ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230632

Robustness Testing for Multi-Agent Reinforcement Learning: State Perturbations on Critical Agents

Ziyuan Zhou[a] **and Guanjun Liu**[a;*]

[a]Department of Computer Science, Tongji University, Shanghai, China
ORCiD ID: Ziyuan Zhou https://orcid.org/0000-0002-2649-8666,
Guanjun Liu https://orcid.org/0000-0002-7523-4827

Abstract. Multi-agent reinforcement learning (MARL) has been widely applied in many fields, such as smart traffic and unmanned aerial vehicles. However, most MARL algorithms are vulnerable to adversarial perturbations on agent states. Robustness testing for a trained model is an essential step for confirming the trustworthiness of the model against unexpected perturbations. This work proposes a novel Robustness Testing framework for MARL that attacks states of Critical Agents (RTCA). The RTCA has two innovations: 1) a differential evolution (DE) based method to select critical agents as victims and to advise the worst-case joint actions on them, and 2) a team cooperation policy evaluation method employed as the objective function for the optimization of DE. Then, adversarial state perturbations of the critical agents are generated based on the worst-case joint actions. This is the first robustness testing framework with varying victim agents. RTCA demonstrates outstanding performance in terms of the number of victim agents and destroying cooperation policies.

1 Introduction

Multi-agent reinforcement learning (MARL) is widely utilized in multi-agent systems (MAS) such as smart transportation [33, 38] and unmanned aerial vehicles [3, 25, 31, 34] as a result of its superior performance in team decision-making problems. As the number of agents increases and the joint-action space of MAS grows exponentially, MARL faces the issues of high combinatorial complexity and poor scalability. Centralized training and decentralized execution (CTDE) is currently the more popular framework to address these problems, in which all agents share global information in the training process. In contrast, in the execution phase, each agent makes independent decisions based on its own perceptions and policy. Value decomposition networks (VDN) [30] and monotonic value function factorization (QMIX) [22] are classical CTDE-based MARL. They introduce a network in the training process to guarantee the Individual-Global-Max (IGM) [26].

However, it is shown that the agents trained by these classical MARL are sensitive to state perturbation [6, 8, 15, 39]. In reality, the states of agents are often perturbed due to the presence of sensor noise and malicious attacks. Furthermore, state perturbations to some of the agents can not only mislead the decision of victims but also impact the cooperative policy of the team. Robustness testing for a model trained by MARL is essential for confirming the trustworthiness of MAS against unexpected perturbations. The perturbation types are so diverse that it is impossible to cover all possible cases during testing. Consequently, it is vital to generate perturbation states via adversarial attacks that significantly impact team collaboration. Intuitively, the more stealthy and disruptive the attack is, the more potential it is for robustness testing.

There has been a lot of robustness testing technique on Single-agent reinforcement learning (SARL), such as using the adversarial attack based on the gradient of the neural network [9, 37] or constructing an adversary as an RL agent [36] to generate the adversarial observation. However, there have been few related research to test the robustness of MARL against state perturbation of agents. Zhou et al. [39] demonstrate that the adversary for MARL can be formulated as the stochastic game (SG), and the joint optimal adversarial state exists. But the influence of individual policy on teams is not considered during the attack in [6, 39]. They only generate adversarial observation misleading the victim to take actions that are not within expectations, which may not lead to the failure of team tasks. The methods in [7, 14, 15] consider the effect of individual actions on teams by constructing the adversary as SARL or MARL agent. However, the adversary is trained at the assumption that the victim is determined. When the victim changes, it needs to be retrained.

Compared with single-agent situations, multi-agent situations face the following challenges:

1) The victims are uncertain, making the robustness testing process unable to be formulated as SG. The testing process becomes more difficult due to this uncertainty.
2) An agent doing a sub-optimal action does not necessarily lead to team failure.
3) The centralized training process is usually unknown and cannot be employed during testing. Besides, centralized training is based on the assumption that all agents make the optimal decision. As a result, it may not accurately evaluate the team reward when the agent takes the sub-optimal action. How to estimate the team reward for the adversary is important.

Facing these challenges, this work aims to propose a stealthy and effective attack method on states of Critical Agents for Robustness Testing (RTCA), which has the following novel contributions:

1) We propose a novel robust testing framework with two steps. The first step, based on differential evolution (DE) aims to select critical agents as victims and determine the worst joint actions they

* Corresponding Author. Email: liuguanjun@tongji.edu.cn

should take to decrease the accumulated reward of the team. The second step is a targeted attack to compute adversarial observation according to the outcomes of the first step.

2) We introduce a Sarsa-based approach for learning the joint action-value function in MARL to evaluate the team cooperation policy. The function clearly represents the relationship between individual actions and team-accumulated rewards and is used as the objective function for optimizing DE.
3) The results demonstrate that RTCA performs better when attacking fewer agents. RTCA is more suitable for robustness testing of models trained by MARL due to its stealthiness and effectiveness.

2 Preliminary

In this section, we introduce decentralized partially observable Markov processes (Dec-POMDPs) and state-adversarial stochastic games (SASG). Important concepts are described as follows:

- Adversarial perturbation aims to mislead the actions of agents to minimize the expected cumulative discount reward of the team.
- Clean observation is generated from the environment state.
- Adversarial observation is an observation after adding an adversarial perturbation to a clean observation.
- The adversary aims to generate the adversarial perturbation.
- Victim is the agent whose observation is perturbed by the adversary, i.e., attacked by the adversary.
- Critical agents are those whose actions impact the team most at a time step.

2.1 Decentralized Partially Observable Markov Processes

A Dec-POMDP [18] is defined as a tuple

$$\left\langle \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \mathcal{N}, Z, p, r, \gamma \right\rangle$$

where $\mathcal{N}$ is the set of agents, the number of agents is $N \triangleq |\mathcal{N}|$, each agent $i \in \mathcal{N} \triangleq \{1, \cdots, N\}$, $\mathcal{S}$ is the state set of the environment, the state $s \in \mathcal{S}$, $a^i \in \mathcal{A}^i$ is the action of agent i, $\mathcal{A}^i$ is the action space of the agent i, $o^i \in \mathcal{O}^i$ drawn according to observation function $Z(s, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathcal{O}^1 \times \cdots \times \mathcal{O}^N$ is the observation of agent i, $\mathcal{O}^i$ is the observation space of agent i, r is the immediate reward, for each agent $r^1 = \cdots = r^N = r$, $R(s, \boldsymbol{a}, s') : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \times \mathcal{S} \to \mathbb{R}$ is the reward function of all agents, $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Delta(\mathcal{S})$ is the transition probability based on the joint action $\boldsymbol{a}$ and $\gamma \in [0, 1]$ is the discount factor over time. The environment is the partial observation. Thus, the observation of the agent is equal to the agent state in this paper.

