Implicit Poisoning Attacks in Two-Agent Reinforcement Learning: Adversarial Policies for Training-Time Attacks

Mohammad Mohammadi*
MPI-SWS
Saarbrücken, Germany
mmohamma@mpi-sws.org

Jonathan Nöther*
Saarland University
Saarbrücken, Germany
s8jonoet@stud.uni-saarland.de

Debmalya Mandal
MPI-SWS
Saarbrücken, Germany
dmandal@mpi-sws.org

Adish Singla
MPI-SWS
Saarbrücken, Germany
adishs@mpi-sws.org

Goran Radanovic
MPI-SWS
Saarbrücken, Germany
gradanovic@mpi-sws.org

ABSTRACT

In targeted poisoning attacks, an attacker manipulates an agent-environment interaction to force the agent into adopting a policy of interest, called *target* policy. Prior work has primarily focused on attacks that modify standard MDP primitives, such as rewards or transitions. In this paper, we study targeted poisoning attacks in a two-agent setting where an attacker *implicitly* poisons the *effective* environment of one of the agents by modifying the policy of its peer. We develop an optimization framework for designing optimal attacks, where the cost of the attack measures how much the solution deviates from the assumed default policy of the peer agent. We further study the computational properties of this optimization framework. Focusing on a tabular setting, we show that in contrast to poisoning attacks based on MDP primitives (transitions and (unbounded) rewards), which are always feasible, it is NP-hard to determine the feasibility of implicit poisoning attacks. We provide characterization results that establish sufficient conditions for the feasibility of the attack problem, as well as an upper and a lower bound on the optimal cost of the attack. We propose two algorithmic approaches for finding an optimal adversarial policy: a model-based approach with tabular policies and a model-free approach with parametric/neural policies. We showcase the efficacy of the proposed algorithms through experiments.

KEYWORDS

Adversarial Policies, Poisoning Attacks, Reinforcement Learning

ACM Reference Format:
Mohammad Mohammadi, Jonathan Nöther, Debmalya Mandal, Adish Singla, and Goran Radanovic. 2023. Implicit Poisoning Attacks in Two-Agent Reinforcement Learning: Adversarial Policies for Training-Time Attacks. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

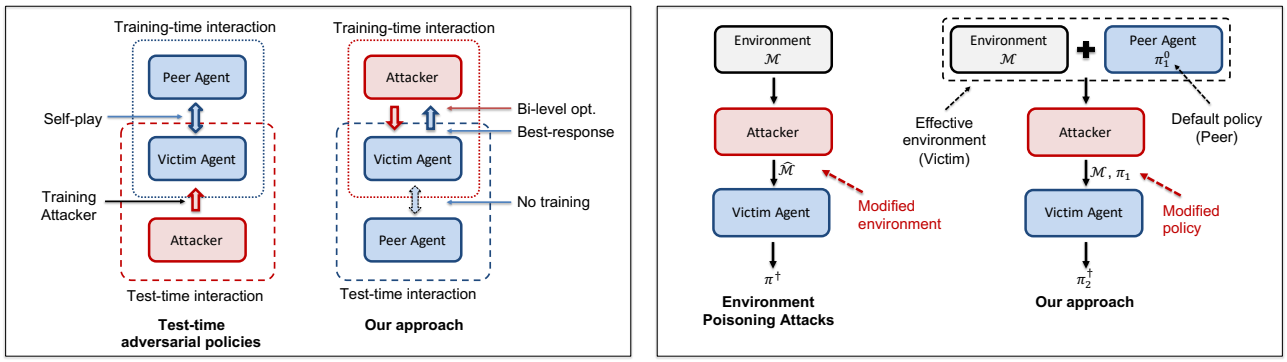
Recent works on adversarial attacks in reinforcement learning (RL) have demonstrated the susceptibility of RL algorithms to various forms of adversarial attacks [14, 17, 22, 42, 43], including poisoning attacks which manipulate a learning agent in its training phase, altering the end result of the learning process, i.e., the agent's policy [23, 26, 37–39, 43, 61]. In order to understand and evaluate the stealthiness of such attacks, it is important to assess the underlying assumptions that are made in the respective attack models. Often, attack models are based on altering the environment feedback of a learning agent. For example, in environment poisoning attacks, the attacker can manipulate the agent's rewards or transitions [26, 37]—these manipulations could correspond to changing the parameters of the model that the agent is using during its training process. However, directly manipulating the environment feedback of a learning agent may not always be practical. For example, rewards are often internalized or are goal specific, in which case one cannot directly poison the agent's rewards. Similarly, direct manipulations of transitions and observations may not be practical in environments with complex dynamics, given the constraints on what can be manipulated by such attacks.

In order to tackle these practical challenges, Gleave et al. [11] introduce a novel class of attack models for a competitive two-agent RL setting with a zero-sum game structure. In particular, they consider an attacker that controls one of the agents. By learning an adversarial policy for that agent, the adversary can force the other, victim agent, to significantly degrade its performance. Gleave et al. [11] focus on test-time attacks that learn adversarial policies for an already trained victim. This idea has been further explored by Guo et al. [13] in the context of more general two-player games, which are not necessarily zero-sum, and by Wang et al. [47] in the context of backdoor attacks, where the action of the victim's opponent can trigger a backdoor hidden in the victim's policy.

In this paper, we focus on *targeted* poisoning attacks that aim to force a learning agent into adopting a certain policy of interest, called *target policy*. In contrast to prior work on targeted poisoning attacks [26, 37, 38], which primarily considered single-agent RL and attack models that directly manipulate MDP primitives (e.g., rewards or transitions), we consider a two-agent RL setting and an attack model that is akin to the one studied in [11, 13, 47], but tailored to environment poisoning attacks. More specifically, in our setting, the attacker *implicitly* poisons the *effective* environment of a victim agent by controlling its peer at training-time. To ensure the stealthiness of the attack, the attacker aims to minimally alter

*Equal contributions

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.



(a) Comparison to test-time adversarial policies

(b) Comparison to environment poisoning attacks

Figure 1: The figure compares our approach to the closely related prior work. Fig. 1a illustrates the differences between our setting and test-time adversarial policies (from [11]). Test-time adversarial policies are attacking a fixed victim (trained in self-play), whereas our approach attacks a victim during training phase. We consider an optimization framework based on bi-level optimization that minimizes the attack cost while ensuring that the victim's best response to our attack is to adopt target policy $\pi_2^\dagger$. As shown in Fig. 1b, this optimization approach is similar to environment poisoning attacks (from [26, 37, 38]). However, it differs from environment poisoning attacks in that the attack only modifies the default policy of the victim's peer, π_1^0, but not the underlying environment $\mathcal{M}$. I.e, our approach implicitly poisons the effective environment of the victim.

the default behavior of the peer, which we model through the cost of the attack.

