Counterexample-Guided Policy Refinement in Multi-Agent Reinforcement Learning

Briti Gangopadhyay
IIT Kharagpur
India
briti_gangopadhyay@iitkgp.ac.in

Pallab Dasgupta
IIT Kharagpur
India
pallab@cse.iitkgp.ac.in

Soumyajit Dey
IIT Kharagpur
India
soumya@cse.iitkgp.ac.in

ABSTRACT

Multi-Agent Reinforcement Learning (MARL) policies are being incorporated into a wide range of safety-critical applications. It is important for these policies to be free of counterexamples and adhere to safety requirements. We present a methodology for the counterexample-guided refinement of an optimized MARL policy with respect to given safety specifications. The proposed algorithm refines a calibrated MARL policy to become safer by eliminating counterexamples found during testing, using targeted gradient updates. We empirically validate our method on different cooperative multi-agent tasks and demonstrate that targeted gradient updates induce safety in MARL policies.

KEYWORDS

Counterexample-Guided Refinement; Multi-Agent Reinforcement Learning; Multi-Agent Proximal Policy Optimization

ACM Reference Format:
Briti Gangopadhyay, Pallab Dasgupta, and Soumyajit Dey. 2023. Counterexample-Guided Policy Refinement in Multi-Agent Reinforcement Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023,* IFAAMAS, 9 pages.

1 INTRODUCTION

Single-agent Deep Reinforcement Learning (DRL) is a popular control technique where the policy controlling agent learns to choose actions that maximize a discounted long-term reward. DRL has been successfully applied for games [25] and complex tasks in simulation environments [9], with performances often exceeding human counterparts. The success of DRL algorithms has motivated their use in multi-agent settings, giving rise to techniques collectively known as Multi-Agent Reinforcement Learning (MARL). MARL involves the study of optimally controlling multiple agents through adaptive interactions with unknown entities in an environment. These methods have been modeled on the mathematical framework of stochastic games expressed as i) fully competitive [19], ii) fully cooperative [18] or iii) mixed setting [20]. MARL is being studied for multiple real-world applications like traffic management [17], autonomous driving [24], and robotic control [15]. Safety-critical MARL applications, like autonomous driving, require agents to behave robustly and reasonably, as visiting an unsafe state during deployment is undesirable. MARL inherits the drawbacks of DRL, such as interpretability, susceptibility to adversarial inputs, and lack of a formal safety guarantee. In general, MARL suffers from a lack of convergence guarantee except for certain special cases. Difficulty in establishing safety in a MARL environment is also attributed to the fact that individual agents need to consider other agents' behaviors so that the joint transition of the system is safe. Safety for MARL environments has been studied through the addition of constraints with the optimization objective [13], or the use of shields [5]. Given an optimized policy, it is safe to use the policy in the training environment. However, if the policy is deployed in an uncertain environment with changing parameters, the existing policy may lead to safety violations. Our objective in this paper is to *refine* the existing policy in a minimal way such that the new counterexamples found during testing are excluded. It is often preferable to modify a tested policy than to learn a new one.

In this work, we propose a *counterexample guided refinement* technique for a MARL policy that has already been fine-tuned to optimize a reward function. The proposed methodology aims to make the optimized policy progressively safer by incorporating targeted gradient-based action shaping specific to a set of counterexample traces obtained from intelligent testing. This accommodates corrections for counterexamples without compromising the quality of the learned policy. Counterexample-guided refinement has been studied for single agent RL [7, 8] but has not been explored for multi-agent settings. In summary, the paper makes the following novel contributions.

(1) We propose the first work on counterexample-guided refinement in a MARL setting. The counterexamples are obtained with respect to multiple safety specifications in a cooperative environment.
(2) We propose two types of refinements. One that finds an alternate path to the goal state while satisfying safety criteria. The other one keeps the system in a safe state when goal and safety cannot be achieved together.
(3) We show that under the proper choice of importance factor for the penalty for violation of safety objective, the updated policy monotonically increases towards safety.
(4) The proposed methodology is studied over environments with continuous state and action spaces on different multi-agent tasks.

Contributions 2 and 3 enhance the single-agent counterexample-guided refinement methodology in general. The paper is organized as follows: Section 2 outlines related works, Section 3 discusses the background concepts, Section 4 presents the overall methodology of policy refinement, Section 5 presents case studies on several MARL environments, and Section 6 provides concluding remarks.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

2 RELATED WORKS

Classical DRL techniques rely purely on maximizing a reward signal for deriving an optimal policy. However, only reward maximization is not enough for real-world applications where reaching unsafe states is not desirable [8]. The domain of safe RL [11] aims to tackle this problem such that the policies respect the notion of some predefined safety both during training and deployment. Designing safe policies for a multi-agent system is a challenging yet pertinent problem. However, limited studies exist for Safe MARL [32]. Safety for RL has been studied on the mathematical model of Constrained Markov Decision Processes (CMDPS) where constraint costs, in addition to rewards, are produced by each state, which is factored into the optimization problem [1]. Constrained policy optimization has been extended for MARL with algorithms like Multi-Agent Constrained Policy Optimization (MACPO) and MACPO-Lagrangian [13]. Optimization with constraints has a positive effect on safety but often generates poor rewards and is susceptible to design errors [21]. In our work, we do not introduce constraint costs, rather, we perform targeted gradient changes based on the margin of safety specification violation on failure traces. The use of external knowledge from safety shields to replace unsafe actions is another popular technique that has been explored for single-agent systems [2] and recently for multi-agent RL [5]. The authors in [5] propose an algorithm that coordinates multiple agents to join factored shields constructed from sub-parts of the state space and given LTL specifications. Safety shields require engineering expertise and continuous monitoring to guarantee safety. They also introduce performance overhead caused by the delay during shield switching.

Counterexample guided learning has been studied for networks [26], which are restricted to only monotonic constraints. Policy refinement through counterexamples has been applied to Recurrent Neural Policies [3] and Apprentice Learning (AL) [33, 34] using model-checking approaches like Probabilistic Computational Tree Logic (PCTL). These model-checking techniques are only applicable to discrete state spaces and suffer from the curse of dimensionality. Our work closely follows the counterexample-guided refinement technique proposed by [8] for single agent RL where authors perform gradient updates based on failure traces to obtain a safer policy. However, [8] does not account for trajectories that cannot reach the goal without violating safety and the quantification of the importance factor. In our work, we extend the refinement methodology to work with MARL environments while also accounting for paths that cannot reach the goal state safely and quantify the penalty importance.

3 PRELIMINARIES

In this Section, we give an overview of the MARL setting and discuss Bayesian Optimization (BO), which is used to uncover counterexamples from a trained policy.

