Revisiting Domain Randomization via Relaxed State-Adversarial Policy Optimization

Yun-Hsuan Lien [1] **Ping-Chun Hsieh** [1] **Yu-Shuen Wang** [1]

Abstract

Domain randomization (DR) is widely used in reinforcement learning (RL) to bridge the gap between simulation and reality by maximizing its *average returns* under the perturbation of environmental parameters. However, even the most complex simulators cannot capture all details in reality due to finite domain parameters and simplified physical models. Additionally, the existing methods often assume that the distribution of domain parameters belongs to a specific family of probability functions, such as normal distributions, which may not be correct. To overcome these limitations, we propose a new approach to DR by rethinking it from the perspective of *adversarial state perturbation*, without the need for reconfiguring the simulator or relying on prior knowledge about the environment. We also address the issue of over-conservatism that can occur when perturbing agents to the worst states during training by introducing a *Relaxed State-Adversarial Algorithm* that simultaneously maximizes the average-case and worst-case returns. We evaluate our method by comparing it to state-of-the-art methods, providing experimental results and theoretical proofs to verify its effectiveness. Our source code and appendix are available at https://github.com/sophialien/RAPPO.

1. Introduction

The use of reinforcement learning (RL) agents in real-world environments is often hindered by the difficulty of collecting data. As a result, many RL agents are trained in simulated environments. However, there is often a significant difference between the simulated and real environments, known as the *"reality gap,"* which can greatly reduce the performance of these agents. To address this issue, domain randomization (DR) methods have been developed to perturb environmental parameters (Tobin et al., 2017; Rajeswaran et al., 2017; Jiang et al., 2021), such as mass and friction coefficient, in order to simulate uncertainty in state transition probabilities and improve the agents' ability to maximize return in various environments. Despite its effectiveness, DR has two major limitations: (1) it requires direct access to the underlying parameters of the simulation, which may not be possible when only off-the-shelf simulation platforms are available, and (2) it relies on prior knowledge of the distribution of environmental parameters, which can greatly affect performance in real-world environments.

To prevent the above limitations, we rethink DR from the perspective of *adversarial state perturbation*, which eliminates the need for reconfiguring the simulator or relying on prior knowledge about the environment. Our method involves perturbing states after nominal state transitions rather than altering transition probabilities. A popular approach from the robust optimization literature (Ben-Tal & Nemirovski, 1998) is to take a *worst-case* viewpoint and perturb the states to nearby states with the lowest long-term expected return under the current policy (Kuang et al., 2022). However, this worst-case strategy can lead to severe over-conservatism in the learned policy, which will not be useful even in nominal environments. We identify that the over-conservative behavior results from the tight coupling between temporal difference (TD) learning in robust RL and the worst-case state perturbation. Specifically: (1) In robust RL, the value functions are learned with bootstrapping in TD methods since finding nearby worst-case states via Monte-Carlo sampling is NP-hard (Ho et al., 2018; Chow et al., 2015; Behzadian et al., 2021). (2) When the state perturbations are in the worst-case scenario, the value function updates are based on the local minimum within a neighborhood of the nominal next state, ignoring the value of the nominal next state. This causes the learner to fail to identify or explore states with high potential returns. To illustrate the issue of over-conservatism, we present a toy example using a grid world environment where the goal is to find the shortest path to a specific location. As shown in Figure 1(a), despite the goal state having a high value, the use of the

*Equal contribution [1]National Yang Ming Chiao Tung University, Hsinchu, Taiwan. Correspondence to: Yun-Hsuan Lien <sophia.yh.lien@gmail.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

