Resilient Multi-Agent Reinforcement Learning with Adversarial Value Decomposition

Thomy Phan,[1] **Lenz Belzner,**[2] **Thomas Gabor,**[1]
Andreas Sedlmeier,[1] **Fabian Ritz,**[1] **Claudia Linnhoff-Popien**[1]
[1]LMU Munich,
[2]MaibornWolff
thomy.phan@ifi.lmu.de

Abstract

We focus on resilience in cooperative multi-agent systems, where agents can change their behavior due to udpates or failures of hardware and software components. Current state-of-the-art approaches to cooperative multi-agent reinforcement learning (MARL) have either focused on idealized settings without any changes or on very specialized scenarios, where the number of changing agents is fixed, e.g., in extreme cases with only one productive agent. Therefore, we propose *Resilient Adversarial value Decomposition with Antagonist-Ratios (RADAR)*. RADAR offers a value decomposition scheme to train competing teams of varying size for improved resilience against arbitrary agent changes. We evaluate RADAR in two cooperative multi-agent domains and show that RADAR achieves better worst case performance w.r.t. arbitrary agent changes than state-of-the-art MARL.

Introduction

Distributed systems consist of multiple separated components that collaborate to accomplish a common task (Tanenbaum and Van Steen 2007). Distributed autonomous systems can be formulated as cooperative *multi-agent system (MAS)*, which can be realized with methods of *reinforcement learning (RL)* (Foerster et al. 2018; Rashid et al. 2018).

Multi-agent RL (MARL) potentially offers better scalability and resilience against changing agents compared to single-agent RL. We define an *agent change* either as update or failure. E.g., some agents may be updated with new software or temporarily be replaced by other versions due to maintainance. In both cases, we would expect the remaining MAS to collaborate with these novel agents. On the other hand, agents might behave erroneously due to hardware or software failures. In this case, we would expect the remaining MAS to degrade gracefully without failing entirely (Stone and Veloso 2000). Intuitively, resilience should improve with more agents due to more available resources for compensation (Tanenbaum and Van Steen 2007).

Although resilience has been long recognized as a main motivation for realizing cooperative MAS (Stone and Veloso 2000; Panait and Luke 2005; Buşoniu, Babuška, and De Schutter 2010), most state-of-the-art approaches to cooperative MARL have focused on optimizing idealized scenarios, where an agent only faces the same or similar agents as during training (Foerster et al. 2018; Gupta, Egorov, and Kochenderfer 2017; Rashid et al. 2018). This bears the risk of *overfitting*, where a MAS can entirely fail when some agents significantly change their behavior, which could be fatal in safety critical environments, where such failures may have catastrophic consquences (Uesato et al. 2019).

Some work on resilient MARL based on adversarial learning (Littman 1994; Li et al. 2019; Phan et al. 2020) has focused on specialized settings with a fixed number of adversary agents, e.g., where a single productive agent remains. These approaches lack flexibility, which is required, when arbitrary portions of the MAS can change. Furthermore, they introduce new tunable hyperparameters like the fraction of adversaries or the degree of adversarial behavior, which further increase sensitivity w.r.t. unexpected scenarios.

In this paper, we propose *Resilient Adversarial value Decomposition with Antagonist-Ratios (RADAR)*. RADAR offers a value decomposition scheme to train competing teams of varying size for improved resilience against arbitrary agent changes. Our main contributions are:

- A simple mechanism to train adversarial agents with variable team sizes during training, which is necessary to create MAS that can potentially cope with arbitrary agent changes. Unlike prior work on resilient MARL, RADAR does not introduce any new hyperparameters, thus can be easily integrated into existing RL frameworks.
- An agent test scheme to consistently evaluate performance and resilience against changing agents in cooperative MAS in a fair way, which is inspired by prior work on single-agent RL (Badia et al. 2020; Jordan et al. 2020).
- An empirical evaluation of RADAR in two cooperative multi-agent domains and a comparison with state-of-the-art MARL w.r.t. the proposed test scheme. While being competitive against state-of-the-art MARL in cooperative settings, RADAR achieves better worst case performance when facing a variable number of previously unknown adversary agents at test time.

Background

Problem Formulation

MAS can be formulated as partially observable *Markov game* $M = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \Omega \rangle$, where $\mathcal{D} = \{1, ..., N\}$ is

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

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, $\mathcal{R}(s_t, a_t) = \langle r_{t,1}, ..., r_{t,N} \rangle \in \mathbb{R}^N$ is the joint reward with $r_{t,i}$ being 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 subsequent joint observation. Each agent i maintains an action-observation *history* $\tau_{t,i} \in (\mathcal{Z} \times \mathcal{A}_i)^t$. $\pi(a_t|\tau_t) = \langle \pi_1(a_{t,1}|\tau_{t,1}), ..., \pi_N(a_{t,N}|\tau_{t,N}) \rangle$ is the *joint policy*, where $\pi_i(a_{t,i}|\tau_{t,i})$ is the *local policy* of agent i. π_i can be evaluated with a *value function* $Q_i^\pi(s_t, a_t) = \mathbb{E}_\pi[G_{t,i}|s_t, a_t]$ for all $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$, where $G_{t,i} = \sum_{k=0}^{\infty} \gamma^k r_{t+k,i}$ is the *return* of agent i, and $\gamma \in [0, 1)$ is the discount factor. We denote the joint action and policies without agent i by $a_{t,-i}$ and π_{-i} respectively. The goal of each agent i is to find a *best response* π_i^* which maximizes Q_i^π given π_{-i}.

Policy Gradient Reinforcement Learning

Policy gradient RL is a popular approach to approximate best responses π_i^* for each agent i. A function approximator $\hat{\pi}_{i,\theta}$ with parameters θ is trained with gradient ascent on an estimate of $J = \mathbb{E}_\pi[G_{1,i}]$. Most policy gradient methods use gradients g of the following form (Sutton et al. 2000):

$$g = A_i^\pi(s_t, a_t) \nabla_\theta log \hat{\pi}_{i,\theta}(a_{t,i}|\tau_{t,i}) \tag{1}$$

where $A_i^\pi(s_t, a_t) = Q_i^\pi(s_t, a_t) - V_i^\pi(s_t)$ is the *advantage function* and $V_i^\pi(s_t) = \mathbb{E}_\pi[G_{t,i}|s_t]$ is the state value function of agent i. *Actor-critic* approaches often approximate $\hat{A}_i \approx A_i^\pi$ by replacing $Q_i^\pi(s_t, a_t)$ with $G_{t,i}$ and V_i^π with $\mathbb{E}_{\pi_i}[Q_i^\pi]$. Q_i^π can be approximated with a *critic* $\hat{Q}_{i,\omega}$ and parameters ω using value-based RL (Watkins and Dayan 1992; Mnih et al. 2015). For simplicity, we omit the parameter indices θ, ω and write $\hat{\pi}_i$, $\hat{Q}_i$ instead.