3.1 Multi Agent Reinforcement Learning

A multi-agent reinforcement learning problem can be formulated in terms of an Agent Environment Cycle (AEC) [29] game which is a sequential version of the Partially Observable Stochastic Game (POSG) model. We use a slightly modified version of AEC formally defined by the tuple:

$\langle \tilde{S}, M, \{A^i\}, \{T^i\}, \{R^i\}, \gamma \rangle$ where $\tilde{S}$ is the set of global states. S^i is the set of local states. $\tilde{S} = \langle S^1, S^2, \ldots, S^M \rangle$, M is the number of agents, A^i represents the set of actions for each agent i, $T^i : S^i \times A^i \rightarrow S^i$ is the agent transition function, R^i is the reward signal corresponding to agent i and $\gamma \in [0, 1)$ is the discount factor. We consider a fully cooperative multi-agent setting. In each iteration, t of the cycle, each agent $i \in M$ observes a local state s_t^i and sequentially chooses one action $a_t^i \in A^i$ generating reward r_t^i. The combination of the local states and actions yield global state $\tilde{s}_t = \langle s_t^1, s_t^2, \ldots, s_t^m \rangle$ and joint action $\tilde{a_t} = \langle a_t^1, a_t^2, \ldots, a_t^m \rangle$. The states may have common observations, i.e., $S^i \cap S^j \neq \emptyset$. When $\tilde{a_t}$ is applied to $\tilde{s_t}$ the system transitions to a new state $\tilde{s}_{t+1} \sim T(\tilde{s_t}, \tilde{a_t})$.

We choose Multi-Agent Proximal Policy Optimization (MAPPO) as the training algorithm. MAPPO is effective in several multi-agent settings with minimum hyper-parameter tuning and architectural modifications [31]. MAPPO also enjoys monotonic improvement guarantees under a proper choice of clipping bounds [27]. Since our environments have homogeneous agents, we use parameter sharing. Policy parameters θ and value parameters φ are shared across all the agents. The policy network $\pi_\theta^i(a^i|s^i)$ for agent i produces action $a_t^i \sim \pi_\theta^i(\cdot|s_t^i)$ based only on its local state $s^i \in S^i$ and receives a reward $r_t^i \in R^i$. The objective of the MAPPO algorithm is to adjust θ via gradient ascent such that the discounted accumulated reward for each agent is maximized $J(\theta_i) = \mathbb{E}_{a_t^i, s_t^i}[\sum_{t=0}^{T} \gamma^t r_t^i(s_t^i, a_t^i)]$. Similar to single agent PPO, MAPPO achieves this by updating the policy parameters in the direction which maximizes a clipped surrogate objective. The clipped independent probability ratio for the policy of each agent i is denoted by

$$\lambda_{\pi'^i} = \frac{\pi'^i(a^i|s^i)}{\pi^i(a^i|s^i)}. \tag{1}$$

Here π'^i is the current policy, and π^i is the old policy that was used to collect samples in a previous iteration. The purpose of $\lambda_{\pi'^i}$ is to trace the impact of the change in actions under π'^i and π^i. The objective function is given by:

$$\begin{aligned} \max_{\pi'^i} \mathbb{E}_{(s^i,a^i)\sim d_{\pi^i}}[min(\lambda_{\pi'^i} * \mathbb{A}^i(s^i, a^i), \\ clip(\lambda_{\pi'^i}, 1+\epsilon, 1-\epsilon) * \mathbb{A}^i(s^i, a^i))] \end{aligned} \tag{2}$$

where $\mathbb{E}$ is the expectation, $\mathbb{A}^i(s^i, a^i) = \sum_{t=0}^{\infty}[r(s_t^i, a_t^i)] - V(s^i)$ is the advantage function, $V(s^i)$ is the value function and ϵ is the clip hyperparameter.

3.2 Uncovering Counterexamples Using Bayesian Optimisation

Bayesian Optimization (BO) is a global optimization technique that is used for optimizing black box functions, which are expensive to evaluate [23]. BO has been extensively used in literature for testing cyber-physical systems [4, 10, 12] and analog circuits due to low sample complexity [14]. BO has been shown to find more counterexamples than techniques like random and grid-search for single agent environments [8]. For finding adversarial counterexamples through BO, the negation of given safety specifications is converted into an objective function $\phi(s)$ over state variables s [10]. A safety specification is expressed as a combination of multiple

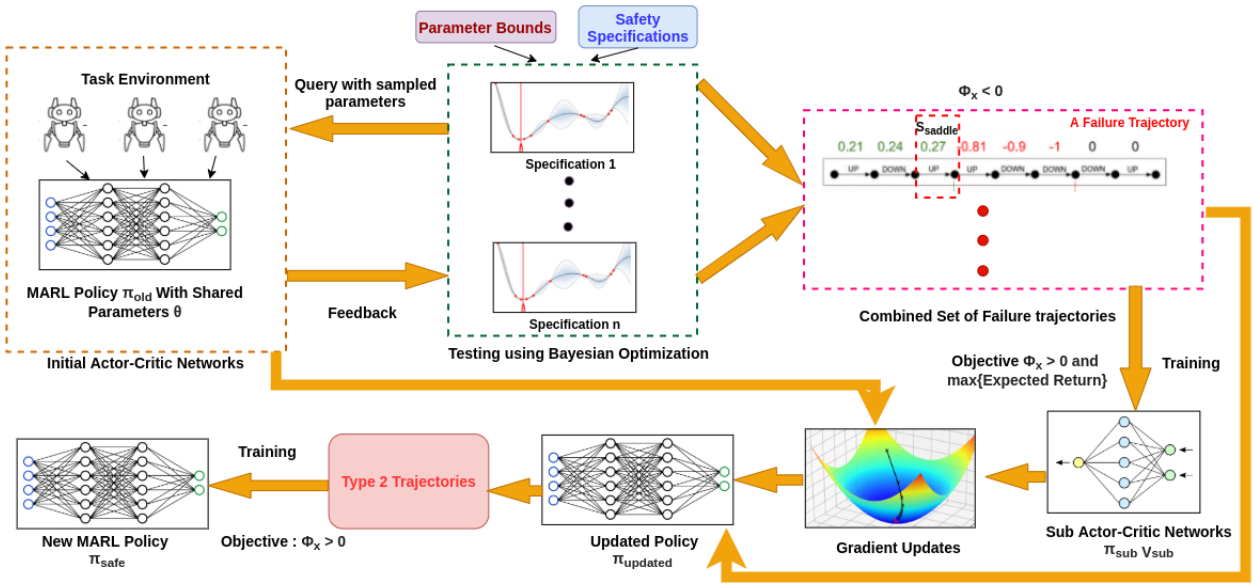

Figure 1: An overview of the counterexample-guided refinement framework for MARL

predicates of the form $\varrho(s) \geq 0$, where $\varrho : S \rightarrow \mathbb{R}$ represents a real-valued function of the state. For each predicate, $\varrho(s) \geq 0$, we use Bayesian optimization to search for $\phi_i = min(\varrho_i(s))$. Since the objective function, ϕ_i, corresponding to the safety predicate, $(\varrho_i(s) \geq 0)$ is real-valued, we need composition operators over reals to arrive at a single objective function when multiple safety predicates are given. The equivalent quantitative semantics for the logical operators is given by: $\neg\phi = -\phi$; $\phi_1 \wedge \phi_2 = min(\phi_1, \phi_2)$ and $\phi_1 \vee \phi_2 = max(\phi_1, \phi_2)$. The system is unsafe if $\phi < 0$ and safe otherwise. Each ϕ_i is estimated using a separate Gaussian Process (GP) model as this uncovers more failures than using a single GP model [12]. The BO algorithm samples parameters p_i from given parameter bounds $[P_{low}, P_{high}]$ over the state variables. When a trajectory ξ initialized with p_i leads to a negative evaluation of ϕ, it is added to the set of failure trajectories ξ_f. To uncover multiple counterexamples, we remove already explored failure regions from the search space using the technique proposed by [10]. The BO loop runs until the entire search space is covered or the maximum sampling budget for BO search has been exhausted.

