBM, Samsung (as part of the MIT-IBM Watson AI Lab initiative), and Kwanjeong Educational Foundation Fellowship. We thank Amazon Web services for computational support.

References

[1] S. Mellouli, "A reorganization strategy to build fault-tolerant multi-agent systems," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2007, pp. 61–72.

[2] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[3] P. Culbertson, J.-J. Slotine, and M. Schwager, "Decentralized adaptive control for collaborative manipulation of rigid bodies," *IEEE Transactions on Robotics*, 2021.

[4] C. Vasile, A. Pavel, and C. Buiu, "Integrating human swarm interaction in a distributed robotic control system," in *2011 IEEE International Conference on Automation Science and Engineering*. IEEE, 2011, pp. 743–748.

[5] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sk2u1g-0-

[6] D. K. Kim, M. Liu, M. D. Riemer, C. Sun, M. Abdulhai, G. Habibi, S. Lopez-Cot, G. Tesauro, and J. How, "A policy gradient algorithm for learning to learn in multiagent reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 5541–5550. [Online]. Available: https://proceedings.mlr.press/v139/kim21g.html

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[8] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 32, no. 1, 2018.

[9] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *arXiv preprint arXiv:1905.10615*, 2019.

[10] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[11] C. Daskalakis, S. Skoulakis, and M. Zampetakis, "The complexity of constrained min-max optimization," *arXiv preprint arXiv:2009.09623*, 2020.

[12] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *CoRR*, vol. abs/1707.09183, 2017. [Online]. Available: http://arxiv.org/abs/1707.09183

[13] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International Conference on Machine Learning (ICML)*, vol. 80, 10–15 Jul 2018, pp. 5571–5580. [Online]. Available: http://proceedings.mlr.press/v80/yang18d.html

[14] Y. Wen, Y. Yang, R. Luo, J. Wang, and W. Pan, "Probabilistic recursive reasoning for multi-agent reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=rkl6As0cF7

[15] D.-K. Kim, M. Liu, S. Omidshafiei, S. Lopez-Cot, M. Riemer, G. Habibi, G. Tesauro, S. Mourad, M. Campbell, and J. P. How, "Learning hierarchical teaching policies for cooperative agents," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 620–628.

[16] J. Perolat, F. Strub, B. Piot, and O. Pietquin, "Learning Nash Equilibrium for General-Sum Markov Games from Batch Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 232–241. [Online]. Available: http://proceedings.mlr.press/v54/perolat17a.html

[17] J. Grau-Moya, F. Leibfried, and H. Bou-Ammar, "Balancing two-player stochastic games with soft q-learning," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 268–274. [Online]. Available: https://doi.org/10.24963/ijcai.2018/37

[18] M. Osborne, *An introduction to game theory*. New York, NY [u.a.]: Oxford Univ. Press, 2004. [Online]. Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+369342747&sourceid=fbw_bibsonomy

[19] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4213–4220.

[20] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1328–1337.

[21] M. Shen and J. P. How, "Robust opponent modeling via adversarial ensemble reinforcement learning," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, no. 1, pp. 578–587, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/16006

[22] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari,

go, chess and shogi by planning with a learned model," *Nature*, vol. 588 7839, pp. 604–609, 2020.

[23] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," *CoRR*, vol. abs/1711.09846, 2017. [Online]. Available: http://arxiv.org/abs/1711.09846

[24] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, and et al., "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, p. 859–865, May 2019. [Online]. Available: http://dx.doi.org/10.1126/science.aau6249

[25] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Sy0GnUxCb

[26] C. Zhang and V. R. Lesser, "Multi-agent learning with policy prediction," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

[27] J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," in *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018, p. 122–130.

[28] J. Foerster, G. Farquhar, M. Al-Shedivat, T. Rocktäschel, E. Xing, and S. Whiteson, "DiCE: The infinitely differentiable Monte Carlo estimator," in *International Conference on Machine Learning (ICML)*, vol. 80, 10–15 Jul 2018, pp. 1524–1533. [Online]. Available: http://proceedings.mlr.press/v80/foerster18a.html

[29] A. Letcher, J. Foerster, D. Balduzzi, T. Rocktäschel, and S. Whiteson, "Stable opponent shaping in differentiable games," in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=SyGjjsC5tQ

[30] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*. Springer, 2017, pp. 97–117.

[31] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5276–5285.

[32] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. T. Vechev, "Fast and effective robustness certification." *NeurIPS*, vol. 1, no. 4, p. 6, 2018.

[33] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," *arXiv preprint arXiv:1809.08098*, 2018.

[34] G. Singh, T. Gehr, M. Püschel, and M. T. Vechev, "Boosting robustness certification of neural networks." in *ICLR (Poster)*, 2019.

[35] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *arXiv preprint arXiv:1811.00866*, 2018.

[36] A. Raghunathan, J. Steinhardt, and P. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," *arXiv preprint arXiv:1811.01057*, 2018.

[37] K. D. Dvijotham, R. Stanforth, S. Gowal, C. Qin, S. De, and P. Kohli, "Efficient neural network verification with exactness characterization," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 497–507.

[38] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robust verification of neural networks," *arXiv preprint arXiv:1902.08722*, 2019.

[39] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *CoRR*, vol. abs/1803.02999, 2018. [Online]. Available: http://arxiv.org/abs/1803.02999

[40] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5286–5295.

[41] K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh, "Automatic perturbation analysis for scalable certified robustness and beyond," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[42] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh, "Towards stable and efficient training of verifiably robust neural networks," *arXiv preprint arXiv:1906.06316*, 2019.

[43] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.