Fig. 1 illustrates the main aspects of the setting considered in this work. In this setting, the default policy of the victim's peer is fixed and the victim is trained to best respond to a corrupted version of the peer's policy.[1] This is different from test-time adversarial policies where the victim is fixed and the adversary controls the victim's opponent/peer at test-time. Our setting corresponds to a practical scenario in which an attacker controls the peer agent at training-time and executes a training-time adversarial policy instead of the peer's default policy. If the victim is trained offline, the attacker can corrupt the offline data instead, e.g., by executing training-time adversarial policies when the offline data is collected or by directly poisoning the data. Note that direct access to the training process of the victim agent is not required to train an adversarial policy. It suffices that the victim agent approximately best respond to the adversarial policy. This is effectively the same assumption that prior work on environment poisoning attacks in offline RL adopts, where the attacker first manipulates the underlying environment, after which the victim agent optimizes its policy in the poisoned environment [26, 37, 38]. Fig. 1 also shows how our setting compares to environment poisoning target attacks. As explained in the figure, there are conceptual differences between the corresponding attack models.

To the best of our knowledge, this is the first work that studies adversarial policies for training-time attacks. Our contributions are summarized below:

- We introduce a novel optimization framework for studying an implicit form of targeted poisoning attacks in a two-agent RL setting, where an attacker manipulates the victim agent by controlling its peer at training-time.
- We then analyze computational aspects of this optimization problem. We show that it is NP-hard to decide whether the optimization problem is feasible, i.e., whether it is possible to force a target policy. This is in contrast to general environment poisoning attacks that manipulate both rewards and transitions [38], which are always feasible.[2]
- We further analyze the cost of the optimal attack, providing a lower and an upper bound on the cost of the attack, as well as a sufficient condition for the feasibility of the attack problem. To obtain the lower bound, we follow the theoretical analysis in recent works on environment poisoning attacks [26, 37, 38] that establish similar lower bounds for the single-agent setting, and we adapt it to the two-agent setting of interest. The theoretical analysis that yields the upper bound does not follow from prior work, since the corresponding proof techniques cannot be directly applied to our setting.
- We propose two algorithmic approaches for finding an efficient adversarial policy. The first one is a model-based approach with tabular policies which outputs a feasible solution to the attack problem, if one is found. It is based on a conservative policy search algorithm that performs efficient policy updates that account for the cost of the attack, while aiming to minimize the margin by which the constraints of the attack problem are not satisfied. The second one is a model-free approach with parametric/neural policies, which is based on a nonconvex-nonconcave minimax optimization.

[1] While the setting is similar to Stackelberg models where the peer agent commits its policy (e.g., [20]), a critical difference is that the adversary can modify this policy. See also the related work section.

[2] Attacks that poison only rewards are always feasible. Attacks that poison only transitions may not be always feasible since transitions cannot be arbitrarily changed [37]. Similarly, when rewards are bounded, Rangi et al. [39] show that reward poisoning attacks may be infeasible. The computational complexity of such attacks have not been formally analyzed.

- Finally, we conduct extensive experiments to demonstrate the efficacy of our algorithmic approaches.

The full version of the paper that includes the Supplementary Material can be found in [28].

1.1 Related Work

Adversarial Attacks in ML. Adversarial attacks on machine learning models have been extensively studied by prior work. We recognize two main attack approaches on machine learning: test-time attacks [6, 29, 31, 32, 44], which do not alter a learning model, but rather they fool the model by manipulating its input, and training-time or data poisoning attacks [7, 21, 27, 52, 53], which manipulate a learning model by, e.g., altering its training points. We also mention backdoor attacks [8, 12, 24], that hide a trigger in a learned model, which can then be activated at test-time.

Adversarial Attacks in RL. Needless to say, such attack strategies have also been studied in RL. [5, 14, 18, 22, 42] consider efficient test-time attacks on agents' observations. In contrast to this line of work, we consider training-time attacks which are not based on state/observation perturbations. [17, 48, 54] consider backdoor attacks on RL policies. These works are different in that backdoor triggers affect the victim's observations; our attack model influences the victim's transitions and rewards. Poisoning attacks in single-agent RL have been studied under different poisoning aims: attacking rewards [26, 37, 39], attacking transitions [37], attacking both rewards and transition [38], attacking actions [23], or attacking a generic observation-action-reward tuple [43]. Reward poisoning attacks have also been studied in multi-agent RL [51]. In contrast to such poisoning attacks, our attack model does not directly poison any of the mentioned poisoning aims. It instead indirectly influences the victim's rewards and transitions. This work, therefore, complements prior work on poisoning attacks in RL and adversarial policies, as already explained.

Other Related Work. We also mention closely related work on robustness to adversarial attacks and settings that have similar formalisms. Much of the works on robustness to these attacks study robustness to test-time attacks [33, 49, 55, 56] and, closer to this paper, poisoning attacks [4, 19, 25, 50, 59, 60]. Out of these, [4] has the formal setting that is the most similar to ours, focusing on defenses against targeted reward poisoning attacks. Our setting is also related to stochastic Stackelberg games and similar frameworks [10, 20, 46] in that we have an attacker who acts as a leader that aims to minimize its cost, while accounting for a rational follower (victim) that optimizes its return. However, in our framework, the cost of the attack is not modeled via a reward function, while the attack goal of forcing a target policy is a hard constraint. Hence, the computational intractability results for Stackelberg stochastic games do not directly apply to our setting. Nonetheless, the reduction that we use to show our NP-hardness result is inspired by the proofs of the hardness results in [20]. Finally, we also mention the line of work on policy teaching [3, 57, 58], whose formal settings are quite similar to those of targeted reward poisoning attacks [26, 38].

2 IMPLICIT POISONING ATTACKS

In this section, we formalize the attack problem of interest: adversarial policies for training-time attacks.

2.1 Multi-Agent Environment

Environment model. We study a reinforcement learning setting formalized by a two-agent Markov Decision Process $\mathcal{M} = (\{1, 2\}, S, A, P, R_2, \gamma, \sigma)$, where 1 is the index of an agent controlled by an attacker, 2 is the index of a learning agent (victim) under attack, S is the state space, $A = A_1 \times A_2$ is the joint action space with A_1 and A_2 defining the action spaces of agents 1 and 2 respectively, $P : S \times A \times S \to [0, 1]$ is the transition model, $R_2 : S \times A \to \mathbb{R}$ is the reward function of the learner, γ is the discount factor, and σ is the initial state distribution. We denote the probability of transitioning to state s' from s by $P(s, a_1, a_2, s')$ and the reward obtained in state s by $R_2(s, a_1, a_2)$, where a_1 and a_2 are the actions of agent 1 and agent 2 taken in state s. In our formal treatment of the problem, we will primarily focus on finite state and action spaces S and A.

Policies. The policy of agent 1 is denoted by π_1 and we assume that it comes from the set of stochastic stationary policies Π^1. That is, policy π_1 is mapping $\pi_1 : S \to \mathcal{P}(A_1)$, where $\mathcal{P}(A_1)$ is the probability simplex over A_1. Analogously, the policy of agent 2 is denoted by π_2. A stochastic stationary policy $\pi_2 \in \Pi^2$ is a mapping $\pi_2 : S \to \mathcal{P}(A_2)$. The set of all deterministic policies in Π^2 is denoted by $\Pi^2_{\text{det}} = \{\pi_2 \in \Pi^2 \text{ s.t. } \pi_2(s, a_2) \in \{0, 1\}\}$.