4 METHODOLOGY

The overview of the refinement methodology is shown in Figure 1. The refinement strategy is broadly divided into the following steps:

(1) Given a MARL policy π_{old} trained by optimizing rewards, we test it against different safety objectives by sampling from parameters of uncertainty P over state variables. We use BO to minimize the objective functions such that the safety specifications are falsified. This generates a set of failure trajectories ξ_f.
(2) Next, we train sub-policy π_{sub} and critic V_{sub} using a combination of reward and penalty only on ξ_f. Using ξ_f and π_{sub} we selectively do gradient updates on π_{old} to construct a new policy $\pi_{updated}$.
(3) If $\pi_{updated}$ has uncorrected trajectories ξ'_f in ξ_f, we again train $\pi_{updated}$ on ξ'_f treating only safety (ϕ) as the optimization objective beyond a saddle state (the penultimate state before specification violation). This generates a new policy π_{safe} such that safety specifications are respected by all the agents in π_{safe}.

Definition 4.1. *Local Safety Specifications: These are defined over the state parameters of a single agent in the system. The objective function has the form $\phi_{local}(s^i_t) < 0, s^i_t \in S^i$. An example of a local objective function is the speed of agent one at time step 10 is less than 5: speed($agent^1_{10}$) - 5 < 0.*

Definition 4.2. *Mutual Safety Specification: These are defined over the state parameters of two agents in the system. The objective function used for finding counterexamples has the form $\phi_{mutual}(s^i_t, s^j_t) < 0, s^i \in S^i, s^j \in S^j \wedge i \neq j$. For example, the distance between two agents < 0.*

Definition 4.3. *Global Safety Specification: These are defined over the state parameters of all the agents in the system, which cooperate to achieve a common goal. The objective function has the form $\phi_{global}(s^0_t, \ldots, s^i_t, \ldots, s^m_t) < 0$ or $\phi_{global}(\tilde{s}_t) < 0, \tilde{s}_t \in \tilde{S}$. For example, the angle of the package carried by multiple agents < 0.*

A trajectory is treated as a failure trajectory if either local, mutual, or global safety specification is violated.

Definition 4.4. *A failure trajectory ξ_{fl} in the AEC setting is defined as $\xi_{fl} = \{(\tilde{s_0}, \tilde{a_0}, \tilde{s_1}, \tilde{a_1}, \ldots, \tilde{s_n})\}$ where $\tilde{s_t} = \langle s^0_t, \ldots, s^m_t \rangle$ and $\tilde{a_t} = \langle a^0_t, \ldots, a^m_t \rangle$ s.t. $(\exists s^i_{t_l}, \phi_{local}(s^i_{t_l}) < 0)$, or $(\exists (s^i_{t_l}, s^j_{t_l}), \phi_{mutual}(s^i_{t_l}, s^j_{t_l}) <$*

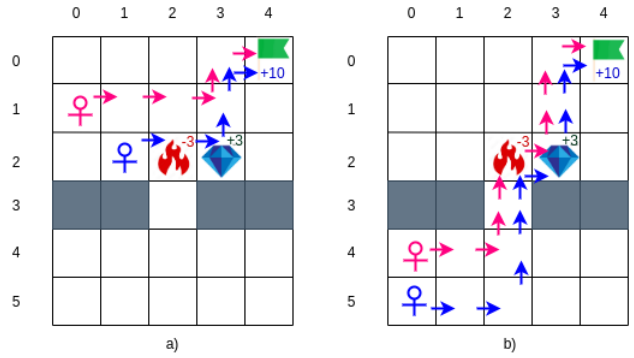

Figure 2: a) Example of a Type-1 trajectory b) Example of a Type-2 trajectory

0), *or* $(\exists \tilde{s}_{t_l}, \phi_{global}(\tilde{s}_{t_l}) < 0)$. ξ_f *denotes the set of failure trajectories and* $s^i_{t_l}$ *denotes the state for* i^{th} *agent at time-step* t *in* l^{th} *failure trajectory* ξ_{fl}.

A failure trajectory may violate one or more safety specifications. For example, in an environment where multiple agents are carrying a package, if one of the agents carrying the package falls, then the package may fall as a result. In this work, we shall consider a trajectory only as part of one specification during refinement.
We start with a motivating example to elucidate our methodology:

Example 4.1. *Consider the toy grid world in Fig 2. The grey tiles are obstacles, the tile with flame is a bad state with a -3 penalty, and the tile with the flag is the goal state with a +10 reward. The environment consists of two agents* $(A1, A2)$*, which are allowed three actions, right for moving right, up for moving up, and no action for staying in the current state. Each movement incurs a -1 penalty, and no action incurs a -2 penalty. The game terminates when both agents reach the goal or when the maximum time budget of 10 steps is exhausted. The agents can start and visit any location in the grid world except the obstacle states. Now let us consider two safety specifications:*
1) No agent should reach a bad state i.e., $(reward(state) + 2 \geq 0)$
2) Agents must not collide i.e., $(distance(A_1, A_2) > 0)$
For a policy trained to only maximize rewards, the following counterexample configurations exist as highlighted in Fig 2:

Trajectory 1

$A1$ $(1,0) \xrightarrow{right} (1,1) \xrightarrow{right} (1,2) \xrightarrow{right} (1,3) \xrightarrow{up} (0,3) \xrightarrow{right} (0,4)$

$A2$ $(2,1) \xrightarrow{right} (2,2) \xrightarrow{right} (2,3) \xrightarrow{up} (1,3) \xrightarrow{up} (0,3) \xrightarrow{right} (0,4)$

Trajectory 2

$A1$ $(4,0) \xrightarrow{right} (4,1) \xrightarrow{right} (4,2) \xrightarrow{up} (3,2) \xrightarrow{up} (2,2) \xrightarrow{right} (2,3) \xrightarrow{up} (1,3) \xrightarrow{up} (0,3) \xrightarrow{right} (0,4)$

$A2$ $(5,0) \xrightarrow{right} (5,1) \xrightarrow{right} (5,2) \xrightarrow{up} (4,2) \xrightarrow{up} (3,2) \xrightarrow{up} (2,2) \xrightarrow{left} (2,3) \xrightarrow{up} (1,3) \xrightarrow{up} (0,3) \xrightarrow{left} (0,4)$

A counterexample trajectory may have three different types of states in its path. They are described as follows:

(1) *States with an alternate path:* Trajectories with some state $s^i_{k_l} \in \xi_{fl}$ where corrective actions a_c can be applied to lead the agents to goal without violating the safety specification. These are Type-1 trajectories.
(2) *States with a path to a safe region:* Trajectories with some state $s^i_{k_l} \in \xi_{fl}$ where agents can reach a safe state but do not have any action that can take them to the goal state. For example, the starting state (4,0) has no trajectory that leads to the goal state. However, the agent can reach states (3,2) or (4,3) which are safe. These are Type-2 trajectories.
(3) *Irrecoverable States:* Trajectories with starting state $s^i_{k_l} \in \xi_{fl}$ which already lie in an unsafe region. For example, if any of the agents start in the state (2,2), then the safety specification is already violated. States like these are flagged as irrecoverable.