Independent Learning

$\hat{Q}_i$ can be learned independently using single-agent RL on $a_{t,i}$ and $\tau_{t,i}$ (Tan 1993; Leibo et al. 2017). These local approximations can be used for each agent's policy $\hat{\pi}_i$ to perform gradient ascent according to Eq. 1 leading to the *independent actor-critic (IAC)* approach (Foerster et al. 2018).

IAC offers optimal scalability w.r.t. N but violates the Markov assumption due to non-stationarity caused by simultaneously learning agents (Laurent et al. 2011).

Centralized Training Decentralized Execution

For many problems, training usually takes place in a laboratory or in a simulated environment, where global information is available. State-of-the-art MARL exploits this fact to approximate centralized value functions $\hat{Q}_i$, which condition on global states s_t and joint actions a_t, and use them as critic in Eq. 1 (Lowe et al. 2017). While $\hat{Q}_i$ is only required during training in order to learn local policies, $\hat{\pi}_i$ itself only conditions on the local history $\tau_{t,i}$, thus it can be executed in a decentralized way. This paradigm is known as *centralized training and decentralized execution (CTDE)*.

$\hat{Q}_i$ can be approximated separately for each agent i while integrating global information, in contrast to IAC, to learn best responses (Lowe et al. 2017). This approach lacks a *multi-agent credit assignment mechanism* for training agent teams, where all agents observe the same reward signal.

COMA approximates a single value function $\hat{Q}$ per team to compute agent-wise counterfactual baselines $V_i^\pi(s_t) = \sum_{a_{t,i} \in \mathcal{A}_i} \hat{\pi}_i(a_{t,i}|\tau_{t,i}) \hat{Q}(s_t, \langle a_{t,i}, a_{t,-i} \rangle)$ for individual credit assignment (Foerster et al. 2018).

The centralized $\hat{Q}$ can be factorized to approximate individual $\hat{Q}_i$ for each agent i in order to update $\hat{\pi}_i$ according to Eq. 1 in a coordinated way. *Value decomposition network (VDN)* is the simplest factorization method, where $\hat{Q}$ is defined by $\sum_{i \in \mathcal{D}} \hat{Q}_i(\tau_{t,i}, a_{t,i})$ (Sunehag et al. 2018). Alternatively, there exist non-linear factorization methods like QMIX or QTRAN (Rashid et al. 2018; Son et al. 2019).

While CTDE mitigates the non-stationarity issue of independent learning due to exploiting the Markov property of states, most approaches based on deep learning require a *fixed* number of agents N due to the predefined input dimension of s_t and a_t required by $\hat{Q}(s_t, a_t)$ (Lowe et al. 2017; Foerster et al. 2018; Rashid et al. 2018; Son et al. 2019).

Adversarial Reinforcement Learning

In *zero-sum games*, there are $N = 2$ agents with opposing goals. The value functions of agent i and j (and analogously the rewards) are defined by $Q_i^\pi = -Q_j^\pi$. A *minimax equilibrium policy* of agent i is defined by $\pi_i^* = argmax_{\pi_i} min_{\pi_j} Q_i^*$, which corresponds to a best response to the worst case, represented by π_j^* (Littman 1994).

Adversarial RL approaches attempt to approximate π_i^* with alternating optimization or reformulation of the minimax objective by applying standard RL techniques to each agent (Littman 1994; Pinto et al. 2017; Li et al. 2019).

Related Work

Adversarial Reinforcement Learning

Adversarial learning is a popular paradigm to train two opponents alternately to improve each other's performance and robustness (Goodfellow et al. 2014; Pinto et al. 2017). *Self-play RL* is the simplest form of adversarial RL, where a single agent is trained to play against itself to ensure an adequate difficulty level and steady convergence to robust policies (Samuel 1959; Tesauro 1995; Silver et al. 2016). In (single-agent) RL, the environment can be modeled as adversary by adding disturbances to confront the original agent with worst case scenarios (Morimoto and Doya 2001; Rajeswaran et al. 2017; Pinto et al. 2017). These adversarial disturbances can be realized, e.g., with RL or coevolutionary approaches (Gabor et al. 2019; Wang et al. 2019).

Our work is mainly based on adversarial learning. In contrast to single-agent RL, where external changes can only occur within the environment, we focus on *agent changes* in cooperative MAS. For that, we integrate adversary agents into the training process in order to improve resilience.

Multi-Agent Reinforcement Learning

MARL is a long-standing AI research area with various approaches (Tan 1993; Panait and Luke 2005; Foerster

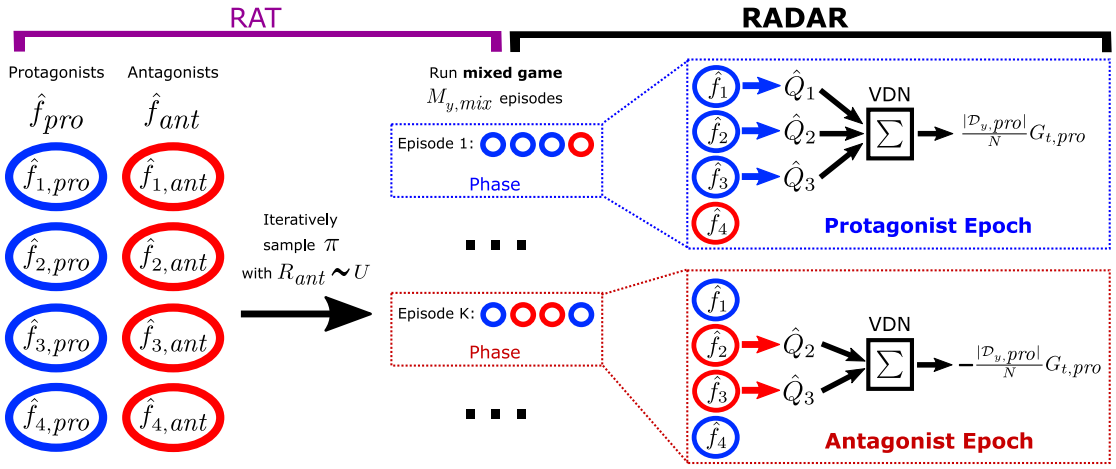


Figure 1: RAT and RADAR scheme for $N = 4$. Left: each agent has a protagonist (blue) and an antagonist (red) representation. Middle: mixed games $M_{y,mix}$ are generated by randomly sampling R_{ant} for each phase. Right: $\hat{f}_i$ are updated in epochs using VDN either for protagonists or antagonists. Note that the number of protagonists and antagonists can vary depending on R_{ant}.