Score & Occupancy Measures. We further consider standard quantities. The (normalized) expected discounted return of agent 2 under policies π_1 and π_2 is defined as

$$\rho_2 = (1 - \gamma) \cdot \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \cdot R_2(s_t, a_{1,t}, a_{2,t}) | \sigma, \pi_1, \pi_2\right],$$

where the expectation is taken over trajectory $(s_1, a_{1,1}, a_{2,1}, ...)$ obtained by executing policy π starting in a state sampled from σ. The return $\rho_2^{\pi_1,\pi_2}$ is equal to

$$\rho_2^{\pi_1,\pi_2} = \sum_{s,a_1,a_2} \psi^{\pi_1,\pi_2}(s, a_1, a_2) \cdot R_2(s, a_1, a_2), \tag{1}$$

where $\psi^{\pi_1,\pi_2}(s, a_1, a_2) = \mu^{\pi_1,\pi_2}(s) \cdot \pi_1(s, a_1) \cdot \pi_2(s, a_2)$ is the state-action occupancy measure, and μ^{π_1,π_2} is the state occupancy measure, i.e., $\mu^{\pi_1,\pi_2}(s) = (1 - \gamma) \cdot \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \cdot \mathbb{1}\left[s_t = s\right] | \sigma, \pi_1, \pi_2\right]$. Note that we do not assume that the underlying MDP is ergodic, i.e., we allow that $\mu^{\pi_1,\pi_2}(s) = 0$ for some states s. Finally, we also define value function $V^{\pi_1,\pi_2} : S \to \mathbb{R}$ as

$$V^{\pi_1,\pi_2}(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \cdot R_2(s_t, a_{1,t}, a_{2,t}) | s_1 = s, \pi_1, \pi_2\right].$$

Remark 1. *To simplify the notation, we often abbreviate summations, e.g., the summation over a_1 and a_2 can be replaced by $R_2(s, \pi_1, \pi_2)$. Furthermore, since in our formal treatment of the problem we focus on a tabular setting with finite state and action spaces, we in part utilize vector notation when convenient. For example, R_2 can be thought of as a vector with $|S| \cdot |A|$ components.*

2.2 Problem Statement

We focus on an attack model that manipulates a *default* policy of the victim's peer, π_1^0, to force a target policy $\pi_2^\dagger$. Following prior work on targeted policy attacks [26, 37, 38], we first consider an optimization problem which models the attack goal as a hard constraint with *deterministic* $\pi_2^\dagger$ and a victim agent that adopts an approximately

optimal deterministic policy[3]:

$$\min_{\pi_1} \quad \textsc{Cost}(\pi_1, \pi_1^0) \qquad \text{s.t.} \quad \textsc{Opt}_2^{\epsilon}(\pi_1) \subseteq \Pi_2^{\dagger}(\pi_1). \tag{P1}$$

Here, $\Pi_2^{\dagger}(\pi_1)$ is a set of policies π_2 that are equal to $\pi_2^{\dagger}$ on visited states, i.e., $\pi_2(s, a_2) = \pi_2^{\dagger}(s, a_2)$ when $\mu^{\pi_1,\pi_2^{\dagger}}(s) > 0$. Furthermore, $\textsc{Opt}_2^{\epsilon}(\pi_1)$ is the set of approximately optimal deterministic policies π_2 given π_1, i.e., $\textsc{Opt}_2^{\epsilon}(\pi_1) = \{\pi_2 \in \Pi^2_{\text{det}} \text{ s.t. } \rho^{\pi_1,\pi_2} > \rho^{\pi_1,\pi_2^{*|\pi_1}} - \epsilon\}$, where $\pi_2^{*|\pi_1} \in \arg\max_{\pi_2} \rho_2^{\pi_1,\pi_2}$, while $\epsilon \geq 0$ is a parameter that controls the sub-optimality of the learner. As standard in this line of work, in our characterization results of (P1), we focus on a norm-based attack cost function:

$$\textsc{Cost}(\pi_1, \pi_1^0) = \left(\sum_s \left(\sum_{a_1} |\pi_1(s, a_1) - \pi_1^0(s, a_1)| \right)^{\frac{1}{p}} \right)^p,$$

where $p \geq 1$. In the next sections, we formally analyze (P1) and propose an algorithm for solving it. We also consider an optimization problem that relaxes the attack goal, but is more amenable to optimization with deep RL and allows *stochastic* target policies $\pi_2^{\dagger}$:

$$\min_{\theta} \max_{\phi} \mathcal{L}_I(\theta, \pi_1^0) - \lambda \cdot \left[\rho_2^{\pi_\theta,\pi_2^{\dagger}} - \rho_2^{\pi_\theta,\pi_\phi} \right]. \tag{P2}$$

Here, π_θ and π_ϕ are parametric policies that respectively correspond to π_1 and π_2, and $\mathcal{L}_I(\theta, \pi_1^0)$ is an imitation learning loss function. The imitation learning loss corresponds to the cost of the attack: we instantiate it with standard cross-entropy imitation objective for deterministic π_1^0 and Kullback–Leibler divergence for stochastic π_1^0. We further motivate (P2) in the next sections.

3 CHARACTERIZATION RESULTS

In this section, we provide a theoretical treatment of the optimization problem (P1) akin to those from prior work on poisoning attacks in RL [26, 37, 38]. We start by analyzing the complexity of the optimization problem, followed by the analysis that provides bounds on the optimal value of (P1). The proofs of our results from this section are provided in the full version of the paper [28].

3.1 Computational Complexity

To study the properties of the optimization problem (P1), let us more explicitly write its constraint using the following set of inequalities:

$$\rho_2^{\pi_1,\pi_2^{\dagger}} \geq \rho_2^{\pi_1,\pi_2} + \epsilon, \quad \forall \pi_2 \in \Pi^2_{\text{det}} \backslash \Pi_2^{\dagger}(\pi_1).$$

At the first glance, the optimization problem (P1) appears to be computationally challenging: the number of inequality constraints is exponential. On the other hand, Lemma 1 from [38] suggests that it suffices to consider *neighbor* policies of the target policy to determine whether a solution is feasible—a neighbor policy $\pi\{s, a\}$ of policy π is equal to π in all states except in s, where it is defined as $\pi\{s, a\}(s, a) = 1.0$. However, given the differences between the setting of Rakhsha et al. [38] and the setting of this paper, in particular, because the latter considers a two-agent and possibly non-ergodic MDP environment, this result does not directly apply. In the full version of the paper [28], we prove a couple of results akin to Lemma 1 from [38], but for the setting of interest. These results allow us to reduce the number of constraints one ought to account for when testing the feasibility of solution π_1. For example, if the MDP environment is ergodic, π_1 is a feasible solution iff

$$\rho_2^{\pi_1,\pi_2^{\dagger}} \geq \rho_2^{\pi_1,\pi_2^{\dagger}\{s,a_2\}} + \epsilon \quad \forall s, a \text{ s.t. } \pi_2^{\dagger}(s, a_2) = 0. \tag{2}$$

While such results are useful as they reduce the number of constraints one ought to account for when testing the feasibility of solution π_1, they do not necessarily imply that the optimization problem is easy to solve. The difficulty lies in the quadratic form of the constraints in Eq. (2). Namely, as can be seen from Eq. (1), they depend on π_1 through policy π_1 itself but also through the state occupancy measure $\mu^{\pi_1,\cdot}$. Our next result verifies this intuition.

