Admissible Policy Teaching through Reward Design

Kiarash Banihashem, Adish Singla, Jiarui Gan, Goran Radanovic

Max Planck Institute for Software Systems
{kbanihas, adishs, jrgan, gradanovic}@mpi-sws.org

Abstract

We study reward design strategies for incentivizing a reinforcement learning agent to adopt a policy from a set of admissible policies. The goal of the reward designer is to modify the underlying reward function cost-efficiently while ensuring that any approximately optimal deterministic policy under the new reward function is admissible and performs well under the original reward function. This problem can be viewed as a dual to the problem of optimal reward poisoning attacks: instead of forcing an agent to adopt a specific policy, the reward designer incentivizes an agent to avoid taking actions that are inadmissible in certain states. Perhaps surprisingly, and in contrast to the problem of optimal reward poisoning attacks, we first show that the reward design problem for admissible policy teaching is computationally challenging, and it is NP-hard to find an approximately optimal reward modification. We then proceed by formulating a surrogate problem whose optimal solution approximates the optimal solution to the reward design problem in our setting, but is more amenable to optimization techniques and analysis. For this surrogate problem, we present characterization results that provide bounds on the value of the optimal solution. Finally, we design a local search algorithm to solve the surrogate problem and showcase its utility using simulation-based experiments.

Introduction

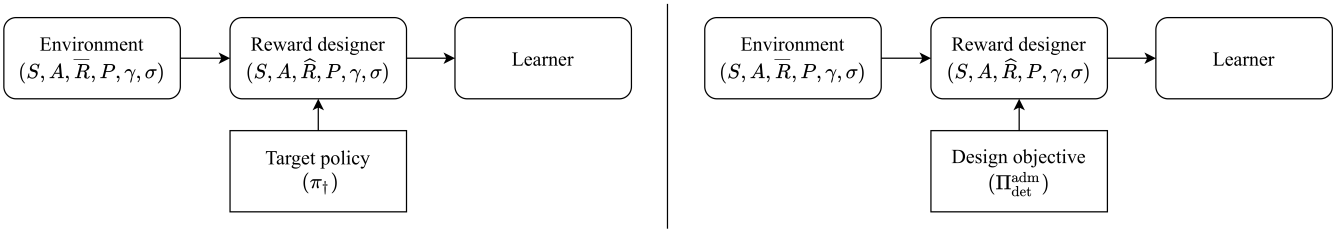
Reinforcement learning (RL) (Sutton and Barto 2018) is a framework for deriving an agent's policy that maximizes its utility in sequential decision making tasks. In the standard formulation, the utility of an agent is defined via its reward function, which determines the decision making task of interest. Reward design plays a critical role in providing sound specifications of the task goals and supporting the agent's learning process (Singh, Lewis, and Barto 2009; Amodei et al. 2016).

There are different perspectives on reward design, which differ in the studied objectives. A notable example of reward design is *reward shaping* (Mataric 1994; Dorigo and Colombetti 1994; Ng, Harada, and Russell 1999) which modifies the reward function in order to accelerate the learning process of an agent. Reward transformations that are similar to or are based on reward shaping are not only used for accelerating learning. For example, reward penalties are often used in safe RL to penalize the agent whenever it violates safety constraints (Tessler, Mankowitz, and Mannor 2018). Similarly, reward penalties can be used in offline RL for ensuring robustness against model uncertainty (Yu et al. 2020), while exploration bonuses can be used as intrinsic motivation for an RL agent to reduce uncertainty (Bellemare et al. 2016).

In this paper, we consider a different perspective on reward design, and study it in the context of (targeted) *policy teaching* and closely related (targeted) *reward poisoning attacks*. In this line of work (Zhang and Parkes 2008; Zhang, Parkes, and Chen 2009; Ma et al. 2019; Rakhsha et al. 2020b,a), the reward designer perturbs the original reward function to influence the choice of policy adopted by an optimal agent. For instance, (Zhang and Parkes 2008; Zhang, Parkes, and Chen 2009) studied policy teaching from a principal's perspective who provides incentives to an agent to influence its policy. In reward poisoning attacks (Ma et al. 2019; Rakhsha et al. 2020b,a), an attacker modifies the reward function with the goal of forcing a specific target policy of interest. Importantly, the reward modifications do not come for free, and the goal in this line of work is to alter the original reward function in a cost-efficient manner. The associated cost can, e.g., model the objective of minimizing additional incentives provided by the principal or ensuring the stealthiness of the attack.

The focus of this paper is on a dual problem to reward poisoning attacks. Instead of forcing a specific target policy, the reward designer's goal is to incentivize an agent to avoid taking actions that are inadmissible in certain states, while ensuring that the agent performs well under the original reward function. As in reward poisoning attacks, the reward designer cares about the cost of modifying the original reward function. Interestingly and perhaps surprisingly, the novel reward design problem leads to a considerably different characterization results, as we show in this paper. We call this problem *admissible policy teaching* since the reward designer aims to maximize the agent's utility w.r.t. the original reward function, but under constraints on admissibility of state-action pairs. These constraints could encode additional knowledge that the reward designer has about the safety and security of executing certain actions. A comparison of our framework with targeted policy teaching can be seen in Figure 1. Our key contributions are:

- We develop a novel optimization framework based on Markov Decision Processes (MDPs) for finding a minimal



(a) Targeted policy teaching **(b) Admissible policy teaching**

Figure 1: Comparison between *targeted* and *admissible* policy teaching. **(a)** depicts the targeted policy teaching framework. In this framework, the reward designer has a *target policy* $\pi_\dagger$ and perturbs the original reward function $\overline{R}$ to the modified reward function $\widehat{R}$ in order to ensure that $\pi_\dagger$ is optimal w.r.t $\widehat{R}$. This would ensure that a learner maximizing its performance w.r.t $\widehat{R}$ would follow $\pi_\dagger$. Since there may be many suitable $\widehat{R}$, the designer typically chooses the one with minimum modification cost. **(b)** shows the admissible policy teaching problem studied in this paper. In contrast to a single targeted policy, the designer has *a set of admissible policies* $\Pi^{\text{adm}}_{\text{det}}$. The goal of the designer is to modify $\overline{R}$ to ensure that the learner chooses an admissible policy. As before, the designer wants to keep modification cost at a minimum. Unlike the previous setting, we are also interested in the performance of the learner w.r.t the original reward function $\overline{R}$.

reward modifications which ensure that an optimal agent adopts a well-performing admissible policy. which ensure that the optimal RL agent adopts an admissible policy (See subsection *Problem Formulation*).

