Learning and Testing Resilience in Cooperative Multi-Agent Systems

Thomy Phan
Thomas Gabor
Andreas Sedlmeier
Fabian Ritz
LMU Munich
thomy.phan@ifi.lmu.de

Bernhard Kempter
Cornel Klein
Horst Sauer
Reiner Schmid
Jan Wieghardt
Marc Zeller
Siemens AG

Claudia Linnhoff-Popien
LMU Munich

ABSTRACT

State-of-the-art multi-agent reinforcement learning has achieved remarkable success in recent years. The success has been mainly based on the assumption that all teammates perfectly cooperate to optimize a global objective in order to achieve a common goal. While this may be true in the ideal case, these approaches could fail in practice, since in multi-agent systems (MAS), all agents may be a potential source of failure. In this paper, we focus on resilience in cooperative MAS and propose an *Antagonist-Ratio Training Scheme (ARTS)* by reformulating the original target MAS as a mixed cooperative-competitive game between a group of protagonists which represent agents of the target MAS and a group of antagonists which represent failures in the MAS. While the protagonists can learn robust policies to ensure resilience against failures, the antagonists can learn malicious behavior to provide an adequate test suite for other MAS. We empirically evaluate ARTS in a cyber physical production domain and show the effectiveness of ARTS w.r.t. resilience and testing capabilities.

KEYWORDS

multi-agent learning; adversarial learning; learning and testing

ACM Reference Format:
Thomy Phan, Thomas Gabor, Andreas Sedlmeier, Fabian Ritz, Bernhard Kempter, Cornel Klein, Horst Sauer, Reiner Schmid, Jan Wieghardt, Marc Zeller, and Claudia Linnhoff-Popien. 2020. Learning and Testing Resilience in Cooperative Multi-Agent Systems. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Distributed systems need to be *resilient* such that they can reliably provide a service, while single components may fail due to hardware failures, software flaws, or malicious attacks [38]. To preserve resilience in such systems, various counter mechanisms like replication or decentralization of functions are implemented to prepare such systems for worst-case scenarios of arbitrary failure.

Multi-agent reinforcement learning (MARL) is a promising way to realize distributed systems, where the components are represented by autonomous agents in a cooperative *multi-agent system (MAS)* [6, 24]. Although state-of-the-art MARL has achieved remarkable success in recent years [14, 42], they have mainly relied on the assumption that all teammates perfectly cooperate to achieve a common goal [6, 21, 28, 33]. While this may be true in the ideal case, these approaches could fail in practice, since in MAS, all agents may be a potential source of failure due to hardware or software flaws, or malicious attacks [38]. This is a critical limitation of current MARL, which can lead to catastrophic failures that could be severe in industrial and safety-critical domains [15, 40].

Current approaches to learning resilience in autonomous systems are based on *adversarial learning*, where the training environment is modeled as an adversary which further challenges the agent by forcing it to develop robust strategies. A majority of these works focus on single-agent systems and adversarial changes to a static environment. However in MAS, agents themselves could fail in an arbitrary fashion during runtime, which may harm the system due to changes in the team dynamics which could differ significantly from the dynamics encountered during training. Preparing each agent for all possible worst-case scenarios (e.g., different team compositions) is an intractable problem and requires more scalable solutions to generalize across different scenarios [14, 40, 42].

Besides learning resilience via adequate training, *testing* of existing systems to uncover flaws is an important aspect. In classical *software engineering (SE)*, testing is already an integral part and typically requires 50% of the total development costs which motivates automation [1, 5]. While systematic methods are available in SE to uncover flaws in software systems, such methods and tools are still lacking for reinforcement learning (RL) systems due to the unstructured and complex nature of methods and problems [3, 40].

There are adversarial approaches to detect flaws in existing RL systems [29, 40], but they are very specialized to a particular system and have to be retrained from scratch, when encountering a new system. In cases of system updates or modifications, *general test suites* are needed to check *any* system (configuration) for flaws.

In this paper, we address both learning and testing of resilience in cooperative MAS. For that, we propose an *Antagonist-Ratio Training Scheme (ARTS)* by reformulating the original target MAS as a mixed cooperative-competitive game between a group of protagonists which represent agents of the target MAS and a group of antagonists which represent failures in the MAS. While training the protagonists can improve resilience in the target MAS, training the antagonists can provide an adequate test suite for other MAS.

Our main contributions are as follows:

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

- We propose a scalable adversarial training scheme for MARL to simultaneously train agents, which exhibit robust behavior in the presence of failures and malicious agents, which can be used as a test suite for other MAS. The proposed scheme can also be used for black box testing of other MAS.
- For the special case of only one functional agent, we propose *QMixMax* as a scalable approximation of Minimax-Q [19], where we exploit the monotonicity constraint of QMIX [28] to compute the minimax term with linear time complexity. QMixMax can be considered as an instantiation of ARTS, and thus is trained with all concepts proposed in ARTS.
- We empirically evaluate ARTS and QMixMax in a cyber physical production domain and show the effectiveness of our adversarial training scheme w.r.t. resilience against failures and testing capabilities.

2 BACKGROUND

2.1 Problem Formulation

General MAS problems can be formulated as *stochastic game* $M_{SG} = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \Omega \rangle$, where $\mathcal{D} = \{1, ..., N\}$ is a set of agents, $\mathcal{S}$ is a set of states s_t, $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_N$ is the set of joint actions $a_t = \langle a_{t,1}, ..., a_{t,N} \rangle$, $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the transition probability, $r_{t,i} = \mathcal{R}_i(s_t, a_t)$ is the reward of agent $i \in \mathcal{D}$, $\mathcal{Z}$ is a set of local observations $z_{t,i}$ for each agent i, and $\Omega(s_t, a_t) = z_{t+1} = \langle z_{t+1,1}, ..., z_{t+1,N} \rangle \in \mathcal{Z}^N$ is the joint observation function. The behavior of a MAS is defined by the (stochastic) *joint policy* $\pi(a_t|z_t) = \langle \pi_1(a_{t,1}|z_{t,1}), ..., \pi_N(a_{t,N}|z_{t,N}) \rangle$, where $\pi_i(a_{t,i}|z_{t,i})$ is the *local policy* or *decentralized policy* of agent i.

$\pi_i(a_{t,i}|z_{t,i})$ can be evaluated with a value function $Q_i^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[G_{t,i}|s_t, a_t]$, where $G_{t,i} = \sum_{k=0}^{\infty} \gamma^k \mathcal{R}_i(s_{t+k}, a_{t+k})$ is the return of agent i, and $\gamma \in [0, 1)$ is the discount factor. We denote the joint actions, observations, and policies without agent i by $a_{t,-i}$, $z_{t,-i}$ and π_{-i} respectively. The goal of each agent i is to find a *best response* π_i to the policies π_{-i} of all other agents which maximizes Q_i^{π}.

In this paper, we focus on two special cases of M_{SG}:

(1) *Cooperative MAS* $M_C = M_{SG}$, where all agents observe the same reward $r_t = \mathcal{R}_i(s_t, a_t) = \mathcal{R}_j(s_t, a_t)$ for all $i, j \in \mathcal{D}$ to maximize a common global objective. We use cooperative MAS to model our original *target system.*

(2) *Zero-sum games* $M_Z = M_{SG}$, where $N = 2$ and $\mathcal{R}_1(s_t, a_t) = -\mathcal{R}_2(s_t, a_t)$. We use zero-sum games to model (worst-case) *failures* in our target system.