We first describe the refinement strategy for trajectories of Type-1 and then subsequently address the refinement of trajectories of Type-2.

4.1 Refinement of Type-1 Trajectories

All counterexample trajectories ξ_f are initially assumed to be Type-1 trajectories. Trajectory 1 in Example 4.1 is a Type-1 trajectory. The desired objective is to modify the policy of $A2$ such that the state (2,2) violating the safety specification $reward(state) + 2 \geq 0$ can be avoided. In a single agent setting, this is achieved by first training a sub-policy π_{sub} with reward + penalty for specification violation. Then the gradients of the original policy π_{old} are updated using the following update rule :

$$\lambda_t(\theta) = \frac{\pi_{old}(a_t|s_t)}{\pi_{sub}(a_t|s_t)} \tag{3}$$

considering an advantage of 1 for the ratio update to enforce the corrections [8]. However, adapting this methodology directly to a multi-agent setting has the following pitfalls.

- Adding only a penalty for one specification during training may lead to the violation of another specification for the same agent. In Trajectory 1 of Example 4.1 we can correct the policy of A2 by changing the action from $(2,1) \xrightarrow{right} (2,2)$ to $(2,1) \xrightarrow{up} (1,1)$. This satisfies the predicate $reward(state) + 2 \geq 0$. However, this change causes a collision with $A1$ and violates the predicate $distance(A_1, A_2) > 0$
- Since the policy uses shared parameters, the environment becomes non-stationary from the perspective of a single agent. An update for a single agent causes an exogenous shift changing the policy for all other agents. Hence, enforcing the actions with the advantage of 1 may cause disruptive changes in the policy for other agents.

To handle the first problem, we account for the penalty for violation of any safety objective. The updated reward function while training the sub-policy becomes $R_t + \beta_t * (\phi_x(s') - t)$ where β_t is the importance of the penalty term at time step t and $\phi_x(s') = \phi_{local}(s') + \phi_{mutual}(s') + \phi_{global}(s')$. To have minimum interference with the reward, a penalty is only added when a safety objective is violated i.e,$\phi_{local \vee mutual \vee global}(s') < 0$ making $\phi_x(s') < 0$. The inclusion of penalties for all violated safety specifications pushes the

Algorithm 1: Policy Refinement for Type-1 Trajectories

Input: Policy π_{old}, Critic V_{old}, Failure Trajectories ξ_f
Function `Refine_Policy_1`($\pi_{old}, V_{old}, \xi_f$):
  π_{sub}, V_{sub} = *random_Initialize*()
  for *each* ξ_{fl} *in* ξ_f **do**
    $traj = []$
    $s^i_{t_l} = s^0_{t_l}$
    while *not*(*env.done*) **do**
      $a^i_{t_l} = \pi_{sub}(s^i_{t_l})$
      $s' = env.apply(a^i_{t_l})$
      if $\phi_{local}(s') \vee \phi_{mutual}(s') \vee \phi_{global}(s') < 0$ **then**
        $r^i_{t_l} = r^i_{t_l} + \beta_t * (\phi_x(s') - t)$
      end
      $traj = traj \cup (s^i_{t_l}, a^i_{t_l}, r^i_{t_l}),\ s^i_{t_l} = s'$
    end
    Train π_{sub}, V_{sub} using $(traj)$
  end
  $\pi_{update}, V_{updated} = Update(\pi_{old}, V_{old}, \pi_{sub}, V_{sub}, \xi_f)$

Algorithm 2: Updating the Old Policy

Input: Old Policy π_{old}, Old Critic V_{old}, Policy π_{sub}, Critic V_{sub}, ξ_f
Function `Update`($\pi_{old}, V_{old}, \pi_{sub}, V_{sub}, \xi_f$):
  for *each* ξ_{fl} *in* ξ_f **do**
    Collect new actions and log probabilities corresponding to ξ_{fl} from π_{sub}
    Calculate Advantage Estimate $\mathbb{A}_l$ using V_{sub}
    if $\mathbb{A}_l$ *is positive* **then**
      Update (π_{old}, V_{old}) to $(\pi_{updated}, V_{updated})$ by maximizing clipped objective, minimizing MSE using (π_{sub}, V_{sub})
    end
  end

policy towards choosing the safest path for all the agents. Following from example, agent $A2$ will now prefer $(2,1) \xrightarrow{no-action} (2,1)$ and then $(2,1) \xrightarrow{up} (1,1)$ as directly taking the up action would garner a negative penalty for collision.

To alleviate the second problem, we consider the advantage of the chosen action for all the agents during the update. The advantage is calculated using the sub-critic network V_{sub}, trained along with the sub-actor network to accommodate both reward and safety. The update rule given in Eq:3 for a particular state belonging to a failure trajectory $s^i_{t_l} \in \xi_{fl}$ is modified as below:

$$\lambda^i_{t_l}(\theta) = \frac{\pi^i_{old}(a^i_{t_l}|s^i_{t_l})}{\pi^i_{sub}(a^i_{t_l}|s^i_{t_l})} \times \mathbb{A}^i_{t_l} \tag{4}$$

Where $\mathbb{A}^i_t = \Sigma_{t=0}^{T} r^i_t(s^i_t, a^i_t) - V_{sub}(s^i_t)$. The update is only allowed when the advantage of taking all actions a_l in policy π_{sub} for failure trajectory ξ_{fl} is positive with respect to the sub critic network V_{sub}. The reason for choosing a sub-critic network for calculating the advantage is further elaborated while discussing Type-2 refinement. The updated network post Type-1 corrections are referred to as $\pi_{updated}$ and contain the trajectories that can reach the goal without violating any safety specification. The refinement process of Type-1 trajectories is described in Algorithm 1, and the update function is given in Algorithm 2.