et al. 2018; Son et al. 2019). While cooperative MARL has achieved impressive results in challenging domains, most approaches have been only evaluated with the same or similar agents as encountered during training. Thus, it remains unclear if these approaches offer resilience against arbitrary agent changes, which are expectable in the real world.

There is some prior work towards resilient MARL: *Minimax-Q* was proposed in (Littman 1994) as an adaptation of Q-Learning for zero-sum games. While guaranteeing convergence to safe policies w.r.t. worst case opponents, Minimax-Q becomes intractable if the (joint) action space of the opponent j is large. (Li et al. 2019) proposes M3DDPG, which considers *extreme cases*, where each agent i considers itself the sole productive agent, while all other agents are modeled as adversaries who attempt to minimize Q_i^π. M3DDPG can lead to poor policies, if the problem is too difficult or even unsolvable for single productive agents, leading to insufficient training signal. (Phan et al. 2020) proposes ARTS, where productive and adversary agents are trained simultaneously according to a fixed adversary ratio, since most CDTE approaches need a predefined input dimension to approximate $\hat{Q} \approx Q^\pi$. ARTS can improve resilience against agent failures with adequately chosen adversary ratios. However, an ideal ratio needs to be known a priori, which is an unrealistic assumption. Furthermore, a fixed ratio can lead to sensitive policies when N is sufficiently large.

We propose an *adversarial value decomposition* scheme, where the number of productive and adversary agents can change *dynamically* during training. Furthermore, we propose an *agent test scheme* to evaluate performance and resilience of MARL approaches in a fair way.

Methods

Terminology

We focus on *mixed (cooperative-competitive) games* M_{mix}, where $\mathcal{D} = \mathcal{D}_{pro} \cup \mathcal{D}_{ant}$ with $\mathcal{D}_{pro} \cap \mathcal{D}_{ant} = \emptyset$ (Lowe et al. 2017; Phan et al. 2020). $\mathcal{D}_{pro}$ is a team of productive or *protagonist* agents, which need to accomplish a certain (cooperative) task. $\mathcal{D}_{ant}$ is a team of *antagonist* agents representing (adversarial) agent changes in the MAS. For all protagonists $i \in \mathcal{D}_{pro}$ and for all antagonists $j \in \mathcal{D}_{ant}$ the corresponding rewards are defined by $r_{t,i} = r_{t,pro} = -r_{t,j}$. The *protagonist return* $G_{t,pro}$ is computed analogously to the individual return $G_{t,i}$ using $r_{t,pro}$ as rewards and γ as discount factor.

Furthermore, we define an *antagonist-ratio* $R_{ant} = \frac{|\mathcal{D}_{ant}|}{|\mathcal{D}|}$. If $R_{ant} = 0$, then M_{mix} is fully cooperative with $\mathcal{D} = \mathcal{D}_{pro}$ and a single global reward $r_{t,i} = r_{t,pro}$ for all agents $i \in \mathcal{D}$.

We use $\hat{f}_i = \langle \hat{\pi}_i, \hat{Q}_i \rangle$ as general notation for the learnable function representation of agent i wherever possible.

Randomized Adversarial Training

Most approaches towards resilient MARL focus on particular failure scenarios with a fixed R_{ant} (Littman 1994; Li et al. 2019; Phan et al. 2020), which has several drawbacks: First, R_{ant} must be known a priori or extensively tuned, which is generally not feasible. Second, a fixed R_{ant} during training can lead to inflexible behavior when facing a variable number of changing agents, which can be expected in real-world scenarios. Third, R_{ant} can have a huge impact on the training quality itself, e.g., if R_{ant} is too large, the MAS problem becomes too difficult to learn any meaningful policy.

Thus, we regard a *randomized adversarial training (RAT)* scheme. Since arbitrary agent changes can occur in a MAS, we provide a protagonist and antagonist representation for *each* agent. We maintain a pool $\hat{f}_{pro} = \langle \hat{f}_{1,pro}, ..., \hat{f}_{N,pro} \rangle$ of protagonist and a pool $\hat{f}_{ant} = \langle \hat{f}_{1,ant}, ..., \hat{f}_{N,ant} \rangle$ of antagonist representations, which are trained in T *phases* similarly to (Pinto et al. 2017). At each phase x, we randomly sample $R_{ant} \in [0, 1)$ from a *uniform distribution* U to run N_e episodes of different mixed games $M_{y,mix}$, where $\lfloor (1 - R_{ant})N \rfloor$ protagonist policies $\hat{\pi}_{i,pro}$ representing $\mathcal{D}_{y,pro}$ and $\lceil R_{ant}N \rceil$ antagonist policies $\hat{\pi}_{j,ant}$ represent-

Algorithm 1 Randomized Adversarial Training (RAT)

