Paper accepted for publication at IJCAI 2023. Author´s copy of camera ready manuscript.

Decentralized Anomaly Detection in Cooperative Multi-Agent Reinforcement Learning

Kiarash Kazari , Ezzeldin Shereen , György Dán
Division of Network and Systems Engineering, School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology, Stockholm, Sweden
{kkazari, eshereen, gyuri}@kth.se

Abstract

We consider the problem of detecting adversarial attacks against cooperative multi-agent reinforcement learning. We propose a decentralized scheme that allows agents to detect the abnormal behavior of one compromised agent. Our approach is based on a recurrent neural network (RNN) trained during cooperative learning to predict the action distribution of other agents based on local observations. The predicted distribution is used for computing a normality score for the agents, which allows the detection of the misbehavior of other agents. To explore the robustness of the proposed detection scheme, we formulate the worst-case attack against our scheme as a constrained reinforcement learning problem. We propose to compute an attack policy via optimizing the corresponding dual function using reinforcement learning. Extensive simulations on various multi-agent benchmarks show the effectiveness of the proposed detection scheme in detecting state of the art attacks and in limiting the impact of undetectable attacks.

1 Introduction

Multi-agent reinforcement learning (MARL) is emerging as an important tool for solving various sequential decision making problems in application areas like 5G networks, unmanned aerial vehicle (UAV) swarms, autonomous driving, power grid control, and Internet of things [Li *et al.*, 2022; Canese *et al.*, 2021] . Inspired by how human beings learn from trial and error, MARL utilizes the concept of rewards in order to teach a team of agents to perform a certain sequential task. The individual agents receive local observations and rewards from the environment, and use those to learn a policy over their possible actions.

MARL problems can be cooperative, competitive, or a mix between the two. In cooperative MARL, the agents need to cooperate to achieve a common goal, thus typically receiving a common reward, while in competitive MARL, the agents are competing against each other. MARL has been shown to perform better on multi-agent decision making problems compared to centralized approaches, especially when the problem is relatively complex [Canese *et al.*, 2021].

Despite the potential of MARL for solving complex decision making problems, a prerequisite for its adoption is that it should withstand faults and adversarial manipulations. For instance, an adversary that compromises a MARL agent could manipulate the agent to take sub-optimal actions, possibly affecting other agents and the team reward. Alternatively, an adversary that compromises the communication link between the environment and the agent could modify the agent's observations, causing the agent to take sub-optimal actions that reduce the team reward [Lin *et al.*, 2020]. Adversarial training, which relies on a known attack model for training robustified ML models [Abusnaina *et al.*, 2021], can mitigate the impact of such adversarial manipulations to some extent, but it does not provide situational awareness.

Thus, even if an MARL algorithm is robustified against attacks, attack detection and identification are essential for timely response, e.g., to be able to promptly evict victim agents. Attack detection can also be used for improving the robustness of MARL, e.g., by ensuring that non-victim agents adjust their policy in a timely manner. Attack detection schemes have thus far been designed for single-agent RL, focusing on anomaly detection in the observations of an agent [Sedlmeier *et al.*, 2020; Zhang *et al.*, 2021]. Anomaly detection in the multi-agent setting has so far been addressed in the control literature (e.g., [Shames *et al.*, 2011] and [Ye *et al.*, 2019]), but these works assume a known dynamical model of the environment and of the control policies of the agents. Instead, in MARL, the policies are approximated by neural networks and the environment is unknown, which makes model-based detection of anomalous behavior infeasible. At the same time, it is unclear how to design model-free approaches for anomaly detection that would scale to large systems and enable decentralized operation.

In this paper, we propose a model-free approach for detecting adversarial attacks against cooperative MARL. The contributions of our paper are as follows:

1. We propose a decentralized detection scheme based on training recurrent neural networks (RNNs) to predict the distribution of actions of other agents conditional on local observations, and use the trained predictors for computing a normality score that quantifies the extent to which agents behave as expected.

2. We propose a dynamic adversary that represents a worst-case attack against our detection scheme and use it to evaluate the robustness of our detector.

3. We carry out extensive simulations of our proposed scheme as well as the worst-case attack utilizing different multi-agent benchmarks, and show that our detection scheme can accurately detect adversarial attacks and limit their impact.

2 Related Work

Most works concerning adversarial attacks against single-agent RL revolve around perturbation of states (or observations). In these works, adversarial example generation algorithms such as FGSM [Goodfellow *et al.*, 2014] and JSMA [Papernot *et al.*, 2016] are used to generate fake observations for the agent. Consequently, the agent will take suboptimal actions as a result of the compromised observation of the environment´s state. Among these papers, [Huang *et al.*, 2017] used FGSM to minimize the probability of taking the best action by the victim. [Pattanaik *et al.*, 2017] utilized the same algorithm but to encourage selecting the worst action by the agent. [Behzadan and Munir, 2017] exploits the transferability of adversarial examples to implement an attack during the training phase. [Russo and Proutiere, 2021] optimized the attack policy with respect to the perturbation budget.

In the context of MARL, [Lin *et al.*, 2020] applied the idea of perturbation of observations to attack one of the agents in c-MARL. As the first step of the attack, they trained an adversarial policy to minimize the long-term team reward. Then, in the second step, a JSMA-based algorithm was used to make the compromised agent follow that adversarial policy. Attacking the actions of an agent is another scenario, which is particularly important in the multi-agent setting. [Gleave *et al.*, 2019] showed that in the competitive MARL, an adversary can fool an agent by controlling another agent's actions. [Guo *et al.*, 2022] is another work in this area, which investigated the robustness of state-of-the-art c-MARL algorithms against attacks on both observations and actions.