2.2 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is a popular approach to learn best responses π_i for each agent i from experience E. Experience is generated from interaction with the environment, where a global *experience tuple* is defined by $e_t = \langle s_t, z_t, a_t, r_t, s_{t+1}, z_{t+1} \rangle \in E$. In this paper, we focus on *value-based MARL*, where the value function $Q_i^{\pi}(s_t, a_t)$ is approximated for each agent in order to derive local policies π_i. *Independent Q-Learning (IQL)* is a simple MARL approach, where $Q_i^{\pi}(s_t, a_t)$ is approximated with $\hat{Q}_i(z_{t,i}, a_{t,i})$, where each agent i is regarding all other agents as part of the environment and applies the following learning rule [37]:

$$\hat{Q}_i(z_{t,i}, a_{t,i}) \leftarrow (1 - \alpha)\hat{Q}_i(z_{t,i}, a_{t,i}) + \alpha V \tag{1}$$

where $V = r_{t,i} + \gamma max_{a_{t+1,i} \in \mathcal{A}_i} \hat{Q}_i(z_{t+1,i}, a_{t+1,i})$ and α is the learning rate. $\pi_i(a_{t,i}|z_{t,i})$ can be realized by sampling $a_{t,i}$ according to $\hat{Q}_i(z_{t,i}, a_{t,i})$ using e.g., ϵ-greedy or Boltzmann exploration [17, 37].

IQL offers good scalability, since the maximization in V is more tractable for $\mathcal{A}_i$ than for the joint action space $\mathcal{A}$ which scales exponentially w.r.t. N. However, IQL suffers from *non-stationarity* caused by simultaneously learning agents [6, 19, 21, 28, 33].

For large-scale domains, Q_i^{π} can be approximated with a function approximator $\hat{Q}_{i,\theta}$ with parameters θ which could be a deep neural network like a *Deep Q-Network (DQN)* [17, 22, 36].

2.2.1 Cooperative MARL.

In cooperative MAS $M_C = M_{SG}$, learning is based on a global reward $r_t = \mathcal{R}(s_t, a_t)$ which makes the deduction of individual agent contributions difficult [4, 44]. To mitigate this *multi-agent credit assignment* problem, factorization of the global value function $Q^{\pi}(s_t, a_t)$ has been proposed [28, 33, 34]. *QMIX* is a state-of-the-art MARL algorithm, which approximates $\hat{Q}_{\theta} \approx Q^{\pi}$ and its decomposition into $\hat{Q}_{i,\theta_i}$ by minimizing the loss $\mathcal{L}(\theta)$ w.r.t. θ and b randomly sampled tuples from previous experiences E using *stochastic gradient descent (SGD)* [28]:

$$\mathcal{L}(\theta) = \frac{1}{b} \sum_{k=1}^{b} [(y_{t,k} - \hat{Q}_{\theta}(s_t, a_t))^2] \tag{2}$$

where $y_{t,k} = r_t + \gamma max_{a_{t+1} \in \mathcal{A}} \hat{Q}_{\theta^-}(s_{t+1}, a_{t+1})$. θ^- are the parameters of a *target network* which are periodically set to $\theta^- \leftarrow \theta$ after C iterations to ensure stable learning [22].

QMIX enforces monotonicity of $\hat{Q}_{\theta}$ in per-agent value $\hat{Q}_{i,\theta_i}$ such that $\frac{\delta \hat{Q}_{\theta}}{\delta \hat{Q}_{i,\theta_i}} \geq 0$ for all $i \in \mathcal{D}$ [28]. This enables a tractable *argmax*-approximation of $\hat{Q}_{\theta}$ by using *argmax* on the local values $\hat{Q}_{i,\theta_i}$ with linear complexity w.r.t. N:

$$argmax_{a_t \in \mathcal{A}} \hat{Q}_{\theta} = \begin{pmatrix} argmax_{a_{t,1} \in \mathcal{A}_1} \hat{Q}_{1,\theta_1} \\ ... \\ argmax_{a_{t,N} \in \mathcal{A}_N} \hat{Q}_{N,\theta_N} \end{pmatrix} \tag{3}$$

A *mixing network* with non-negative weights, which are produced by hypernetworks, is used to learn a non-linear decomposition of $\hat{Q}_{\theta}$. QMIX adopts the paradigm of *centralized training and decentralized execution (CTDE)*, where learning usually takes place in a laboratory or in a simulated environment. Thus, global information can be integrated into the training process (e.g., into the hypernetworks of QMIX) to learn coordinated policies for decentralized execution in the locally observable world [6, 21, 28, 33].

2.2.2 Zero-Sum MARL.

In zero-sum games $M_Z = M_{SG}$, where two players or agents $i = 1$ and $j = 2$ have opposing goals, the Q-Learning update from Eq. 1 can be modified for agent i (and analogously agent j) by adjusting the variable $V = V_{minimax}$ [19]:

$$V_{minimax} = r_{t,i} + max_{\pi_i} min_{a_{t,j} \in \mathcal{A}_j} \sum_{a_{t,i} \in \mathcal{A}_i} \pi_i(a_{t,i}|z_{t,i}) \hat{Q}_i(s_t, a_t) \tag{4}$$

This algorithm, is called *Minimax-Q*. Note that $\hat{Q}_j = -\hat{Q}_i$, thus theoretically *only one* value function (Q_i^{π} or Q_j^{π}) needs to be approximated. Also note, that global information like the state s_t and the joint action a_t are required to compute $V_{minimax}$ in Eq. 4.

3 RELATED WORK

3.1 Adversarial Learning and Co-Evolution

Adversarial learning is typically modeled as a zero-sum game, where two opponents are trained in a minimax fashion to improve each other's performance and robustness [11, 23, 26]. *Generative adversarial networks* are a popular example, where a generator is trained to produce data in order to fool a discriminator which in turn is trained to distinguish between real and generated data [11]. The generator can be trained with unsupervised learning or RL [9, 11].

Self-play RL is the most simple way of adversarial RL, where a single agent is trained to play against itself [2, 26, 30–32, 39]. This ensures an adequate difficulty level to improve steadily and can lead to complex behavioral strategies emerging from simple rules.

In H_∞ control, disturbances and errors in the training environment are modeled to implement robust controllers [45]. This concept was adopted for model-based RL to confront the controller or agent with adversarial disturbances or with difficult environment settings in order to learn robust policies for the real world [23, 27]. *Robust Adversarial RL (RARL)* is a model-free approach, where the training environment itself is modeled as an RL agent which is able to manipulate forces to further challenge the original agent [26].

Co-evolution is another adversarial learning paradigm, where the environment can be adversarially modified by a metaheuristic search or optimization algorithm, which can provide a population of challenging environment instances for the learning agent [7, 8, 43].

While adversarial learning forms the basis of our approach, we focus on *cooperative MAS*, where arbitrary agent failures may occur. Instead of the environment, we adversarially modify the *team* itself to train robust policies for improved *resilience* in the target MAS.

3.2 Testing in Reinforcement Learning

Several machine learning approaches have been shown to be vulnerable to adversarial examples due to bias and insufficient generalization [3, 12, 35]. Some works propose search algorithms like importance sampling to determine difficult settings for RL agents to uncover flaws in learned policies [29, 40]. RL can also be used to adversarially determine flaws in fixed RL agents [10, 41].

In contrast to these existing works which are very specialized and focus on static settings of the environment (e.g., alignment of obstacles or adversarial input), we focus on *generalized testing* of the *dynamic aspects* in MAS, where *all agents* are a potential source of failure. While failures are common in distributed systems, a cooperative MAS should be resilient against such arbitrary failures.

3.3 Multi-Agent Reinforcement Learning

Minimax-Q was proposed in [19] as an adaptation of Q-Learning for zero-sum games. While guaranteeing convergence to safe policies w.r.t. worst-case opponents, the minimization in Eq. 4 is expensive if the action space $\mathcal{A}_j$ of the opponent j is large [18].