Theorem 1. *It is NP-hard to decide if the optimization problem* (P1) *is feasible, i.e., whether there exists a solution π_1 s.t. the constraints of the optimization problem are satisfied.*

The proof of the claim can be found in [28], and is based on a polynomial time reduction of the Boolean 3-SAT problem to our optimization framework. Hence, the tractability of the optimization problem (P1) would imply that NP=P. To conclude, despite the similarities between our implicit poisoning attack model and the general environment poisoning attacks from [38], which are always feasible, determining the feasibly of implicit poisoning attacks is computationally challenging.

3.2 Bounds on the Optimal Value

Lower Bound. Next, we aim to bound the value of the optimal solution. We first focus on a lower bound on the cost of optimal solution. In particular, we follow the recent line of work on poisoning attacks in RL [26, 37, 38], and adapt their proof techniques to our problem setting in order to establish a lower bound on the cost of the optimal attack. To state the main theorem, we define a state-action dependent quantity $\bar{\chi}_{\epsilon'}(s, a)$ similar to the one from [38], but adapted to the setting of the paper. In particular, we define[4] $\overline{\chi}_{\epsilon'}(s, a_2) = \left[\frac{\rho_2^{\pi_1^0,\pi_2^{\dagger}\{s,a_2\}} - \rho_2^{\pi_1^0,\pi_2^{\dagger}} + \epsilon'}{\mu^{\pi_1^0,\pi_2^{\dagger}\{s,a_2\}}(s)} \right]^+$ if $\mu^{\pi_1,\pi_2^{\dagger}}(s) > 0$ for all π_1 and $\pi_2^{\dagger}(s, a_2) = 0$, while $\overline{\chi}_{\epsilon'}(s, a_2) = 0$ otherwise.[5] $\overline{\chi}_{\epsilon'}(s, a)$ is a measure of the utility gap between the target policy $\pi_2^{\dagger}$ and the neighbor policy $\pi_2^{\dagger}\{s, a\}$ given the default policy π_1^0 and some offset ϵ'. Together with R_2 and $V^{\pi_1^0,\pi_2^{\dagger}}$, $\overline{\chi}_{\epsilon'}$ can be used to obtain the following lower bound.

Theorem 2. *The attack cost of any solution to the optimization problem* (P1)*, if it exists, satisfies*

$$\textit{Cost}(\pi_1, \pi_1^0) \geq \frac{1-\gamma}{2} \cdot \frac{\left\|\overline{\chi}_0\right\|_\infty}{\|R_2\|_\infty + \gamma \cdot \left\|V^{\pi_1^0,\pi_2^{\dagger}}\right\|_\infty}.$$

The lower bound in Theorem 2 is similar to the corresponding lower bound for general environment poisoning attacks [38], albeit not being fully comparable given the differences between the

[3] Similar learner models have been considered in prior work that analyzes a dual to optimal reward poisoning attacks [3].

[4] Note that $[x]^+ = \max(0, x)$.

[5] The first condition can be verified for any given state s by optimizing over π_1 a reward function that is strictly negative in state s, and is equal to 0 otherwise. The condition is satisfied iff the optimal value is 0. In general, the condition holds if the underlying Markov chain is ergodic for $\pi_2^{\dagger}$ and every policy π_1 (see Theorem 3).

settings and the definitions of $\overline{\chi}$. One notable difference is that the bound in Theorem 2 additionally depends on the reward vector R_2 because the adversary only influences rewards through its actions.

Upper Bound. Compared to environment poisoning attacks, providing an interpretable upper bound in our setting is more challenging since the attack model of this paper cannot in general successfully force a target policy $\pi_2^\dagger$. This is in stark contrast to, e.g., reward poisoning attacks, which remain feasible even under the restriction that rewards obtained by following $\pi_2^\dagger$ are not modified. Additionally, as per Theorem 1, the feasibility of the attack problem is computationally intractable. Due to the latter challenge, we consider a special case when transitions are independent of policy π_1 (i.e., $P(s, a_1, a_2) = P(s, a_1', a_2)$ for all a_1 and a_1') and the Markov chain induced by $\pi_2^\dagger$ and any policy π_1 is ergodic. In [28], we show that (P1) can be efficiently solved in this case.

To state our formal result, we first define two quantities, $\alpha_2^{\pi_1}(s,a) = \rho_2^{\pi_1,\pi_2^\dagger} - \rho_2^{\pi_1,\pi_2^\dagger\{s,a\}}$, and $\alpha_2^* = \sup_{\pi_1} \min_{s,a} \alpha_2^{\pi_1}(s,a)$. Intuitively, $\alpha_2^{\pi_1}$ measures the utility gap between $\pi_2^\dagger$ and its neighbor policy $\pi_2^\dagger\{s,a\}$ for a given policy π_1, whereas α_2^* denotes the optimal guaranteed gap that can be achieved. Note that there exists π_1^* s.t. $\alpha_2^{\pi_1^*} = \alpha_2^*$, and in [28] we provide a linear program for finding π_1^*.

Theorem 3. *Assume that $P(s, a_1, a_2) = P(s, a_1', a_2)$ for all a_2 and a_1', and that for $\pi_2^\dagger$ and every policy π_1 the underlying Markov chain is ergodic, i.e., $\mu^{\pi_1,\pi_2^\dagger}(s) > 0$ for all π_1. If $\alpha_2^* \geq \epsilon$, the optimization problem* (P1) *is feasible and the cost of an optimal solution satisfies*

$$\textsc{Cost}(\pi_1, \pi_1^0) \leq 2 \cdot \left\| \frac{\overline{\chi}_\epsilon}{\chi_2^* + \overline{\chi}_\epsilon} \right\|_\infty \cdot |S|^{1/p}$$

with the element-wise division (equal to 0 if $\chi_\epsilon(s,a_2) = \chi_2^*(s,a_2) = 0$), where $\chi_2^*(s,a_2) = \frac{\alpha_2^*(s,a_2) - \epsilon}{\mu^{\pi_1^0,\pi_2^\dagger\{s,a_2\}}(s)}$.

As with the lower bound, the upper bound is not directly comparable to the bounds obtained in prior work [38]. In the full version of the paper [28], we analyze another special case, when both π_1 and π_2 do not influence transitions, and obtain a slightly tighter bound. In that case, we obtain the upper bound $2 \cdot \left\| \frac{\overline{\chi}_\epsilon}{\chi_2^* + \overline{\chi}_\epsilon} \right\|_{p,\infty}$, where $\overline{\chi}_\epsilon$ and χ_2^* are now treated as matrices with $|A_2| \times |S|$ entries. We leave for the future work whether it is possible to improve the result in Theorem 3 and match this bound.

4 ALGORITHMS

In this section, we study two algorithmic approaches for solving the optimization problem (P1): a model-based approach with tabular policies for solving (P1), and a model-free approach with neural policies for solving (P2).

4.1 Conservative Policy Search for Implicit Attacks

In this subsection, we propose an algorithm for (P1). To simplify the exposition, we focus on a version of the algorithm that applies to ergodic environments—in the full version of the paper [28], we provide an extension to non-ergodic environments.