It is NEXP-complete to solve Dec-POMDPs due to the combinatorial problem [2]. CTED is a potential learning paradigm to solve Dec-POMDPs. A centralized controller exists to evaluate the team cooperation policy based on the environmental state in the training process. And in the execution process, each agent takes action according to its observation. One of the most classical solutions is via Q-function factorization, including VDN [30] and QMIX [22]. The factorization needs to satisfy the IGM condition [26], i.e.,

$$\arg\max_{\boldsymbol{a}} Q_{jt}(\boldsymbol{\tau}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{a^1} Q_1\left(\tau^1, a^1\right) \\ \cdots \\ \arg\max_{a^N} Q_N\left(\tau^N, a^N\right) \end{pmatrix}, \quad (1)$$

where $\boldsymbol{\tau} \in \mathcal{T}^1 \times \cdots \times \mathcal{T}^N$ is the joint action-observation histories, $\mathcal{T}^i$ is the set of action-observation histories of agent i, $Q_{jt} : \mathcal{T}^1 \times \cdots \times \mathcal{T}^N \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$ is the joint action-value function to evaluate the team cooperation policy.

VDN and QMIX are methods to provide additivity and monotonicity, two sufficient conditions for IGM, respectively:

$$Q_{jt}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{N} Q^i\left(\tau^i, a^i\right), \quad (2)$$

$$\frac{\partial Q_{jt}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q^i(\tau^i, a^i)} \geq 0, 0 \leq i < N. \quad (3)$$

2.2 State-Adversiral Stochastic Game

An SASG [39] is defined as a tuple

$$\left\langle \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\mathcal{B}^i\}_{i\in\mathcal{M}}, \{r^i\}_{i\in\mathcal{N}}, \mathcal{M}, \mathcal{N}, p, \gamma \right\rangle$$

where $\mathcal{B}^i$ is the set of adversarial states of agent i, $\mathcal{M} \subseteq \mathcal{N}$ is the set of victim agents, and the number of victims is $M \triangleq |\mathcal{M}|$. Assume that the adversarial perturbation $v^i(s)$ of agent i is deterministic $v^i : \mathcal{S} \to \mathcal{B}^i$. The value and action-value functions of SASG are

$$\widehat{V}^i_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s) = \mathbb{E}_{\boldsymbol{\pi}\circ\boldsymbol{v},p}\left(\sum_{k=0}^{\infty} \gamma^k r^i_{t+k+1} | s_t = s\right), \quad (4)$$

$$\widehat{Q}^i_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}\circ\boldsymbol{v},p}\left(\sum_{k=0}^{\infty} \gamma^k r^i_{t+k+1} | s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\right), \quad (5)$$

where $\boldsymbol{\pi} \in \prod^1 \times \cdots \times \prod^N : \mathcal{S} \to \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ is the joint policy, $\pi^i \in \prod^i : \mathcal{S} \to \mathcal{A}^i$ is the policy of agent i, $\prod^i$ is the policy space of agent i, $\boldsymbol{v} \triangleq \left(v^i, \cdots, v^M\right)$ denotes the joint adversarial perturbation and $\boldsymbol{\pi} \circ \boldsymbol{v}$ denotes the joint policy under the joint adversarial perturbation.

The value function and the action-value function can respectively be written as follows:

$$\widehat{V}^i_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}(s) = \min_{v^i} \widehat{V}^i_{\boldsymbol{\pi}\circ\left(v^i, \boldsymbol{v}_*^{-i}\right)}(s), \quad (6)$$

$$\widehat{Q}^i_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}(s, \boldsymbol{a}) = \min_{v^i} \widehat{Q}^i_{\boldsymbol{\pi}\circ\left(v^i, \boldsymbol{v}_*^{-i}\right)}(s, \boldsymbol{a}), \quad (7)$$

where $\boldsymbol{v}_* \triangleq \left(v^1_*, \cdots, v^M_*\right)$ is the joint optimal adversarial perturbation, v^i is an arbitrary valid adversarial perturbation and $\boldsymbol{v}_*^{-i} \triangleq \left(v^1_*, \cdots, v^{i-1}_*, v^{i+1}_*, \cdots, v^M_*\right)$. The joint optimal adversarial perturbation aims to minimize every victim agent's expected cumulative discount reward.

In [39], the properties of SASG are discussed, including the existence and contraction of the joint optimal adversarial perturbation. And they point out that solving the joint optimal adversarial perturbation is equal to solving an SG, which MARL can solve. Therefore, there is some research to solve this one via MARL [7, 14, 15]. However, the victim agents are certain in the training process. If the victims are changing, MARL models have to retrain. We solve this problem in the next section.

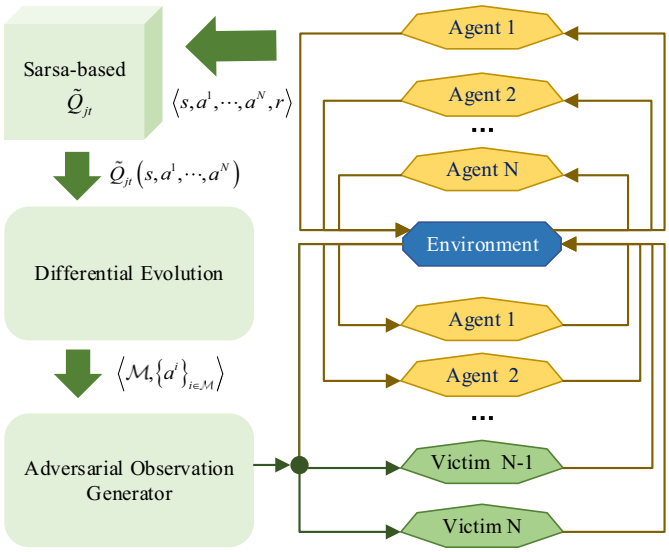

Figure 1. The illustration of RTCA.

3 Robustness Testing on Critical Agent States

In this section, we define agent State Adversarial Dec-POMDPs (SA-Dec-POMDPs) and propose a novel framework to solve it.

Definition 1 (SA-Dec-POMDPs) *An SA-Dec-POMDP is defined as a tuple*

$$\left\langle \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \{\mathcal{B}^i\}_{i\in\mathcal{M}}, \mathcal{N}, \mathcal{M}, Z, r, p, \gamma \right\rangle$$

where $\mathcal{B}$ is the set of the adversarial observation of agent i. Assume that the adversarial perturbation $v^i(\tau^i)$ of agent i is a deterministic function $v^i : \mathcal{T}^i \to \mathcal{B}^i$.