Recently, deep learning approaches to MARL have been proposed to tackle more complex games. A general CTDE framework was proposed in [21], where $Q_i^\pi(s_t, a_t)$ is approximated for each agent i to learn best responses for each agent simultaneously in a centralized fashion. That framework was extended for minimax optimization, where the non-linear critic is approximated by a local linear function to handle the expensive minimization in Eq. 4 [18].

Population-based training (PBT) has become a popular alternative to the CTDE paradigm, where agents are trained independently with different objectives which are learned or shaped dynamically [14, 16, 20]. Training agents with a diverse population of potential teammates or opponents can lead to robust policies that generalize well across different MAS settings and team compositions [14, 42].

Our approach can be regarded as a *combination* of CTDE and PBT. We integrate global information into our training process to learn decentralized policies. To consider potential failures of *each agent*, we maintain a pool representing the agents of the cooperative target MAS and a pool of adversaries representing the "failed" counterparts of each agent. During training, we sample different team compositions to train each agent against a random mixture of original and failed agents to ensure generalization of both pools w.r.t. varying team compositions and failure scenarios. This way, we can train *resilient* cooperative MAS and *challenging test suites* represented by the adversary pool. In addition, we propose a specialized approach which approximates Minimax-Q in a scalable way by exploiting the monotonicity constraint of QMIX (Eq. 3).

4 TRAINING AND TESTING RESILIENT MAS

4.1 Terminology

Our *target system* is a cooperative MAS $M_C = M_{SG}$, where all N agents maximize a common global objective.

For learning and testing, we formulate a *mixed cooperative-competitive game* M_X [21] with two competing agent teams $\mathcal{D}_X = \mathcal{D}_{pro} \cup \mathcal{D}_{ant}$ with $\mathcal{D}_{pro} \cap \mathcal{D}_{ant} = \emptyset$ and $|\mathcal{D}_X| = N$. The *protagonists* $i \in \mathcal{D}_{pro} = \{1, ..., N_{pro}\}$ represent functional agents, which maximize the original objective, while the *antagonists* $j \in \mathcal{D}_{ant} = \{1, ..., N_{ant}\}$ represent malicious agents, which attempt to minimize that objective. Thus, $M_X = M_Z$ can be regarded as a zero-sum game between a team of protagonists $\mathcal{D}_{pro}$ and a team of antagonists $\mathcal{D}_{ant}$, while the teams themselves are internally cooperative since all protagonists or antagonists share a common goal respectively. While all protagonists i observe a common reward of $r_{t,i} = \mathcal{R}_i(s_t, a_t)$, all antagonists j observe a common reward of $r_{t,j} = \mathcal{R}_j(s_t, a_t) = -r_{t,i}$ which is the negative of the protagonist reward. Therefore, we only regard the protagonist reward $r_{t,i} = r_t = \mathcal{R}(s_t, a_t)$ of M_C.

4.2 Antagonist-Ratio Training Scheme (ARTS)

Our goal is to simultaneously train robust *protagonists* for resilient cooperative MAS and *antagonists* as a challenging test suite.

In practice, all N agents in a MAS are a potential source of failure due to hardware or software flaws, or malicious attacks. Given a fixed *antagonist-ratio*[1] $R_{ant} = \frac{N_{ant}}{N}$, we would theoretically need $\binom{N-1}{N \cdot R_{ant}}$ antagonist teams per agent $i \in \mathcal{D} = \{1, ..., N\}$ to robustly train each of the N agents against all possible failure compositions.

To alleviate this scalability issue, we assume different observation sets $\mathcal{Z}_X = \mathcal{Z}_{pro} \cup \mathcal{Z}_{ant}$ with $\mathcal{Z}_{pro} \cap \mathcal{Z}_{ant} = \emptyset$, where the protagonists are unable to explicitly distinguish between protagonists and antagonists, while the antagonists are able to do so. When using

[1] A variable antagonist-ratio during training would be useful for curriculum learning and generalization [14, 16]. However, a systematic evaluation as proposed in this paper would be much harder, thus we defer an analysis with variable R_{ant} to future work.

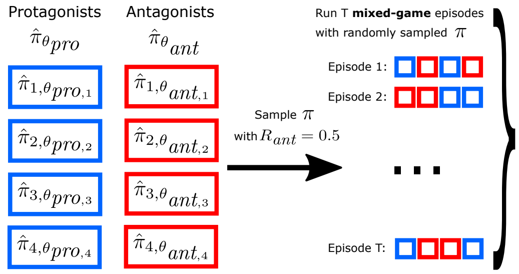

Protagonists
$\hat{\pi}_{\theta_{pro}}$
Antagonists
$\hat{\pi}_{\theta_{ant}}$
$\hat{\pi}_{1,\theta_{pro,1}}$
$\hat{\pi}_{2,\theta_{pro,2}}$
$\hat{\pi}_{3,\theta_{pro,3}}$
$\hat{\pi}_{4,\theta_{pro,4}}$
$\hat{\pi}_{1,\theta_{ant,1}}$
$\hat{\pi}_{2,\theta_{ant,2}}$
$\hat{\pi}_{3,\theta_{ant,3}}$
$\hat{\pi}_{4,\theta_{ant,4}}$
Sample π
with $R_{ant} = 0.5$
Run T **mixed-game** episodes
with randomly sampled π
Episode 1:
Episode 2:
Episode T:
Train policies
or evaluate
test results

Figure 1: Overview of the components and an example process of ARTS with $R_{ant} = 0.5$ and $N = 4$.

function approximation (e.g., deep learning), the antagonists should be able to generalize across different protagonist settings, while the protagonists have to learn strategies that are robust against varying team compositions without "blindly trusting" each other, in contrast to prior works. Thus, given our cooperative target MAS M_C with N protagonists $i \in \mathcal{D} = \{1, ..., N\}$, we additionally train one corresponding antagonist for each protagonist i, requiring us only to train $2N$ agents in total as shown in Fig. 1.

ARTS *combines* CTDE and PBT: The value function Q^π is approximated in a *centralized fashion* with global information to learn decentralized protagonist and antagonist policies. The corresponding joint policies can be regarded as *pools* $\hat{\pi}_{\theta_{pro}}$ and $\hat{\pi}_{\theta_{ant}}$. In each iteration, a mixed game M_X is created by defining π consisting of a *protagonist part* of $N(1 - R_{ant})$ policies sampled from $\hat{\pi}_{\theta_{pro}}$ and an *antagonist part* of $N \cdot R_{ant}$ policies sampled from $\hat{\pi}_{\theta_{ant}}$. π may only contain $\hat{\pi}_{i,\theta_{pro,i}}$ and $\hat{\pi}_{j,\theta_{ant,j}}$, if $i \neq j$ for all $i, j \in \mathcal{D}$ (Fig. 1).

The M_X is then run with π in one *episode* which consists of h time steps to generate *experience samples* $e_t = \langle s_t, z_t, a_t, r_t, s_{t+1}, z_{t+1} \rangle$, which are stored in a central *experience buffer* E. Note that the joint action a_t consists of the protagonist joint action $a_{t,pro}$ and the antagonist joint action $a_{t,ant}$. After the episode, a function approximator $\hat{Q}_\omega \approx Q^\pi$ with parameters ω is updated for each $e_t \in E$ by minimizing the following loss w.r.t. ω:

$$\mathcal{L}(\omega) = \frac{1}{h} \sum_{t=1}^{h} [(y_t - \hat{Q}_\omega(s_t, a_t))^2] \tag{5}$$

where $y_t = r_t + \gamma \hat{Q}_{\omega^-}(s_{t+1}, a_{t+1})$ and $a_{t+1} \sim \pi(a_{t+1}|z_{t+1})$. Similarly to QMIX, a target network with parameters ω^- is used [22, 28].