Algorithm 3: Policy Refinement for Type-2 Trajectories

Input: Policy $\pi_{updated}$, Critic $V_{updated}$, ξ'_f
Function `Refine_Policy_2`($\pi_{updated}, V_{updated}, \xi'_f$):
  $\pi_{safe}, V_{safe} = \pi_{updated}, V_{updated}$
  for *each* ξ_{fl} *in* ξ'_f **do**
    Calculate saddle state $s^i_{k_l}$
    $traj = []$
    $s^i_{t_l} = s^i_{k_l}$
    while *not*(*env.done*) **do**
      $a^i_{t_l} = \pi_{safe}(s^i_{t_l})$
      $s' = env.apply(a^i_{t_l})$
      if $\phi_{local}(s') \vee \phi_{mutual}(s') \vee \phi_{global}(s') < 0$ **then**
        $r^i_{t_l} = \beta_t * (\phi_x(s') - t)$
      end
      if $\phi_x(s) \approx 0$ **then**
        $r^i_{t_l} = \delta$
      end
      $traj = traj \cup (s^i_{t_l}, a^i_{t_l}, r^i_{t_l}),\ s^i_{t_l} = s'$
    end
    Train π_{safe}, V_{safe} using $(traj)$
  end

4.2 Refinement of Type-2 Trajectories

Type-2 trajectories have no safe path to the goal state but have a path to a safe region. All counterexample trajectories not corrected by Type-1 refinement are treated as Type-2 trajectories. Trajectory 2 in Example 4.1 is a Type-2 trajectory. It is important to note that an agent can have multiple safe regions. For example, agents $A1$ and $A2$ can also take *{no-action}* repeatedly and be safe. We shall choose the updated trajectory that is closest to the original trajectory. With minimum deviation from the original policy, Trajectory 2 will be updated as follows:

Updated Trajectory 2

A1 $(4,0) \xrightarrow{right} (4,1) \xrightarrow{right} (4,2) \xrightarrow{up} (3,2) \xrightarrow{no-act} (3,2) \ldots \xrightarrow{no-act} (3,2)$

A2 $(5,0) \xrightarrow{right} (5,1) \xrightarrow{right} (5,2) \xrightarrow{up} (4,2) \xrightarrow{no-act} (4,2) \ldots \xrightarrow{no-act} (4,2)$

On Type-2 failure trajectories, which were not corrected after the first update, we train on $\pi_{updated}$ to obtain new policy π_{safe}. π_{safe} and critic V_{safe} are trained using rewards, $R_t = r_t$ until the agent moves close to the violation of the specification but does not violate it i.e, $\phi_x(s') > 0$. If $\phi_x(s')$ is close to 0, the policy receives a positive reward of $R_t = \delta$. Close to zero denotes the last real positive value

obtained by $\phi_x(s')$ before it becomes 0. When valuation $\phi_x(s') < 0$ then the policy receives only penalties $R_t = \beta_t * (\phi_x(s') - t)$. The penalty increases with an increase in the margin of violation with each time step t. The reward of the trajectory is not included beyond $\phi_x(s') \approx 0$ as the primary objective of π_{safe} is to find safe regions rather than reach the goal. To minimize the deviation from the original trajectory π_{safe} is initialized with $\pi_{updated}$. The clipped ratio training using $\pi_{updated}$ ensures that the trajectories of π_{safe} have a marginal deviation from $\pi_{updated}$. The advantage is calculated using V_{safe} i.e, $\mathbb{A}_t^i = Q_{safe}(s_t^i, a_t^i) - V_{safe}(s_t^i)$. The reason for calculating the advantage with respect to the sub-critic is explained with the following example:

Example 4.2. *Let us consider the policy of only agent A1 for Trajectory 2 from Ex. 4.1 with $\gamma = 1$ from saddle state $s = (3, 2)$ and with only one specification, $reward(state) + 2 \geq 0$. From state $(3, 2)$ the reward of taking {up} action is given by: $Q_{reward}(s, up) = -3 + 3 - 1 - 1 - 1 + 10 = 7$. Now changing the action {up} to {no-action} will generate the following Q value in terms of the reward, $Q_{reward}(s, \text{no-action}) = -2-2-2-2-2-2 = -10$, which will generate a negative advantage when calculated in terms of the reward critic, V_{old}. However, when calculated in terms of the safety critic, V_{safe}, that is learned on the function, $\phi(s) = reward(state) + 2$, we obtain $Q_{safe}(s, \text{no-action}) = \delta + \delta + \delta + \delta + \delta + \delta = 6\delta$. This gives a positive advantage when calculated with V_{safe}. Similarly, V_{sub} will generate a positive advantage for actions that respect safety along with reward. Hence, the advantage is calculated in terms of the critic trained via incorporation of reward and penalty in V_{sub} for Type-1 trajectories and in terms of penalty only in V_{safe} for Type-2 trajectories.*

Further action shaping using ratio update is not required in the case of Type-2 trajectories as the updated policy $\pi_{updated}$ is changed directly during training. It is important to note that training a separate sub-policy is required in the case of Type-1 trajectories because we want the policy to explore alternate safer paths towards the goal, which may not be close to the original policy. The refinement process of Type-2 trajectories is described in Algorithm 3. We show that even though trajectories in π_{safe} have less reward, under a proper choice of β_t, the refinement improves the network monotonically towards safety. Monotonic improvement means $\eta(\pi') > \eta(\pi)$ at each update step where η [16] is the performance measure with respect to safety.

Proposition 4.5. *A modified trajectory ξ'_{f_l} in π_{safe} has equal or less reward corresponding to its failure trajectory ξ_{f_l} in π_{old}*
Assumption: π_{old} converges to an optimal reward for all trajectories.
Proof: For the saddle state $s^i_{k_l}$ the action $a^i_{k_l}$, proposed by π_{old}, takes the agent to an unsafe state in ξ_{f_l}. An alternate action $a'^i_{k_l}$ is proposed by π_{safe} for $s^i_{k_l}$ modifying the trajectory to ξ'_{f_l}. Since π_{old} is optimal $Q_{old}(s^i_{k_l}, a'^i_{k_l}) \leq Q_{old}(s^i_{k_l}, a^i_{k_l})$. In general, such repeated refinements lead π_{safe} to have trajectories with less reward than π_{old}.

Proposition 4.6. *π_{old} monotonically increases towards safety through the update steps if $\beta_h > \frac{r_h}{-(\phi_x(s')-h)}$, where H is the horizon of the trajectory following old policy π_{old} and $h \in 0 \ldots H$ and s' is a next state violating a safety specification i.e, $s' : \phi_{local}(s') \vee \phi_{mutual}(s') \vee \phi_{global}(s') < 0$*

Proof: Let us first consider a single-agent setting. The policy performance of another policy π' in terms of original policy π_{old} or π can be expressed using the identity given by [16].