Algorithm 1 Conservative Policy Search for Implicit Attacks and Ergodic Environments

Input: $\mathcal{M} = (\{1,2\}, S, A, P, R, \gamma, \sigma)$, ϵ, δ_ϵ, π_1^0, λ, p
Output: Policy of the adversary, π_1
  Initialize $t = 0$
  for $t = 0$ to $T - 1$ **do**
    Calculate state occupancy measures $\mu^{\pi_1^t,\pi_2^\dagger}$ and $\mu^{\pi_1^t,\pi_2^\dagger\{s,a_2\}}$
    Evaluate the gap $\epsilon_{\pi_1^t} = \min_{\epsilon'} \epsilon'$ s.t. $\rho_2^{\pi_1^t,\pi_2^\dagger} \geq \rho_2^{\pi_1^t,\pi_2^\dagger\{s,a_2\}} + \epsilon'$
    Solve the optimization problem (P1') to obtain π_1^{t+1}
    if $\pi_1^{t+1} = \pi_1^t$ **then**
      break
    end if
  end for
  Set the result π_1 to solution π_1^t that minimizes $\left\|\pi_1^t - \pi_1^0\right\|_{1,p}$ while satisfying $\epsilon_{\pi_1^t} \geq \epsilon$

To design an efficient algorithmic procedure for finding a solution to (P1), we utilize the fact that (P1) can be efficiently solved when policy π_1 does not affect the transition dynamics. Inspired by conservative policy iteration [16] and similar approaches in RL [40], we propose an algorithm that alternates between two phases.

(1) In the first phase, we obtain the occupancy measures of the current solution π_1^t and policies $\pi_2^\dagger$ and $\pi_2^\dagger\{s,a\}$. That is, we calculate $\mu^{\pi_1^t,\pi_2^\dagger}$ and $\mu^{\pi_1^t,\pi_2^\dagger\{s,a\}}$.
(2) In the second phase, we update the current solution π_1^t by solving a relaxed version of (P1), i.e.,

$$\min_{\pi_1 \in \mathcal{B}(\pi_1^t,\delta),\epsilon'} \quad \textsc{Cost}(\pi_1, \pi_1^0) - \lambda \cdot \min\{\epsilon', \epsilon \cdot (1+\delta_\epsilon)\}$$
$$\text{s.t.} \quad \hat{\rho}_2^{\pi_1,\pi_2^\dagger} \geq \hat{\rho}_2^{\pi_1,\pi_2^\dagger\{s,a_2\}} + \epsilon', \qquad \text{(P1')}$$

for all s s.t. $\mu^{\pi_1,\pi_2^\dagger}(s) > 0$ and a_2 s.t. $\pi_2^\dagger(s,a_2) = 0$. Here, $B(\pi_1^t,\delta) = \{\pi_1 \text{ s.t. } |\pi_1(s,a_1) - \pi_1^t(s,a_1)| \leq \delta\}$, $\hat{\rho}_2^{\pi_1,\pi_2}$ is obtained via Eq. (1) but by using $\mu^{\pi_1^t,\pi_2}$ instead of μ^{π_1,π_2}, and $\delta_\epsilon \geq 0$ is a positive offset which adjusts ϵ (and whose role is explained later in the text).

The optimization problem (P1') is a relaxation of (P1) since we optimize over the margin parameter ϵ', which can take negative values. Hence, (P1') is always feasible. Critically, when solving (P1'), the state occupancy measures are fixed to $\mu^{\pi_1^t,\pi_2^\dagger}$ and $\mu^{\pi_1^t,\pi_2^\dagger\{s,a\}}$, which implies that we can solve (P1') efficiently since the objective is convex, while the constraints are linear in π_1 and ϵ'. The conservative update is reflected in the constraint $\pi_1 \in B(\pi_1^t,\delta)$, which ensures that solutions to (P1') approximately satisfy the constraints of the original problem (P1) (e.g., see Lemma 14.1 in [1]). We can control the quality of this approximation through the hyperparameter δ_ϵ: for higher values of δ_ϵ and $\epsilon' \geq \epsilon \cdot (1+\delta_\epsilon)$, solution π_1 to (P1') is more likely to be be a feasible solution to (P1).

The final step of each iteration is to evaluate the true gap $\rho_2^{\pi_1^t,\pi_2^\dagger} - \rho_2^{\pi_1^t,\pi_2}$ that each solution π_1^t achieves. The output of the algorithm is the solution that minimizes the cost while ensuring that the target gap ϵ is achieved.

Algorithm 1 summarizes the main steps of conservative policy search for ergodic environments.[6] The algorithm assumes access to the model of the environment, i.e., the corresponding MDP parameters (rewards and transition probabilities), needed for obtaining relevant quantities, such as occupancy measures. The algorithm also takes the learner's parameter ϵ as its inputs; in practice, one can use a conservative estimate of the true parameter instead.

4.2 Alternating Policy Updates for Implicit Attacks

We now turn to (P2). First, note that we can view (P2) as a parametric relaxation of (P1'). Namely, (P2) is equivalent to the bi-level optimization problem:

$$\min_{\theta} \mathcal{L}_I(\theta, \pi_1^0) - \lambda \cdot \left[\rho_2^{\pi_\theta, \pi_2^\dagger} - \rho_2^{\pi_\theta, \pi_{\phi^*}} \right] \qquad \text{(P2')}$$
$$\text{s.t.} \qquad \pi_{\phi^*} \in \arg\max_{\phi} \rho_2^{\pi_\theta, \pi_\phi}.$$

The second term, $\rho_2^{\pi_\theta, \pi_2^\dagger} - \rho_2^{\pi_\theta, \pi_{\phi^*}}$, measures the sub-optimality gap of the target policy, and corresponds to parameter ϵ' in (P1'), while the first term corresponds to the cost of the attack. This bi-level structure also motivates our algorithmic approach for finding an optimal θ.

In general, the objective of (P2) is nonconvex-nonconcave, so the order of min and max is important (e.g., see [15]). To solve the optimization problem (P2), we alternate between minimizing the loss function $\mathcal{L}(\theta, \phi)$ over parameters θ while keeping parameters ϕ fixed, and maximizing $\rho_2^{\pi_\theta, \pi_\phi}$ over parameters ϕ while keeping θ fixed. Each optimization subroutine optimizes for a few episodes, with the latter one using more episodes. As shown by [36], this type of alternating optimization can be more effective in solving game-theoretic bi-level optimization problems in RL similar to (P2') than a gradient descent-ascent approach that, in our setting, would simultaneously update θ and ϕ. Algorithm 2 summarizes the main steps of our alternating policy updates (APU) approach. In our implementation, we pre-train policy π_ϕ for ϕ-pretrain timesteps, which is typically larger than the number of timesteps (ϕ-update timesteps) used for updating π_ϕ in each epoch (ϕ-pretrain timesteps = 10000 and ϕ-update timesteps = 5000 in our experiments).

5 EXPERIMENTS

In this section, we demonstrate the efficacy of our algorithmic approaches through simulation-based experiments. As explained in the introduction, our setting differs from those studied in prior work, so our algorithms are not directly comparable to approaches from prior work. Hence, we compare our algorithms against their simplified versions and naive baselines. Additional results and implementation details, including running times and training parameters, are provided in the full version of the paper [28].[7]

[6] While the algorithm is well defined for any π_1^t, in the experiments we only consider π_1^t that are fully stochastic, i.e., $\pi_1(s, a_1) > 0$ for any s and a_1. In this case, the set of states s s.t. $\mu^{\pi_1^t, \pi_2^\dagger}(s)$ does not change over time, and can be precalculated.