With E, the protagonist and the antagonist pool can be updated via general cooperative MARL methods Ψ_{pro} and Ψ_{ant} respectively by regarding their respective joint action parts and the correct sign of r_t and $\hat{Q}_\omega$. $\hat{Q}_\omega$ can be used as target network for value-based methods [28, 33], or as critic for policy gradient methods [6, 21].

CTDE approaches to cooperative MARL typically require the joint action of *all agents* to approximate a global value function (e.g., Eq. 2) [6, 21, 28, 33]. Since only a sampled fraction of the protagonist or antagonist pool participated in the episode, we use the current local policies of the respective "unused" agents to estimate the joint action but do not perform an update on these "unused" policies (e.g., by setting the corresponding gradients to zero).

The complete formulation of ARTS is given in Algorithm 1, where $\mathcal{D}$ is the set of agents in the cooperative target MAS M_C, N is the number of agents in M_C, Ψ_{pro} is the protagonist MARL algorithm, $\hat{\pi}_{\theta_{pro}}$ is the approximator of the protagonist joint policy (the target joint policy for M_C), Ψ_{ant} is the antagonist MARL algorithm, $\hat{\pi}_{\theta_{ant}}$ is the approximator of the antagonist joint policy (the failure representation for M_C), $\hat{Q}_\omega$ is the global value function approximator, $Tr = 1$ if ARTS is used for training (otherwise for testing), and $R_{ant} \in [0, 1)$ is the antagonist-ratio. ARTS returns a statistic Δ which can contain domain relevant data (e.g., the average return or domain specific violations), and the (trained) parameters θ_{pro}, θ_{ant}, and ω of the respective function approximators.

Algorithm 1 Antagonist-Ratio Training Scheme (ARTS)

```
procedure ARTS($\mathcal{D}, N, \Psi_{pro}, \hat{\pi}_{\theta_{pro}}, \Psi_{ant}, \hat{\pi}_{\theta_{ant}}, \hat{Q}_\omega, Tr, R_{ant}$)
    Initialize $\theta_{pro}$ for $\hat{\pi}_{\theta_{pro}}$, $\theta_{ant}$ for $\hat{\pi}_{\theta_{ant}}$, and $\omega$ for $\hat{Q}_\omega$
    Create statistic $\Delta$
    for $x = 1, T$ do
        Create $\mathcal{D}_{ant}$: randomly draw $N \cdot R_{ant}$ agents from $\mathcal{D}$
        Create $\mathcal{D}_{pro}$: select $\{i \in \mathcal{D} | i \notin \mathcal{D}_{ant}\}$
        for $i = 1, N$ do    ▷ Create $M_X$ with $\mathcal{D}_X = \mathcal{D}_{pro} \cup \mathcal{D}_{ant}$
            if $i \in \mathcal{D}_{ant}$ then $\pi_i \leftarrow \hat{\pi}_{i,\theta_{ant,i}}$
            if $i \in \mathcal{D}_{pro}$ then $\pi_i \leftarrow \hat{\pi}_{i,\theta_{pro,i}}$
        $\pi \leftarrow \langle \pi_1, ..., \pi_N \rangle$
        Run one $M_X$ episode with joint policy $\pi$
        Store all experiences $e_t = \langle s_t, z_t, a_t, r_t, s_{t+1}, z_{t+1} \rangle$ in $E$
        Update statistic $\Delta$ w.r.t. $E$
        if $Tr = 1$ then
            Update $\omega$ w.r.t. Eq. 5
            Update $\theta_{pro}$ with $\Psi_{pro}(\theta_{pro}, E, \mathcal{D}_{pro})$ w.r.t. $\hat{Q}_\omega$
            Update $\theta_{ant}$ with $\Psi_{ant}(\theta_{ant}, E, \mathcal{D}_{ant})$ w.r.t. $-\hat{Q}_\omega$
    return $\langle \Delta, \theta_{pro}, \theta_{ant}, \omega \rangle$
```

ARTS is strongly inspired by RARL [26] which was devised for single-agent problems. The main difference is that ARTS focuses on training *resilient MAS*, where arbitrary parts of the MAS themselves can act adversarially. For that, we maintain *two pools* of agents from which we sample a MAS setting which mixes protagonists and antagonists in an arbitrary way. While the protagonists have to learn robust policies to ensure resilience against failures in the target MAS, the antagonists have to generalize their adversarial strategies which can be used as *test suites* of other MAS. Also unlike RARL which trains both parties alternatingly, we train protagonists and antagonists *simultaneously* and try to synchronize training with a global value function $\hat{Q}_\omega$. We also regard the *testing capabilities* of ARTS w.r.t. existing systems as described in Section 4.4.

4.3 QMixMax

In addition to ARTS formulated in Section 4.2, we propose an approximated Minimax-Q scheme which we call *QMixMax* for the special case of having $N - 1$ antagonists in the system, which can be considered the *most extreme* but theoretically still solvable worst-case scenario in most MAS settings. QMixMax can be viewed as an instantiation of ARTS and its structure is shown in Fig. 2.

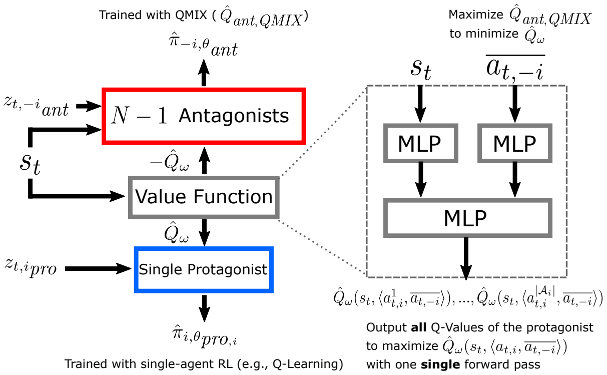

Trained with QMIX ($\hat{Q}_{ant,QMIX}$)
$\hat{\pi}_{-i,\theta_{ant}}$
$z_{t,-i_{ant}}$
$N-1$ Antagonists
s_t
$-\hat{Q}_\omega$
Value Function
$\hat{Q}_\omega$
$z_{t,i_{pro}}$
Single Protagonist
$\hat{\pi}_{i,\theta_{pro,i}}$
Trained with single-agent RL (e.g., Q-Learning)
Maximize $\hat{Q}_{ant,QMIX}$
to minimize $\hat{Q}_\omega$
s_t
$\overline{a_{t,-i}}$
MLP
MLP
MLP
$\hat{Q}_\omega(s_t, \langle a^1_{t,i}, \overline{a_{t,-i}} \rangle), ..., \hat{Q}_\omega(s_t, \langle a^{|\mathcal{A}_i|}_{t,i}, \overline{a_{t,-i}} \rangle)$
Output **all** Q-Values of the protagonist
to maximize $\hat{Q}_\omega(s_t, \langle a_{t,i}, \overline{a_{t,-i}} \rangle)$
with one **single** forward pass

Figure 2: Structure and information flow between protagnists, antagonists, and value function $\hat{Q}_\omega$ in QMixMax.