$$\eta(\pi') - \eta(\pi) = \mathbb{E}_{\tau\sim\pi'}[\Sigma_{t=0}^{\infty}\gamma^t \mathbb{A}(s_t, a_t)] \tag{5}$$

where the expected discounted reward $\eta(\pi) = \mathbb{E}[\Sigma_{t=0}^{\infty}\gamma^t r(s_t)]$. It is guaranteed that any policy update that yields a non-negative expected advantage i.e, $\Sigma_a \pi'(a|s) \times \mathbb{A}_\pi \geq 0$, increases the policy performance η [22]. We shall start our analysis from a saddle state sampled from failure trajectory $s_k \sim \xi_f$. Eq: 5 changes as follows:

$$\begin{aligned}
&\eta(\pi') - \eta(\pi)\\
=&\mathbb{E}_{s_k\sim\xi_f}[\Sigma_{t=k}^{\infty}\gamma^{t-k}\mathbb{A}(s_t, a_t)]\\
=&\mathbb{E}_{s_k\sim\xi_f}[\Sigma_{t=k}^{\infty}\gamma^{t-k} r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)]\\
=&\mathbb{E}_{s_k\sim\xi_f}[\Sigma_{t=k}^{\infty}\gamma^{t-k} r(s_t, a_t) + \Sigma_{t=k+1}^{\infty}\gamma^{t-k+1} V^{\pi}(s_t) - \Sigma_{t=k}^{\infty}\gamma^{t-k} V^{\pi}(s_t)]\\
=&\mathbb{E}_{s_k\sim\xi_f}[\Sigma_{t=k}^{\infty}\gamma^{t-k} r(s_t, a_t) - V^{\pi}(s_k)]\\
=&\mathbb{E}_{s_k\sim\xi_f}[\Sigma_{t=k}^{\infty}\gamma^{t-k} r(s_t, a_t) - \{\Sigma_{h=k}^{H}\gamma^{h-k}(r(s_h) + \beta_h \times (\phi_x(s') - h))\}]
\end{aligned}$$

where H is the horizon of the trajectory following old policy π. Since we want $\eta(\pi') - \eta(\pi) > 0$, the expectation can be removed as it is an average over the length of the trajectory. Now, $[\Sigma_{t=k}^{\infty}\gamma^{t-k} r(s_t, a_t)]$ is a positive quantity as either it reaches goal or gets a reward of δ from Algo 1 or Algo 3. In order to have policy improvement we need to ensure $\gamma^{h-k}(-r(s_h) - (\beta_h \times (\phi_x(s') - h))) > 0$. From this it follows:

$$\beta_h > \frac{r_h}{-(\phi_x(s') - h)} \tag{6}$$

The result also holds for a multi-agent setting if independent ratios in Eq. 1 are bound under a centralized trust region ([27], theorem 4.3). The independent ratio bounded between $\lambda_i \in [1 - \frac{\alpha}{M}, 1 + \frac{\alpha}{M}]$, where M is the number of agents and α is the trust region constraint, is a sufficient condition to enforce the centralized trust region constraint ([27], Eq. 10).

5 EMPIRICAL STUDIES

To evaluate our strategy, we use the multi-walker environment from [28], custom Cooperative ACC environment, and Multi-agent Ant environment from Safe Multi-Agent Mujoco environments [13]. The experiments were run on a machine with AMD Ryzen 4600h six-core processor and GeForce GTX 1660 Graphics unit. For calculating the distance between the policies π_{old} and π_{safe}, we sample n=1000 random trajectories ξ from π_{old} and their corresponding trajectories ξ' from π_{safe} along with the failure trajectories and the updated trajectories. We then measure the difference of states visited between the trajectories through metric D_v proposed in [8].

$$D_v(\pi_{old} \,||\, \pi_{safe}) = \frac{1}{n}\sum_{\xi_i, \xi'_i \in \xi}\sqrt{\sum_{s_i, s'_i \in \xi_i, \xi'_i}\left|(s_i)_{\pi_o} - (s'_i)_{\pi_s}\right|^2} \tag{7}$$

The environments, parameter bounds on the uncertain variables, safety specifications, number of original counterexample traces, number of counterexamples traces after Type-1 and Type-2 refinement, and the mean variation distance, along with the standard deviation, between the original and refined policy are summarised in Table 1. The environments are briefly described below:

Multi-Walker Environment: The environment has a set of bipedal robots with a package placed on top of them. The goal is to

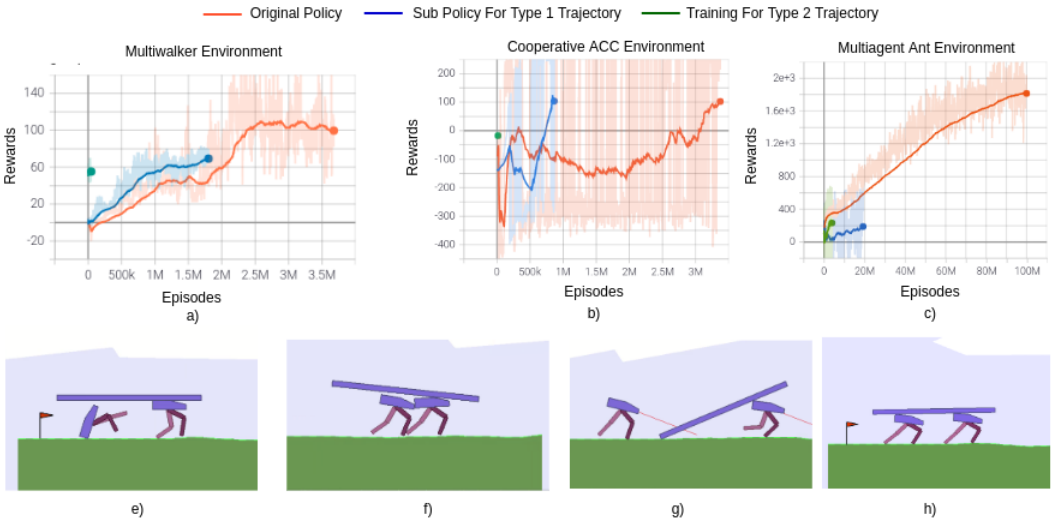

Figure 3: Reward and training time plots for original, sub-policy, and refinement of Type-2 trajectory for a) Multi-walker Environment b) Cooperative ACC Environment c) Multi-agent Ant Environment. A counterexample for e) Local Specification, f) Mutual Specification, h) Global Specification h) A refined trajectory from Type-2 refinement where the walkers prefer balancing the pole and standing.

Table 1: Parameter bounds on uncertain variables, safety specifications, counterexamples after BO testing, counterexamples after Type-1 (T1) and Type-2 (T2) refinement, and distance between the original and the refined policies.

Environment	Parameter Bounds	Safety Specification Type	Counterexamples	T1	T2	Distance (n=1000)
MultiWalker	agent.hull_angle : $(0, 2 * \pi)$	Local : walker[i].hull_angle >0.2	133	6	0	2.97 ± 1.33
	agent.velocity_x : (-1,1)	Mutual : distance(walker[i],walker[j]) ≥ 0	12	1	0	
	agent.velocity_y : (-1,1)	Global : package.angle ≥ 0.25	11	1	0	
Cooperative ACC	mid_car.pos_y : (0,5)	Local : car[i].path_error <0.5	25	0	–	7.09 ± 0.18
	car.velocity : (-1,1)	Mutual : distance(car[i],car[j]) >0	50	5	0	
	rear_car.pos_y : (0,3)	Global : $\forall i, j \in car\ dist(i, j) > 0 \implies speed_{platoon} > 0$	21	0	–	
Multiagent Ant	torso_pos : (-1,1)	Local : $\text{wall\_distance}_{left}$ <-1.8 ∨ $\text{wall\_distance}_{right}$ >1.8	115	10	0	5.093 ± 1.574
	torso_velocity : (-1,1)	Global : Contact_{right} >1 ∧ Contact_{left} >1	70	22	10	

carry the package as far right as possible through coordination. We consider two agents with 6 six sources of uncertainty as mentioned in Table 1. The original MAPPO policy is tested against the following safety specifications: 1) Local: The hull angle of each walker is greater than 0.2; 2) Mutual: The walkers must not collide; and 3) Global: The package is not touching the ground. The visualization for a counterexample for each specification is shown in Fig 3 (e,f,g). Fig 3h) shows a trace obtained as a result of secondary training where the walkers prefer to balance and stand rather than take an unsafe action to reach the goal.