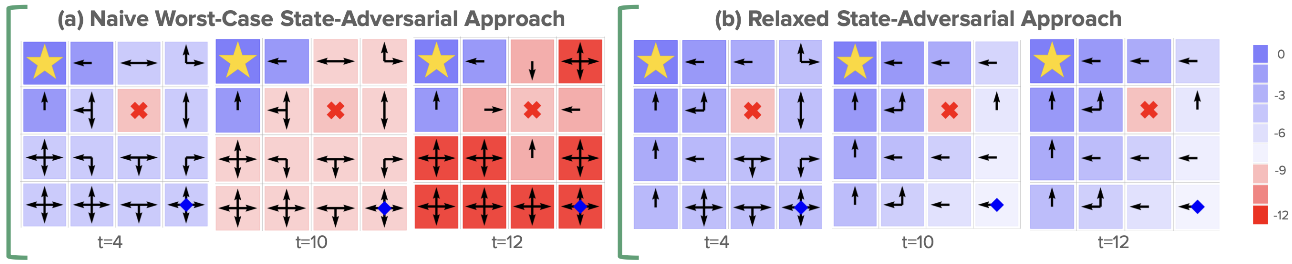

Figure 1. We show the over-conservatism issue in the worst-case state-adversarial policy optimization using a grid world environment for shortest path navigation. In this environment, we have a goal, a trap and an initial state represented by a star, a cross and a dot respectively. The rewards for reaching the trap and the goal are -10 and 0 respectively. We use arrows to indicate the action that has the highest value at each state, and multiple arrows in a state indicate that the actions have equal Q-values. We also use color to indicate the value of the best action at each state. In (a), the agent trained with the worst-case state-adversarial approach fails to learn how to reach the goal state, since TD updates lead the agent to ultimately move towards the trap state after 12 training iterations. In (b), our relaxed state-adversarial approach overcomes this issue by considering both average-case and worst-case environments. For more details on the step-by-step evolution of the value functions, we refer readers to Appendix A.

worst-case state-adversarial method prevents TD updates from propagating this value to other states. Additionally, the agent ultimately learns to move towards the "trap state" due to the combined effect of TD updates and worst-case state-adversarial perturbations. This problem is not limited to the grid world environment, but also commonly occurs in various RL tasks, such as locomotion tasks in MuJoCo with various termination conditions. This motivates us to answer a question in robust RL: *How to fully unleash the power of the state-adversarial model in robustifying RL algorithms without suffering from over-conservatism?*

To answer this question, we introduce relaxed state-adversarial perturbations. This approach involves: (1) Considering both average-case and worst-case scenarios during training, allowing TD updates to propagate the values of high-return states and prevent over-conservatism (as shown in Figure 1(b)). (2) Using a relaxed state-adversarial transition kernel, which allows for easy representation of average-case environments through interpolation of nominal and worst-case environments. Each interpolation coefficient corresponds to a distribution of state adversaries. (3) Theoretically quantifying the performance gap between average-case and worst-case environments and proving that maximizing average-case performance also improves worst-case performance. (4) Implementing *Relaxed State-Adversarial Policy Optimization*, a bi-level framework that optimizes rewards for both cases through alternating and iterative updates. One level updates the policy to maximize average-case performance, while the other updates the interpolation coefficient of the relaxed state-adversarial transition kernel to increase the lower bound of worst-case environment returns.

2. Related Work

Robust Markov Decision Process (MDP) and Robust RL. Robust MDP is a method that aims to maximize rewards in the worst-case scenarios if the testing environment deviates from the training environment (Nilim & El Ghaoui, 2005; Iyengar, 2005; Wiesemann et al., 2013). However, due to the large search space, the complexity of robust MDP increases rapidly as the dimensionality increases. To address this issue, (Tamar et al., 2014) developed an approximation of dynamic programming to scale up the robust MDP paradigm. (Roy et al., 2017) extended the method to nonlinear estimation and ensured convergence to a regional minimum. Later, (Wang & Zou, 2021; Badrinath & Kalathil, 2021) studied the convergence rate when applying function approximations under certain assumptions. (Derman et al., 2021) showed that regularized MDPs are a specific subset of robust MDPs that have uncertain rewards. They chose to solve regularized MDPs as they have lower computational complexity as compared to robust MDPs. (Clement & Kroer, 2021) developed efficient proximal updates to solve the distributionally robust MDP via gradient descent, improving the convergence rate. However, despite these approximations, these model environments are still too restrictive to be applied to real-world problems.

Adversary in Observations. Deep neural networks are highly sensitive to small changes in input, making them vulnerable to adversarial attacks (Huang et al., 2017). To mitigate this issue, various methods have been proposed to train agents in environments with adversarial attacks to improve their robustness (Kos & Song, 2017; Pattanaik et al., 2018). Later, (Wang et al., 2019; Lütjens et al., 2020) adopted the concept of certified defense which is commonly used in classification problems, to guarantee a minimum level of performance. They applied it to agents that take discrete actions and showed that the agents are robust to adversaries in observations within a specific distance. As many real-world problems require agents to take continuous actions, researchers have also developed methods for these scenarios (Weng et al., 2019; Zhang et al., 2020; Oikarinen et al., 2021; Zhang et al., 2021).