- We show that finding an optimal solution to the reward design problem for admissible policy teaching is computationally challenging, in particular, that it is NP-hard to find a solution that approximates the optimal solution.
- We provide characterization results for a surrogate problem whose optimal solution approximates the optimal solution to our reward design problem. For a specific class of MDPs, which we call *special* MDPs, we present an exact characterization of the optimal solution. For *general* MDPs, we provide bounds on the optimal solution value.
- We design a local search algorithm for solving the surrogate problem, and demonstrate its efficacy using simulation-based experiments.

Related Work

Reward design. A considerable number of works is related to designing reward functions that improve an agent's learning procedures. The optimal reward problem focuses on finding a reward function that can support computationally bounded agents (Sorg, Singh, and Lewis 2010; Sorg, Lewis, and Singh 2010). Reward shaping (Mataric 1994; Dorigo and Colombetti 1994), and in particular, potential-based reward shaping (Ng, Harada, and Russell 1999) and its extensions (e.g., (Devlin and Kudenko 2012; Grześ 2017; Zou et al. 2019)) densify the reward function so that the agent receives more immediate signals about its performance, and hence learns faster. As already mentioned, similar reward transformations, such as reward penalties or bonuses, are often used for reducing uncertainty or for ensuring safety constraints (Bellemare et al. 2016; Yu et al. 2020; Tessler, Mankowitz, and Mannor 2018). Related to safe and secure RL are works that study reward specification problem and negative side affects of reward misspecification (Amodei et al. 2016; Hadfield-Menell et al. 2017). The key difference between the above papers and our work is that we focus on policy teaching rather than on an agent's learning procedures.

Teaching and steering. As already explained, our work relates to prior work on policy teaching and targeted reward poisoning attacks (Zhang and Parkes 2008; Zhang, Parkes, and Chen 2009; Ma et al. 2019; Huang and Zhu 2019; Rakhsha et al. 2020b,a; Zhang et al. 2020b; Sun, Huo, and Huang 2021). Another line of related work is on designing steering strategies. For example, (Nikolaidis et al. 2017; Dimitrakakis et al. 2017; Radanovic et al. 2019) consider two-agent collaborative settings where a dominant agent can exert influence on the other agent, and the goal is to design a policy for the dominant agent that accounts for the imperfections of the other agent. Similar support mechanisms based on providing advice or helpful interventions have been studied by (Amir et al. 2016; Omidshafiei et al. 2019; Tylkin, Radanovic, and Parkes 2021). In contrast, we consider steering strategies based on reward design. When viewed as a framework for supporting an agent's decision making, this paper is also related to works on designing agents that are robust against adversaries (Pinto et al. 2017; Fischer et al. 2019; Lykouris et al. 2019; Zhang et al. 2020a, 2021a,b; Banihashem, Singla, and Radanovic 2021). These works focus on agent design and are complementary to our work on reward design.

Problem Setup

In this section we formally describe our problem setup.

Environment

The environment in our setting is described by a discrete-time Markov Decision Process (MDP) $M = (S, A, R, P, \gamma, \sigma)$, where S and A are the discrete finite state and action spaces respectively[1], $R : S \times A \to \mathbb{R}$ is a reward function, P :

[1]This setting can encode the case where states have different number of actions (e.g., by adding actions to the states with smaller number of actions and setting the reward of newly added state-action pairs to $-\infty$).

$S \times A \times S \to [0, 1]$ specifies the transition dynamics with $P(s, a, s')$ denoting the probability of transitioning to state s' from state s by taking action a, $\gamma \in [0, 1)$ is the discounted factor, and σ is the initial state distribution. A deterministic policy π is a mapping from states to actions, i.e., $\pi : S \to A$, and the set of all deterministic policies is denoted by Π_{det}.

Next, we define standard quantities in this MDP, which will be important for our analysis. First, we define the *score* of a policy π as the total expected return scaled by $1 - \gamma$, i.e., $\rho^{\pi,R} = \mathbb{E}\left[(1-\gamma)\sum_{t=1}^{\infty}\gamma^{t-1}R(s_t, a_t)|\pi, \sigma\right]$. Here states s_t and actions a_t are obtained by executing policy π starting from state s_1, which is sampled from the initial state distribution σ. Score $\rho^{\pi,R}$ can be obtained through state occupancy measure μ^{π} by using the equation $\rho^{\pi,R} = \sum_s \mu^{\pi}(s) \cdot R(s, \pi(s))$. Here, μ^{π} is the expected discounted state visitation frequency when π is executed, given by $\mu^{\pi}(s) = \mathbb{E}\left[(1-\gamma)\sum_{t=1}^{\infty}\gamma^{t-1}\mathbb{1}\left[s_t = s\right]|\pi, \sigma\right]$. Note that $\mu^{\pi}(s)$ can be equal to 0 for some states. Furthermore, we define $\mu^{\pi}_{\text{min}} = \min_{s|\mu^{\pi}(s)>0} \mu^{\pi}(s)$—the minimum always exists due to the finite state and action spaces. Similarly, we denote by μ_{min} the minimal value of μ^{π}_{min} across all deterministic policies, i.e., $\mu_{\text{min}} = \min_{\pi \in \Pi_{\text{det}}} \mu^{\pi}_{\text{min}}$.

We define the state-action value function, or Q values as $Q^{\pi,R}(s, a) = \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^{t-1}R(s_t, a_t)|\pi, s_1 = s, a_1 = a\right]$, where states s_t and actions a_t are obtained by executing policy π starting from state $s_1 = s$ in which action $a_1 = a$ is taken. State-action values $Q(s, a)$ relate to score ρ via the equation $\rho^{\pi,R} = \mathbb{E}_{s\sim\sigma}\left[(1-\gamma)\cdot Q^{\pi,R}(s, \pi(s))\right]$, where the expectation is taken over possible starting states.

Agent and Reward Functions

We consider a reinforcement learning agent whose behavior is specified by a deterministic policy π, derived offline using an MDP model given to the agent. We assume that the agent selects a deterministic policy that (approximately) maximizes the agent's expected utility under the MDP model specified by a reward designer. In other words, given an access to the MDP $M = (S, A, R, P, \gamma, \sigma)$, the agent chooses a policy from the set $\text{OPT}^{\epsilon}_{\text{det}}(R) = \{\pi \in \Pi_{\text{det}} : \rho^{\pi,R} > \max_{\pi' \in \Pi_{\text{det}}} \rho^{\pi',R} - \epsilon\}$, where ϵ is a strictly positive number. It is important to note that the MDP model given to the agent might be different from the true MDP model of the environment. In this paper, we focus on the case when only the reward functions of these two MDPs (possibly) differ.