We propose to train the antagonists with $\Psi_{ant} = QMIX$ (Algorithm 1) and approximate the minimization in Eq. 4, which requires the *argmax* of $\hat{Q}_{ant,QMIX}$ (which is the *argmax* of the local antagonist values $\hat{Q}_{ant,QMIX,i}$ from Eq. 3) which can be computed with linear complexity. This changes $V_{minimax} = V_{QMixMax}$ of Eq. 4 to:

$$V_{QMixMax} = r_{t,i} + max_{a_{t,i} \in \mathcal{A}_i} \hat{Q}_\omega(s_t, \langle a_{t,i}, \overline{a_{t,-i}} \rangle) \tag{6}$$

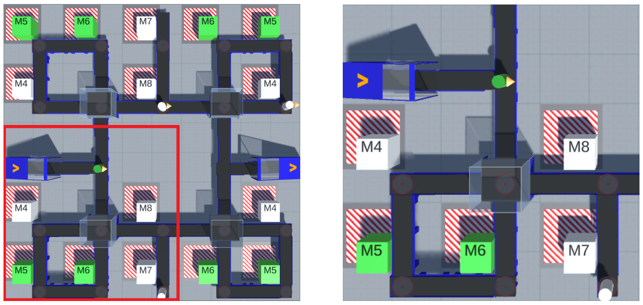
where $\overline{a_{t,-i}} = argmax_{a_{t,-i}}(\hat{Q}_{ant,QMIX})$. $\hat{Q}_{ant,QMIX}$ must not be confused with $\hat{Q}_\omega$ from Algorithm 1. $\hat{Q}_{ant,QMIX}$ is the approximated global value function of the *antagonists* and uses $-\hat{Q}_\omega$ as a target network and $-r_t$ as reward (Fig. 2). With Eq. 6, the single protagonist i (according to Algorithm 1, we need to train N protagonists) is able to approximate the Minimax-Q Learning rule in a scalable way by using e.g., $\Psi_{pro} = Q\text{-}Learning$ of Eq. 1 with $V = V_{QMixMax}$.

4.4 Testing Resilience in MAS

Since testing can also be modeled as an adversarial process [10, 29, 40, 41], we propose to use the ARTS formulation in Algorithm 1 for testing two given joint policies $\hat{\pi}_{\theta_{pro}}$ and $\hat{\pi}_{\theta_{ant}}$, where $\hat{\pi}_{\theta_{pro}}$ represents the MAS to be tested and $\hat{\pi}_{\theta_{ant}}$ represents the test suite. R_{ant} represents the *test difficulty*, which determines "how much antagonism" $\hat{\pi}_{\theta_{pro}}$ will face during testing. To run the test process Tr needs to be set to zero ($Tr \neq 1$ according to Algorithm 1), to avoid modification of $\hat{\pi}_{\theta_{pro}}$ and $\hat{\pi}_{\theta_{ant}}$ during testing. The statistic Δ, which is returned at the end of ARTS, can be used to log the test results (e.g., domain specific violations) for evaluation afterwards.

ARTS can be considered as a *black box test approach*, where arbitrary failures are simulated by randomly replacing protagonists of the target joint policy $\hat{\pi}_{\theta_{pro}}$ with antagonists from the test suite $\hat{\pi}_{\theta_{ant}}$, without requiring further domain or system knowledge.

Note that ARTS cannot be used for verification, since the purpose of testing is to *detect* flaws in a *feasible* way. A good test suite should be able to detect flaws (e.g., significant drops in performance or domain specific violations) with *high probability* in faulty systems. If the test fails to find any flaw, the tested system might *still* be faulty. Thus, we consider a system to be *resilient* (rather than correct), when it is able to pass a variety of test suites without failure.



(a) CPPS overview **(b) bottom left corner of the CPPS**

Figure 3: (a) Overview of the whole CPPS with $N = 4$, a blue entry on the left, and a blue exit on the right. The green and white cylinders represent agents. The machines are represented by the numbered boxes with each number after the 'M' stands for the machine type. The green agent i has *tasks*$_i$ = [{5, 6}] with the corresponding machines being colored green. The black paths represent bidirectional conveyor belts and the transparent boxes represent hubs. (b) Zoomed view of the bottom left corner of the CPPS which is marked by the red rectangle in Fig. 3a.

5 EXPERIMENTAL SETUP

5.1 Cyber Physical Production System (CPPS)

Our *Cyber Physical Production System (CPPS)* domain consists of a 5 × 5 grid of machines as shown in Fig. 3 and N agents with each having a list of four random tasks $tasks_i$ organized in two *buckets*. An example is shown in Fig. 3a. All agents start at the *blue entry* on the left (Fig. 3a) and are able to enqueue at their current machine, move north, south, west, east, or do nothing. At every time step, each machine processes one agent in its queue. If a task in its current bucket matches with the machine type, the task is removed from the agent's task list with a global reward of +1. An agent i is *complete*, if $tasks_i = \emptyset$ and it reaches the *blue exit* on the right (Fig. 3a), yielding another reward of +1. For each incomplete agent, a global reward of -0.1 is given at every time step. All agents only know their own tasks while being able to perceive each other's positions. All agents are only allowed to move along the black paths which represent bidirectional conveyor belts and may only share the same position at the transparent boxes which represent hubs. Thus, all agents ideally have to coordinate to avoid conflicts or collisions to ensure fast completion of all tasks.

5.2 Learning Algorithms

To evaluate ARTS, we implemented IQL (Eq. 1) with deep neural networks, which we refer to as *DQN*, and *QMIX* as MARL algorithms for Ψ_{pro} and Ψ_{ant} in Algorithm 1. We experimented with antagonist-ratios of $R_{ant} \in \{0, 0.25, 0.5, 0.75\}$. If $R_{ant} = 0$, then ARTS is equivalent to the original cooperative MARL algorithm Ψ_{pro}, since no antagonist will be sampled for training (Algorithm 1 and Fig. 1). We always put the antagonist-ratio used during training in brackets e.g., *DQN (0.5)* is used for agents that were trained with ARTS using $R_{ant} = 0.5$ and $\Psi_{pro} = \Psi_{ant} = DQN$ (Algorithm 1).

For *QMixMax*, we use *QMIX* as Ψ_{ant} and a single-agent *DQN* for Ψ_{pro} to approximate Eq. 6 as suggested in Section 4.3. For comparison, we also implemented a Minimax-Q version using *DQN* as Ψ_{ant} instead of *QMIX*, which we simply call *Minimax*. All algorithms use indpendent ϵ-greedy exploration with a linear decay of 0.00002 per training time step until a minimum value of 0.01 is reached.

5.3 Neural Network Architecture

We used deep neural networks to implement $\hat{Q}_{\omega}$, $\hat{Q}_{ant,QMIX}$ for QMIX (Section 4.3), and $\hat{Q}_{i,\theta_i}$ for each agent i. An experience buffer E was implemented to store the last 20,000 transitions and to sample minibatches of size 64 to perform SGD using ADAM with a learning rate of 0.0005. We set $\gamma = 0.95$. We also used target networks, which were updated every $C = 4,000$ SGD steps [22].

Since CPPS is a gridworld, the states and local observations are encoded as multi-channel image as proposed in [13, 25]. We implemented all networks as multilayer perceptron (MLP) and flattened the multi-channel images before feeding them into the networks. $\hat{Q}_{i,\theta_i}$ for DQN and QMIX has two hidden layers of 64 units. The output of $\hat{Q}_{i,\theta_i}$ has one linear unit per local action. The mixing network of QMIX has a single hidden layer of 64 units and a single linear output unit. $\hat{Q}_{\omega}$ has two input streams for s_t and a_t with each having a hidden layer of 128 units which are concatenated and fed into a common MLP with one hidden layer of 256, another hidden layer of 128 units, and a single linear output unit ($|\mathcal{A}_i| = 6$ units for *QMixMax*, Fig. 2). All hidden layers use ELU activation.

6 RESULTS

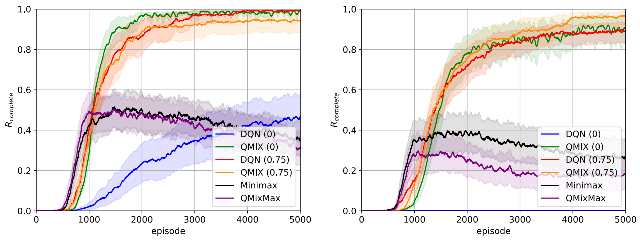
We conducted various CPPS experiments with $N = 4, 8$ agents and ARTS using different MARL algorithms and R_{ant}. An *episode* is reset after $h = 50$ time steps or when all protagonists are complete. A training *run* of each setting according to Algorithm 1 consists of $T = 5,000$ episodes and is repeated 50 times.