Anomaly Detection for sequential data is the focus of works such as [Oh and Iyengar, 2019; Malhotra *et al.*, 2016; Wang *et al.*, 2021]. However, as argued in [Müller *et al.*, 2022], anomaly detection in RL has gained less attention compared to domains like video, audio, and text. In this context, [Zhang *et al.*, 2021] proposed a framework to detect anomalous state observations based on a Gaussian approximation of the state representation space. Authors in [Sedlmeier *et al.*, 2020] proposed an entropy-based anomaly detector for the detection of out-of-distribution observations, though not in an adversarial setting. None of these works addressed anomaly detection in a multi-agent setting, and applying these single-agent schemes to MARL would either require centralized tracking of all agents or implementing anomaly detection locally (each agent for itself). The first case might not be feasible in decentralized multi-agent systems, as collecting data from all agents is very resource intensive. The second case is also suboptimal too because it does not account for adversaries who have complete control over the actions of the victim agent. In our approach, however, anomaly detection is done by other agents interacting with the environment. Accordingly, irrespective of whether the misbehavior of the victim is the result of the perturbation of its observations or its actions, it can be detected. Furthermore, some existing works have proposed decentralized defences against adversarial attacks for control systems. For instance, [Shames *et al.*, 2011] addressed decentralized fault detection, and [Ye *et al.*, 2019] proposed a fault-tolerant control scheme for various control systems. However, these approaches rely on a known model of the system and cannot be applied to general MARL applications, where the environment is typically unknown.

3 System Model

3.1 c-MARL Model

We consider a decentralized POMDP with N agents. Such a model can be represented by a tuple $M = (\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, R, P, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \gamma)$, where $\mathcal{N} = \{1, 2, ..., N\}$ is the set of agents, $\mathcal{S}$ is the state space, and $\mathcal{A}^i$ and $\mathcal{O}^i$ are the set of actions and observations of agent i, respectively. Furthermore, R, P, and γ denote the reward function, the state transition probability, and the discount factor, respectively. At each time step t, agent i receives an observation $o_t^i \in \mathcal{O}^i$ from the system and takes an action $a_t^i \in \mathcal{A}^i$. According to the joint action $\mathbf{a}_t = \{a_t^i\}$ taken by all agents, the state of the system changes from s_t to s_{t+1} based on the (unknown) transition probability $P(s_{t+1}|s_t, \mathbf{a}_t)$, and a shared reward $R_t = R(s_t, \mathbf{a}_t)$ is obtained by the agents. The objective of the agents is to maximize the long-term discounted average reward $\sum_{t=1}^{\infty} \gamma^{t-1} R_t$.

We assume that the agents have been trained using a c-MARL algorithm that involves decentralized execution; this can be done by the centralized-training decentralized-execution paradigm (e.g., QMIX [Rashid *et al.*, 2018] and VDN [Sunehag *et al.*, 2017] algorithms) or by independent learning (e.g., IQL [Tampuu *et al.*, 2017]). As a result, agent i follows an independent policy π^i. If we denote by $\Gamma^i \triangleq (\mathcal{O}^i \times \mathcal{A}^i)^*$ the set of all possible observation-action histories of agent i, the policy π^i maps the history $\tau_t^i \triangleq (o_1^i, a_1^i, ..., o_t^i) \in \Gamma^i$ to the action $a_t^i \in \mathcal{A}^i$.

3.2 Threat Model

We consider an adversary that can manipulate the actions of one agent, whom we refer to as the victim agent, denoted by index $v \in \mathcal{N}$. Although we assume that there is only one compromised agent, our proposed scheme can be applied to the case of multiple victims too. At each time step t, the adversary receives the observation o_t^{adv} and according to its policy π^{adv} selects an action $a_t^{adv} \in \mathcal{A}^v$ to be taken by the victim instead of a_t^v. In general, o_t^{adv} is not necessarily the same as the observation of the victim (o_t^v); it may contain more (or less) information. Note that besides the attacks aiming directly at the actions (such as in [Gleave *et al.*, 2019] and [Guo *et al.*, 2022]), this model can account for misbehavior caused by a perturbation of the victim's observations (such as in [Lin *et al.*, 2020]) too. The reason for this is that from the other agents' point of view, which is the main focus of

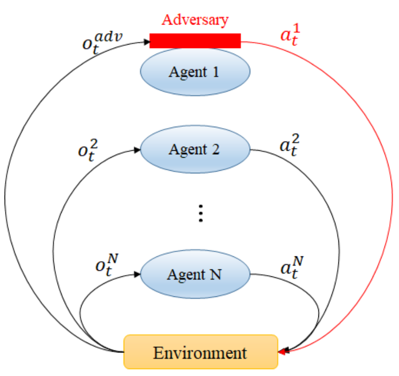


Figure 1: Illustration of the system model, considering agent 1 as the victim.

this paper, the anomalous behavior of the victim is perceptible only through its actions. Recent work has shown that this attack model can cause up to 100% performance loss to the system [Guo *et al.*, 2022], therefore the detection of such attacks is of crucial importance.