1: **procedure** *RAT*($\mathcal{D}, N, \hat{f}_{pro}, \hat{f}_{ant}, \Psi$)
2: Initialize parameters of $\hat{f}_{pro}$ and $\hat{f}_{ant}$
3: **for** phase $x = 1, T$ **do**
4: Sample $R_{ant} \sim U$ ▷ uniform sampling
5: **for** episode $y = 1, N_e$ **do**
6: $\mathcal{D}_{y,ant} \leftarrow$ sample $\lceil R_{ant} N \rceil$ agents from $\mathcal{D}$
7: $\mathcal{D}_{y,pro} \leftarrow \{i \in \mathcal{D} | i \notin \mathcal{D}_{y,ant}\}$
8: **for** $i = 1, N$ **do** ▷ Create $M_{y,mix}$
9: **if** $i \in \mathcal{D}_{y,ant}$ **then**
10: $\pi_i \leftarrow \hat{\pi}_{i,ant}$ ▷ from $\hat{f}_{i,ant}$
11: **if** $i \in \mathcal{D}_{y,pro}$ **then**
12: $\pi_i \leftarrow \hat{\pi}_{i,pro}$ ▷ from $\hat{f}_{i,pro}$
13: $\pi \leftarrow \langle \pi_1, ..., \pi_N \rangle$
14: Run one $M_{y,mix}$ episode with joint policy π
15: Store $e_{y,t} = \{\langle s_t, z_t, a_t, r_{t,pro} \rangle\}$ and $\mathcal{D}_{y,pro}$
16: **if** $x \bmod 2 = 1$ **then**
17: Update $\hat{f}_{i,pro}$ with Ψ $\forall i \in \mathcal{D}_{y,pro}$ w.r.t. $e_{y,t}$
18: **else**
19: Update $\hat{f}_{i,ant}$ with Ψ $\forall i \in \mathcal{D}_{y,ant}$ w.r.t. $e_{y,t}$

ing $\mathcal{D}_{y,ant}$ with $i \neq j$ are randomly selected. Each episode y can be considered a zero-sum game between $\mathcal{D}_{y,pro}$ and $\mathcal{D}_{y,ant}$. After each phase, either $\hat{f}_{pro}$ or $\hat{f}_{ant}$ is updated in alternating epochs w.r.t. the generated experience $e_{y,t} = \{\langle s_t, z_t, a_t, r_{t,pro} \rangle\}$ while the other pool is kept fixed.

The complete formulation of RAT is given in Algorithm 1, where $\mathcal{D}$ is the set of agents of the original MAS (given $R_{ant} = 0$), $N = |\mathcal{D}|$ is the number of agents, $\hat{f}_{pro}$ and $\hat{f}_{ant}$ are the learnable protagonist and antagonist representations respectively, and Ψ is an optimization or MARL algorithm. Due to its simplicity, we do not consider RAT a major contribution but a necessary preliminary and baseline for RADAR, which is introduced in the next section.

Resilient Adversarial Value Decomposition

Ψ in Algorithm 1 can be easily set to IAC or other independent learning algorithms, since RAT requires Ψ to be flexible w.r.t. the number of protagonists and antagonists, which can vary between each phase depending on R_{ant}. Using independent learning for RAT comes with the non-stationarity issue and the lacking credit assignment w.r.t. to (protagonist and antagonist) agent teams. Most CTDE approaches require a fixed team size due to the predefined input dimension of $\hat{Q} \approx Q^{\pi}$ depending on s_t and the joint action a_t (Lowe et al. 2017; Foerster et al. 2018; Rashid et al. 2018).

Therefore, we propose RADAR, a CTDE scheme to approximate protagonist and antagonist policies with variable R_{ant} based on VDN (Sunehag et al. 2018). VDN approximates $Q^{\pi}(s_t, a_t)$ with $\hat{Q}(\tau_t, a_t) = \sum_{i \in \mathcal{D}} \hat{Q}_i(\tau_{t,i}, a_{t,i})$ for cooperative MAS, where $\tau_t = \langle \tau_{t,1}, ..., \tau_{t,N} \rangle$ is the joint history. Although we focus on mixed games $M_{y,mix}$, Q^{π} can be obviously only approximated with cooperating agents[1]. Thus, we approximate $\hat{Q}_{pro}$ and $\hat{Q}_{ant}$ for protagonists and antagonists respectively using *separate* VDN instances.

Given RAT in Algorithm 1, we can approximate $\hat{Q}_{pro}$ in *protagonist epochs* (line 17) with the following term:

$$\sum_{i \in \mathcal{D}_{y,pro}} \hat{Q}_{i,pro}(\tau_{t,i}, a_{t,i}) = \mathbb{E}_{y,\pi}[\frac{|\mathcal{D}_{y,pro}|}{N} G_{t,pro} | s_t, a_t] \quad (2)$$

where $G_{t,pro}$ is the protagonist return and $\frac{|\mathcal{D}_{y,pro}|}{N}$ is used to normalize $G_{t,pro}$ w.r.t. the number of participating protagonists in episode y, because the scale of $G_{t,pro}$ could give more weight to settings where R_{ant} is small.

Analogously, we can approximate $\hat{Q}_{ant}$ in *antagonist epochs* (line 19 of Algorithm 1) with the following term:

$$\sum_{i \in \mathcal{D}_{y,ant}} \hat{Q}_{i,ant}(\tau_{t,i}, a_{t,i}) = \mathbb{E}_{y,\pi}[-\frac{|\mathcal{D}_{y,pro}|}{N} G_{t,pro} | s_t, a_t] \quad (3)$$

which approximates $\hat{Q}_{ant} = -\hat{Q}_{pro}$.

The terms of Eq. 2 and 3 can be approximated via end-to-end training of $\hat{Q}_{i,pro}$ and $\hat{Q}_{i,ant}$ using backpropagation (Sunehag et al. 2018). $\hat{Q}_{i,pro}$ and $\hat{Q}_{i,ant}$ can be used to derive the corresponding local policies, either via multi-armed bandits applied to the values or via policy gradient methods according to Eq. 1. The main components of RADAR and their integration into RAT are visualized in Fig. 1.

Although non-linear factorization methods have been proposed in (Rashid et al. 2018; Son et al. 2019), VDN offers some advantages in our context: Regarding Eq. 2 and 3, VDN just approximates a sum which is not bounded by a specific number of agents, thus being able to deal with variable team sizes of $\mathcal{D}_{y,pro}$ and $\mathcal{D}_{y,ant}$. Normalizing returns w.r.t. R_{ant} is straightforward in VDN due to the linear decomposition unlike in the non-linear case. Furthermore, VDN does neither introduce additional learnable parameters (e.g., additional neural networks) nor hyperparameters, which improves efficiency w.r.t. computation and tuning. We are still aware that adapting non-linear factorization methods for RAT could lead to even more powerful approaches towards resilient MARL, which we defer to future work.

Discussion of RAT and RADAR

RAT and RADAR offer simple mechanisms to train resilient MAS and can be easily combined with existing RL algorithms (value- or policy-based) to train $\hat{\pi}_i$. Neither RAT nor RADAR introduce new hyperparameters (RAT uses uniform sampling for R_{ant} and RADAR uses VDN without additional approximators or objectives), thus the tuning complexity completely depends on the underlying RL algorithm.