Therefore, in our notation, we differentiate the reward function that the reward designer specifies to the agent, denoting it by $\widehat{R}$, from the original reward function of the environment, denoting it by $\overline{R}$. A generic reward function is denoted by R, and is often used as a variable in our optimization problems. We also denote by π^* a deterministic policy that is optimal with respect to $\overline{R}$ for any starting state, i.e., $Q^{\pi^*,\overline{R}}(s, \pi^*(s)) = \max_{\pi \in \Pi_{\text{det}}} Q^{\pi,\overline{R}}(s, \pi(s))$ for all states s.

Reward Designer and Problem Formulation

We take the perspective of a reward designer whose goal is to design a reward function, $\widehat{R}$, such that the agent adopts a policy from a class of admissible deterministic policies $\Pi^{\text{adm}}_{\text{det}} \subseteq \Pi_{\text{det}}$. Ideally, the new reward function $\widehat{R}$ would be close to the original reward function $\overline{R}$, thus reducing the cost of the reward design. At the same time, the adopted policy should perform well under the original reward function $\overline{R}$, since this is the performance that the reward designer wants to optimize and represents the objective of the underlying task. As considered in related works (Ma et al. 2019; Rakhsha et al. 2020b,a), we measure the *cost* of the reward design by L_2 distance between the designed $\widehat{R}$ and the original reward function $\overline{R}$. Moreover, we measure the agent's performance with the score $\rho^{\pi,\overline{R}}$, where the agent's policy π is obtained w.r.t. the designed reward function $\widehat{R}$. Given the model of the agent discussed in the previous subsection, and assuming the worst-case scenario (w.r.t. the tie-breaking in the policy selection), the following optimization problem specifies the reward design problem for *admissible policy teaching* (APT):

$$\min_{R} \max_{\pi} \left\|\overline{R} - R\right\|_2 - \lambda \cdot \rho^{\pi,\overline{R}} \qquad \text{(P1-APT)}$$
$$\text{s.t.} \quad \text{OPT}^{\epsilon}_{\text{det}}(R) \subseteq \Pi^{\text{adm}}_{\text{det}}$$
$$\pi \in \text{OPT}^{\epsilon}_{\text{det}}(R),$$

where $\lambda \geq 0$ is a trade-off factor.[2] While in this problem formulation $\Pi^{\text{adm}}_{\text{det}}$ can be any set of policies, we will primarily focus on admissible policies that can be described by a set of admissible actions per state. More concretely, we define sets of admissible actions per state denoted by $A^{\text{adm}}_s \subseteq A$. Given these sets A^{adm}_s, the set of admissible policies will be identified as $\Pi^{\text{adm}}_{\text{det}} = \{\pi | \pi(s) \in A^{\text{adm}}_s \vee \mu^{\pi}(s) = 0 \text{ for } s \in S\}$.[3] In other words, these policies must take admissible actions for states that have non-zero state occupancy measure.

We conclude this section by validating the soundness of the optimization problem (P1-APT). The following proposition shows that the optimal solution to the optimization problem (P1-APT) is always attainable.

Proposition 1. *If $\Pi^{\text{adm}}_{\text{det}}$ is not empty, there always exists an optimal solution to the optimization problem* (P1-APT).

In the following sections, we analyze computational aspects of this optimization problem, showing that it is intractable in general and providing characterization results that bound the value of solutions. The proofs are provided in the full version of the paper (Banihashem et al. 2022).

Computational Challenges

We start by analyzing computational challenges behind the optimization problem (P1-APT). To provide some intuition, let us first analyze a special case of (P1-APT) where $\lambda = 0$, which reduces to the following optimization problem:

$$\min_{R} \left\|\overline{R} - R\right\|_2 \qquad \text{(P2-APT}_{\lambda=0}\text{)}$$
$$\text{s.t.} \quad \text{OPT}^{\epsilon}_{\text{det}}(R) \subseteq \Pi^{\text{adm}}_{\text{det}}.$$

This special case of the optimization problem with $\lambda = 0$ is a generalization of the reward poisoning attack from

[2] Since we are taking a worst-case view w.r.t tie-breaking among policies, we maximize over π.

[3] In practice, we can instead put the constraint that $\mu^{\pi}(s)$ is greater than or equal to some threshold. For small enough threshold, our characterization results qualitatively remain the same.

(Rakhsha et al. 2020b,a). In fact, the reward poisoning attack of (Rakhsha et al. 2020a) can be written as

$$\min_{R} \left\|\overline{R} - R\right\|_2 \qquad \text{(P3-ATK)}$$
$$\text{s.t.} \quad \text{OPT}^{\epsilon}_{\text{det}}(R) \subseteq \{\pi | \pi(s) = \pi_\dagger(s) \text{ if } \mu^{\pi_\dagger}(s) > 0\},$$

where $\pi_\dagger$ is the target policy that the attacker wants to force. However, while (P3-ATK) is tractable in the setting of (Rakhsha et al. 2020a), the same is not true for (P2-APT$_{\lambda=0}$); see Remark 1. Intuitively, the difficulty of solving the optimization problem (P2-APT$_{\lambda=0}$) lies in the fact that the policy set $\Pi^{\text{adm}}_{\text{det}}$ (in the constraints of (P2-APT$_{\lambda=0}$)) can contain exponentially many policies. Since the optimization problem (P2-APT$_{\lambda=0}$) is a specific instance of the optimization problem (P1-APT), the latter problem is also computationally intractable. We formalize this result in the following theorem.

Theorem 1. *For any constant $p \in (0, 1)$, it is NP-hard to distinguish between instances of* (P2-APT$_{\lambda=0}$) *that have optimal values at most ξ and instances that have optimal values larger than $\xi \cdot \sqrt{(|S| \cdot |A|)^{1-p}}$. The result holds even when the parameters ϵ and γ in* (P2-APT$_{\lambda=0}$) *are fixed to arbitrary values subject to $\epsilon > 0$ and $\gamma \in (0, 1)$.*

The proof of the theorem is based on a classical NP-complete problem called EXACT-3-SET-COVER (X3C) (Karp 1972; Garey and Johnson 1979). The result implies that it is unlikely (assuming that P = NP is unlikely) that there exists a polynomial-time algorithm that always outputs an approximate solution whose cost is at most $\sqrt{(|S| \cdot |A|)^{1-p}}$ times that of the optimal solution for some $p > 0$.