Domain Randomization. Uncertainty in transition probabilities can be introduced in the environments. To simulate this scenario, one can perturb the environmental parameters of a simulator to reasonably change transition probabilities when training agents (Huang et al., 2021; Tobin et al., 2017; Jiang et al., 2021; Igl et al., 2019; Cobbe et al., 2019). Specifically, (Tobin et al., 2017) randomly sampled environmental variables and optimized the agents' average reward. Since excessive perturbation may hinder training, (Cobbe et al., 2019) gradually increased the level of difficulty when training agents. (Jiang et al., 2021) further considered the expected return in the optimal case and introduced monotonic robust policy optimization to maximize both the average-case and worst-case returns simultaneously. However, perturbing transition probabilities through environmental parameters requires prior knowledge, so (Kuang et al., 2022) transferred states to nearby local minima based on gradients obtained from the value function to imitate environmental disturbance. (Igl et al., 2019) injected selective noise based on a variational information bottleneck and value networks to prevent models from overfitting to the training environment. This regularization helps agents resist the uncertainty of state transition probabilities.

Our method perturbs states through the gradients of the value function, as (Kuang et al., 2022) did. However, pushing states toward the nearby local minimum will make agents over-conservative because they consider only the worst-case scenarios. We present the relaxed state adversarial perturbation and optimize both the average-case and worst-case environments to overcome this problem.

3. Preliminaries

A Robust MDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{U}, R, \mu, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is action space, $\mathcal{U}$ is the *uncertainty set* that contains all possible transition kernels, $R : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$ is the reward function, μ is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. Let $P_0 \in \mathcal{U}$ be the *nominal transition kernel*, which characterizes the transition dynamics of the nominal environment without perturbation. We define the total expected return under a policy π and a transition kernel $P \in \mathcal{U}$ as

$$J(\pi|P) := \mathbb{E}_{s_0\sim\mu, a_t\sim\pi(\cdot|s_t), s_{t+1}\sim P(\cdot|s_t,a_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]. \tag{1}$$

For ease of exposition, we also define the value function under policy π and transition kernel P as $V^\pi(s|P) := \mathbb{E}_{a_t\sim\pi(\cdot|s_t), s_{t+1}\sim P(\cdot|s_t,a_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s \right]$. To learn a policy in a robust MDP, the DR approaches are built on two major design principles: (1) *Construction of an uncertainty set*: DR presumes that one could have access to the environment parameters of the simulator. The uncertainty set $\mathcal{U}$ is constructed by specifying the possible range of one or multiple environment parameters, typically based on some domain knowledge. (2) *Average-case perspective*: DR resorts to maximizing the average performance with respect to some pre-configured distribution $\mathcal{D}$ over the uncertainty set $\mathcal{U}$, i.e., $\mathbb{E}_{P\sim\mathcal{D}}[J(\pi|P)]$.

4. Relaxed State-Adversary Algorithm

Conventional DR methods enforce attacks on state transitions by perturbing the environment parameters of a simulator. This can be replaced by perturbing the state after each nominal transition (Kuang et al., 2022): Let (s, a) be a state-action pair, and $\Gamma : \mathcal{S} \to \mathcal{S}$ be a state perturbation function. In a nominal environment, the probability of transitioning to state s' under s, a is $P(s'|s, a)$. Under the state perturbation Γ, the probability becomes $P(\Gamma(s')|s, a)$. However, this approach is too effective as a value function considers the expected future return, and a modification to an early state may significantly influence later states, leading to the over-conservatism problem. To address this issue, we present a relaxed state-adversarial policy optimization and prove that the relaxed MDP enjoys two advantages: (1) It helps capture the average performance of the uncertainty set. (2) It enables policy improvement guarantees in the performance of the worst-case MDP. Further, we prove that a specific average-case MDP corresponds to a relaxation parameter. Accordingly, we propose an algorithm for adapting the relaxation parameter during training.

4.1. State-Adversarial MDPs and Uncertainty Sets

State-adversarial attacks perturb the current states to neighboring states with the lowest values. This perturbation process can be captured by a state-adversarial transition kernel, which connects the nominal MDP and the resulting state-adversarial MDP. For ease of exposition, for each state $s \in \mathcal{S}$, we define $\mathcal{N}_\sigma(s) := \{s' | d(s, s') \le \sigma\}$ to be the σ-neighborhood of s, where $d(s, s')$ can be any distance metric. In this study, we use L_∞-norm and use $\|\cdot\|$ to denote the L_∞-norm throughout the paper.