3.3 Problem Formulation

Our focus is on detecting the abnormal behavior of a (victim) agent using the sequence of observations of another agent. Formally, corresponding to agent i as the observer and agent j ($j \neq i$) as the possible victim, we want to find a function $M^{ij} : (\mathcal{O}^i \times \mathcal{A}^j)^* \to \{0,1\}$, i.e., one that takes the history $o_1^i, a_1^j, o_2^i, a_2^j, ..., o_t^i, a_t^j$ at time t as input, and the corresponding output M^{ij} equals 0 if the sequence is normal and equals 1 if it is abnormal.

4 Decentralized Anomaly Detection

In what follows, we propose a decentralized scheme for the detection and identification of an attack against an agent. The detector we propose is comprised of two parts: 1) a predictor that predicts the distribution of agent j's actions at time step t based on $o_1^i, o_2^i, ..., o_t^i$, and 2) an algorithm for computing a normality score based on the actions $a_1^j, a_2^j, ..., a_t^j$ taken by agent j and the outputs of the predictor. Next, we describe these two parts in details.

Prediction of action distribution: Each agent i maintains a predictor ϕ^{ij} that takes τ_t^i as input and produces a distribution (PMF) over the possible actions of agent $j \neq i$. That is $\phi^{ij}(\tau_t^i) = \boldsymbol{p}_t^{ij}$, where $\boldsymbol{p}_t^{ij} = \{p_t^{ij}(a^j|\tau_t^i)\}$, and $p_t^{ij}(a^j|\tau_t^i)$ is the probability of taking action a^j by agent j from agent i's point of view. In practice, the implementation of the function ϕ^{ij} can be regarded as a sequence classification problem, where τ_t^i is the input and possible actions of the victim are the classes (labels). We propose to use recurrent neural networks to predict these labels. By using a softmax layer as the output activation, the PMF $\boldsymbol{p}_t^{ij}$ is obtained. Figure 2b shows our realization of these predictors using a gated recurrent unit (GRU) structure. Note that training these predictors should be done after training the agents using c-MARL so that the policies π^i can be considered unchanged.

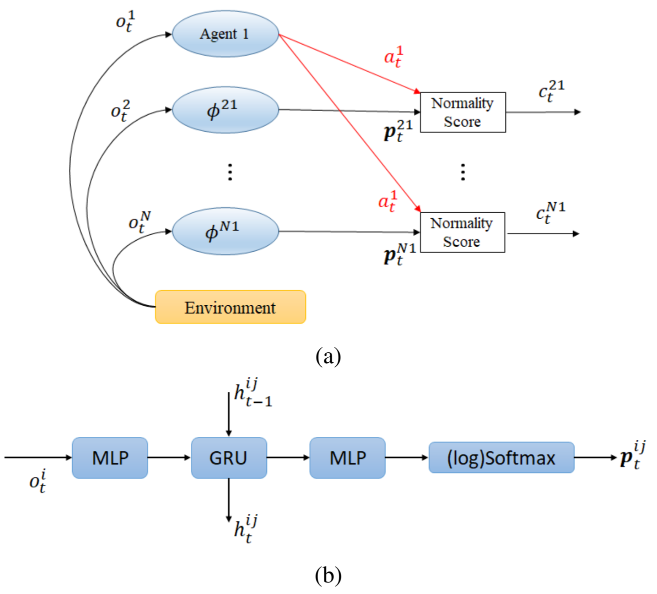


Figure 2: (a) Decentralized detection scheme, considering agent 1 as the possible victim. (b) Realization of ϕ^{ij}.

Computing normality score: Given the predictions $\boldsymbol{p}_t^{ij}$, the second step is to find a sequential test to determine the normality of a given sequence of agent j's actions. The test takes the sequence $\boldsymbol{p}_1^{ij}, a_1^j, \boldsymbol{p}_2^{ij}, a_2^j, ..., \boldsymbol{p}_t^{ij}, a_t^j$ and outputs the (ab)normality of this sequence. Observe that well-known sequential tests, such as Pearson's chi-square test (see e.g., [Lehmann *et al.*, 2005]) cannot be applied as the normality test here as the target distribution $\boldsymbol{p}_t^{ij}$ depends on the state, and thus it changes over time.

Thus, we propose to compute a normality score for the observed sequence of actions and compare it to a predefined threshold. The normality score we propose is given by

$$c_t^{ij} = \begin{cases} \frac{1}{t}\sum_{l=1}^{t} \log\left(\frac{p_l^{ij}(a_l^j)}{\max_{a^j} p_l^{ij}(a^j)}\right), & 1 \leq t < w \\ \frac{1}{w}\sum_{l=t-w+1}^{t} \log\left(\frac{p_l^{ij}(a_l^j)}{\max_{a^j} p_l^{ij}(a^j)}\right), & t \geq w \end{cases} \tag{1}$$

where w is the window size. The rationale behind this normality score is the following. At every time step, the score takes into account the predicted probability of the action taken by agent j, normalized by $\max_{a^j} p_l^i(a^j)$, i.e., the predicted probability of taking the most probable action. The normalization in (1) makes sure to take the confidence of the predictions into account, and the reason for using the *log* function is to facilitate the computations.

At each time step, agent i computes c_t^{ij} and compares it to a predefined threshold β^{ij}. Agent i considers agent j as compromised (i.e., $M^{ij} = 1$) if

$$c_t^{ij} < \beta^{ij} \tag{2}$$

for some time step t. Overall, each agent needs to maintain $||\mathcal{N}||-1$ predictors, i.e., the number of predictors is quadratic in the number of agents.