[7] The code for this paper is available at https://github.com/gradanovic/rl-implicit-poisoning-attacks.

Algorithm 2 Alternating Policy Updates for Implicit Attacks

Input: *epochs*, λ, ϵ, $\pi_2^\dagger$, π_1^0, ϕ-pretrain timesteps, ϕ-update timesteps

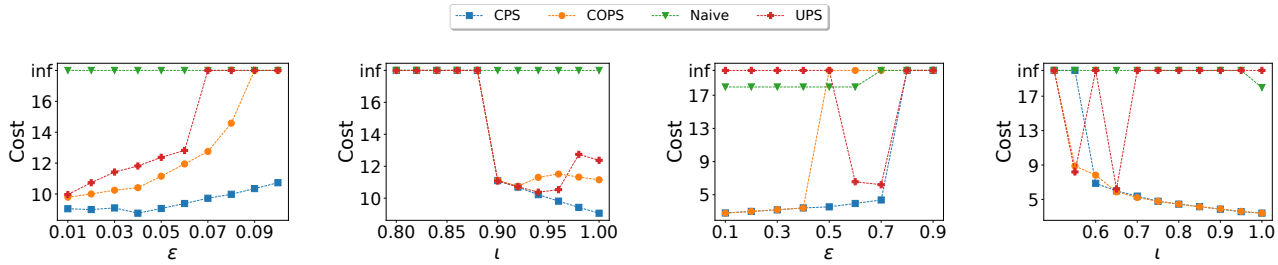
Initialize π_θ and π_ϕ
Train π_ϕ for ϕ-pretrain timesteps with PPO that optimizes performance under R_2 and π_θ
for epoch = 0, 1, ... **do**
  Update π_ϕ for ϕ-update timesteps with PPO that optimizes performance under R_2 and π_θ
  Collect trajectory τ^ϕ with π_θ and π_ϕ
  Collect trajectory $\tau^\dagger$ with π_θ and $\pi_2^\dagger$
  $bc\_loss \leftarrow cost\_fn(\pi_1^0, \pi_\theta)$ {either cross entropy or kl-divergence}
  $loss^\dagger \leftarrow \mathcal{L}^{PPO}(\tau^\dagger, \pi_\theta)$
  $loss^\phi \leftarrow \mathcal{L}^{PPO}(\tau^\phi, \pi_\theta)$
  $policy\_loss \leftarrow \frac{1}{|\tau^\phi| + |\tau^\dagger|} \cdot (loss^\phi - loss^\dagger)$ {where $|\tau|$ is the length of trajectory τ, i.e., the number of timesteps in τ}
  $loss \leftarrow bc\_loss + \lambda \cdot policy\_loss$
  Update π_θ with gradients of $loss$
  Update critic network of adversary with τ^ϕ and $\tau^\dagger$
end for

5.1 Experiments for Conservative Policy Search

We consider two environments based on or inspired by prior work [34, 38], but modified to fit the two-agent setting of this paper.

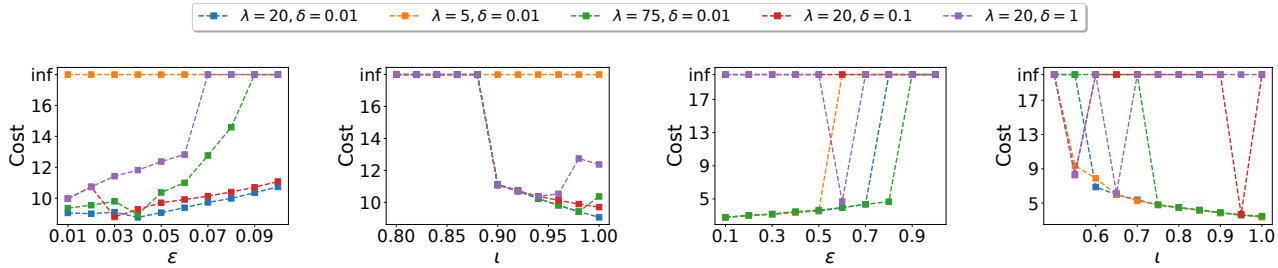
Navigation Environment. This environment is based on the navigation environment from [38], developed for testing environment poisoning attacks on a single RL agent in a tabular setting. We refer the reader to [38] for the description of the original environment and to [28] for the full description of the two-agent variant that we introduce. The original environment is ergodic, contains 9 states and the action space of an agent specifies in which direction ("left" or "right") the agent should move. The two-agent variant has an extended action space to include the actions of the attacker, who has the same action space as the victim agent. Rewards and transitions primarily depend on whether the actions of the two agents match, e.g., if the agents' actions match, the victim agent moves in the desired direction with high probability and obtains a positive reward. The default policy of the attacker is to always take "left", while the target policy is that the victim takes action "right" in each state.

Inventory Management. We consider a modified version of the inventory management environment from [34], with two agents. As in the original version, we have a manager of a warehouse that decides on the current inventory of a warehouse (the number of stocks/items in the warehouse). In our two agent version of the environment, the victim agent is controlling the amount of stock in the inventory and the attacker is controlling the demand. The victim's actions are "buy" actions that select between 0 and $M-1$ items. The attacker's actions are "create demand" of 0 to $M-1$ items. In our experiments, we set $M = 10$ and $\gamma = 0.9$. The default policy of the attacker is to select the demand uniform at random over all possible values. The target policy is defined by the following rule: if there are more than $k = 7$ items, do not buy anything, otherwise



(a) Navigation, ϵ (b) Navigation, Influence (c) Inventory, ϵ (d) Inventory, Influence

Figure 2: The cost of the attack as a function of the victim's sub-optimality and the adversary's influence over the victim's peer agent. We use the same cost function as for the characterization results. The default value of ϵ is 0.05 for Navigation and 0.4 for Inventory. When inf is reached, no solution was found. As explained in the text, ϵ parameter controls the sub-optimality of the learner. Fig. 2a and Fig. 2c show that the more sub-optimal the learner is, the harder it is to force a target policy. We further vary the influence of the attacker ι by executing policy $\pi_1^\iota(s,a) = (1-\iota)\cdot\pi_1^0(s,a) + \iota\cdot\pi_1(s,a)$ instead of π_1. Fig. 2b and Fig. 2d show that the lower the influence of the attacker is, the harder it is to force a target policy. The default value of ι is 1.0.



(a) Navigation, ϵ (b) Navigation, Influence (c) Inventory, ϵ (d) Inventory, Influence

Figure 3: The effect of hyperparameters λ and δ on the performance of conservative policy search (CPS). The plots correspond to those from Figure 2. The results suggest that λ and δ affect the success rate of CPS in finding a feasible solution. To find a suitable hyperparameters, one can run a meta-search over hyperparameters.

buy $k - s$ items. Other details of this environment are explained in [28]. Note that this is a non-ergodic environment.