We proceed by introducing a surrogate problem (P4-APT), which is more amenable to optimization techniques and analysis since the focus is put on optimizing over policies rather than reward functions. In particular, the optimization problem takes the following form:

$$\min_{\pi \in \Pi^{\text{adm}}_{\text{det}}, R} \left\|\overline{R} - R\right\|_2 - \lambda \cdot \rho^{\pi, \overline{R}} \qquad \text{(P4-APT)}$$
$$\text{s.t.} \quad \text{OPT}^{\epsilon}_{\text{det}}(R) \subseteq \{\pi' | \pi'(s) = \pi(s) \text{ if } \mu^{\pi}(s) > 0\}.$$

Note that (P4-APT) differs from (P3-ATK) in that it optimizes over all admissible policies, and it includes performance considerations in its objective. The result in Theorem 1 extends to this case as well, so the main computational challenge remains the same.

The following proposition shows that the solution to the surrogate problem (P4-APT) is an approximate solution to the optimization problem (P1-APT), with an additive bound. More precisely:

Proposition 2. *Let $\widehat{R}_1$ and $\widehat{R}_2$ be the optimal solutions to* (P1-APT) *and* (P4-APT) *respectively and let $l(R)$ be a function that outputs the objective of the optimization problem* (P1-APT)*, i.e.,*

$$l(R) = \max_{\pi \in \text{OPT}^{\epsilon}_{\text{det}}(R)} \left\|\overline{R} - R\right\|_2 - \lambda \rho^{\pi, \overline{R}}. \qquad (1)$$

Then $\widehat{R}_2$ satisfies the constraints of (P1-APT)*, i.e,* $\text{OPT}^{\epsilon}_{\text{det}}(\widehat{R}_2) \subseteq \Pi^{\text{adm}}_{\text{det}}$*, and*

$$l(\widehat{R}_1) \leq l(\widehat{R}_2) \leq l(\widehat{R}_1) + \frac{\epsilon}{\mu_{\min}} \cdot \sqrt{|S| \cdot |A|}.$$

Due to this result, in the following sections, we focus on the optimization problem (P4-APT), and provide characterization for it. Using Proposition 2 we can obtain analogous results for the optimization problem (P1-APT).

Remark 1. *The optimization problem* (P3-ATK) *is a strictly more general version of the optimization problem studied in (Rakhsha et al. 2020a) since* (P3-ATK) *does not require $\mu^{\pi}(s) > 0$ for all π and s. This fact also implies that the algorithmic approach presented in (Rakhsha et al. 2020a) is not applicable in our case. We provide an efficient algorithm for finding an approximate solution to* (P3-ATK) *with provable guarantees in the full version of the paper (Banihashem et al. 2022).*

Characterization Results for Special MDPs

In this section, we consider a family of MDPs where an agent's actions do not influence transition dynamics, or more precisely, all the actions influence transition probabilities in the same way. In other words, the transition probabilities satisfy $P(s, a, s') = P(s, a', s')$, for all s, a, a', s'. We call this family of MDPs *special* MDPs, in contrast to *general* MDPs that are studied in the next section. Since an agent's actions do not influence the future, the agent can reason myopically when deciding on its policy. Therefore, the reward designer can also treat each state separately when reasoning about the cost of the reward design. Importantly, the hardness result from the previous section does not apply for this instance of our setting, so we can efficiently solve the optimization problems (P1-APT) and (P4-APT).

Forcing Myopic Policies

We first analyze the cost of forcing a target policy $\pi_\dagger$ in special MDPs. The following lemma plays a critical role in our analysis.

Lemma 1. *Consider a* special *MDP with reward function $\overline{R}$, and let $\pi^*_{\text{adm}}(s) = \arg\max_{a \in \Pi^{\text{adm}}_{\text{det}}} \overline{R}(s, a)$. Then the cost of the optimal solution to the optimization problem* (P3-ATK) *with $\pi_\dagger = \pi^*_{\text{adm}}$ is less than or equal to the cost of the optimal solution to the optimization problem* (P3-ATK) *with $\pi_\dagger = \pi$ for any $\pi \in \Pi^{\text{adm}}_{\text{det}}$.*

In other words, Lemma 1 states that in special MDPs it is easier to force policies that are myopically optimal (i.e., optimize w.r.t. the immediate reward) than any other policy in the admissible set $\Pi^{\text{adm}}_{\text{det}}$. This property is important for the optimization problem (P4-APT) since its objective includes the cost of forcing an admissible policy.

Analysis of the Reward Design Problem

We now turn to the reward design problem (P4-APT) and provide characterization results for its optimal solution. Before stating the result, we note that for special MDPs $\mu^{\pi}(s)$ is independent of policy π, so we denote it by $\mu(s)$.

Theorem 2. *Consider a special MDP with reward function*

$\overline{R}$. *Define* $\widehat{R}(s,a) = \overline{R}(s,a)$ *for* $\mu(s) = 0$ *and otherwise*

$$\widehat{R}(s,a) = \begin{cases} x_s + \frac{\epsilon}{\mu(s)} & \text{if } a = \pi^*_{\text{adm}}(s) \\ x_s & \text{if } a \neq \pi^*_{\text{adm}}(s) \wedge \overline{R}(s,a) \geq x_s \\ \overline{R}(s,a) & \text{otherwise} \end{cases},$$

where x_s *is the solution to the equation*

$$\sum_{a \neq \pi^*_{\text{adm}}(s)} \left[\overline{R}(s,a) - x\right]^+ = x - \overline{R}(s, \pi^*_{\text{adm}}(s)) + \frac{\epsilon}{\mu(s)}.$$

Then, $(\pi^*_{\text{adm}}, \widehat{R})$ *is an optimal solution to* (P4-APT).

Theorem 2 provides an interpretable solution to (P4-APT): for each state-action pair $(s, a \neq \pi^*_{\text{adm}}(s))$ we reduce the corresponding reward $\overline{R}(s,a)$ if it exceeds a state dependent threshold. Likewise, we increase the rewards $\overline{R}(s, \pi^*_{\text{adm}}(s))$.

Characterization Results for General MDPs

In this section, we extend the characterization results from the previous section to general MDPs for which transition probabilities can depend on actions. In contrast to the previous section, the computational complexity result from Theorem 1 showcase the challenge of deriving characterization results for general MDPs that specify the form of an optimal solution. We instead focus on bounding the value of an optimal solution to (P4-APT) relative to the score of an optimal policy π^*. More specifically, we define the relative value Φ as

$$\Phi = \underbrace{\left\| \overline{R} - \widehat{R}_2 \right\|_2}_{\text{cost}} + \lambda \cdot \underbrace{[\rho^{\pi^*, \overline{R}} - \rho^{\pi_2, \overline{R}}]}_{\text{performance reduction}},$$

where $(\pi_2, \widehat{R}_2)$ is an optimal solution to the optimization problem (P4-APT). Intuitively, Φ expresses the optimal value of (P4-APT) in terms of the cost of the reward design and the agent's performance reduction.