5 Dynamic Adversary

In order to evaluate the effectiveness of the proposed detection scheme in a worst case scenario, we now consider a dynamic adversary that knows the detection scheme and aims to bypass it while attacking the system. Intuitively, if the detector performs well against an adversary with perfect knowledge of the detector then it can perform well against attackers with less knowledge. In what follows, we first formulate the problem faced by the dynamic adversary, and then we show how to compute an adversarial policy as a solution to this problem.

5.1 Dynamic Attacker Problem Formulation

Consider an attacker that has access to $\boldsymbol{p}_t^{iv}$ and knows β^{iv} for every agent i. The attacker seeks to minimize the long-term team reward, while being expectedly undetectable, in the following sense.

Definition 1. *An attack policy π^{adv} is expectedly undetectable if there is no agent $i \neq v$ such that $\mathbb{E}\left[c_t^{iv}(\pi^{adv})\right] < \beta^{iv}$ for any time step $t > 0$.*

We define $r_t^{adv} \triangleq -R_t$ as the adversarial reward, so minimizing the long-term team reward is equivalent to maximizing the long-term adversarial reward. Accordingly, if we define $V(\pi^{adv}) \triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t^{adv}\right]$ then the adversary's objective is to find a policy π^{*adv} that solves

$$\begin{aligned} &\max_{\pi^{adv}} V(\pi^{adv}) \\ &\text{s.t. } \mathbb{E}\left[c_t^{iv}\right] \geq \beta^{iv}, \ {\scriptstyle t=1,2,\ldots,\ \forall i \in \mathcal{N}\setminus\{v\}}. \end{aligned} \tag{P}$$

The problem (P) has a non-Markovian structure and is intractable to solve. Therefore, we propose a Markovian approximation of (P), which we call (P').

Let $z_t^i \triangleq \log(\frac{p_t^i(a_t^v)}{\max_{a^v} p_t^i(a^v)})^1$. We define the extended state $\bar{s}_t \in \bar{\mathcal{S}} \triangleq \mathcal{S} \times \Gamma^1 \times \Gamma^2 \times \ldots \times \Gamma^N$ as $\bar{s}_t = (s_t, \tau_t^1, \ldots, \tau_t^N)$. Then, we can rewrite $r^{adv}(s_t, a_t^v)$ as $r^{adv}(\bar{s}_t, a_t^v)$ and $z^i(\tau_t^i, a_t^v)$ as $z^i(\bar{s}_t, a_t^v)$ (note that we had to extend the state because we could not write z_t^i as $z^i(s_t, a_t^v)$). Now, let us define (P') as a constrained RL problem

$$\begin{aligned} &\max_{\pi^{adv}} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r^{adv}(\bar{s}_t, \pi^{adv}(\bar{s}_t))\right] \\ &\text{s.t. } C^i(\pi^{adv}) \geq \frac{\beta^{iv}}{1-\gamma}, \ {\scriptstyle \forall i \in \mathcal{N}\setminus\{v\}}, \end{aligned} \tag{P'}$$

where $C^i(\pi^{adv}) \triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} z^i(\bar{s}_t, \pi^{adv}(\bar{s}_t))\right]$. In the following proposition we show that the solution to (P') is an upper bound to the solution to (P).

Proposition 1. *Let π^{*adv} and $\bar{\pi}^{*adv}$ be policies such that $V(\pi^{*adv})$ and $V(\bar{\pi}^{*adv})$ are the solutions to (P) and (P'), respectively. Then $V(\bar{\pi}^{*adv}) \geq V(\pi^{*adv})$.*

Proof. Let F and F' be the feasible regions in (P) and (P'), respectively. Since the objective function is the same in both problems, it is sufficient to show that $F \subseteq F'$. Suppose that π is an arbitrary policy in F. We show that $\pi \in F'$. For that purpose, we need to prove that $C^i(\pi) \geq \frac{\beta^{iv}}{1-\gamma}$.

Let $S_T^i(\pi) \triangleq \mathbb{E}[\frac{1}{T}\sum_{t=1}^{T} z_t^i]$. Any $T \in \mathbb{N}$ can be written as $T = nw + m$, where $m, n \in \mathbb{N} \cup \{0\}$ and $m < w$. Thus, we can write

$$\begin{aligned} S_T^i(\pi) &= \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{m} z_t^i + \sum_{t=1+m}^{w+m} z_t^i + \ldots + \sum_{t=1+(n-1)w+m}^{nw+m} z_t^i\right] \\ &= \frac{1}{T}\mathbb{E}[m c_m^{iv} + w c_{w+m}^{iv} + \ldots + w c_{nw+m}^{iv}] \end{aligned} \tag{3}$$

Since $\pi \in F$, we know that $\mathbb{E}[c_t^{iv}] \geq \beta^{iv}$ for any $t \geq 1$. Therefore, we can conclude that

$$S_T^i(\pi) \geq \frac{1}{T}(m + nw)\beta^{iv} = \beta^{iv} \tag{4}$$

Now, we can use the following lemma to conclude the proof.