Definition 1 (State Perturbation Matrix)**.** Given a nominal MDP with transition kernel P_0, a policy π, and a perturbation parameter $\sigma \ge 0$, the state perturbation matrix Z^π_σ with respect to π is defined as follows: for each pair of states $i, j \in \mathcal{S}$,

$$Z^\pi_\sigma(i, j) := \begin{cases} 1, & \text{if } j = \arg\min_{s\in\mathcal{N}_\sigma(i)} V^\pi(s|P_0), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The reasoning behind our choice of the surrogate perturbation model is twofold: (1) it can be seen as a way to create adversarial examples for true states; and (2), it is closely connected to the perturbation of environmental parameters, which serve as the standard machinery in the canonical DR formulation, as described in (Kuang et al., 2022).

Remark 1. In continuous state spaces, the $\arg\min$ in Equation 2 can be computed by adapting the fast gradient sign method (FGSM) (Goodfellow et al., 2015). Let V be a value function (i.e., network) with parameter ϕ, s be a state, and ϵ be the strength of perturbation. FGSM finds the perturbed state $\Gamma(s) = s - \epsilon \cdot \text{sign}(\nabla_s V(\phi, s))$ that has the minimum value, where $||s - \Gamma(s)|| \leq \epsilon$, and the gradient at s is computed using back-propagation.

Remark 2. The state-adversarial perturbation does not change the states in a simulator during training because TD learning only considers the reward at the current state and the value at the next adversarial state. The value of each state s is updated repeatedly using $V(s) = r(s,a) + \gamma V(\Gamma(s'))$. Hence, unlike the conventional DR, the state-adversarial perturbation does not require reconfiguring the simulator.

Definition 2 (State-Adversarial MDP)**.** For any policy π, the corresponding state-adversarial MDP with respect to π is defined as a tuple $(\mathcal{S}, \mathcal{A}, P_\sigma^\pi, R, \mu, \gamma)$, where the state-adversarial transition kernel P_σ^π is defined as

$$P_\sigma^\pi(\cdot|s,a) := [Z_\sigma^\pi]^\top P_0(\cdot|s,a), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad (3)$$

and P_0 is the nominal transition kernel. We use the notation $P_\sigma^\pi = [Z_\sigma^\pi]^\top P_0$ in the later paragraphs for simplicity. Note that the state-adversarial transition matrix Z_σ^π depends on the strength of perturbation. Each perturbation radius σ results in a unique state-adversarial MDP P_σ^π.

Remark 3. The state-adversarial MDP, as defined in Definition 2, modifies the true states, rather than the observations, which is fundamentally different from (Zhang et al., 2020).

Definition 3 (Uncertainty Set)**.** Given a radius $\epsilon > 0$, the uncertainty set induced by state-adversarial perturbations, denoted by $\mathcal{U}_\epsilon^\pi$, is defined as

$$\mathcal{U}_\epsilon^\pi := \{P_\sigma^\pi : P_\sigma^\pi = [Z_\sigma^\pi]^\top P_0 \text{ and } \sigma \leq \epsilon\}. \quad (4)$$

Agents trained using the state adversarial MDP P_ϵ^π would prevent themselves from falling into the worst situation (Kuang et al., 2022). However, a large ϵ will make agents too conservative as the high value cannot be propagated to neighboring states by the TD updates (cf. Figure 1). While using a small ϵ can ease the problem, agents would be completely oblivious of the risks outside the bounding area. Moreover, this strategy can be infeasible in an environment with a discrete state space due to the inherent lower bound of ϵ. For example, the agent's movement in the grid world is at least one hop and cannot be further reduced.

Lemma 1 (Monotonicity of Average Value in Perturbation Strength)**.** Under the setting of state-adversarial MDP, the value of the local minimum monotonically decreases as the bounded radius σ increases. Let x be a positive real number. Under any policy π, the total expected return J satisfies

$$J(\pi|P_\sigma^\pi) \geq J(\pi|P_{\sigma+x}^\pi). \quad (5)$$

The proof is in Appendix C. Notably, Lemma 1 indicates that among the transition kernels in the uncertainty set $\mathcal{U}_\epsilon^\pi$, the worst-case occurs when $\sigma = \epsilon$.