Similar to SASG, the adversary aims to minimize the expected cumulative discount reward of the team via manipulating the observation of the victim. The value function and the action-value function of the adversary can be written as follows:

$$\widehat{V}^i_{\pi^i \circ v^i_*}\left(\tau^i\right) = \min_{v^i} \widehat{V}^i_{\pi^i \circ v^i}\left(\tau^i\right), \tag{8}$$

$$\widehat{Q}^i_{\pi^i \circ v^i_*}\left(\tau^i, a^i\right) = \min_{v^i} \widehat{Q}^i_{\pi^i \circ v^i}\left(\tau^i, a^i\right). \tag{9}$$

Obviously, solving the optimal joint policy for the adversary in SA-Dec-POMDPs is equivalent to solving the optimal joint policy for the agent in Dec-POMDPs. The action space of the agent in Dec-POMDPs is the adversarial observation space of the adversary in SA-Dec-POMDPs. Given this, we can leverage CTDE-based MARL to address this problem. However, it should be noted that victim agents are known with certainty, including the number of victims M and their respective indexes. When the value of M changes, the SA-Dec-POMDP also changes, it is necessary to retrain the adversary agent via CTDE-based MARL.

Not only in SA-Dec-POMDPs but SG and SASG, the set of agents $\mathcal{N}$ is a certainty. And learning a MARL model to generate adversarial observations is challenging when the space of one $\mathcal{B}^M$ is large. In this paper, we propose a novel robustness testing framework, i.e., solving of SA-Dec-POMDPs named RTCA as shown in Figure 1 that can choose critical agents from the set of $\mathcal{N}$ as the set of victims $\mathcal{M}$ and provide guidance on the worst joint action of victims, followed by generating adversarial observations on them. Furthermore, our RTCA does not require any training, allowing for a variable set of victims $\mathcal{M}$ to be used.

3.1 Selection of critical agents and worst actions

We present a method based on DE for selecting critical agents and worst joint actions in MARL. The goal of this method is to identify those agents and their actions that have the greatest influence on the overall performance of the system.

Motivated by CTDE, if the joint action-value function is known for the adversary, the objective of the adversary is as follows:

$$v_{de}\left(\boldsymbol{\tau}\right) = \arg \min_{\mathcal{M}, \{a^i_{i\in\mathcal{M}}\}} Q_{jt}\left(\boldsymbol{\tau}, \boldsymbol{a}^M, \boldsymbol{a}^{-M}\right) \tag{10}$$

where $\boldsymbol{a}^M$ and $\boldsymbol{a}^{-M}$ are the joint action of the victim and the other agents, respectively, we employ DE to accomplish this objective. The DE algorithm is a global optimization algorithm [27, 4, 19] that the concept of population evolution for exploring the global optimal solution and is extensively applied to solve complex optimization problems. It maintains a population of candidate solutions represented as vectors in a high-dimensional search space. During each iteration of the DE algorithm, new candidate solutions are generated by perturbing the vectors in the population and combining them using a simple arithmetic operator. The fitness of candidate solutions is then evaluated using the objective function, and the best ones are selected to form the next generation of the population.

The main advantage of the DE algorithm is the use of a differential mutation operator, which perturbs the candidate solutions in a way that is guided by the difference between two randomly selected vectors from the population. This operator allows the algorithm to explore the search space efficiently and converge to a high-quality solution relatively quickly.

In this paper, we encode the critical agent and the corresponding worst joint action into a candidate solution. One candidate solution contains the actions of M victims, and that is a tuple of $2M$ elements: the indexes of critical agents and their joint worst actions $a^i{}_{i\in\mathcal{M}}$. We use the same setting with [28]. At the start of the algorithm, a population of 400 candidate solutions D is generated. During every iteration, an additional 400 candidate solutions are generated utilizing the following standard DE formula:

$$\begin{gathered} D_j\left(g+1\right) = D_{d_1}\left(g\right) + F\left(D_{d_2}\left(g\right) - D_{d_3}\left(g\right)\right) \\ d_1 \neq d_2 \neq d_3, \end{gathered} \tag{11}$$

where g is the current generation index, and each element D_j of the candidate solution is combined with three randomly chosen elements using a scaling parameter F set to 0.5. The indices of the randomly chosen elements are represented by d_1, d_2, and d_3. The whole process is presented in Algorithm 1.

By doing so, we can identify which agents are most critical to the team performance, and which actions on them are most likely to lead to failure or sub-optimal outcomes. The advantages of using DE to compute the index of critical agents and their joint worst actions are as follows:

- Less information. The joint action-value function is not necessarily differentiability. In fact, the joint action-value network of QMIX is non-differentiability, making it difficult for the adversary to estimate the team reward. Furthermore, we consider the

Algorithm 1: Selection of critical agents and worst actions

Input: Objective function Q_{jt}, the joint action-observation histories $\boldsymbol{\tau}$, the joint policy of agents $\boldsymbol{\pi}$, the value $Q_{jt}(\boldsymbol{\tau}, \boldsymbol{a})$, population size 400, scaling factor $F = 0.5$, crossover rate CR, maximum number of iterations T

Output: $D_{best} \in D \triangleq \{\mathcal{M}, \{\hat{a}^i\}_{i\in\mathcal{M}}\}$

Initialize population D with 400 random solutions; and Evaluate the fitness of each solution in the population based on Q_{jt}; **while** $t = 1, 2, \cdots, T$ **do**
  for *each solution* $x_j \in D$ **do**
    Randomly select three distinct solutions D_{d_1}, D_{d_2} and D_{d_3} from D;
    Generate a mutant vector D_j according to (11);
    Generate a trial vector u_j by performing a binary crossover between x_j and D_j with crossover rate CR;
    Generate joint action of the victim agent $\boldsymbol{a}_u^M$ and $\boldsymbol{a}_D^M$ based on u_j and D_j, respectively.
    Evaluate the fitness of u_j;
    if $Q_{jt}(\boldsymbol{\tau}, \boldsymbol{a}_u^M, \boldsymbol{a}^{-M}) < Q_{jt}(\boldsymbol{\tau}, \boldsymbol{a}_D^M, \boldsymbol{a}^{-M})$ **then**
      Replace D_j with u_j in the population D;
  $t = t + 1$;
Select and return the best solution found, D_{best}, from P.

case of a discrete action space and the situation where a subset of agents becomes the victim. Even if the joint action-value network is differentiable, such as in VDN, gradient-based attack methods cannot be used to solve the worst-case joint action. DE is ideal for addressing this type of problem.