Lemma [Theorem 13.29 in [Maschler *et al.*, 2020]]: *Let $\{x_t\}_{t=1}^{\infty}$ be a bounded sequence of real numbers and S_T be the average of the first T elements of it: $S_T = \frac{1}{T}\sum_{t=1}^{T} x_t$. Also, let $\alpha_T(\gamma)$ denote $(1-\gamma)^2\gamma^{T-1}T$. Then, for any $\gamma \in [0,1)$ we have $\sum_{T=1}^{\infty} \alpha_T(\gamma) = 1$, and also*

$$(1-\gamma)\sum_{t=1}^{\infty} \gamma^{t-1} x_t = \sum_{T=1}^{\infty} \alpha_T(\gamma) S_T.$$

By applying the above lemma with z_t as x_t and $S_t^i(\pi)$ as S_t, and taking the expectation we get

$$(1-\gamma)C^i(\pi) = \sum_{T=1}^{\infty} \alpha_T(\gamma) S_T(\pi) \geq \beta^{iv} \sum_{T=1}^{\infty} \alpha_T(\gamma) = \beta^{iv}. \tag{5}$$

We thus have $C^i(\pi) \geq \frac{\beta^{iv}}{1-\gamma}$, which completes the proof. □

A conclusion from Proposition 1 would be that if the defender finds a solution $V(\bar{\pi}^{*adv})$ to (P'), then it can be sure that there can not be any other expectedly undetectable attack with an impact higher than $V(\bar{\pi}^{*adv})$. Next, we will discuss how $V(\bar{\pi}^{*adv})$ can be found.

5.2 Finding the Adversarial Policy

Although (P') is a non-convex optimization problem, it can be shown that if r^{adv} and z^i are bounded and Slater's condition holds, it has zero duality gap [Paternain *et al.*, 2019]. We assume that there is an $\epsilon > 0$ such that $p_t^{iv}(a^v) \geq \epsilon$ for any action $a^v \in \mathcal{A}^v$. Consequently, z_t^i will be bounded. Moreover, Slater's condition states that the feasible region has an interior point. In practice, choosing proper values as thresholds (β^{iv}) causes the no-attack case (where $\pi^{adv} = \pi^v$) to be an interior point of the feasible region of (P'), in which case Slater's condition holds. As a result, we can find the policy via optimizing the dual problem. The Lagrangian of (P') can be defined as

$$\mathcal{L}(\pi^{adv}, \boldsymbol{\lambda}) = V(\pi^{adv}) + \sum_{i \neq v} \lambda_i (C^i(\pi^{adv}) - \frac{\beta^{iv}}{1-\gamma}), \tag{6}$$

$^1 z_t$ is related to both agents i and v, but for the ease of notation, we omit the superscript v.

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. Accordingly, the dual function is defined as

$$d(\boldsymbol{\lambda}) = \max_{\pi^{adv}} \mathcal{L}(\pi^{adv}, \boldsymbol{\lambda}), \tag{7}$$

and the objective would be to find $\min_{\boldsymbol{\lambda} \in \mathbb{R}_+^{N-1}} d(\boldsymbol{\lambda})$. Note that for a given $\boldsymbol{\lambda}$, computing the dual function corresponds to

$$\max_{\pi^{adv}} \mathcal{L}(\pi^{adv}, \boldsymbol{\lambda}) = \max_{\pi^{adv}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t [r_t^{adv} + \sum_{i \neq v} \lambda_i z_t^i]\right], \tag{8}$$

which can be considered as finding the optimal policy in an RL problem with the reward defined as $r_{\boldsymbol{\lambda}} = r_t^{adv} + \sum_{i \neq v} \lambda_i z_t^i$.

Following the dual descent approach in [Paternain *et al.*, 2019], if we parameterize the policy π as π_θ, the Lagrangian can be represented by

$$\mathcal{L}(\theta, \boldsymbol{\lambda}) = V(\theta) + \sum_{i \neq v} \lambda_i (C^i(\theta) - \frac{\beta^{iv}}{1-\gamma}). \tag{9}$$

Then, the optimal policy can be found by the following iterative algorithm:

1. In each iteration, we update the parameters θ such that

   $$\theta^{(k+1)} \approx \arg\max_{\theta} \mathcal{L}(\theta, \boldsymbol{\lambda}^{(k)}). \tag{10}$$

   This is done by training an RL algorithm with the reward defined by $r_{\boldsymbol{\lambda}^{(k)}}$.

2. Updating $\boldsymbol{\lambda}^{(k)}$ by using a gradient descent step

   $$\lambda_i^{(k+1)} = [\lambda_i^{(k)} - \eta(C^i(\theta^{(k+1)}) - \frac{\beta^{iv}}{1-\gamma})]_+ \ , \ \forall i \in \mathcal{N} \backslash \{v\}, \tag{11}$$

   where η is a step size parameter.

As shown in [Paternain *et al.*, 2019], if the RL algorithm used for solving (10) can find a "good" solution (according to Assumption 1 in [Paternain *et al.*, 2019]) then for given threshold values the iterative execution of (10) and (11) will converge to an optimal solution. Note that the value of λ_i computed using the above procedure trades off between the detectability of the dynamic attack by agent i and the impact of the attack.

6 Numerical Results[2]

In this section, we evaluate our detection scheme against state-of-the-art adversarial attacks, as well as the dynamic attack proposed in Section 5.