The characterization results in this section provide bounds on Φ and are obtained by analyzing two specific policies: an optimal admissible policy $\pi^*_{\text{adm}} \in \arg\max_{\pi \in \Pi^{\text{adm}}_{\text{det}}} \rho^{\pi, \overline{R}}$ that optimizes for performance ρ, and a min-cost policy $\pi_{\min_c}$ that minimizes the cost of the reward design and is a solution to the optimization problem (P4-APT) with $\lambda = 0$. As we show in the next two subsections, bounding the cost of forcing π^*_{adm} and $\pi_{\min_c}$ can be used for deriving bounds on Φ. Next, we utilize the insights of the characterization results to devise a local search algorithm for solving the reward design problem, whose utility we showcase using experiments.

Perspective 1: Optimal Admissible Policy

Let us consider an optimal admissible policy $\pi^*_{\text{adm}} \in \arg\max_{\pi \in \Pi^{\text{adm}}_{\text{det}}} \rho^{\pi, \overline{R}}$. Following the approach presented in the previous section, we can design $\widehat{R}$ by (approximately) solving the optimization problem (P3-ATK) (see Remark 1) with the target policy $\pi_\dagger = \pi^*_{\text{adm}}$. While this approach does not yield an optimal solution for general MDPs, the cost of its solution can be bounded by a quantity that depends on the gap between the scores of an optimal policy π^* and an optimal admissible policy π^*_{adm}.

In particular, for any policy π we can define the performance gap as $\Delta^\pi_\rho = \rho^{\pi^*, \overline{R}} - \rho^{\pi, \overline{R}}$. As we will show, the cost of forcing policy π can be upper and lower bounded by terms that linearly depend on Δ^π_ρ. Consequently, this means that one can also bound Φ with terms that linearly depend on $\Delta_\rho = \min_\pi \Delta^\pi_\rho$, which is nothing else but the performance gap of $\pi = \pi^*_{\text{adm}}$. Formally, we obtain the following result.

Theorem 3. *The relative value* Φ *is bounded by*

$$\alpha_\rho \cdot \Delta_\rho \leq \Phi \leq \beta_\rho \cdot \Delta_\rho + \frac{\epsilon}{\mu_{\min}} \cdot \sqrt{|S| \cdot |A|},$$

where $\alpha_\rho = \left(\lambda + \frac{1-\gamma}{2}\right)$ *and* $\beta_\rho = \left(\lambda + \frac{1}{\mu_{\min}}\right)$.

Note that the bounds in the theorem can be efficiently computed from the MDP parameters. Moreover, the reward design approach based on forcing π^*_{adm} yields a solution to (P4-APT) whose value (relative to the score of π^*) satisfies the bounds in Theorem 3. We use this approach as a baseline.

Perspective 2: Min-Cost Admissible Policy

We now take a different perspective, and compare Φ to the cost of the reward design obtained by forcing the min-cost policy $\pi_{\min_c}$. Ideally, we would relate Φ to the the smallest cost that the reward designer can achieve. However, this cost is not efficiently computable (due to Theorem 1), making such a bound uninformative.

Instead, we consider Q values: as Ma et al. (2019) showed, the cost of forcing a policy can be upper and lower bounded by a quantity that depends on Q values. We introduce a similar quantity, denoted by Δ_Q and defined as

$$\Delta_Q = \min_{\pi \in \Pi^{\text{adm}}_{\text{det}}} \max_{s \in S^\pi_{\text{pos}}} \left(Q^{\pi^*, \overline{R}}(s, \pi^*(s)) - Q^{\pi^*, \overline{R}}(s, \pi(s)) \right),$$

where $S^\pi_{\text{pos}} = \{s | \mu^\pi(s) > 0\}$ contains the set of states that policy π visits with strictly positive probability. In the full version of the paper, we present an algorithm called QGREEDY that efficiently computes Δ_Q. The QGREEDY algorithm also outputs a policy π_{qg} that solves the corresponding min-max optimization problem. By approximately solving the optimization problem (P3-ATK) with $\pi_\dagger = \pi_{\text{qg}}$, we can obtain reward function $\widehat{R}$ as a solution to the reward design problem. We use this approach as a baseline in our experiments, and also for deriving the bounds on Φ relative to Δ_Q provided in the following theorem.

Theorem 4. *The relative value* Φ *is bounded by*

$$\alpha_Q \cdot \Delta_Q \leq \Phi \leq \beta_Q \cdot \Delta_Q + \frac{\epsilon}{\mu_{\min}} \sqrt{|S| \cdot |A|},$$

where $\alpha_Q = \left(\lambda \cdot \mu_{\min} + \frac{1-\gamma}{2}\right)$ *and* $\beta_Q = \left(\lambda + \sqrt{|S|}\right)$.

The bounds in Theorem 4 are obtained by analyzing the cost of forcing policy π_{qg} and the score difference $(\rho^{\pi^*, \overline{R}} - \rho^{\pi_{\text{qg}}, \overline{R}})$. The well-known relationship between $\rho^{\pi, R} - \rho^{\pi', R}$ and $Q^{\pi, R}$ for any two policies π, π' (e.g., see (Schulman et al. 2015)) relates the score difference $(\rho^{\pi^*, \overline{R}} - \rho^{\pi_{\text{qg}}, \overline{R}})$ to $Q^{\pi^*, \overline{R}}$, so the crux of the analysis lies in upper and lower bounding the cost of forcing policy π_{qg}. To obtain the corresponding

bounds, we utilize similar proof techniques to those presented in (Ma et al. 2019) (see Theorem 2 in their paper). Since the analysis focuses on π_{qg}, the approach based on forcing π_{qg} outputs a solution to (P4-APT) whose value (relative to the score of π^*) satisfies the bounds in Theorem 4.

Practical Algorithm: CONSTRAIN&OPTIMIZE

In the previous two subsections, we discussed characterization results for the relative value Φ by considering two specific cases: optimizing performance and minimizing cost. We now utilize the insights from the previous two subsections to derive a practical algorithm for solving (P4-APT). The algorithm is depicted in Algorithm 1, and it searches for a well performing policy with a small cost of forcing it.