- More flexibility. It is unnecessary to know the structure of the joint action-value function; only its inputs and outputs are required. Therefore, networks trained in any way can be attacked. In addition, the number of victims M only needs to remain constant during the current time step and can be changed arbitrarily without retraining.

3.2 Sarsa-based joint action-value function

In Section 4.1, we assume that the joint action-value function is known to the adversary. However, in reality, this component is not being deployed. We cannot directly compute Q_{jt}, and more importantly, the training of Q_{jt} in CTDE is based on the assumption that all agents make the optimal decision. As a result, the joint action-value function in VDN or QMIX may not evaluate the team reward. To solve this problem, we train a joint action-value network based on Sarsa [1] during the execution process.

Since the $\boldsymbol{\pi}$ is fixed in the execution process, we use ϵ-greedy to explore all situations possible, not just the good trajectories. We construct the joint action-value function $\tilde{Q}_{jt}$ as a neural network with parameters θ. Its input is the environment state s and the joint action of agent $\boldsymbol{a}$. Learning object of $\tilde{Q}_{jt}$ is to minimize:

$$L_{jt}(\theta) = \sum_{k=1}^{BS} \left[R_k + \gamma \tilde{Q}_{jt}\left(s_k', \boldsymbol{a}_k'\right) - \tilde{Q}_{jt}\left(s_k, \boldsymbol{a}_k\right) \right], \tag{12}$$

where BS is the batch size. The training process is shown in Algorithm 2.

Algorithm 2: Sarsa-based joint action-value function

Initialize $\tilde{Q}_{jt}^{\theta}$ and replay buffer $\mathcal{R}$;
for $t = 1, 2, \ldots, T$ **do**
  Get the current state of the environment s and the observation of each agent $\left[o^1, \cdots, o^N\right]$;
  Take the ϵ-greedy joint action $\boldsymbol{a}$ according to $\left[Q_1\left(\tau^1, a^1\right), \cdots, Q_N\left(\tau^N, a^N\right)\right]$ and get the next state s' and observation $\left[o'^1, \cdots, o'^N\right]$;
  Put $\langle s, \boldsymbol{a}, r, s' \rangle$ in $\mathcal{R}$;
  if $|\mathcal{R}| \geq BS$ **then**
    Sample BS transitions from replay buffer $\mathcal{R}$ $\{\langle s_k, \boldsymbol{a}_k, r_k, s_k' \rangle\}_{k=1,2,\cdots,BS}$;
    Update the parameters θ via minimizing (12).

Algorithm 3: RTCA

Initialize the environment.
for $t = 1, 2, \ldots, T$ **do**
  Get the current state of the environment s and the observation of each agent $\left[o^1, \cdots, o^N\right]$;
  Generate the victim index and the worst action according to Algorithms 1 and 2;
  Generate the adversarial state according to (14);
  Take action based on the clean observation for the normal agent and based on the adversarial observation for the victim agent.

3.3 Generation of adversarial observations

After obtaining the index of the critical agent and the worst joint action, we aim to generate adversarial perturbation using these to induce the victim to take the worst actions.

In the classification tasks, there are many methods of targeted attacks[5, 12, 13, 17], which add the perturbation to the sample to make the neural network output the targeted label. In our setting, this type of method can be used to generate adversarial perturbations based on the policy of the victim agents, misleading them to output the targeted action. This process can be achieved by optimizing the following objective function,

$$L_{sa}\left(\theta^i, \hat{o}^i\right) = \min_{\hat{o}^i \in \mathcal{B}^i} \Bigg\{ - \sum_{a^j \in \mathcal{A}^i} \left[\hat{a}^i = a^j\right] \log\left(\pi_{\theta^i}\left(a^j | \hat{o}^i\right)\right) + \sum_{a^j \in \mathcal{A}^i} \left[z^i = a^j\right] \log\left(\pi_{\theta^i}\left(a^j | \hat{o}^i\right)\right) \Bigg\}, \tag{13}$$

where z^i is the action of victim agent i based on the clean observation, θ^i is the parameter of the policy network of victim agent i, $\hat{a}^i$ is the target action of victim agent i and $\hat{o}^i$ is the perturbation observation of agent i. This optimization makes the victim take the action which is closer to the target action and farther to the true one. We use Fast Gradient Sign Method (FGSM) [5] to address this optimization problem. FGSM uses the one-step calculation of gradient descent as follows:

$$\hat{o}^i = \text{clip}\left(\hat{o}^i + \alpha \text{sgn}\left(\nabla_{\hat{o}^i} L_{sa}\left(\theta^i, \hat{o}^i, \hat{a}^i\right)\right)\right), \tag{14}$$

where α is the step size. Our RTCA is presented in Algorithm 3.

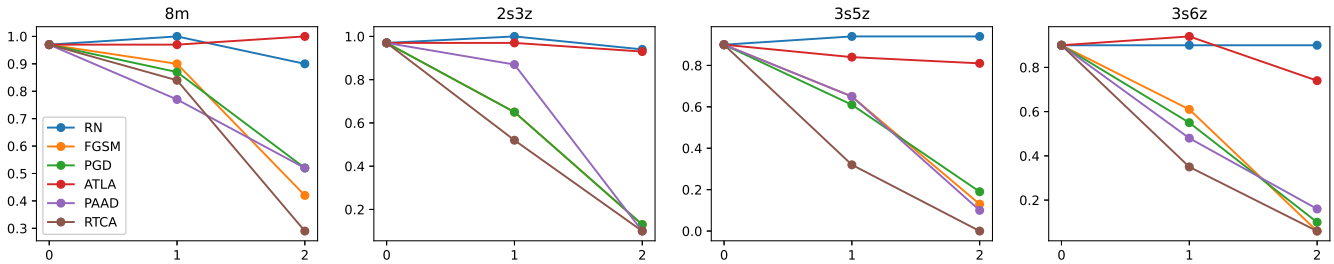

Figure 2. The illustration of robustness test for QMIX. The horizontal coordinate indicates the number of victims, and the vertical coordinate indicates the winning rate.