Given the uniform sampling of R_{ant} and $2N$ agent representations (N for the protagonists and N for the antagonists), the expected computational complexity of RADAR is $\mathcal{O}(N)$ because $\mathbb{E}_{R_{ant} \sim U}[R_{ant}] = 0.5$. The worst case complexity is $\mathcal{O}(2N)$ (when $R_{ant} = 0$ in each protagonist epoch and close to 1 in each antagonist epoch). Thus, RADAR scales similarly to other CTDE approaches in expectation with some overhead due to training additional antagonists.

[1] Naively integrating adversary value functions into $\hat{Q}$ could lead to trivial solutions, since $\hat{Q} = 0$ according to zero-sum games.

Testing Performance and Resilience in MAS

Most tests are conducted after training, where antagonists are trained on the final version of $\hat{f}_{pro}$ to determine flaws (Littman 1994; Li et al. 2019; Gleave et al. 2019). We propose an *online test* method, where we enable consistent tests during training. For that, we use *predefined* test cases that integrate novel agents which ideally have not been encountered during training before[2]. Such novel agents could be new cooperative agents (e.g., when the MAS is tested for compatibility with unknown agents) or antagonists which represent failures (e.g., due to flaws or malicious attacks). A MAS should behave resiliently in both cases.

We regard a *test suite* $\mathcal{T}$ consisting of test cases $c = \langle \hat{f}', R'_{ant} \rangle \in \mathcal{T}$. For a given $\hat{f}_{pro}$, we run a single test case by generating random mixed games similar to RAT (Algorithm 1) with antagonist-ratio R'_{ant} and $\hat{f}_{ant} = \hat{f}'$ to evaluate the average performance with g_c (e.g., the protagonist return $G_{t,pro}$ or some domain-specific value). $\mathcal{T}$ can contain the following disjoint subsets as schematically shown in Fig. 2:

- $\langle \hat{f}_{pro}, 0 \rangle \in \mathcal{T}_{ideal}$ only involves protagonists $\hat{f}_{i,pro}$ encountered during training. The majority of work on cooperative MAS only reports such idealized test cases to evaluate sample efficiency and raw performance (Lowe et al. 2017; Foerster et al. 2018; Gupta, Egorov, and Kochenderfer 2017; Rashid et al. 2018; Son et al. 2019).
- $\langle \hat{f}'_{pro}, R'_{ant} \rangle \in \mathcal{T}_{cooperation}$ integrates new protagonists $\hat{f}'_{i,pro} \neq \hat{f}_{i,pro}$ that were obtained from a *different* training process. $\mathcal{T}_{cooperation}$ evaluates the ability of $\hat{f}_{pro}$ to collaborate with unknown agents. In this paper, we set $R'_{ant} = \frac{1}{2}$.
- $\langle \hat{f}'_{ant}, \chi \rangle \in \mathcal{T}_{failure,\chi}$ integrates new antagonists $\hat{f}'_{i,ant} \neq \hat{f}_{i,ant}$ that were obtained from a *different* training process. $\mathcal{T}_{failure,\chi}$ evaluates the resilience of $\hat{f}_{pro}$ against unknown failure scenarios according to different $R'_{ant} = \chi$.

$\mathcal{T}_{cooperation}$ and $\mathcal{T}_{failure,\chi}$ enable us to compare different MARL approaches with each other in a fair way because they face the *same* test agents in $\mathcal{T}$. Furthermore, using a *single* test suite $\mathcal{T}$ is practically more efficient than separately performing an adversarial test on *each* MARL result as previously proposed in (Littman 1994; Li et al. 2019).

Naively aggregating the performance values g_c of each $c \in \mathcal{T}$ could lead to the domination of test cases with small R'_{ant}, since more protagonists contribute to the success of the MAS. Thus, a normalized value $\overline{g}_c \in \mathbb{R}$ is required to scale g_c according to the number of protagonists $|\mathcal{D}_{pro}| = \lfloor (1 - R'_{ant})N \rfloor$ to ensure a meaningful evaluation w.r.t. different R'_{ant} (Jordan et al. 2020). In this paper, we focus on the following measures[3] as depicted in Fig. 2:

- **Cooperation performance** estimates $\mathbb{E}_{c \in \mathcal{T}_{cooperation}}[\overline{g}_c]$ w.r.t. to the number of protagonists (the integrated test agents do not contribute to $\overline{g}_c$) to evaluate the compatibility with $\frac{N}{2}$ new cooperatively trained test agents.
- **Worst case performance** estimates the (degraded) protagonist performance $min_{c \in \mathcal{T}_{cooperation} \cup (\bigcup_\chi \mathcal{T}_{failure,\chi})} \{\overline{g}_c\}$ in the worst case w.r.t. arbitrary agent changes.

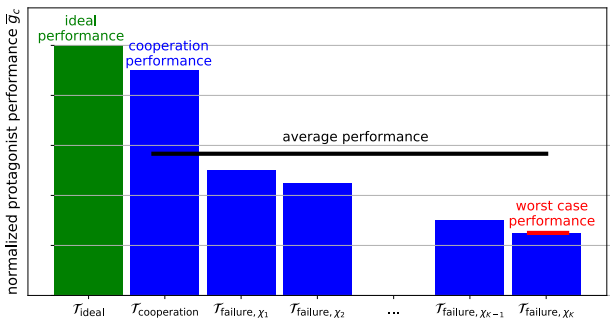


Figure 2: The introduced test sets: $\mathcal{T}_{ideal}$ involves the *same* protagonists encountered during training. $\mathcal{T}_{cooperation}$ and $\mathcal{T}_{failure,\chi}$ integrate protagonists or antagonists of *different* training processes than the MAS to be tested. While prior work mainly focused on ideal performance (green), we regard cooperation (blue) and worst case (red) performance.

The average performance of all $c \in \mathcal{T}_{cooperation} \cup (\bigcup_\chi \mathcal{T}_{failure,\chi})$ as shown in Fig. 2 could put more emphasis on test cases, where $\hat{f}_{pro}$ performs especially well, reducing the significance of our evaluation (Jordan et al. 2020). Thus, we focus on the performance, which we can *at least* expect from $\hat{f}_{pro}$ given arbitrary agent changes (Badia et al. 2020).

Experiments[4]

Evaluation Domains

We implemented a *predator-prey (PP)* and a *cyber-physical production system (CPPS)* domain with N agents. An episode is reset after 50 time steps for each domain. We define a normalized performance value $\overline{g}_c$ for each domain as quality measure for $\hat{f}_{pro}$.