Algorithm 1: CONSTRAIN&OPTIMIZE

Input: MDP $\overline{M}$, admissible set $\Pi^{\text{adm}}_{\text{det}}$
Output: Reward function $\widehat{R}$, policy π_{co}

1: $\pi_{\text{co}} \leftarrow \arg\max_{\pi \in \Pi^{\text{adm}}_{\text{det}}} \rho^{\pi,\overline{R}}$
2: $cost_{\text{co}} \leftarrow$ approx. solve (P3-ATK) with $\pi_\dagger = \pi_{\text{co}}$
3: set $\Pi_{\text{co}} \leftarrow \Pi^{\text{adm}}_{\text{det}}$
4: **repeat**
5: output$_{\text{new}}$ $\leftarrow$false
6: **for** s in *priority-queue*$(S^{\pi_{\text{co}}}_{\text{pos}})$ **do**
7: $\Pi' \leftarrow \{\pi | \pi \in \Pi_{\text{co}} \wedge \pi(s) \neq \pi_{\text{co}}(s)\}$
8: $\pi' \leftarrow \arg\max_{\pi \in \Pi'} \rho^{\pi,\overline{R}}$
9: $cost' \leftarrow$ approx. solve (P3-ATK) with $\pi_\dagger = \pi'$
10: **if** $cost' - \lambda\rho^{\pi',\overline{R}} < cost_{\text{co}} - \lambda\rho^{\pi_{\text{co}},\overline{R}}$ **then**
11: set $\pi_{\text{co}} \leftarrow \pi'$, $cost_{\text{co}} \leftarrow cost'$, and $\Pi_{\text{co}} \leftarrow \Pi'$
12: set output$_{\text{new}}$ $\leftarrow$true and **break**
13: **end if**
14: **end for**
15: **until** output$_{\text{new}}$ = false
16: $\widehat{R} \leftarrow$ approx. solve (P3-ATK) with $\pi_\dagger = \pi_{\text{co}}$

The main blocks of the algorithm are as follows:

- **Initialization (lines 1-2).** The algorithm selects π^*_{adm} as its initial solution, i.e., $\pi_{\text{co}} = \pi^*_{\text{adm}}$, and evaluates its cost by approximately solving (P3-ATK).
- **Local search (lines 4-15).** Since the initial policy π_{co} is not necessarily cost effective, the algorithm proceeds with a local search in order to find a policy that has a lower value of the objective of (P4-APT). In each iteration of the local search procedure, it iterates over all states that are visited by the current π_{co} (i.e., $S^{\pi_{\text{co}}}_{\text{pos}}$), prioritizing those that have a higher value of $Q^{\pi^*,\overline{R}}(s,\pi^*(s)) - Q^{\pi^*,\overline{R}}(s,\pi_{\text{co}}(s))$ (obtained via *priority-queue*). The intuition behind this prioritization is that this Q value difference is reflective of the cost of forcing action $\pi_{\text{co}}(s)$ (as can be seen by setting $\lambda = 0$ in the upper bound of Theorem 4). Hence, deviations from π_{co} that are considered first are deviations from those actions that are expected to induce high cost.
- **Evaluating a neighbor solution (lines 7-12).** Each visited state s defines a neighbor solution in the local search. To find this neighbor, the algorithm first defines a new admissible set of policies Π' (line 7), obtained from the current one by making action $\pi_{\text{co}}(s)$ inadmissible. The neighbor solution is then identified as $\pi' \in \arg\max_{\pi \in \Pi'} \rho^{\pi,\overline{R}}$ (line 8) and the costs of forcing it is calculated by approximately solving (P3-ATK) with $\pi_\dagger = \pi'$ (line 9). If π' yields a better value of the objective of (P4-APT) than π_{co} does (line 10), we have a new candidate policy and the set of admissible policies is updated to Π' (lines 11-12).
- **Returning solution (line 16).** Once the local search finishes, the algorithm outputs π_{co} and the reward function $\widehat{R}$ found by approximately solving (P3-ATK) with $\pi_\dagger = \pi_{\text{co}}$.

In each iteration of the local search (lines 5-14), the algorithm either finds a new candidate (output$_{\text{new}}$=true) or the search finishes with that iteration (output$_{\text{new}}$=false). Notice that the former cannot go indefinitely since the admissible set reduces between two iterations. This means that the algorithm is guaranteed to halt. Since the local search only accepts new policy candidates if they are better than the current π_{co} (line 10), the output of CONSTRAIN&OPTIMIZE is guaranteed to be better than forcing an optimal admissible policy (i.e., approx. solving (P3-ATK) with $\pi_\dagger = \pi^*_{\text{adm}}$).

Numerical Simulations

We analyze the efficacy of CONSTRAIN&OPTIMIZE in solving the optimization problem (P4-APT) and the policy it incentivizes, π_{co}. We consider three baselines, all based on approximately solving the optimization problem (P3-ATK), but with different target policies $\pi_\dagger$: a) forcing an optimal policy, i.e., $\pi_\dagger = \pi^*$, b) forcing an optimal admissible policy, i.e., $\pi_\dagger = \pi^*_{\text{adm}}$, c) forcing the policy obtained by the QGREEDY algorithm, i.e., $\pi_\dagger = \pi_{\text{qg}}$.[4] We compare these approaches by measuring their performance w.r.t. the objective value of (P4-APT)—lower value is better. By default, we set the parameters $\gamma = 0.9$, $\lambda = 1.0$ and $\epsilon = 0.1$.[5]

Experimental Testbeds

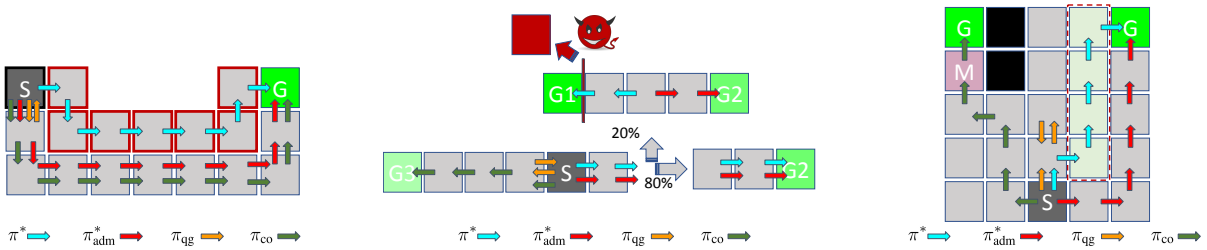
As an experimental testbed, we consider three simple navigation environments, shown in Figure 2. Each environment contains a start state S and goal state(s) G. Unless otherwise specified, in a non-goal state, the agent can navigate in the left, right, down, and up directions, provided there is a state in that direction. In goal states, the agent has a single action which transports it back to the start state.

Cliff environment (Figure 2a). This environment depicts a scenario where some of the states are potentially unsafe due to model uncertainty that the reward designer is aware of. More concretely, the states with "red" cell boundaries in Figure 2a represent the edges of a cliff and are unsafe; as such, all actions leading to these states are considered inadmissible. In this environment, the action in the goal state yields a reward $\overline{R}$ of 20 while all other actions yield a reward $\overline{R}$ of -1.