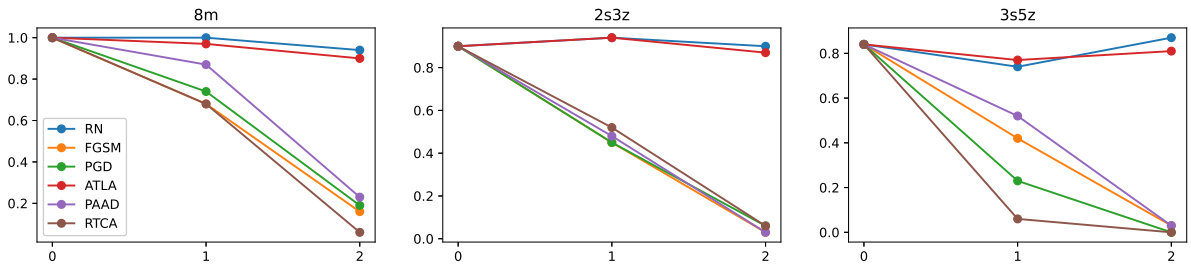

Figure 3. The illustration of robustness test for VDN. The horizontal coordinate indicates the number of victims, and the vertical coordinate indicates the winning rate.

4 Experiment Results and Analysis

In this section, we demonstrate the outstanding performance of RTCA in generating adversarial observation from the perspective of the number of victims and decreasing the team reward. The code is available at: https://github.com/zhou-ziyuan/RTCA.

4.1 Experiment settings

4.1.1 Environment settings

We evaluate our robustness testing framework on the StarCraft multi-agent challenge (SMAC) [23]. SMAC is a distributed real-time strategy game widely used to evaluate the performance of RL. The main task is to defeat the opponent through cooperation among agents. We conduct experiments on four maps containing 8 Marines (8m), 2 Stalkers & 3 Zealots (2s3z), 3 Stalkers & 5 Zealots (3s5z), and 3 Stalkers & 6 Zealots (3s6z).

- Observation space: At each time step, the agents obtain information about their field of view, including the following information about their opponent and teammates: distance, relative x, y, health, shield, and unit type. The state of the environment contains information about all units on the map. Specifically, the coordinates of all agents relative to the center of the map, as well as the elements present in their observations are contained. The global state is only used in the centralized training process. During the testing process, we only attack the observations of the agents.
- Action space: In the SMAC environment, each agent has access to four possible actions: move, attacking opponent, stop, and no-op. Movement is restricted to four cardinal directions: north, south, east, or west. The attack can only be performed if the enemy unit is within the shooting range. The stop is to do nothing and the no-op can take when the agent is dead.
- Reward: The overall goal is to increase the win rate of the agents as much as possible. SMAC uses shaped reward by default, and at each time step, the agents receive a reward based on the hit point damage. In addition, the agent receives an additional reward for each opponent it defeats and for winning by defeating all opponents. All of these rewards are normalized to ensure that the maximum cumulative reward earned in an episode is 20.

4.1.2 Benchmark methods

The victim agent is trained via VDN and QMIX with two million steps. We compare RTCA with the following state-of-art robustness testing method.

- Random noise (RN). The adversary adds some random perturbations to the observation of the victim agent. We use uniform distribution noise as a way to interfere with the decision-making of the victim.
- FGSM and PGD [5, 17]. The adversary generates the adversarial perturbation according to the gradient on the decision-making network of the victim. It does not consider the effect of the perturbation on team cumulative reward, but only the disruption of individual policy, i.e., making the victim do actions that are inconsistent with those in the clean observation. We set the step of PGD as 20.
- ATLA [36]. In the single-agent scenario, the adversary is trained via PPO [24]. We use MAPPO [35] as adversaries in the multi-agent scenario. For MAPPO agents, the input is the clean observation, the action is the adversarial observation, and the reward is the negative of victimization teams. In the training process, we set all agents in the environment are victims.
- PAAD [29]. The adversary is trained in two steps: the director gives advice for the worst joint action, and the actor generates the adversarial observation based on this action. The director is a PPO

Table 1. The performance of QMIX and VDN in the number of victims is 0, 1, and 2 under RN. WR is the winning rate.

Envs	Number of victims	0		1		2	
	Victim	WR	Reward	WR	Reward	WR	Reward
8m	QMIX	0.97	19.74±1.45	1.00	20.00	0.90	19.31±2.10
	VDN	1.00	20.00	1.00	20.00	0.94	19.48±1.99
2s3z	QMIX	0.97	19.73±1.47	1.00	20.00	0.94	19.72±1.08
	VDN	0.90	19.58±1.41	0.94	19.64±1.43	0.90	19.55±1.43
3s5z	QMIX	0.90	19.52±1.52	0.94	19.73±1.09	0.94	19.69±1.28
	VDN	0.84	19.22±1.94	0.74	18.90±2.09	0.87	19.44±1.61
3s6z	QMIX	0.90	19.48±1.64	0.90	19.45±1.70	0.90	19.37±1.94

Table 2. The results of robustness testing for QMIX and VDN. WR is the winning rate. M is the number of victims. The smaller the WR and reward, the better. Bold scores represent the best performance.

Envs	Adversary	FGSM		PGD		ATLA		PAAD		RTCA		M
	Victim	WR	Reward	WR	Reward	WR	Reward	WR	Reward	WR	Reward	
8m	QMIX	0.90	19.25±2.30	0.87	18.98±2.65	0.97	19.79±1.18	**0.77**	18.18±3.45	0.84	18.86±2.61	1
	VDN	**0.68**	17.24±4.04	0.74	17.78±3.81	0.97	19.75±1.34	0.87	19.04±2.49	**0.68**	17.36±3.87	
	QMIX	0.42	15.57±3.82	0.52	16.22±3.94	1.00	20.00	0.52	16.05±4.15	**0.29**	14.20±3.83	2
	VDN	0.16	12.30±3.62	0.19	12.38±4.05	0.90	19.31±2.12	0.23	13.15±3.86	**0.06**	10.78±2.79	
2s3z	QMIX	0.65	17.87±3.13	0.65	17.97±2.82	0.97	19.87±0.79	0.87	19.08±2.41	**0.52**	16.93±3.43	1
	VDN	**0.45**	16.64±3.34	**0.45**	16.21±3.67	0.94	19.67±1.44	0.48	16.89±3.29	0.52	16.97±3.37	
	QMIX	0.13	13.29±2.96	0.13	13.76±2.95	0.93	19.52±1.85	**0.10**	14.27±2.32	**0.10**	13.75±2.73	2
	VDN	**0.03**	11.81±1.98	0.06	12.22±2.47	0.87	19.25±2.02	**0.03**	12.33±2.29	0.06	12.49±2.48	
3s5z	QMIX	0.65	17.92±3.03	0.61	18.01±2.68	0.84	19.26±1.79	0.65	18.19±2.63	**0.32**	16.35±2.91	1
	VDN	0.42	16.48±3.39	0.23	15.84±3.01	0.77	19.21±1.76	0.52	17.49±3.12	**0.06**	14.25±2.69	
	QMIX	0.13	13.83±2.83	0.19	15.02±2.68	0.81	19.13±1.85	0.10	13.23±2.72	**0.00**	12.70±1.50	2
	VDN	0.03	10.64±2.69	**0.00**	11.61±2.02	0.81	19.49±1.32	0.03	11.95±3.01	**0.00**	9.71±1.51	
3s6z	QMIX	0.61	18.11±2.66	0.55	17.32±3.29	0.94	19.71±1.12	0.48	17.15±3.10	**0.35**	16.48±3.02	1
		0.06	13.15±2.56	0.10	13.78±2.68	0.74	18.97±2.02	0.16	14.08±3.05	**0.06**	13.40±2.58	2

Table 3. The impact of changing the size of the population in the 3s5z scenario.