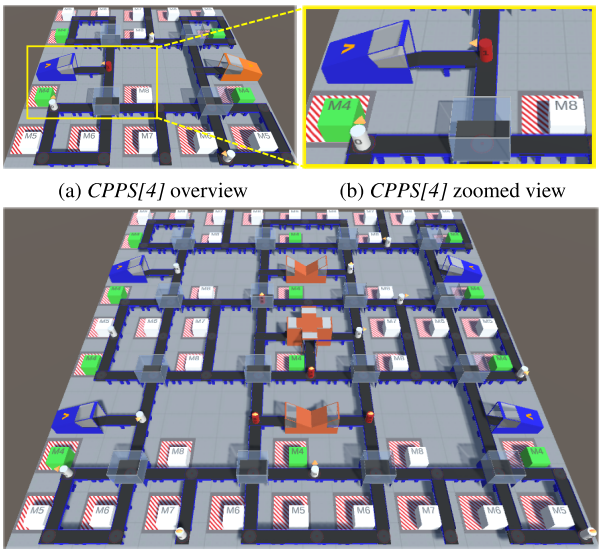
PP[K,N] consists of a $K \times K$ grid with N learning predator agents and $\frac{N}{2}$ randomly moving prey agents. Each agent starts at a random position, is able to move north, south, west, east, or do nothing, and has a 5×5 field of view. A prey is captured with a global reward of +1, when at least one predator i occupies the same position as *main capturer* with another predator $j \neq i$ being within sight of i, which is recorded as $\kappa = \langle i, j \rangle$. Captured preys respawn at random positions. We define $\overline{g}_c = \frac{1}{|\mathcal{D}_{pro}|} \sum_{\kappa = \langle i,j \rangle} \mathbb{1}[i \in \mathcal{D}_{pro}]$ as the normalized number of *protagonist main captures*.

CPPS[N] consists of a machine grid as shown in Fig. 3. Each agent has a list of four random tasks $tasks_i$ organized in two *buckets*. All agents start at a *blue entry* 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

[2] These agents can be trained with any (adversarial) MARL algorithm (e.g., RADAR, M3DDPG, ARTS) independently of $\hat{f}_{pro}$.

[3] Although none of these measures is actually new (Crandall and Goodrich 2005; Powers, Shoham, and Vu 2007), they are widely neglected in current deep MARL in favor of the idealized case.

[4] Code available at https://github.com/thomyphan/resilient-marl



(a) *CPPS[4]* overview (b) *CPPS[4]* zoomed view

(c) *CPPS[16]* overview

Figure 3: Two CPPS instances with $R_{ant} = \frac{1}{4}$. The white and red cylinders represent protagonists and antagonists respectively. (a, b) *CPPS[4]* as 4-agent setting with 1 antagonist. (c) *CPPS[16]* as 16-agent setting with 4 antagonists.

matches with the machine type, the task is removed from the agent's task list with a reward of +1. An agent i is *complete*, if $tasks_i = \emptyset$ and it reaches an *orange exit*, yielding another reward of +1. For each incomplete agent, a reward of -0.01 is given at every time step. Each agent has a 5×5 field of view without knowing the tasks of other agents. 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 have to coordinate to avoid conflicts or collisions to ensure fast completion. We define $\overline{g}_c = \frac{|\{i \in \mathcal{D}_{pro} | tasks_i = \emptyset\}|}{|\mathcal{D}_{pro}|}$ as the *protagonist completion rate*.

Learning Algorithms and Test Cases

We implemented an actor-critic algorithm with $A_i^\pi(s_t, a_t) = G_{t,i} - V_i^\pi(s_t)$ and M3DDPG. The critic $V_i^\pi = \mathbb{E}_{\pi_i}[Q_i^\pi]$ is approximated via IAC, COMA, QMIX, or RADAR.

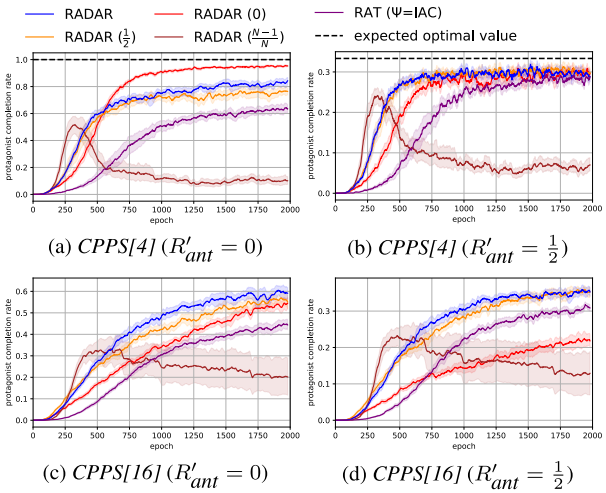
To study the effect of variable antagonist-ratios and adversarial value decomposition, we implemented ablations with fixed $R_{ant} = \chi \in \{0, \frac{1}{2}, \frac{N-1}{N}\}$ denoted by RADAR (χ) and a RAT instantiation with $\Psi = \text{IAC}$. $R_{ant} = \frac{N-1}{N}$ represents the extreme case with a single protagonist agent.

Prior to training, we generated $\mathcal{T}_{cooperation}$ with RADAR (0) and $\mathcal{T}_{failure,\chi}$ with RADAR ($\chi \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$). For each test set, we trained 10 test cases which are used to consistently evaluate resilience of all implemented MARL approaches.

Neural Network Architectures

We used deep neural networks to implement $\hat{f}_i = \langle \hat{\pi}_i, \hat{Q}_i \rangle$ for each agent i. The neural networks are updated every 1000 time steps using ADAM with a learning rate of 0.001. We set $\gamma = 0.95$, $T = 4000$, and $N_e = 10$ (Algorithm 1).

Since PP and CPPS are gridworlds, states and observations are encoded as multi-channel image as proposed in (Gupta, Egorov, and Kochenderfer 2017; Phan et al. 2018). We implemented all neural networks as multilayer perceptron (MLP) and flattened the multi-channel images before feeding them into the networks. $\hat{\pi}_i$ and $\hat{Q}_i$ have two hidden layers of 64 units with ELU activation. The output of $\hat{\pi}_i$ has $|\mathcal{A}_i|$ units with softmax activation (gumbel softmax for M3DDPG (Lowe et al. 2017)). The output of $\hat{Q}_i$ has $|\mathcal{A}_i|$ linear units. The centralized $\hat{Q}$-networks of COMA, QMIX, and M3DDPG are MLPs having two hidden layers of 128 units with ELU activation and one linear output unit ($|\mathcal{A}_i|$ output units for COMA (Foerster et al. 2018)).