Measuring performance with the return $G_t = \sum_{k=0}^{\infty} \gamma^k r_t$ is not sufficient to evaluate generalization and to compare between different ARTS settings, since fully cooperative MAS without any antagonism ($R_{ant} = 0$) would always achieve a higher return than MAS with $R_{ant} > 0$. Thus, we use the average *completion rate* of the protagonists $R_{complete} = \mathbb{E}[\frac{N_{pro,complete}}{N_{pro}}]$, where $N_{pro} = N(1 - R_{ant})$ is the total number of protagonists in M_X and $N_{pro,complete}$ is the number of complete protagonists after an episode. The testing performance is measured with the average *flaw probability* $P_{flaw} = 1 - R_{complete}$, which is the chance of preventing faulty protagonists from completing. Both are recorded in Δ (Algorithm 1).

6.1 ARTS w.r.t. the Antagonist-Ratio

We analyzed the effect of different values of $R_{ant} \in \{0, 0.25, 0.5, 0.75\}$ in ARTS using *DQN* or *QMIX* for $\Psi_{pro} = \Psi_{ant}$ for training ($Tr = 1$ in Algorithm 1). We also trained ARTS in extreme scenarios with only one protagonist and $N - 1$ antagonists using *Minimax* with $\Psi_{ant} = DQN$ and *QMixMax* with $\Psi_{ant} = QMIX$, where the protagonists were trained with $\Psi_{pro} = DQN$ to approximate Eq. 6.

The learning progress of the protagonists trained with ARTS using $R_{ant} \in \{0, 0.75\}$ with *DQN* or *QMIX* as well as *Minimax* and *QMixMax* is shown in Fig. 4. In the 4-agent setting, *DQN (0.75)*, *QMIX (0)*, and *QMIX (0.75)* progress similarly well, with each protagonist converging to a completion rate $R_{complete}$ close to 1, while *DQN (0)* progresses much slower. In the first 1,000 episodes, *Minimax* and *QMixMax* progress slightly faster than the other approaches, but their performance significantly detoriates after 3,000 episodes. The 8-agent settings show a similar trend, with *DQN (0)* failing to complete any protagonist during training.



(a) training with 4 agents

(b) training with 8 agents

Figure 4: Learning progress of 50 runs of *DQN* and *QMIX* using ARTS with $R_{ant} \in \{0, 0.75\}$, *Minimax*, and *QMixMax*. Shaded areas show the 95 % confidence interval.

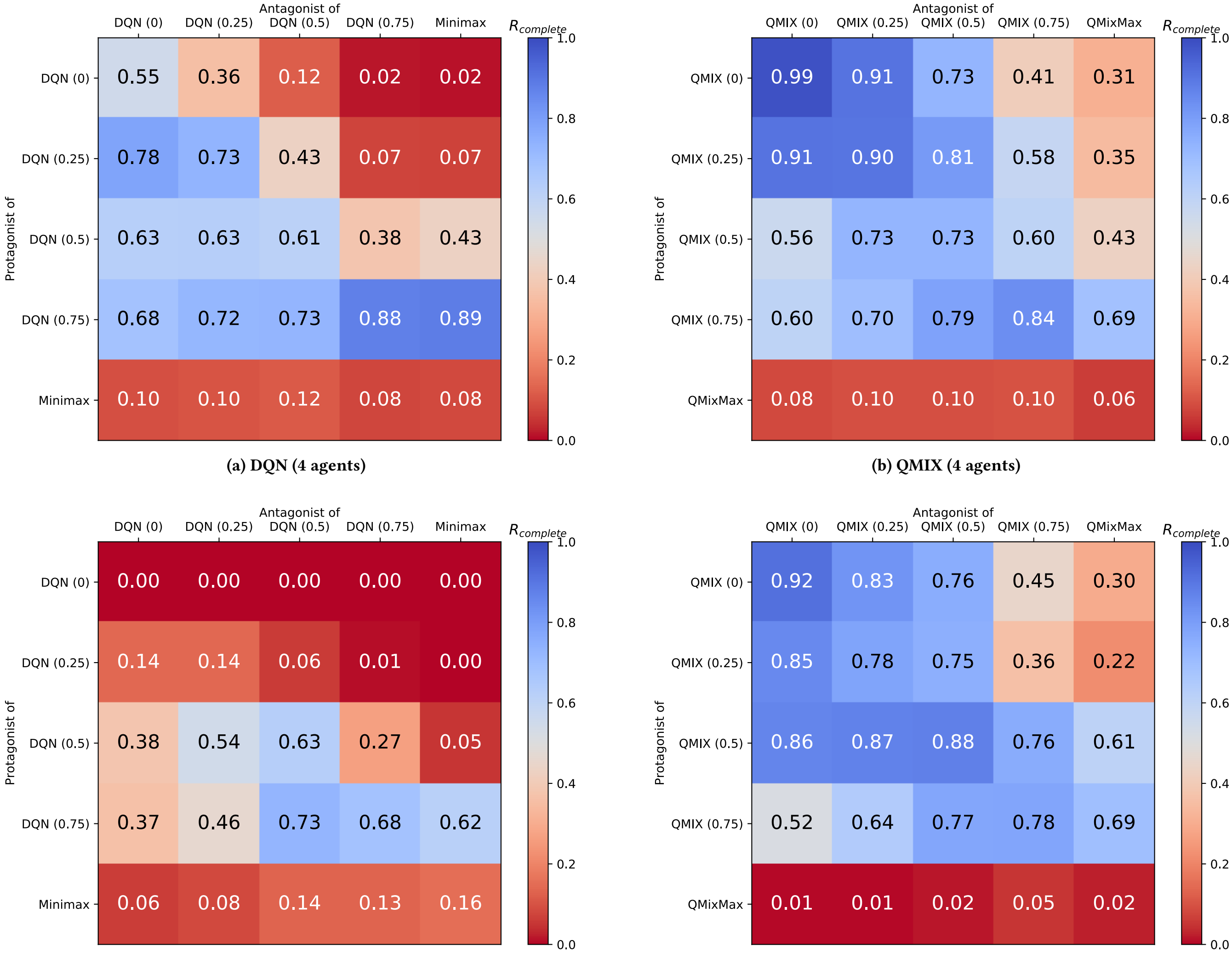
The generalization and resilience of each *DQN* and *QMIX* setting w.r.t. R_{ant} for $N = 4, 8$ agents was tested by randomly drawing 50 protagonist joint policies $\hat{\pi}_{\theta_{pro}}$ for each setting from their 50 training runs (Section 6) and tested them against antagonist joint policies $\hat{\pi}_{\theta_{ant}}$ which were also drawn randomly from their corresponding runs. With each drawn $\langle \hat{\pi}_{\theta_{pro}}, \hat{\pi}_{\theta_{ant}} \rangle$-pair, we ran $T = 100$ random episodes using ARTS for testing ($Tr = 0$ in Algorithm 1) with the antagonist-ratio R_{ant} used for training $\hat{\pi}_{\theta_{ant}}$ to compute $R_{complete}$.

For example, if $\hat{\pi}_{\theta_{pro}}$ of *DQN (0.25)* was sampled against $\hat{\pi}_{\theta_{ant}}$ of *DQN (0.75)*, then $R_{ant} = 0.75$ was used for testing. If $\hat{\pi}_{\theta_{ant}}$ was trained with *DQN (0)* or *Random* (see Section 6.2), then $\hat{\pi}_{\theta_{pro}}$ was tested against the *protagonists* of the other setting with a 50:50 mixture. Even if $\hat{\pi}_{\theta_{pro}}$ and $\hat{\pi}_{\theta_{ant}}$ were both trained with *DQN (0)*, the joint policies would likely originate from *different* training runs.