6.1 Evaluation Methodology

We use three test environments for evaluating the proposed detector: StarCraft II Multi-Agent Challenge (SMAC) [Samvelyan *et al.*, 2019], Multi Particle Environment (MPE) [Mordatch and Abbeel, 2017], and Level-Based Foraging (LBF) [Papoudakis *et al.*, 2021]. In SMAC, a team consists of a number of agents and has the objective of defeating an opponent team, which is governed by the StarCraft built-in AI algorithm. Our selected scenarios in this environment are "2s3z" and "MMM". In "2s3z" both teams consist of 5 units (agents), 2 Stalkers and 3 Zealots, and in "MMM" both teams consist of 10 agents, seven Marines, two Marauders, and one Medivac Dropship unit. Among the MPE scenarios, we selected "Simple Tag" (also, known as predator-prey), where a group of 3 agents has to hunt a prey. In LBF, agents cooperate to collect food items distributed in a rectangular grid environment. In our selected scenario, there are 5 agents and 4 food items in an 8-by-8 grid.In each scenario, we choose one of the team's agents as the victim and the detection is performed by the other agents. The procedure for conducting the experiments is as follows.

Attack	SMAC-2s3z	SMAC-MMM	MPE-Tag	LBF
ACT	1	1	0.980	0.979
OBS	0.999	0.999	0.960	0.873

Table 1: AUC score of the proposed scheme against ACT and OBS attacks in all environments.

1. **Training the c-MARL algorithm:** We used QMIX as the underlying c-MARL algorithm for the SMAC and MPE environments, and MAA2C [Papoudakis *et al.*, 2021] for LBF. Using the implementation provided by the PyMARL [Samvelyan *et al.*, 2019] and EPyMARL [Papoudakis *et al.*, 2021] Python frameworks, we trained the agents.

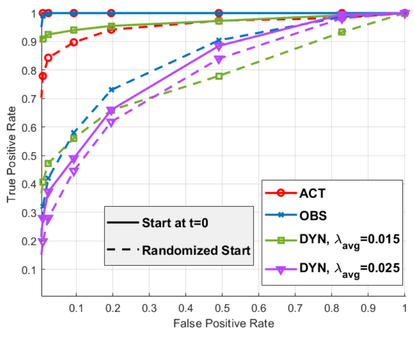
2. **Training the detector:** We used an RNN with the structure shown in Figure 2b as the predictor (one predictor per agent). The hidden state dimension of the GRU layer was 64 for *SMAC-2s3z* and 128 for the other scenarios. We trained the predictors for 20,000 episodes during which agents applied the policies learned in Step 1.

3. **Training the attack:** Once the predictors for detection were trained, we trained the dynamic attacks based on (10). We trained multiple attacks (corresponding to different vectors $\boldsymbol{\lambda}$). For each attack we used a single-agent DRQN algorithm [Hausknecht and Stone, 2015] for 20,000 episodes to obtain the adversary. In the sequel, we characterize each attack by λ_{avg}, the average of the entries of the $\boldsymbol{\lambda}$ vector.

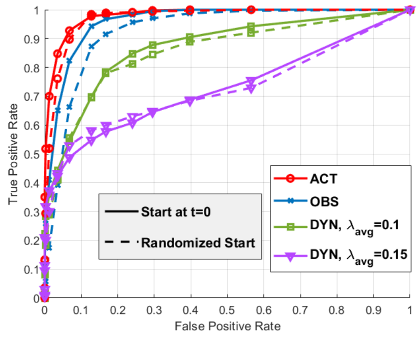
4. **Evaluation:** We applied the learned attacks to the victim, and used the detection scheme at the non-victim agents. For each attack, we evaluated 400 episodes. In our evaluations, once a detection happens by any of the non-victim agents, we consider the attack as detected at that time step.

In addition to our proposed dynamic attack (DYN), we tested the proposed detection scheme against the following attack schemes from the literature:
Observation Attack (OBS): The two-step attack proposed

[2]Code available at https://github.com/kiarashkaz/anomaly-detection-in-cMARL

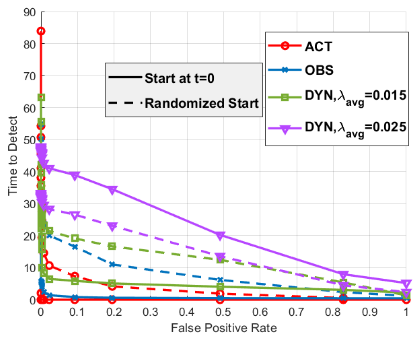


(a) *SMAC-MMM*

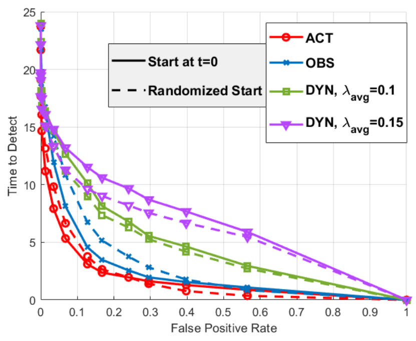


(b) *MPE-Tag*

Figure 3: ROC curve of the proposed detector with $w = \infty$ for the action, observation and dynamic attacks in scenarios SMAC-MMM and MPE-Tag.



(a) *SMAC-MMM*



(b) *MPE-Tag*

Figure 4: Time to detect vs. false positive rate for the action, observation and dynamic attacks in scenarios SMAC-MMM and MPE-Tag for $w = \infty$.

by [Lin *et al.*, 2020] is used to perturb the victim's observations. We used JSMA for cpmputing the perturbations as described in [Lin *et al.*, 2020].
Action Attack (ACT): This attack (studied in [Guo *et al.*, 2022]) manipulates the victim's actions to minimize the team reward. ACT is a special case of DYN with $\boldsymbol{\lambda} = 0$.