Action hacking environment (Figure 2b). This environment depicts a scenario when some of the agent's actions

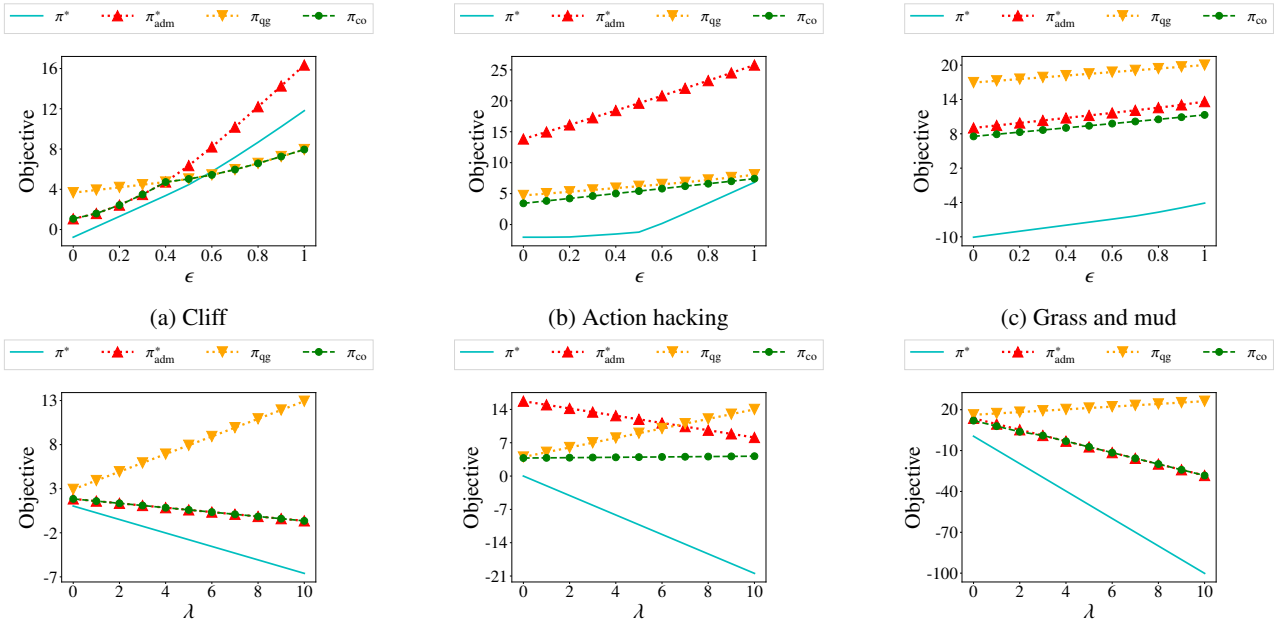
[4] π^* might not be admissible; also, even though π^* is an optimal policy, there is still a cost of forcing it to create the required gap.

[5] For details regarding the experiments and code, please refer to the full version of our paper (Banihashem et al. 2022).



(a) Cliff environment (b) Action hacking environment (c) Grass and mud environment

Figure 2: **Qualitative assessment.** (**a**) π_{co} is the same as π^*_{adm}, taking the longer path to the goal state than π^* in order to avoid the cliff edge, while π_{qg} simply alternates between the starting state and the state below it; (**b**) π_{co} takes the path to the goal state G3—this behavior is less costly to incentivize than navigating to G2 since in the former case the reward designer mainly needs to compensate for the difference in rewards between G3 and G2 (G1 is reachable only 20% of the time from S); (**c**) π^* takes a path through the grass states, π^*_{adm} takes a different but admissible path towards the same goal, while π_{co} navigates the agent towards the other goal (with a lower cumulative reward, but the corresponding policy is less costly to force). **Quantitative assessment.** The objective values of (P4-APT) for the approaches based on forcing π^*, π^*_{adm}, π_{qg}, and π_{co} are respectively: (**a**) 0.27, 1.59, 3.93, and 1.59; (**b**) -2.04, 14.96, 5.00, and 3.82; and (**c**) -9.54, 9.46, 17.26, and 7.92.



(a) Cliff (b) Action hacking (c) Grass and mud

(d) Cliff (e) Action hacking (f) Grass and mud

Figure 3: Effect of λ and ϵ on the objective value of (P4-APT) for different approaches. **(a, b, c)** vary ϵ with $\lambda = 1.0$; **(d, e, f)** vary λ with $\epsilon = 0.1$. Lower values on the y-axis denote better performance. Note that π^* is not an admissible policy in these environments; importantly, the objective value for π_{co} is consistently better (lower) than for π^*_{adm} and π_{qg}, highlighting its efficacy.

could be hacked at the deployment phase, taking the agent to a bad state. The reward designer is aware of this potential hacking and seeks to design a reward function so that these actions are inadmissible. More concretely, the action leading the agent to G1 is considered inadmissible. In this environment, we consider the reward function $\overline{R}$ and dynamics P as follows. Whenever an agent reaches any of the goal states (G1, G2, or G3), it has a single action that transports it back to the starting state and yields a reward of 50, 10, and 5 for G1, G2, and G3 respectively. In all other states, the agent can take either the left or right action and navigate in the corresponding direction, receiving a reward of -1. With a small probability of 0.20, taking the right action in the state next to S results in the agent moving up instead of right.

Grass and mud environment (Figure 2c). This environment depicts a policy teaching scenario where the reward designer and the agent do not have perfectly aligned preferences (e.g., the agent prefers to walk on grass, which the reward designer wants to preserve). The reward designer wants to incentivize the agent not to step on the grass states,

so actions leading to them are considered inadmissible. In addition to the starting state and two goal states, the environment contains four grass states, one mud state, and 16 ordinary states, shown by "light green", "light pink", and "light gray" cells respectively. The "black" cells in the figure represent inaccessible blocks. The reward function $\overline{R}$ is as follows: the action in the goal states yields a reward of 50; the actions in the grass and mud states yield rewards of 10 and -2 respectively; all other actions have a reward of -1.