The results are shown in Fig. 5. We only tested *fully trained* policies and did not update them further as described in Section 4.4. The protagonists of *Minimax* and *QMixMax* perform poorly in all settings. Protagonists trained with *QMIX* always achieve a completion rate of $R_{complete} > 0.5$, when tested with $R_{ant} \leq 0.5$ (Fig. 5b & 5d). The protagonists of *QMIX (0.5)* and *QMIX (0.75)* also show relatively high resilience when being tested with $R_{ant} > 0.5$. Protagonists trained with *DQN* are more resilient towards the antagonistic settings, the larger R_{ant} was during training in the 4-agent setting (Fig. 5a). In the 8-agent setting, *DQN (0)* fails to complete any protagonist. All other *DQN* settings perform rather poorly when being tested with $R_{ant} < 0.5$. Protagonists of *DQN (0.75)* show the highest resilience towards tests with $R_{ant} \geq 0.5$, where they achieve a completion rate of $R_{complete} > 0.6$ (Fig. 5c).

6.2 ARTS w.r.t. different MAS

To further evaluate the generalization and resilience of each setting trained in Section 6.1, we conducted a cross-validation between protagonist-antagonist setups of a random baseline (*Random*), *DQN (0)*, *QMIX (0)*, *DQN (0.75)*, *QMIX (0.75)*, *Minimax*, and *QMixMax*.



(a) DQN (4 agents)

(b) QMIX (4 agents)

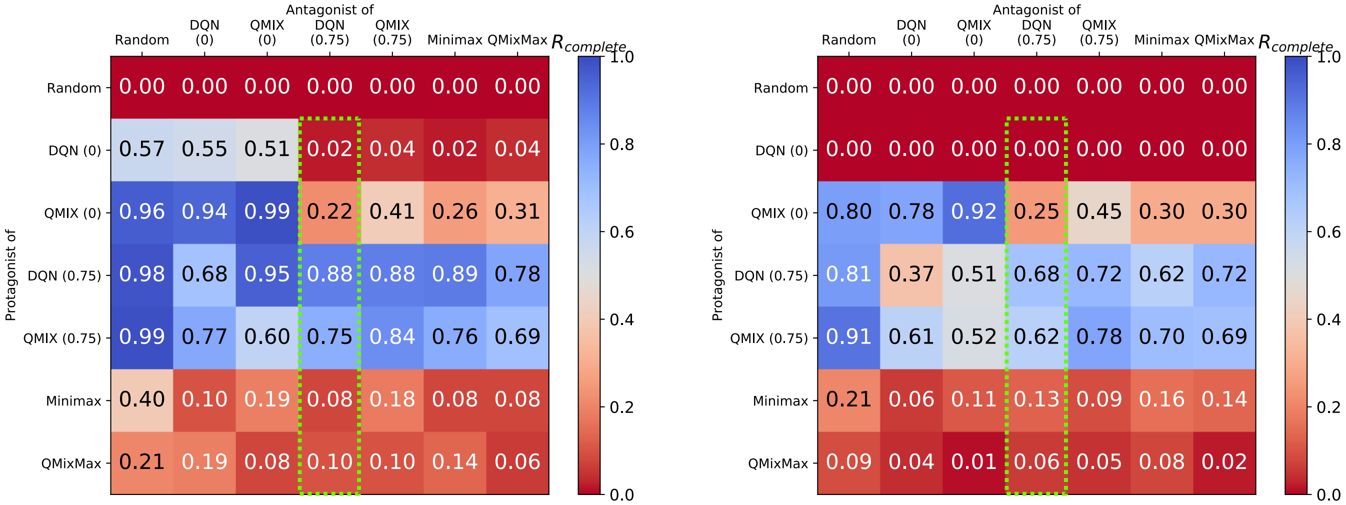
(c) DQN (8 agents)

(d) QMIX (8 agents)

Figure 5: Cross-validation of protagonists and antagonists trained with different R_{ant} using ARTS with *DQN* or *QMIX*. The protagonists were tested against $N \cdot R_{ant}$ antagonists, where R_{ant} for testing with ARTS ($Tr = 0$ in Algorithm 1) is the ratio originally used for training the antagonists. The numbers in the grid cells represent the average completion rates $R_{complete}$ of 50 randomly drawn protagonist-antagonist pairs, which each being tested in $T = 100$ random episodes.

The protagonists and antagonists were similarly picked as described in Section 6.1 for Fig. 5 and the results are shown in Fig. 6. In all protagonist settings, *Random* fails to complete, while *Minimax* and *QMixMax* perform poorly (although better than *Random*, Fig. 6). Protagonists of *DQN (0)* generalize relatively poorly - even in fully cooperative settings with $R_{ant} = 0$, in contrast to *QMIX (0)*. Both cooperatively trained protagonists never achieve a completion rate higher than 0.5, when being tested with $R_{ant} \geq 0.75$. The protagonists of *DQN (0.75)* and *QMIX (0.75)* are highly resilient against tests with $R_{ant} \geq 0.75$. However, both adversarially trained protagonists have a lower completion rate when being tested with fully cooperatively trained agents (Fig. 6b).

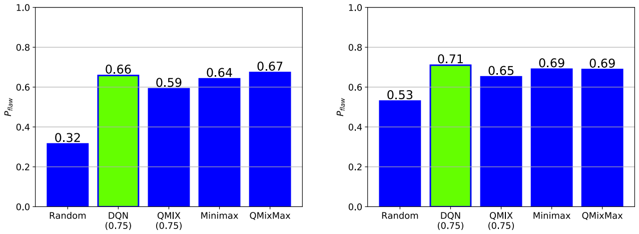
To quantify the ability of the antagonists of *Random*, *DQN (0.75)*, *QMIX (0.75)*, *Minimax*, and *QMixMax* to detect flaws in different protagonist settings, we averaged all $R_{complete}$-values in the columns (Fig. 6, except of *Random*, since it was not trained with MARL) of each corresponding antagonist setting and computed P_{flaw}, as shown in Fig. 7. The antagonists of *Random* have the lowest P_{flaw} which is only greater than 50% in the 8-agent setting. All other antagonists detect flaws with $P_{flaw} > 0.6$ except for *QMIX (0.75)* in the 4-agent setting. The antagonists of *QMixMax* have the highest chance of detecting flaws with $P_{flaw} = 0.67$ in the 4-agent setting, while *DQN (0.75)* has the highest chance with $P_{flaw} = 0.71$ in the 8-agent setting. While the antagonists of *QMixMax* always have a



(a) cross validation (4 agents)

(b) cross validation (8 agents)

Figure 6: Cross-validation of protagonists and antagonists trained with different MARL algorithms using ARTS. The protagonists were tested against $N \cdot R_{ant}$ antagonists, where R_{ant} for testing with ARTS is the ratio originally used for training the antagonists. The numbers in the grid cells represent the average completion rates $R_{complete}$ of 50 randomly drawn protagonist-antagonist pairs, which each being tested in $T = 100$ random episodes. The green dotted rectangles are explained in Fig. 7.



(a) testing with 4 agents

(b) testing with 8 agents

Figure 7: Test performance of different antagonists averaged over the corresponding test settings in Fig. 6. For example, P_{flaw} for *DQN (0.75)* (green bars) was computed for the values in the corresponding green dotted rectangle in Fig. 6.

higher chance of detecting flaws than *QMIX (0.75)* in all settings, *DQN (0.75)* has always a higher chance than *Minimax*.

7 DISCUSSION

In this paper, we proposed ARTS, a scalable adversarial training and testing scheme for MARL. ARTS reformulates the target MAS as a mixed cooperative-competitive game between a group of protagonist agents and a group of antagonist agents.