To further explore the factors affecting detectability, for each attack, we consider a variant, where the corresponding attack starts from a time step chosen uniform at random during the first half of the episode, instead of starting in the first time step of the episode.

6.2 Detection Performance

As the detectability depends on the selected thresholds β^{ij}, we use the receiver operating characteristic (ROC) curve to evaluate the performance of the proposed detector. The ROC curve shows the true positive rate as a function of the false positive rate, and is obtained by using different detection thresholds β^{iv}. We define the true positive rate as the fraction of attacked episodes that are detected, and the false positive rate as the fraction of unattacked episodes that are incorrectly classified as attacked. The detection performance can be quantified by the area under the ROC curve (AUC). The higher the AUC, the more accurate the detector.

Table 1 shows the AUC score of our proposed detector (with $w = \infty$) against ACT and OBS attacks. This table confirms that the proposed scheme is very efficient in detecting non-dynamic state-of-the-art attacks. Figure 3 shows the ROC curve of the proposed detection scheme for various attacks for scenarios SMAC-MMM and MPE-Tag (due to space limit, we do not include the ROC curve for other scenarios). The figure shows that randomizing the start time of the attack does not have a significant effect on detection in MPE-Tag while this effect is considerable in SMAC-MMM. The reason for such a difference is that in MPE-Tag, the distribution of the actions at the first few time steps is almost uniform, leading to normality scores around 0 irrespective of whether there is an attack. Thus, starting the attack after a few time steps would result in a similar normality score as starting it at the beginning of the episode. Figure 3 also shows that the detection for the dynamic attack depends on the the value of λ_{avg}; a higher value of λ_{avg} makes the attack less likely to be detected, as increasing λ_i makes the adversary more cautious.

6.3 Time to Detection

An important performance metric of a detector is the average time required for detection. We define the time to detect as the average number of time steps between the start of an at-

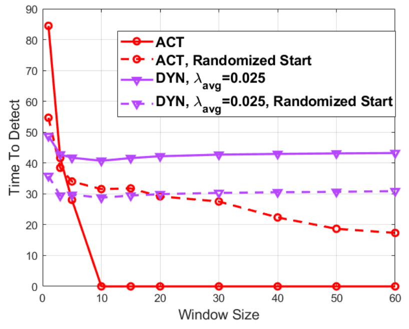


Figure 5: Time to detect as a function of the window size for a false positive rate of 0.01, obtained in the SMAC-MMM environment.

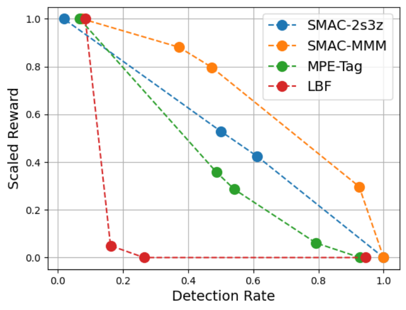


Figure 6: Attack impact vs. detectability for the DAA attack, obtained by using different values of $\boldsymbol{\lambda}$ for a fixed false positive rate.

tack and its detection (or the end of the episode if it is not detected). Figure 4 shows the time to detect as a function of the false positive rate in the SMAC-MMM and MPE-Tag environments for $w = \infty$. We can observe that in the SMAC-MMM environment, the ACT, OBS, and DYN($\lambda_{avg} = 0.015$) attacks can be detected in less than 10 time steps on average with a false-positive rate close to 0. The corresponding time to detection for the randomized-start version of these attacks is below or around 20 time steps, which is very good considering that the maximum episode length in this scenarios is 150. In the MPE-Tag environment, detection (of the non-dynamic attacks) takes longer (relative to the episode length of 25), on average. This can be explained by the distribution of actions, especially in the first time steps. In the SMAC environment, there are more than 10 possible actions, and the detector assigns a very low probability to some of them. A non-dynamic attack, which likely chooses one of those can thus be easily detected. However, in MPE-Tag there are only 5 actions and in the first time steps (where everything is somewhat random) they are almost equally likely to be taken from the detector's perspective. Thus, even for non-dynamic attacks, more time steps are needed before detection happens.

Figure 5 shows the detection time as a function of the window size for a false positive rate of 0.01 in the SMAC-MMM environment (we do not include all scenarios and attacks for brevity). The figure shows that the window size should be long enough to facilitate quick detection with a reasonably low false positive rate. If the window size is short, a low detection threshold has to be chosen for having a small false positive rate (e.g., 0.01), and this low threshold would result in a high time to detect. Figure 5 also shows that randomizing the start time can increase or decrease the detection time depending on the attack. For ACT, the normality score drops significantly as soon as the attack starts, but the drop is more difficult to detect due to averaging if the attack starts during the episode, leading to a higher detection time. For DYN($\lambda = 0.025$), on the other hand, the normality score does not drop significantly when the attack starts, and since the normality score is highest at the beginning of the episode, the detection time in the case of randomized start is lower .

6.4 Impact-Detectability Trade-Off

Recall that attacks obtained using a high value of $\boldsymbol{\lambda}$ are difficult to detect. However, these attacks typically have a lower impact on the performance of the multi-agent system. In other words, there should be a trade-off between attack impact and attack detectability. Figure 6 illustrates this trade-off by showing the team "scaled reward" as a function of the detection rate for DYN attacks in all environments, obtained using different values of λ. The scaled reward is computed using the win rate of the allied team in the SMAC environments and the total episodic reward in MPE-Tag and LBF as the measure of success. We scaled this measure of success such that 1 corresponds to the no-attack case and 0 corresponds to the attack with the highest impact (i.e., ACT).