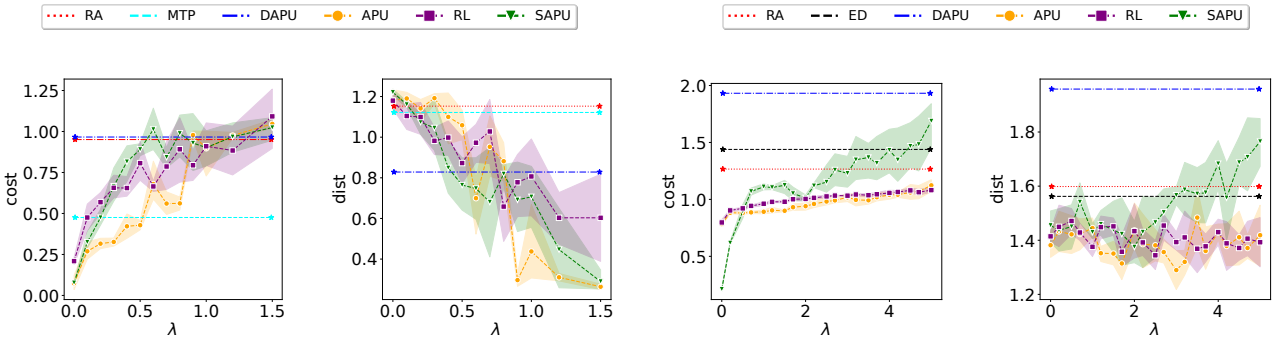
Results. In order to show the efficacy of our conservative policy search algorithm, we consider 4 different algorithms: *Naive* baseline–in the Navigation environment it sets π_1 to always take "right", and in the Inventory Management, π_1 buys 7 items; b) *Conservative PS* (**CPS**)–the policy search algorithm from the previous section that sets $\lambda = 20$, $\delta = 0.01$, and $\delta_\epsilon = 0.1$; c) *Constraints Only PS* (**COPS**)–a modification of CPS that ignores Cost; d) *Unconservative PS* (**UPS**)–a modification of CPS that sets $\delta = 1$.[8]

Fig. 2 compares these algorithmic approaches along two dimensions. We test the effect of the victim's sub-optimality on the cost and the effect the attacker's influence over the victim's peer on the cost. The results show that our algorithmic approach can lead to a significant cost reduction compared to the baselines. These results demonstrate the importance of having: a) a cost-guided search that does not only aim to satisfy the constraint of the optimization problem, but also minimizes the attack cost (CPS outperforms COPS), b) conservative updates that account for the change in occupancy measures when adopting a new solution (CPS outperforms UPS). Fig. 3 shows the effect that the hyperparameters have on the performance of CPS. These results confirm that conservative updates are important, especially in non-ergodic environments (Fig. 3c and Fig. 3d for $\delta = 0.1$ and $\delta = 1.0$), where the performance critically depends on ϵ and ι. We observe similar instabilities for UPS in Fig. 2c and Fig. 2d.

[8]To solve (P1'), we use CVXPY solver in our experiments (see [2, 9]). We provide additional details in [28].

5.2 Experiments for Alternating Policy Updates

Push Environments. We consider two multi-agent RL environments inspired by prior work [30, 45]. We refer to them as Push environments. Both of them have a continuous state space, and are modifications of environments from [45]. In Push environments, the victim is rewarded based on the distance to a given goal location. The target policy stands still if the distance to the goal is within a certain interval, and otherwise moves towards this area. The default policy of the adversary moves towards the goal and stays there. We consider two variants. In 1D Push, the agents can move move left, right, or stand still, on a line segment. In 2D Push, the



(a) 1D Push: Cost, λ (b) 1D Push: Distance, λ (c) 2D Push: Cost, λ (d) 2D Push: Distance, λ

Figure 4: Figures show the test-time performance of each adversary in 1D and 2D Push, for different values of λ. Two empirical performance metrics are plotted, cost and dist, which reflect the cost of the attack and the distance to the attack goal, respectively. More concretely, for a given adversary-victim pair (θ, ϕ), we sample K trajectories τ_k of length T and calculate cost$(\{\tau_k\}) = \frac{1}{T \cdot K} \sum_{s \in \tau_k} \left\| \pi_\theta(s, \cdot) - \pi_1^0(s, \cdot) \right\|_1$ and dist$(\{\tau_k\}) = \frac{1}{T \cdot K} \sum_{s \in \tau_k} \left\| \pi_\phi(s, \cdot) - \pi_2^\dagger(s, \cdot) \right\|_1$. For each λ, we train 5 adversarial policies (using 5 different random seeds). For each adversarial policy, we train 5 victim policies (using 5 different random seeds) against this adversarial policy. The results show the mean and 95% confidence intervals of the obtained data points. In [28], we provide the confidence intervals for baselines whose behavior does not change with λ.

agents have two additional actions, up and down, and are located in a plane. Compared to the 1D version, the reward of the victim has an additional penalty term since the adversary cannot easily "block" the learner from reaching the goal. Note that in 2D Push the target policy is stochastic and encodes the direction to the goal (while minimizing its support). I.e., outside of the annulus where the victim should stay still, the target policy is identified by the vector that connects the victim's position and the closest point of the annulus. We specify other details in [28].

Results. To test the efficacy of our alternating policy updates approach, we consider 4 different algorithms trained with Proximal Policy Optimization (PPO) [41]:[9] a) *Random Adversary* (**RA**) baseline–the adversary takes actions uniformly at random; b) *Move to Target Position* (**MTP**) baseline for 1D Push–the adversary follows a hard-coded policy that moves to the target position; c) *Equal Distance* (**ED**) baseline for 2D Push–the adversary follows a hard-coded policy that keeps the same distance to the victim and goal; d) Alternating Policy Updates (**APU**)–our approach from the previous section, where PPO is used for policy updates and the victim is trained for 5 times as many episodes per epoch as the adversary; e) Random Learner (**RL**)–a modification of APU which fixes the victim's parameters ϕ to random values; f) Symmetric APU (**SAPU**)–a modification of APU in which θ and ϕ are updated in a symmetric manner, i.e., using the same number of episodes; g) Distance-only APU (**DAPU**)–a modification of APU that does not use the imitation learning loss. Fig. 4 compares the test-time performance of these approaches for different values of λ. For larger values of λ, our approach outperforms naive baselines (RA, MTP, ED) both in terms of the attack cost and success; only MTP has a comparable attack costs for large λ in 1D Push. In terms of the attack cost, APU achieves similar performance as its modifications in most cases, while outperforming SAPU in 2D Push. However, in terms of the success rate, it outperforms most of them for large enough λ. One exception is RL, which achieves similar performance in 2D Push. These results suggest that: a) it is important to train (the model of) the victim alongside the attacker (APU vs. RL in 1D Push), b) it is important to have asymmetric update rules that more conservatively update the adversary's policy (APU vs. SAPU), c) it is important to have a cost guided optimization that does not only aim to optimize the attack success (APU vs. DAPU).

[9] We use the implementation from stable-baselines3 [35]. We provide additional training details in [28].