(a) *CPPS[4]* ($R'_{ant} = 0$) (b) *CPPS[4]* ($R'_{ant} = \frac{1}{2}$)

(c) *CPPS[16]* ($R'_{ant} = 0$) (d) *CPPS[16]* ($R'_{ant} = \frac{1}{2}$)

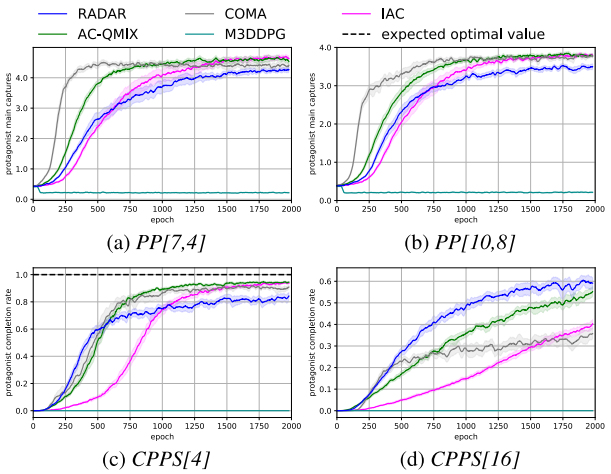
Figure 4: Protagonist completion rates for RADAR and its ablations on *CPPS[4]* and *CPPS[16]* for test cases with $R'_{ant} \in \{0, \frac{1}{2}\}$. Shaded areas show the 95 % confidence interval. The legend at the top applies across all plots.

Results

For each MARL approach, we performed 20 training runs of 40,000 episodes (2 million time steps in total). After each epoch of $N_e = 10$ episodes, a full test was performed on $\hat{f}_{pro}$ by running each $c \in \mathcal{T}$ for 50 times. For *CPPS[4]*, we provide the *expected optimal value* based on antagonists that prevent other agents from entering the CPPS by blocking the entry path as shown in Fig. 3a and 3b.

The test performance for $R'_{ant} \in \{0, \frac{1}{2}\}$ of RADAR and its ablations is shown in Fig. 4. In *CPPS[4]*, all RADAR variants except RADAR ($\frac{N-1}{N}$) outperform RAT. In *CPPS[16]*, RAT outperforms RADAR (0), when $R'_{ant} > 0$. RADAR is competitive or superior to RADAR ($\frac{1}{2}$) and outperforms RADAR (0), when $R'_{ant} > 0$ or when $N = 16$. RADAR ($\frac{N-1}{N}$) improves fastest in all settings but its performance gradually decreases after reaching a peak.

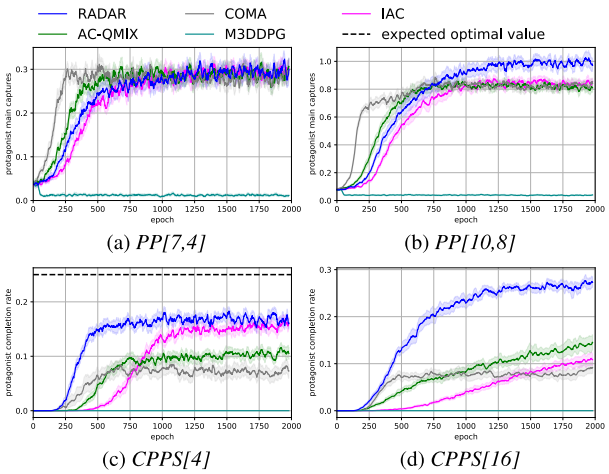
The cooperation performance of RADAR, IAC, COMA, AC-QMIX, and M3DDPG is shown in Fig. 5. COMA, AC-



(a) *PP[7,4]* (b) *PP[10,8]*

(c) *CPPS[4]* (d) *CPPS[16]*

Figure 5: Cooperation performance for RADAR and state-of-the-art MARL. Shaded areas show the 95 % confidence interval. The legend at the top applies across all plots.



(a) *PP[7,4]* (b) *PP[10,8]*

(c) *CPPS[4]* (d) *CPPS[16]*

Figure 6: Worst case performance for RADAR and state-of-the-art MARL. Shaded areas show the 95 % confidence interval. The legend at the top applies across all plots.

QMIX, and IAC achieve the best cooperation performance except in *CPPS[16]*, where RADAR performs best. However, RADAR is only slightly outperformed by the cooperative MARL approaches in *PP[7,4]*, *PP[10,8]*, and *CPPS[4]*. M3DDPG performs worst in all settings.

The worst case performance of RADAR, IAC, COMA, AC-QMIX, and M3DDPG is shown in Fig. 6. RADAR clearly achieves the best worst case performance in all settings except in *PP[7,4]*, where COMA, AC-QMIX, and IAC are competitive. M3DDPG performs worst in all settings.

Discussion

We presented RADAR, an adversarial value decomposition scheme for resilient MAS. RADAR trains competing teams of protagonist and antagonist agents of varying size to improve resilience against arbitrary agent changes.

According to our ablation study, the value decomposition scheme offers a significant advantage over independent learning: RADAR and most fixed antagonist-ratio variants clearly outperform RAT (Ψ = IAC), because RAT (Ψ = IAC) lacks a credit assignment mechanism, which is important to learn coordinated protagonist and antagonist policies.

Fig. 4 indicates that training with fixed antagonist-ratios strongly depends on the concrete setting and must be tuned, e.g., in *CPPS[4]*, RADAR (0) outperforms RADAR ($\frac{1}{2}$) on average but in *CPPS[16]*, RADAR ($\frac{1}{2}$) is clearly superior (although the nature of tasks is the same in both CPPS instances). RADAR does not require such tuning and performs at least second best in all CPPS instances.