Results

Figure 2 provides an assessment of different approaches by visualizing the agent's policies obtained from the designed reward functions $\widehat{R}$. For these results, we set the parameters $\lambda = 1.0$ and $\epsilon = 0.1$. In order to better understand the effect of the parameters λ and ϵ, we vary these parameters and solve (P4-APT) with the considered approaches. The results are shown in Figure 3 for each environment separately. We make the following observations based on the experiments. First, the approaches based on forcing π^* and π^*_{adm} benefit more from increasing λ. This is expected as these two policies have the highest scores under $\overline{R}$; the scores of π_{qg} and π_{co} is less than or equal to the score of π^*_{adm}. Second, the approaches based on forcing π_{qg} and π_{co} are less susceptible to increasing ϵ. This effect is less obvious, and we attribute it to the fact that QGREEDY and CONSTRAIN&OPTIMIZE output π_{qg} and π_{co} respectively by accounting for the cost of forcing these policies. Since this cost clearly increases with ϵ—intuitively, forcing a larger optimality gap in (P3-ATK) requires larger reward modifications—we can expect that increasing ϵ deteriorates more the approaches based on forcing π^* and π^*_{adm}. Third, the objective value of (P4-APT) is consistently better (lower) for π_{co} than for π^*_{adm} and π_{qg}, highlighting the relevance of CONSTRAIN&OPTIMIZE.

Conclusion

The characterization results in this paper showcase the computational challenges of optimal reward design for admissible policy teaching. In particular, we showed that it is computationally challenging to find minimal reward perturbations that would incentivize an optimal agent into adopting a well-performing admissible policy. To address this challenge, we derived a local search algorithm that outperforms baselines which either account for only the agent's performance or for only the cost of the reward design. On the flip side, this algorithm is only applicable to tabular settings, so one of the most interesting research directions for future work would be to consider its extensions based on function approximation. In turn, this would also make the optimization framework of this paper more applicable to practical applications of interest, such as those related to safe and secure RL.

Acknowledgments

Jiarui Gan was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 945719).

References

Amir, O.; Kamar, E.; Kolobov, A.; and Grosz, B. 2016. Interactive teaching strategies for agent training. In *IJCAI*, 804–811.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Banihashem, K.; Singla, A.; Gan, J.; and Radanovic, G. 2022. Admissible Policy Teaching through Reward Design. *CoRR*, abs/2201.02185.

Banihashem, K.; Singla, A.; and Radanovic, G. 2021. Defense Against Reward Poisoning Attacks in Reinforcement Learning. *CoRR*, abs/2102.05776.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. *NeurIPS*, 29: 1471–1479.

Devlin, S. M.; and Kudenko, D. 2012. Dynamic potential-based reward shaping. In *AAMAS*, 433–440.

Dimitrakakis, C.; Parkes, D. C.; Radanovic, G.; and Tylkin, P. 2017. Multi-View Decision Processes: The Helper-AI Problem. In *NeurIPS*, 5443–5452.

Dorigo, M.; and Colombetti, M. 1994. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence*, 71(2): 321–370.

Fischer, M.; Mirman, M.; Stalder, S.; and Vechev, M. 2019. Online robustness training for deep reinforcement learning. *CoRR*, abs/1911.00887.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Grześ, M. 2017. Reward Shaping in Episodic Reinforcement Learning. In *AAMAS*, 565–573.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S.; and Dragan, A. D. 2017. Inverse reward design. In *NeurIPS*, 6768–6777.

Huang, Y.; and Zhu, Q. 2019. Deceptive Reinforcement Learning Under Adversarial Manipulations on Cost Signals. In *GameSec*, 217–237.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of computer computations*, 85–103. Springer.

Lykouris, T.; Simchowitz, M.; Slivkins, A.; and Sun, W. 2019. Corruption robust exploration in episodic reinforcement learning. *CoRR*, abs/1911.08689.

Ma, Y.; Zhang, X.; Sun, W.; and Zhu, J. 2019. Policy poisoning in batch reinforcement learning and control. In *NeurIPS*, 14543–14553.

Mataric, M. J. 1994. Reward functions for accelerated learning. In *ICML*, 181–189.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 278–287.

Nikolaidis, S.; Nath, S.; Procaccia, A. D.; and Srinivasa, S. 2017. Game-theoretic modeling of human adaptation in human-robot collaboration. In *HRI*, 323–331.

Omidshafiei, S.; Kim, D.-K.; Liu, M.; Tesauro, G.; Riemer, M.; Amato, C.; Campbell, M.; and How, J. P. 2019. Learning to teach in cooperative multiagent reinforcement learning. In *AAAI*, 6128–6136.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *ICML*, 2817–2826.

Radanovic, G.; Devidze, R.; Parkes, D.; and Singla, A. 2019. Learning to collaborate in markov decision processes. In *ICML*, 5261–5270.

Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; and Singla, A. 2020a. Policy Teaching in Reinforcement Learning via Environment Poisoning Attacks. *CoRR*, abs/2011.10824.

Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; and Singla, A. 2020b. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. In *ICML*, 7974–7984.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *ICML*, 1889–1897.

Singh, S.; Lewis, R. L.; and Barto, A. G. 2009. Where do rewards come from. In *the Annual Conference of the Cognitive Science Society*, 2601–2606.

Sorg, J.; Lewis, R. L.; and Singh, S. 2010. Reward design via online gradient ascent. *NeurIPS*, 2190–2198.

Sorg, J.; Singh, S.; and Lewis, R. 2010. Internal rewards mitigate agent boundedness. In *ICML*, 1007–1014.

Sun, Y.; Huo, D.; and Huang, F. 2021. Vulnerability-Aware Poisoning Mechanism for Online RL with Unknown Dynamics. In *ICLR*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward Constrained Policy Optimization. In *ICLR*.

Tylkin, P.; Radanovic, G.; and Parkes, D. C. 2021. Learning robust helpful behaviors in two-player cooperative Atari environments. In *AAMAS*, 1686–1688.

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J. Y.; Levine, S.; Finn, C.; and Ma, T. 2020. MOPO: Model-based Offline Policy Optimization. In *NeurIPS*, 14129–14142.

Zhang, H.; Chen, H.; Boning, D.; and Hsieh, C.-J. 2021a. Robust reinforcement learning on state observations with learned optimal adversary. *CoRR*, abs/2101.08452.

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Boning, D.; and Hsieh, C.-J. 2020a. Robust deep reinforcement learning against adversarial perturbations on observations. *CoRR*, abs/2003.08938.

Zhang, H.; and Parkes, D. C. 2008. Value-Based Policy Teaching with Active Indirect Elicitation. In *AAAI*, 208–214.

Zhang, H.; Parkes, D. C.; and Chen, Y. 2009. Policy teaching through reward function learning. In *EC*, 295–304.

Zhang, X.; Chen, Y.; Zhu, X.; and Sun, W. 2021b. Robust policy gradient against strong data corruption. *CoRR*, abs/2102.05800.

Zhang, X.; Ma, Y.; Singla, A.; and Zhu, X. 2020b. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. In *ICML*, 11225–11234.

Zou, H.; Ren, T.; Yan, D.; Su, H.; and Zhu, J. 2019. Reward shaping via meta-learning. *CoRR*, abs/1901.09330.