Cooperative Adaptive Cruise Control (CACC) Environment: CACC is an extension of the Adaptive Cruise Control problem where the aim is to drive a platoon of vehicles in a harmonized manner [30] (details in Appendix A [6]). We consider a platoon of 3 vehicles controlled as an AEC game. We consider four sources of disturbances in the system. We want the following specifications to be maintained: 1) Local: Each car should not deviate from its path predefined through way-points, 2) Mutual: No vehicle collides and 3) Global: If all the cars are at a safe distance, then the platoon should be moving.

Multi-agent Ant Environment: In this environment from [13], the objective is to control a robotic ant through different legs, each treated as an agent while keeping a safe distance from nearby obstacles. For testing, the initial torso position and velocity are chosen to be uncertain. The safety specifications ensure that locally none of the legs hit the obstacle, and globally the torso does not get imbalanced. We observe that after the secondary training, ten counterexamples remain uncorrected. The trajectories for these counterexamples start at irrecoverable states.

We choose a reward-optimized policy to uncover maximum counterexamples for refinement. In Fig 3, we report the reward per training step plots for the original policies, sub-policies π_{sub}, and secondary training for π_{safe}. π_{sub} takes less time to train as it is only trained on counterexample traces and converges to a reward less than or equal to the original policy supporting proposition 4.5. In our experiments, we required 25 updates to the original policy for multi-walker, 10 updates for CACC, and 40 updates for multi-ant on the original policy for Type-1 refinement. For Type-2 trajectories, training π_{safe} takes further less time as the number of counterexamples reduces post-first refinement. Our experiments

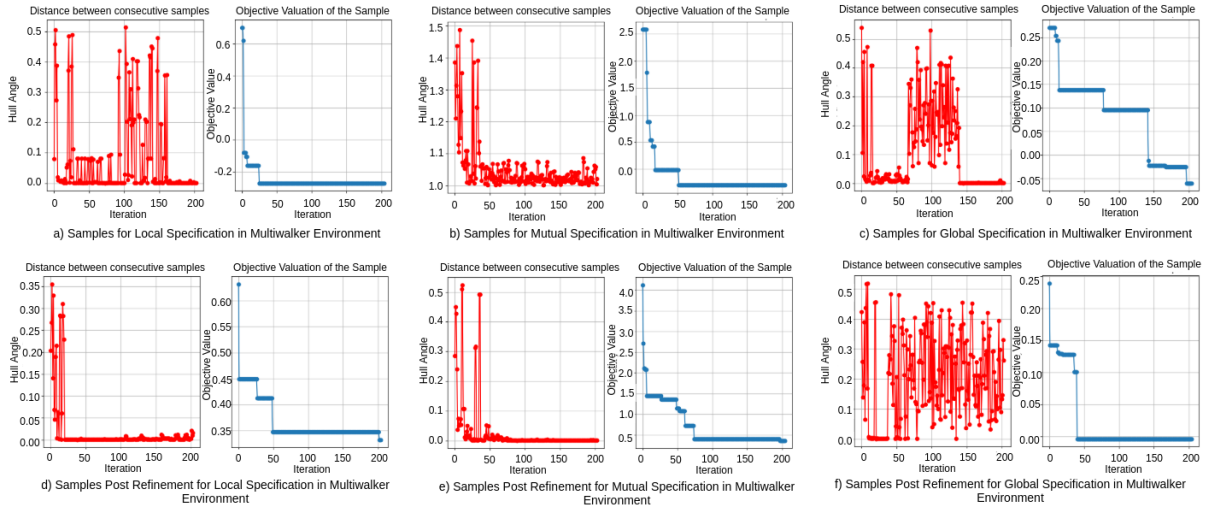

Figure 4: Plots of Bayesian Optimization samples collected and the objective function value for different a) Local b) Mutual c) Global specifications in the multi-walker environment. Plots d) e) f) illustrate samples collected from π_{safe} for each specification.

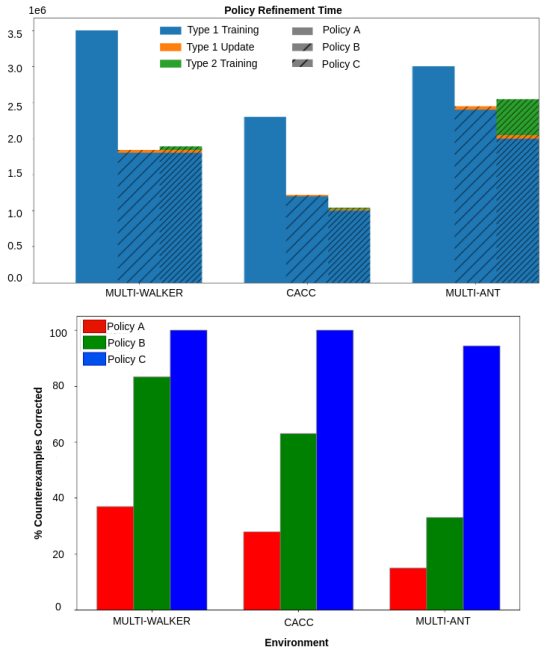

Figure 5: Comparison with baselines w.r.t training time and percentage of counterexamples corrected.

took 45K steps for multi-walker, 18K steps for CACC, and 4M steps for multi-ant (Appendix B [6]).

We consider the following baselines for comparison. 1) Policy A: A MAPPO policy trained from scratch with a combined penalty for specification violation and counterexample traces without importance β_t after one iteration of testing with BO 2) Policy B: A MAPPO policy refined only based on the individual specification violated and advantage set to 1 without secondary training as used in [8] 3) Policy C: Policy refined through the strategies proposed in this paper. The training time for each policy, along with the percentage of counterexamples corrected, is reported in Figure 5. We observe Policy A takes the highest time to train as it is trained from scratch but corrects the lowest amount of counterexamples due to the non-inclusion of β. Policy B has less training time as there is no secondary training involved but reports uncorrected counterexamples. Policy C, which is our strategy, has the highest number of counterexamples corrected in all three environments. Policy B and C are both refinement techniques and thus have similar training times. We report the value of the samples selected and the corresponding objective function evaluation for different specifications in the multi-walker environment in Figure 4 a), b), c). Though multiple state variables contribute to the violation of the specification, only samples of hull angle are plotted for comprehensibility. To demonstrate that the refinement methodology corrects the old counterexamples without introducing any new counterexamples to the policy, we retest the refined policy π_{safe} with 200 BO iterations. Figure 4 d), e), f) illustrates that all objective evaluations are positive for all specifications in π_{safe}.