Size	M	**WR**	**Reward**	**Times(s)**
10	1	0.39	16.15±3.23	31
	2	0.03	12.49±2.33	32
400	1	0.32	16.35±2.91	78
	2	0.00	12.70±1.50	87
1000	1	0.29	15.62±3.24	170
	2	0.03	12.94±2.11	172

agent in the single-agent scenario. We use the same method (i.e., VDN or QMIX) with victims to train MARL directors and set all agents in the environment are victims.

4.1.3 Perturbation settings and Population Size

The perturbation is set as a ℓ_∞ with a range of 0.1. In our RTCA, the victim set is changed at each time step. For fairness, it is randomly changed in the benchmark methods. We conduct 32 episodes and use winning rates and average team cumulative rewards to evaluate the performance of those attack methods.

The population size is a hyper-parameter that affects the search process. As shown in Table 3, a larger population offers broader search coverage, enabling a more comprehensive exploration of the solution space, albeit at higher computational costs. Conversely, a smaller population narrows the search range and may be suitable for resource-constrained scenarios. However, a smaller population may not guarantee to find the optimal solution. Therefore, we set the population size as 400.

4.2 Results and discussions

The experiment results are shown in Tables 1 and 2. Figures 2 and 3 provide a more intuitive presentation of the experimental results. The robustness of VDN is not evaluated in the 3s6z scenario as its performance is lacking.

In the 8m scenario, the agents are homogeneous and play the same role in the team. The purpose of IGM in MARL is to solve the problem of lazy agents and make the contribution of each agent to the team as equal as possible. However, our experimental results show that the overall effect of selecting critical agents as victims is better than random selection. There might exist more than one critical agent because each on the team in the 8m scenario performs the same job, and there is an opportunity that the crucial agent can be chosen at random. As a result, when attacking just one agent, the PAAD outperforms the RTCA regarding QMIX. However, the average team cumulative reward based on PAAD has a high variance, indicating that there is some randomness in which the agent is chosen as a critical one. Our strategy demonstrates superior performance in attacking QMIX in scenarios with heterogeneous agents. For the robustness testing of VDN, RTCA only performs poorly in the 2s3z scenario but is similar to PAAD and FGSM. The performance of the model

trained by VDN in the 2s3z scenario is poor, which causes the $\tilde{Q}_{jt}$ learned to be worse, making it difficult for the adversary to compute the worst joint action accurately.

The attack results of random noise are very poor and even enhance the winning rate of the victim, which indicates that some noise has a positive effect on the decision-making of MARL agents. This is not appropriate for finding the robustness fault of MARL. Likewise, the ATLA method has poor results, similar to those of random noise. We analyze the reason for this and believe that it is because the joint observation space of agents grows exponentially with the number of agents in a multi-agent task. This is very difficult for MAPPO agents to learn the relationship between clean observations and their joint actions (i.e., joint adversarial observations). Thus, its effectiveness is poor. Overall, we can conclude that:

- Random noise does not consider the policy of the victim. Therefore it is a weak attack method.
- FGSM and PGD only consider the policy of the victim, not the team cooperation. Bad actions by the victim do not necessarily lead to the worst for the team.
- ATLA has a large action space, making generating the adversarial observation difficult.
- PAAD considers the effect of victims on the team, but the victim must be certain.
- RTCA is the most suitable for robustness testing of MARL due to its flexibility and strong attack. Furthermore, this kind of attack is more stealthy because the victim agent changes at every time step.

Table 4. Ablation study. The robustness of QMIX and VDN is evaluated based on their action-value functions. M is the number of victims

Envs	Q_{jt}	VDN/QMIX		$\tilde{Q}_{jt}$		M
	Victim	WR	Reward	WR	Reward	
8m	QMIX	0.74	18.15±3.14	0.84	18.86±2.61	1
	VDN	0.84	18.60±3.19	0.68	17.36±3.87	
	QMIX	0.35	14.49±4.21	0.29	14.20±3.83	2
	VDN	0.16	12.23±3.65	0.06	10.78±2.79	
2s3z	QMIX	0.52	17.02±3.26	0.52	16.93±3.43	1
	VDN	0.87	19.41±1.54	0.52	16.97±3.37	
	QMIX	0.10	13.44±2.53	0.10	13.75±2.73	2
	VDN	0.39	15.90±3.73	0.06	12.49±2.48	
3s5z	QMIX	0.48	17.08±3.11	0.32	16.35±2.91	1
	VDN	0.42	16.69±3.27	0.06	14.25±2.69	
	QMIX	0.03	13.14±2.08	0.00	12.70±1.50	2
	VDN	0.00	11.70±2.17	0.00	9.71±1.51	
3s6z	QMIX	0.39	17.06±2.75	0.35	16.48±3.02	1
		0.00	13.45±1.97	0.06	13.40±2.58	2

4.3 Ablation study

We use the value decomposition network and mixing network in VDN and QMIX, respectively, to indicate the impact of the action-value function $\tilde{Q}_{jt}$. The results are demonstrated in Table 4. The joint action-value function of QMIX more accurately represents the quality of the joint policy than VDN. Therefore, critical agents can be chosen as victims and their worst joint actions can be calculated using Q_{jt} of QMIX. According to the experimental results, it can be seen that using $\tilde{Q}_{jt}$ achieves results comparable to QMIX, and even better in complex scenarios (3s5z and 3s6z). On the other hand, because Q_{jt} of VDN cannot accurately evaluate the quality of the joint action well, it is not advisable to employ it as the objective function of DE.

Table 5. Transferability of $\tilde{Q}_{jt}$. M is the number of victims.