The figure shows that,in general, an adversary can become stealthy by following a policy close to the original policy of the victim, but doing so increases the team's chances to accomplish its task. Such a trade-off can be observed in all environments except for LBF, where it was possible to train an attack with the same impact as the ACT attack while the detection rate was around 0.2. This observation points out the limitations of the proposed detector. Such limitations can be due to two reasons. First, if the observations of different agents do not have enough correlation with each other, it cannot be expected that they are able to detect anomalies in each other's behavior. Second, if the original task of the multi-agent system requires the agents to cooperate in a very specific way then it is possible to impose a significant loss to the system by a small change in the victim's policy. Such small anomalies might be difficult to detect by the detector.

7 Conclusion

In this paper, we proposed a decentralized approach to detect the anomalous behavior of agents in c-MARL. Our proposed scheme utilizes the observations that agents obtain from the environment to predict the action distribution of other agents. We proposed a low-complexity anomaly score computed based on the predictions compared to the actual actions taken by agents. We also proposed a method for computing a worst case attack against our detector, which allows to explore its robustness. We evaluated the detector against our proposed worst case attack as well as state-of-the-art attacks from the literature. Our numerical results show that if the window size is large enough then the proposed detector is able to detect attacks effectively. We observed that only relatively ineffective attacks could remain undetected, except for in environments in which the agents' observations have little correlation or when a specific way of coordination is required for the accomplishment of the c-MARL task.

Acknowledgments

This work was partly funded by the Swedish Foundation for Strategic Research through the CLAS project (grant RIT17-0046) and by the Swedish Research Council through project 2020-03860.

References

[Abusnaina *et al.*, 2021] Ahmed Abusnaina, Yuhang Wu, Sunpreet Arora, Yizhen Wang, Fei Wang, Hao Yang, and David Mohaisen. Adversarial example detection using latent neighborhood graph. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7667–7676, 2021.

[Behzadan and Munir, 2017] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *Proceedings of International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[Canese *et al.*, 2021] Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11), 2021.

[Gleave *et al.*, 2019] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Guo *et al.*, 2022] 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*, pages 115–122, 2022.

[Hausknecht and Stone, 2015] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposium series*, 2015.

[Huang *et al.*, 2017] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[Lehmann *et al.*, 2005] Erich Leo Lehmann, Joseph P Romano, and George Casella. *Testing statistical hypotheses*, volume 3. Springer, 2005.

[Li *et al.*, 2022] Tianxu Li, Kun Zhu, Nguyen Cong Luong, Dusit Tao Niyato, Qi hui Wu, Yang Zhang, and Bing Chen. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 24:1240–1279, 2022.

[Lin *et al.*, 2020] Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *Proceedings of IEEE Security and Privacy Workshops (SPW)*, pages 62–68, 2020.

[Malhotra *et al.*, 2016] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[Maschler *et al.*, 2020] Michael Maschler, Shmuel Zamir, and Eilon Solan. *Game theory*. Cambridge University Press, 2020.

[Mordatch and Abbeel, 2017] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[Müller *et al.*, 2022] Robert Müller, Steffen Illium, Thomy Phan, Tom Haider, and Claudia Linnhoff-Popien. Towards anomaly detection in reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1799–1803, 2022.

[Oh and Iyengar, 2019] Min-hwan Oh and Garud Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & data mining*, pages 1480–1490, 2019.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387, 2016.

[Papoudakis *et al.*, 2021] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of NeurIPS*, 2021.

[Paternain *et al.*, 2019] Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Proceedings of NeurIPS*, 2019.

[Pattanaik *et al.*, 2017] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of ICML*, pages 4295–4304, 2018.

[Russo and Proutiere, 2021] Alessio Russo and Alexandre Proutiere. Towards optimal attacks on reinforcement learning policies. In *Proceedings of American Control Conference (ACC)*, pages 4561–4567, 2021.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[Sedlmeier *et al.*, 2020] Andreas Sedlmeier, Robert Müller, Steffen Illium, and Claudia Linnhoff-Popien. Policy entropy for out-of-distribution classification. In *Proceedings of International Conference on Artificial Neural Networks*, pages 420–431, 2020.

[Shames *et al.*, 2011] Iman Shames, André M.H. Teixeira, Henrik Sandberg, and Karl H. Johansson. Distributed fault detection for interconnected second-order systems. *Automatica*, 47(12):2757–2764, 2011.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Tampuu *et al.*, 2017] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[Wang *et al.*, 2021] Zhiwei Wang, Zhengzhang Chen, Jingchao Ni, Hui Liu, Haifeng Chen, and Jiliang Tang. Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3726–3734, 2021.

[Ye *et al.*, 2019] Dan Ye, Meng-Meng Chen, and Hai-Jiao Yang. Distributed adaptive event-triggered fault-tolerant consensus of multiagent systems with general linear dynamics. *IEEE Trans. on Cybernetics*, 49(3):757–767, 2019.

[Zhang *et al.*, 2021] Hongming Zhang, Ke Sun, Bo Xu, Linglong Kong, and Martin Müller. A simple unified framework for anomaly detection in deep reinforcement learning. *arXiv preprint arXiv:2109.09889*, 2021.