6 CONCLUSION

Testing and refinement of MARL policies are critical for their deployment in real-world applications. To the best of our knowledge, we present the first counterexample-guided refinement strategy for a cooperative multi-agent learning setting. The refinement algorithm incorporates both paths that can reach the goal safely and paths that can terminate in a safe region through targeted gradient updates. A limitation of this work lies in assuming the safety predicates to be smooth and continuous functions of the trajectory to estimate their valuations through Gaussian Process. In the future, the authors would like to extend the strategy for providing formal guarantees of safety through formal verification methods and apply the methodology in competitive and mixed settings.

ACKNOWLEDGMENTS

The authors would like to thank TCS Research Scholarship for partially supporting this project.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17)*. JMLR.org, Sydney, NSW, Australia, 22–31.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, New Orleans, Louisiana, USA, Article 326, 10 pages.

[3] Steven Carr, Nils Jansen, Ralf Wimmer, Alexandru Constantin Serban, Bernd Becker, and Ufuk Topcu. 2019. Counterexample-Guided Strategy Improvement for POMDPs Using Recurrent Neural Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, Macao, China, 5532–5539. https://doi.org/10.24963/ijcai.2019/768

[4] Jyotirmoy Deshmukh, Marko Horvat, Xiaoqing Jin, Rupak Majumdar, and Vinayak S. Prabhu. 2017. Testing Cyber-Physical Systems through Bayesian Optimization. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 170 (Sept. 2017), 18 pages. https://doi.org/10.1145/3126521

[5] Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. 2021. Safe Multi-Agent Reinforcement Learning via Shielding. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Virtual Event, United Kingdom) *(AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 483–491.

[6] Briti Gangopadhyay. 2023. Counterexample-Guided Policy Refinement in Multi-Agent Reinforcement Learning (Code and Supplement) [Available online]. https://github.com/britig/Counterexample-Guided-Policy-Refinement-in-Multi-Agent-Reinforcement-Learning.

[7] Briti Gangopadhyay et al. 2022. Refinement Of Reinforcement Learning Algorithms Guided By Counterexamples. In *2022 IEEE Women in Technology Conference (WINTECHCON)*. IEEE, Bangalore,India, 1–6. https://doi.org/10.1109/WINTECHCON55229.2022.9832063

[8] Briti Gangopadhyay and Pallab Dasgupta. 2021. Counterexample Guided RL Policy Refinement Using Bayesian Optimization. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., Sydney, Australia, 22783–22794.

[9] Briti Gangopadhyay, Pallab Dasgupta, and Soumyajit Dey. 2022. Safe and Stable RL (S2RL) Driving Policies Using Control Barrier and Control Lyapunov Functions. *IEEE Transactions on Intelligent Vehicles* (2022), 1–1. https://doi.org/10.1109/TIV.2022.3160202

[10] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings. 2019. Identification of Test Cases for Automated Driving Systems Using Bayesian Optimization. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, Auckland, New Zealand, 1961–1967. https://doi.org/10.1109/ITSC.2019.8917103

[11] Javier García and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *J. Mach. Learn. Res.* 16, 1 (jan 2015), 1437–1480.

[12] S. Ghosh, F. Berkenkamp, G. Ranade, S. Qadeer, and A. Kapoor. 2018. Verifying Controllers Against Adversarial Examples with Bayesian Optimization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Brisbane, Australia, 7306–7313. https://doi.org/10.1109/ICRA.2018.8460635

[13] Shangding Gu, Jakub Grudzien Kuba, Muning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois C. Knoll, and Yaodong Yang. 2021. Multi-Agent Constrained Policy Optimisation. *CoRR* abs/2110.02793 (2021). arXiv:2110.02793 https://arxiv.org/abs/2110.02793

[14] Hanbin Hu, Peng Li, and Jianhua Z. Huang. 2018. Parallelizable Bayesian optimization for analog and mixed-signal rare failure detection with high coverage. In *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018*, Iris Bahar (Ed.). ACM, San Diego, CA, USA, 98. https://doi.org/10.1145/3240765.3240835

[15] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. 2021. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research* 40, 4-5 (2021), 698–721.

[16] Sham Kakade and John Langford. 2002. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML '02)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 267–274.

[17] Prabuchandran K.J., Hemanth Kumar A.N, and Shalabh Bhatnagar. 2014. Multi-agent reinforcement learning for traffic signal control. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Qingdao, 2529–2534. https://doi.org/10.1109/ITSC.2014.6958095

[18] Martin Lauer and Martin A. Riedmiller. 2000. An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 535–542.

[19] Michael L. Littman. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning* (New Brunswick, NJ, USA) *(ICML'94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 157–163.

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6382–6393.

[21] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning.

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 1889–1897. https://proceedings.mlr.press/v37/schulman15.html

[23] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175. https://doi.org/10.1109/JPROC.2015.2494218

[24] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *CoRR* abs/1610.03295 (2016). arXiv:1610.03295 http://arxiv.org/abs/1610.03295

[25] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550 (10 2017), 354–359. https://doi.org/10.1038/nature24270

[26] Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck. 2020. Counterexample-Guided Learning of Monotonic Neural Networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1001, 13 pages.

[27] Mingfei Sun, Sam Devlin, Katja Hofmann, and Shimon Whiteson. 2022. Monotonic Improvement Guarantees under Non-stationarity for Decentralized PPO. *CoRR* abs/2202.00082 (2022). arXiv:2202.00082 https://arxiv.org/abs/2202.00082

[28] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall Williams, Yashas Lokesh, and Praveen Ravi. 2021. PettingZoo: Gym for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., Sydney, Australia, 15032–15043. https://proceedings.neurips.cc/paper/2021/file/7ed2d3454c5eea71148b11d0c25104ff-Paper.pdf

[29] Justin K. Terry, Nathaniel Grammel, Benjamin Black, Ananth Hari, Caroline Horsch, and Luis Santos. 2020. Agent Environment Cycle Games. *ArXiv* abs/2009.13051 (2020).

[30] Ziran Wang, Guoyuan Wu, and Matthew J. Barth. 2018. A Review on Cooperative Adaptive Cruise Control (CACC) Systems: Architectures, Controls, and Applications. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE Press, Maui, HI, USA, 2884–2891. https://doi.org/10.1109/ITSC.2018.8569947

[31] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. 2021. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. *ArXiv* abs/2103.01955 (2021).

[32] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2019. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv* abs/1911.10635 (2019), 1–72. arXiv:1911.10635

[33] Junchen Zhao and Francesco Belardinelli. 2022. Safety-AwareMulti-Agent Apprenticeship Learning. *arXiv preprint arXiv:2201.08111* abs/2201.08111 (2022).

[34] Weichao Zhou and Wenchao Li. 2018. Safety-Aware Apprenticeship Learning. In *Computer Aided Verification*, Hana Chockler and Georg Weissenbacher (Eds.). Springer International Publishing, Cham, 662–680.