Envs	$\tilde{Q}_{jt}$	QMIX		VDN		M
	Victim	WR	Reward	WR	Reward	
8m	QMIX	0.84	18.86±2.61	0.81	18.56±2.96	1
	VDN	0.29	13.83±4.11	0.68	17.36±3.87	
	QMIX	0.29	14.20±3.83	0.48	15.96±3.98	2
	VDN	0.10	11.63±2.95	0.06	10.78±2.79	
2s3z	QMIX	0.52	16.93±3.43	0.55	17.11±3.29	1
	VDN	0.48	16.25±3.81	0.52	16.97±3.37	
	QMIX	0.10	13.75±2.73	0.03	13.37±1.98	2
	VDN	0.00	12.06±1.64	0.06	12.49±2.48	
3s5z	QMIX	0.32	16.35±2.91	0.48	16.95±3.23	1
	VDN	0.13	14.05±3.02	0.06	14.25±2.69	
	QMIX	0.00	12.70±1.50	0.03	13.52±1.91	2
	VDN	0.00	10.00±1.65	0.00	9.71±1.51	

Table 6. The scalability of RTCA. M is the number of victims.

Adversary	**PAAD**		**RTCA**		M
Victim	WR	Reward	WR	Reward	
QMIX	0.00	10.58±1.97	0.00	10.26±0.98	4
VDN	0.00	8.29±1.51	0.00	7.57±1.06	
QMIX	0.00	8.37±1.96	0.00	9.14±1.05	6
VDN	0.00	6.00±1.11	0.00	6.76±1.27	
QMIX	0.00	8.26±1.18	0.00	9.13±1.07	8
VDN	0.00	4.86±1.06	0.00	6.02±0.89	

4.4 Transferability of $\tilde{Q}_{jt}$

The purpose of this section is to evaluate the transferability of the $\tilde{Q}_{jt}$. In this context, transferability is defined as the ability of the $\tilde{Q}_{jt}$ trained using the data sampled by one MARL to transfer the experience to another and successfully attack it. The results present in Table 5 indicate that the $\tilde{Q}_{jt}$ learned via QMIX policy sampling can be successfully attacked against VDN agents in a majority of the cases considered. Correspondingly, the $\tilde{Q}_{jt}$ trained with VDN policy sampling demonstrated high success rates in successfully attacking QMIX agents in most of the cases. Such observations hold significant implications for improving the applicability of adversarial attacks.In a word, the Sarsa-based $\tilde{Q}_{jt}$ exhibits advantageous transferability qualities and possesses considerable prospects for usage as a black-box attack.

4.5 Scalability of RTCA

To ensure attack stealthiness, adversaries aim to target a minimal number of critical agents, leading to poorer overall performance. Simultaneously, attacking fewer victim agents makes it more challenging to detect the attack. In our experiments, we specifically select only 1 or 2 critical agents as victims. For instance, in the 3s5z scenario with 8 agents, the distinction between critical and non-critical agents becomes irrelevant when all agents become victims (e.g., with 8 victim agents). We conduct experiments in the 3s5z scenario to demonstrate the scalability of RTCA, as shown in Table 6. As the number of victims increases, the advantage of selecting critical agents diminishes, and RTCA still achieves good results.

5 Related Work

5.1 Adversarial attacks on SARL

An extensive body of research has been conducted on methods related to generating adversarial examples in classification tasks. Furthermore, recent studies have emerged that explore adversarial attacks in the context of SARL. Based on a survey [10], adversarial attacks on SARL can be classified into four distinct categories, including perturbations to the state space, the reward function, the action space, and the model space. Huang et al. [9] utilize FGSM for creating adversarial examples of agent input states. Their findings illustrate the efficacy of adversarial attacks for the model trained by RL. Pattanaik et al. [20] propose three types of methods containing random noise, gradient-based, and stochastic gradient decrease. The sample efficient model-based adversarial attack is introduced by Weng et al. [32]. To achieve this, they propose a two-step attack framework, including the learning for the dynamic environment model and the generation of the adversarial state based on the environment model. Huang et al [37] propose the State-Adversarial Markov Decision Process (SA-MDP) which indicates the optimal adversary exists. Besides, they improve the gradient-based attack in [20] and propose a robust Sarsa attack. They use Sarsa to learn the critic network in continuous action space, while we use Sarsa to train the joint action-value network in a discrete one. [36] and [29] are introduced in Section 4.

5.2 Adversarial attacks on MARL

There are few studies on adversarial attacks in MARL. Lin et al. [15] propose the method of generating adversarial states for MARL; they use a two-step attack similar to [29], which reduces the team reward by perturbing the state of only a fixed agent. Pham et al. [21] extend [32] to multi-agent setting. Guo et al. [6] propose a comprehensive robustness testing framework named MARLSafe from three aspects: state, action, and reward. In these methods, the set of victim agents is fixed, while in RTCA, the set of victim agents is variable.

6 Conclusion

In this paper, we present a robust testing framework for a model trained by the state-of-the-art MARL. In our framework, based on DE, the critical agents are selected as victims, and their worst joint actions are advised. Moreover, in order to evaluate the team cooperation policy, we present a Sarsa-based method to learn the joint action-value function in MARL. When the indexes of victims and their joint worst actions are known, we use the target attack method to generate the adversarial observation on them. The results clearly indicate the superiority of our RTCA over the existing ones. Our next work tends to apply RTCA to continuous action spaces such as MADDPG [16] and MAAC [11] to test their robustness against observation perturbations of critical agents.

Acknowledgements

This research was supported by the Shanghai Science and Technology Committee (No. 22511105500), the National Nature Science Foundation of China (Nos. 62172299,62032019), and the Fundamental Research Funds for the Central Universities (No. 2023-4-YB-05). We thank Ms. Weiran Guo for providing the code for ATLA.