RADAR is able to achieve competitive cooperation performance compared to state-of-the-art MARL like COMA, AC-QMIX, and IAC. We assume that the additional training of randomly integrated antagonists causes some overhead, which sacrifices a little performance regarding the special case of cooperative test agents. However, RADAR is able to achieve superior worst case performance w.r.t. arbitrary agent changes including failure scenarios. Fig. 6 indicates that RADAR is more resilient than cooperative state-of-the-art MARL w.r.t. the size of the MAS, since RADAR achieves significantly better worst case performance in *PP[10,8]* and *CPPS[16]* compared to *PP[7,4]* and *CPPS[4]* respectively. In contrast to RADAR, cooperative state-of-the-art MARL approaches achieve especially poor worst case performance in *CPPS[16]* compared to *CPPS[4]*, which contradicts our intuition that a MAS should be actually *more resilient* when more agents are available (Tanenbaum and Van Steen 2007).

Although RADAR ($\frac{N-1}{N}$) and M3DDPG focus on extreme cases, they perform poorly in all settings, indicating that if a domain is too difficult (or not solvable at all) for a single protagonist, such specializations are not sufficient for learning resilient behavior. Despite of RADAR ($\frac{N-1}{N}$) improving fastest in the beginning (Fig. 4), the antagonists eventually learn to stall the single protagonist in the CPPS, thus leading to the performance decrease. M3DDPG models adversarial behavior within its $\hat{Q}$ objective, leading to very sparse training signal right at the beginning of training.

Significant advantages of RADAR are its algorithmic simplicity and its flexibility w.r.t. to team sizes. Since it uses uniform sampling and linear value decomposition, it does not introduce any new hyperparameters to be tuned (the integrated antagonists have exactly the same hyperparameters as the protagonists), thus being less sensitive than state-of-the-art MARL, when facing a variety of adversarial settings. Like other CTDE approaches, RADAR scales linearly in expectation and worst case, thus offering a feasible and easy extension to existing RL approaches w.r.t. resilience.

For future work, we want to extend RADAR to non-linear value decomposition like QMIX and use adaptive sampling mechanisms for R_{ant} to further improve performance and resilience. We also aim to provide adequate agent test sets for other established domains similarly to our experiments, which we regard as an important step towards consistent and fair evaluation of future cooperative MARL approaches.

Acknowledgments

We would like to thank Cornel Klein, Horst Sauer, Reiner Schmid, and Jan Wieghardt from Siemens AG for helpful discussions on this project.

References

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, Z. D.; and Blundell, C. 2020. Agent57: Outperforming the Atari Human Benchmark. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 507–517. PMLR.

Buşoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-Agent Reinforcement Learning: An Overview. In *Innovations in Multi-Agent Systems and Applications-1*, 183–221. Springer.

Crandall, J. W.; and Goodrich, M. A. 2005. Learning to Compete, Compromise, and Cooperate in Repeated General-Sum Games. In *Proceedings of the 22nd International Conference on Machine Learning*, 161–168.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence* 32(1).

Gabor, T.; Sedlmeier, A.; Kiermeier, M.; Phan, T.; Henrich, M.; Pichlmair, M.; Kempter, B.; Klein, C.; Sauer, H.; Schmid, R.; and Wieghardt, J. 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, 898–906. Association for Computing Machinery.

Gleave, A.; Dennis, M.; Kant, N.; Wild, C.; Levine, S.; and Russell, S. 2019. Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 27, 2672–2680. Curran Associates, Inc.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative Multi-Agent Control using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems*, 66–83. Springer.

Jordan, S.; Chandak, Y.; Cohen, D.; Zhang, M.; and Thomas, P. 2020. Evaluating the Performance of Reinforcement Learning Algorithms. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 4962–4973. PMLR.

Laurent, G. J.; Matignon, L.; Fort-Piat, L.; et al. 2011. The world of independent learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems* 15(1): 55–64.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-Agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, AAMAS '17, 464–473. International Foundation for Autonomous Agents and Multiagent Systems.

Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; and Russell, S. 2019. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4213–4220.

Littman, M. L. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Machine Learning Proceedings 1994*, 157–163. Elsevier.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, 6379–6390.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518(7540): 529–533.

Morimoto, J.; and Doya, K. 2001. Robust Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 1061–1067.

Panait, L.; and Luke, S. 2005. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multiagent Systems* 11(3): 387–434.

Phan, T.; Belzner, L.; Gabor, T.; and Schmid, K. 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*, AAMAS '18, 730–738. International Foundation for Autonomous Agents and Multiagent Systems.

Phan, T.; Gabor, T.; Sedlmeier, A.; Ritz, F.; Kempter, B.; Klein, C.; Sauer, H.; Schmid, R.; Wieghardt, J.; Zeller, M.; et al. 2020. Learning and Testing Resilience in Cooperative Multi-Agent Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '20, 1055–1063. International Foundation for Autonomous Agents and Multiagent Systems.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust Adversarial Reinforcement Learning. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 2817–2826. PMLR.

Powers, R.; Shoham, Y.; and Vu, T. 2007. A General Criterion and an Algorithmic Framework for Learning in Multi-Agent Systems. *Machine Learning* 67(1-2): 45–76.

Rajeswaran, A.; Ghotra, S.; Ravindran, B.; and Levine, S. 2017. EPOpt: Learning Robust Neural Network Policies using Model Ensembles. In *International Conference on Learning Representations*.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4295–4304. PMLR.

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

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529(7587): 484–489.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 5887–5896. PMLR.

Stone, P.; and Veloso, M. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots* 8(3): 345–383.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; 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)*, AAMAS '18, 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Solla, S.; Leen, T.; and Müller, K., eds., *Advances in Neural Information Processing Systems*, volume 12, 1057–1063. MIT Press.

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

Tanenbaum, A. S.; and Van Steen, M. 2007. *Distributed Systems: Principles and Paradigms*. Prentice-Hall.

Tesauro, G. 1995. Temporal Difference Learning and TD-Gammon. *Communications of the ACM* 38(3): 58–69.

Uesato, J.; Kumar, A.; Szepesvari, C.; Erez, T.; Ruderman, A.; Anderson, K.; Heess, N.; Kohli, P.; et al. 2019. Rigorous Agent Evaluation: An Adversarial Approach to Uncover Catastrophic Failures. *International Conference on Learning Representations* .

Wang, R.; Lehman, J.; Clune, J.; and Stanley, K. O. 2019. POET: Open-Ended Coevolution of Environments and their Optimized Solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, 142–151. Association for Computing Machinery.

Watkins, C. J.; and Dayan, P. 1992. Q-Learning. *Machine Learning* 8(3-4): 279–292.