Remark 2. *Alternating Policy Updates can also be applied to Navigation and Inventory Management, and we report the experimental results for these two environments in the full version of the paper [28].*

6 CONCLUSION

In this paper, we studied a novel form of poisoning attacks in reinforcement learning based on adversarial policies. In this attack model, the attacker utilizes the presence of another agent to influence the behavior of a learning agent. We showed that such an implicit form of poisoning differs from the standard environment poisoning attack models in RL. In particular, the implicit attack model appears to be more restrictive in that it is not always feasible, while determining its feasibility is a computationally challenging problem. In contrast, and as argued by prior work, this type of attack may be more practical as the aspects that are controlled by an attacker are expressed through an agency, i.e., the learner's peer. Hence, we believe that our results contribute valuable insights important for understanding trade-offs between different attack models. One of the most interesting future research directions is to consider settings with more than two agents. In such settings, an attacker has to reason about the agents' equilibrium behavior, which brings additional computational challenges. On the other hand, the attacker could potentially use the conflicting goals of the agents in its own favor, which may decrease the cost of the attack.

7 ACKNOWLEDGEMENTS

This research was, in part, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 467367360.

REFERENCES

[1] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. 2019. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep* (2019).

[2] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2018. A rewriting system for convex optimization problems. *Journal of Control and Decision* 5, 1 (2018), 42–60.

[3] Kiarash Banihashem, Adish Singla, Jiarui Gan, and Goran Radanovic. 2022. Admissible policy teaching through reward design. *arXiv preprint arXiv:2201.02185* (2022).

[4] Kiarash Banihashem, Adish Singla, and Goran Radanovic. 2021. Defense against reward poisoning attacks in reinforcement learning. *arXiv preprint arXiv:2102.05776* (2021).

[5] Vahid Behzadan and Arslan Munir. 2017. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344* (2017).

[6] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. 387–402.

[7] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *International Conference on Machine Learning*. 1467–1474.

[8] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[9] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research* 17, 1 (2016), 2909–2913.

[10] Christos Dimitrakakis, David C Parkes, Goran Radanovic, and Paul Tylkin. 2017. Multi-view decision processes: the helper-AI problem. *Advances in neural information processing systems* (2017), 5449–5458.

[11] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*.

[12] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[13] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. 2021. Adversarial policy learning in two-player competitive games. In *International Conference on Machine Learning*. 3910–3919.

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

[15] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. 2020. What is local optimality in nonconvex-nonconcave minimax optimization?. In *International Conference on Machine Learning*. 4880–4889.

[16] Sham Kakade and John Langford. 2002. Approximately Optimal Approximate Reinforcement Learning. In *International Conference on Machine Learning*. 267–274.

[17] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. 2020. TrojDRL: evaluation of backdoor attacks on deep reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6.

[18] Jernej Kos and Dawn Song. 2017. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452* (2017).

[19] Aounon Kumar, Alexander Levine, and Soheil Feizi. 2021. Policy Smoothing for Provably Robust Reinforcement Learning. In *International Conference on Learning Representations*.

[20] Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles L Isbell. 2012. Computing optimal strategies to commit to in stochastic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1380–1386.

[21] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems* (2016), 1885–1893.

[22] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 3756–3762.

[23] Guanlin Liu and Lifeng Lai. 2021. Provably efficient black-box action poisoning attacks against reinforcement learning. *Advances in Neural Information Processing Systems* (2021), 12400–12410.

[24] Yuntao Liu, Yang Xie, and Ankur Srivastava. 2017. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*. 45–48.

[25] Thodoris Lykouris, Max Simchowitz, Alex Slivkins, and Wen Sun. 2021. Corruption-robust exploration in episodic reinforcement learning. In *Conference on Learning Theory*. 3242–3245.

[26] Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. 2019. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems* (2019), 14543–14553.

[27] Shike Mei and Xiaojin Zhu. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2871–2877.

[28] Mohammad Mohammadi, Jonathan Nöther, Debmalya Mandal, Adish Singla, and Goran Radanovic. 2023. Implicit Poisoning Attacks in Two-Agent Reinforcement Learning: Adversarial Policies for Training-Time Attacks. *arXiv preprint arXiv:2302.13851* (2023).

[29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[30] Igor Mordatch and Pieter Abbeel. 2018. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1495–1502.

[31] Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 427–436.

[32] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

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

[34] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[35] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. 2019. Stable baselines3.

[36] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. 2020. A game theoretic framework for model based reinforcement learning. In *International conference on machine learning*. 7953–7963.

[37] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2020. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*. 7974–7984.

[38] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. 2021. Policy teaching in reinforcement learning via environment poisoning attacks. *Journal of Machine Learning Research* 22, 210 (2021), 1–45.

[39] Anshuka Rangi, Haifeng Xu, Long Tran-Thanh, and Massimo Franceschetti. 2022. Understanding the Limits of Poisoning Attacks in Episodic Reinforcement Learning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*. 3394–3400.

[40] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

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

[42] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. 2020. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5883–5891.

[43] Yanchao Sun, Da Huo, and Furong Huang. 2020. Vulnerability-Aware Poisoning Mechanism for Online RL with Unknown Dynamics. In *International Conference on Learning Representations*.

[44] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

[45] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. 2021. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* (2021), 15032–15043.

[46] Yevgeniy Vorobeychik and Satinder Singh. 2012. Computing stackelberg equilibria in discounted stochastic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1478–1484.

[47] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. 2021. BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning. In *30th International Joint Conference on Artificial Intelligence*. 3699–3705.

[48] Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. 2021. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Transactions on Information Forensics and Security* 16 (2021), 4772–4787.

[49] Fan Wu, Linyi Li, Zijian Huang, Yevgeniy Vorobeychik, Ding Zhao, and Bo Li. 2021. CROP: Certifying Robust Policies for Reinforcement Learning through Functional Smoothing. In *International Conference on Learning Representations*.

[50] Fan Wu, Linyi Li, Huan Zhang, Bhavya Kailkhura, Krishnaram Kenthapadi, Ding Zhao, and Bo Li. 2021. COPA: Certifying Robust Policies for Offline Reinforcement Learning against Poisoning Attacks. In *International Conference on Learning Representations*.

[51] Young Wu, Jermey McMahan, Xiaojin Zhu, and Qiaomin Xie. 2022. Reward Poisoning Attacks on Offline Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2206.01888* (2022).

[52] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. 2015. Is feature selection secure against training data poisoning?. In *international conference on machine learning*. 1689–1698.

[53] Han Xiao, Huang Xiao, and Claudia Eckert. 2012. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence*. 870–875.

[54] Zhaoyuan Yang, Naresh Iyer, Johan Reimann, and Nurali Virani. 2019. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972* (2019).

[55] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. 2021. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452* (2021).

[56] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. 2020. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems* (2020), 21024–21037.

[57] Haoqi Zhang and David Parkes. 2008. Value-based policy teaching with active indirect elicitation. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 1*. 208–214.

[58] Haoqi Zhang, David C Parkes, and Yiling Chen. 2009. Policy teaching through reward function learning. In *Proceedings of the 10th ACM conference on Electronic commerce*. 295–304.

[59] Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. 2021. Robust policy gradient against strong data corruption. In *International Conference on Machine Learning*. 12391–12401.

[60] Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. 2022. Corruption-robust offline reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*. 5757–5773.

[61] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. 2020. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*. 11225–11234.