Our experiments show that standard MARL approaches without any antagonism can fail in the presence of failures due to overfitting to their teammate behavior and that the training progress itself is not sufficient for resilience evaluation, since the performance can significantly degrade when arbitrarily replacing agents (Fig. 5 and Fig. 6). Also, the training progress gives no insight about the quality of the antagonists, since protagonists could also fail due to the difficulty of the domain itself as indicated by *DQN (0)* in Fig. 4.

The results show that ARTS can improve resilience against failures in cooperative MAS, when training protagonists with a sufficient antagonist-ratio. Although the adversarially trained protagonists show resilience against a high antagonist-ratio, they seem to behave somewhat conservatively, since they cannot distinguish between protagonists and antagonists. Training robust agents, which can safely adapt at runtime would be interesting for future work.

ARTS generates test suites, which are able to detect flaws more reliably than simply replacing protagonists by random agents (Fig. 6). Analogously to the resilience case, it is much harder for test suites to uncover flaws when the protagonists behave conservatively as the antagonists try to approach the protagonists to block their paths, while the protagonists try to avoid any other agent. While *QMixMax* was unable to produce productive protagonists, it produced strong antagonists which were able to detect flaws more reliably than antagonists of *QMIX* and *Minimax* (Fig. 5 and Fig. 7).

Since the 8-agent CPPS is significantly more difficult than the 4-agent CPPS due to more potential conflicts and collisions, and due to a more difficult multi-agent credit assignment, the protagonists of *DQN (0)* fail to learn any meaningful policy. The protagonists of *QMIX (0)* perform slightly worse in the 8-agent setting than in the 4-agent setting. Therefore, it is easier to detect flaws in the 8-agent setting. This statement is supported in Fig. 7, where all antagonists (especially *Random*) have a higher chance of uncovering flaws in the 8-agent setting, than in the 4-agent setting. Thus, we recommend *QMixMax* for uncovering flaws in seemingly easy MAS settings.

ARTS could be further improved with adaptive sampling strategies for protagonists and antagonists, and variable antagonist-ratios during training based on episode outcomes or the value function.

REFERENCES

[1] Saswat Anand, Edmund K Burke, Tsong Yueh Chen, John Clark, Myra B Cohen, Wolfgang Grieskamp, Mark Harman, Mary Jean Harrold, Phil Mcminn, Antonia Bertolino, et al. 2013. An Orchestrated Survey of Methodologies for Automated Software Test Case Generation. *Journal of Systems and Software* 86, 8 (2013).

[2] Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking Fast and Slow with Deep Learning and Tree Search. In *Advances in Neural Information Processing Systems*. 5360–5370.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning*. 274–283.

[4] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. 2004. All Learning is Local: Multi-Agent Learning in Global Reward Games. In *Advances in neural information processing systems*. 807–814.

[5] Jon EDVARDSSON. 1999. A Survey on Automatic Test Data Generation. In *Proceedings of the 2nd Conference on Computer Science and Engineering, Linkoping, 1999*. 21–28.

[6] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *AAAI*.

[7] Thomas Gabor, Andreas Sedlmeier, Marie Kiermeier, Thomy Phan, Marcel Henrich, Monika Pichlmair, Bernhard Kempter, Cornel Klein, Horst Sauer, Reiner Schmid, and Jan Wieghardt. 2019. Scenario Co-evolution for Reinforcement Learning on a Grid World Smart Factory Domain. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19)*. ACM, New York, NY, USA, 898–906. https://doi.org/10.1145/3321707.3321831

[8] Thomas Gabor, Andreas Sedlmeier, Thomy Phan, Fabian Ritz, Marie Kiermeier, Lenz Belzner, Bernhard Kempter, Cornel Klein, Horst Sauer, Reiner Schmid, Jan Wieghardt, Marc Zeller, and Claudia Linnhoff-Popien. 2020. The Scenario Co-Evolution Paradigm: Adaptive Quality Assurance for Adaptive Systems. *International Journal on Software Tools and technology Transfer* (2020).

[9] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. 2018. Synthesizing Programs for Images using Reinforced Adversarial Learning. In *International Conference on Machine Learning*. 1652–1661.

[10] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. 2019. Adversarial Policies: Attacking Deep Reinforcement Learning. *arXiv preprint arXiv:1905.10615* (2019).

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in neural information processing systems*. 2672–2680.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. *ICLR 2015* (2015).

[13] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-Agent Control using Deep Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 66–83.

[14] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. 2019. Human-Level Performance in 3D Multiplayer Games with Population-based Reinforcement Learning. *Science* 364, 6443 (2019).

[15] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Julien Perolat, David Silver, Thore Graepel, et al. 2017. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

[16] Joel Z Leibo, Julien Perolat, Edward Hughes, Steven Wheelwright, Adam H Marblestone, Edgar Duéñez-Guzmán, Peter Sunehag, Iain Dunning, and Thore Graepel. 2019. Malthusian Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 1099–1107.

[17] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-Agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS.

[18] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. 2019. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. In *AAAI*.

[19] Michael L Littman. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Machine learning proceedings 1994*. Elsevier, 157–163.

[20] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. 2019. Emergent Coordination through Competition. *ICLR 2019* (2019).

[21] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* (2015).

[23] Jun Morimoto and Kenji Doya. 2001. Robust Reinforcement Learning. In *Advances in Neural Information Processing Systems*. 1061–1067.

[24] Frans A Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Vol. 1. Springer.

[25] Thomy Phan, Lenz Belzner, Thomas Gabor, and Kyrill Schmid. 2018. Leveraging Statistical Multi-Agent Online Planning with Emergent Value Function Approximation. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, 730–738.

[26] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust Adversarial Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2817–2826.

[27] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. 2017. EPOpt: Learning Robust Neural Network Policies using Model Ensembles. *ICLR 2017*.

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. 4292–4301.

[29] Avraham Ruderman, Richard Everett, Bristy Sikder, Hubert Soyer, Jonathan Uesato, Ananya Kumar, Charlie Beattie, and Pushmeet Kohli. 2018. Uncovering Surprising Behaviors in Reinforcement Learning via Worst-case Analysis. (2018).

[30] Arthur L Samuel. 1959. Some Studies in Machine Learning using the Game of Checkers. *IBM Journal of research and development* 3, 3 (1959), 210–229.

[31] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* (2016).

[32] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the Game of Go without Human Knowledge. *Nature* 550, 7676 (2017), 354.

[33] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. 5887–5896.

[34] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. 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 (Extended Abstract)*. IFAAMAS.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing Properties of Neural Networks. *ICLR 2014* (2013).

[36] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *PloS one* 12, 4 (2017), e0172395.

[37] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 330–337.

[38] Andrew S Tanenbaum and Maarten Van Steen. 2007. *Distributed Systems: Principles and Paradigms*. Prentice-Hall.

[39] Gerald Tesauro. 1995. Temporal Difference Learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–69.

[40] Jonathan Uesato, Ananya Kumar, Csaba Szepesvari, Tom Erez, Avraham Ruderman, Keith Anderson, Nicolas Heess, Pushmeet Kohli, et al. 2019. Rigorous Agent Evaluation: An Adversarial Approach to Uncover Catastrophic Failures. *ICLR 2019*.

[41] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aaron Oord. 2018. Adversarial Risk and the Dangers of Evaluating Against Weak Attacks. In *International Conference on Machine Learning*. 5032–5041.

[42] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature* (2019), 1–5.

[43] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. 2019. POET: Open-Ended Coevolution of Environments and their Optimized Solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 142–151.

[44] David H Wolpert and Kagan Tumer. 2002. Optimal Payoff Functions for Members of Collectives. In *Modeling complexity in economic and social systems*. World Scientific, 355–369.

[45] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. 1996. *Robust and Optimal Control*. Vol. 40. Prentice hall New Jersey.