References

[1] *On-line Q-learning using connectionist systems*, volume 37, 1994.
[2] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein, 'The complexity of decentralized control of markov decision processes', *Mathematics of operations research*, **27**(4), 819–840, (2002).
[3] Zhiliang Bi, Xiwang Guo, Jiacun Wang, Shujin Qin, and Guanjun Liu, 'Deep reinforcement learning for truck-drone delivery problem', *Drones*, **7**(7), (2023).
[4] Bilal, Millie Pant, Hira Zaheer, Laura Garcia-Hernandez, and Ajith Abraham, 'Differential evolution: A review of more than two decades of research', *Engineering Applications of Artificial Intelligence*, **90**, 103479, (2020).
[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', *arXiv preprint arXiv:1412.6572*, (2014).
[6] Jun Guo, Yonghong Chen, Yihang Hao, Zixin Yin, Yin Yu, and Simin Li, 'Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 115–122, (June 2022).
[7] Songyang Han, Sanbao Su, Sihong He, Shuo Han, Haizhao Yang, and Fei Miao, 'What is the solution for state adversarial multi-agent reinforcement learning?', *arXiv preprint arXiv:2212.02705*, (2022).
[8] Sihong He, Songyang Han, Sanbao Su, Shuo Han, Shaofeng Zou, and Fei Miao. Robust multi-agent reinforcement learning with state uncertainties, 2023.
[9] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel, 'Adversarial attacks on neural network policies', *arXiv preprint arXiv:1702.02284*, (2017).
[10] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato, 'Challenges and countermeasures for adversarial attacks on deep reinforcement learning', *IEEE Transactions on Artificial Intelligence*, **3**(2), 90–109, (2022).
[11] Shariq Iqbal and Fei Sha, 'Actor-attention-critic for multi-agent reinforcement learning', in *Proceedings of the 36th International Conference on Machine Learning*, eds., Kamalika Chaudhuri and Ruslan Salakhutdinov, volume 97 of *Proceedings of Machine Learning Research*, pp. 2961–2970. PMLR, (09–15 Jun 2019).
[12] Yang Jiao, Kai Yang, and Dongjin Song, 'Distributed distributionally robust optimization with non-convex objectives', in *Advances in Neural Information Processing Systems*, eds., S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, volume 35, pp. 7987–7999. Curran Associates, Inc., (2022).
[13] Maosen Li, Cheng Deng, Tengjiao Li, Junchi Yan, Xinbo Gao, and Heng Huang, 'Towards transferable targeted attack', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (June 2020).
[14] Simin Li, Jun Guo, Jingqiao Xiu, Pu Feng, Xin Yu, Jiakai Wang, Aishan Liu, Wenjun Wu, and Xianglong Liu, 'Attacking cooperative multi-agent reinforcement learning by adversarial minority influence', *arXiv preprint arXiv:2302.03322*, (2023).
[15] Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot, 'On the robustness of cooperative multi-agent reinforcement learning', in *2020 IEEE Security and Privacy Workshops (SPW)*, pp. 62–68, (2020).
[16] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch, 'Multi-agent actor-critic for mixed cooperative-competitive environments', in *Advances in Neural Information Processing Systems*, eds., I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, volume 30. Curran Associates, Inc., (2017).
[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, 'Towards deep learning models resistant to adversarial attacks', in *6th International Conference on Learning Representations, ICLR 2018*, Conference Track Proceedings, Vancouver, BC, Canada, (30 Apr – 3 May 2018). OpenReview.net.
[18] Frans A Oliehoek and Christopher Amato, *A concise introduction to decentralized POMDPs*, Springer, 2016.
[19] Karol R. Opara and Jarosław Arabas, 'Differential evolution: A survey of theoretical analyses', *Swarm and Evolutionary Computation*, **44**, 546–558, (2019).
[20] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan,

and Girish Chowdhary, 'Robust deep reinforcement learning with adversarial attacks', in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, p. 2040–2042, Richland, SC, (2018). International Foundation for Autonomous Agents and Multiagent Systems.

[21] Nhan H Pham, Lam M Nguyen, Jie Chen, Hoang Thanh Lam, Subhro Das, and Tsui-Wei Weng, 'Evaluating robustness of cooperative marl: A model-based approach', *arXiv preprint arXiv:2202.03558*, (2022).

[22] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson, 'Monotonic value function factorisation for deep multi-agent reinforcement learning', *J. Mach. Learn. Res.*, **21**(1), (jan 2020).

[23] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson, 'The starcraft multi-agent challenge', in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, p. 2186–2188, Richland, SC, (2019). International Foundation for Autonomous Agents and Multiagent Systems.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, 'Proximal policy optimization algorithms', *arXiv preprint arXiv:1707.06347*, (2017).

[25] Haoran Shi, Guanjun Liu, Kaiwen Zhang, Ziyuan Zhou, and Jiacun Wang, 'Marl sim2real transfer: Merging physical reality with digital virtuality in metaverse', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **53**(4), 2107–2117, (2023).

[26] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi, 'QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning', in *Proceedings of the 36th International Conference on Machine Learning*, eds., Kamalika Chaudhuri and Ruslan Salakhutdinov, volume 97 of *Proceedings of Machine Learning Research*, pp. 5887–5896. PMLR, (09–15 Jun 2019).

[27] Rainer Storn and Kenneth Price, 'Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces', *J. of Global Optimization*, **11**(4), 341–359, (dec 1997).

[28] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai, 'One pixel attack for fooling deep neural networks', *IEEE Transactions on Evolutionary Computation*, **23**(5), 828–841, (2019).

[29] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang, 'Who is the strongest enemy? towards optimal and efficient evasion attacks in deep RL', in *International Conference on Learning Representations*, (2022).

[30] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel, 'Value-decomposition networks for cooperative multi-agent learning based on team reward', in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, p. 2085–2087, Richland, SC, (2018). International Foundation for Autonomous Agents and Multiagent Systems.

[31] Liang Wang, Kezhi Wang, Cunhua Pan, Wei Xu, Nauman Aslam, and Lajos Hanzo, 'Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing', *IEEE Transactions on Cognitive Communications and Networking*, **7**(1), 73–84, (2021).

[32] Tsui-Wei Weng, Krishnamurthy (Dj) Dvijotham*, Jonathan Uesato*, Kai Xiao*, Sven Gowal*, Robert Stanforth*, and Pushmeet Kohli, 'Toward evaluating robustness of deep reinforcement learning with continuous control', in *International Conference on Learning Representations*, (2020).

[33] Tong Wu, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu, 'Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks', *IEEE Transactions on Vehicular Technology*, **69**(8), 8243–8256, (2020).

[34] Min Yang, Guanjun Liu, Ziyuan Zhou, and Jiacun Wang, 'Partially observable mean field multi-agent reinforcement learning based on graph attention network for uav swarms', *Drones*, **7**(7), (2023).

[35] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and YI WU, 'The surprising effectiveness of ppo in cooperative multi-agent games', in *Advances in Neural Information Processing Systems*, eds., S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, volume 35, pp. 24611–24624. Curran Associates, Inc., (2022).

[36] Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh, 'Robust reinforcement learning on state observations with learned optimal adversary', in *International Conference on Learning Representations*, (2021).

[37] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh, 'Robust deep reinforcement learning against adversarial perturbations on state observations', in *Advances in Neural Information Processing Systems*, eds., H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, volume 33, pp. 21024–21037. Curran Associates, Inc., (2020).

[38] Ziyuan Zhou, Guanjun Liu, and Ying Tang, 'Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges', *arXiv preprint arXiv:2305.10091*, (2023).

[39] Ziyuan Zhou, Guanjun Liu, and Mengchu Zhou, 'A robust mean-field actor-critic reinforcement learning against adversarial perturbations on agent states', *IEEE Transactions on Neural Networks and Learning Systems*, 1–12, (2023).