4.2. Relaxed State-Adversarial MDPs

We propose a relaxation approach to tackle the problem of over-conservatism, which is detailed as follows:

Relaxed State-Adversarial Transition Kernel. Given $\epsilon > 0$ and $\alpha \in [0,1]$, the α-relaxed state-adversarial transition kernel is defined as a convex combination of the nominal and the state-adversarial transition kernels, i.e.,

$$P_\epsilon^{\pi,\alpha}(\cdot|s,a) = \alpha P_0(\cdot|s,a) + (1-\alpha) P_\epsilon^\pi(\cdot|s,a). \quad (6)$$

Connecting Relaxed State-Adversarial MDPs with DR. DR methods demand a prior distribution for computing the average case performance. Let $\mathcal{D}$ be a distribution over the uncertainty set $\mathcal{U}_\epsilon^\pi$. In the following, we show that applying DR with respect to $\mathcal{D}$ is equivalently cast as optimizing an objective under a relaxed state-adversarial transition kernel.

Lemma 2 (Relaxation parameter α as a prior distribution $\mathcal{D}$ in DR)**.** For any distribution $\mathcal{D}$ over the state-adversarial uncertainty set $\mathcal{U}_\epsilon^\pi$, there must exist an $\alpha \in [0,1]$ such that

$$\mathbb{E}_{P \sim \mathcal{D}}[J(\pi|P)] = J(\pi|P_\epsilon^{\pi,\alpha}). \quad (7)$$

The proof is in Appendix D. It is worth noting that different values of α represent different prior assumptions. For example, $\alpha = 1$ implies that the prior probability of nominal MDP is 1, whereas $\alpha = 0$ indicates that the prior probability of the worst-case MDP is 1. In other words, we can control the value of α to represent different distributions $\mathcal{D}$ and train the policies under various environments. To achieve this goal, we quantify the gap between the *average* performance $\mathbb{E}_{P\sim\mathcal{D}}[J(\tilde{\pi}|P)]$ and *the worst-case* performance $J(\tilde{\pi}|P_\epsilon^\pi)$ when updating the current policy π to a new policy $\tilde{\pi}$, and then apply an optimization technique to maximize both of them. Based on the analysis in (Jiang et al., 2021), one can obtain a lower bound as follows.

Theorem 1 (A Direct Connection Between the Average–Case and the Worst-Case Returns)**.** Given a nominal MDP with transition kernel P_0 along with a state-adversarial uncertainty set $\mathcal{U}_\epsilon^\pi$, for any distribution $\mathcal{D}$ over $\mathcal{U}_\epsilon^\pi$, upon an update from the current policy π to a new policy $\tilde{\pi}$, the following bound holds (Jiang et al., 2021):

$$J(\tilde{\pi}|P_\epsilon^\pi) \geq \mathbb{E}_{P\sim\mathcal{D}}[J(\tilde{\pi}|P)] - 2R_{\max}\frac{\gamma\mathbb{E}_{P\sim\mathcal{D}}[d_{\mathrm{TV}}(P_\epsilon^\pi\|P)]}{(1-\gamma)^2} - 4R_{\max}\frac{d_{\mathrm{TV}}(\pi\|\tilde{\pi})}{(1-\gamma)^2}, \quad (8)$$

where $d_{\mathrm{TV}}(\pi\|\tilde{\pi})$ indicates the total variation divergence between π and $\tilde{\pi}$, and P_ϵ^π is the worst-case state-adversarial transition kernel.

Theorem 1 indicates that the gap between the *average*- and the *worst*- case performance can be expressed using the MDP shift $\mathbb{E}_{P\sim\mathcal{D}}[d_{\mathrm{TV}}(P_\epsilon^\pi\|P)]$ and the policy evolution $d_{TV}(\pi,\tilde{\pi})$. For completeness, we provide the proof of Theorem 1 in Appendix E.

Issues with the lower bound in Theorem 1. The issues with Equation 8 are mainly two-fold: (1) *The bound in Theorem 1 can be loose*: This results from the second term of the right hand side (RHS) of Equation 8, where the maximum possible immediate reward $R_{\max}$ could result in a too conservative lower bound. Specifically, the shift in transition kernel $\mathbb{E}_{P\sim\mathcal{D}}[d_{\mathrm{TV}}(P_\epsilon^\pi\|P)]$ is multiplied by the maximum possible total return $\frac{R_{\max}}{1-\gamma}$, which can be very large in many benchmark RL environments (e.g., MuJoCo) and therefore does not well capture the true effect of state-adversarial perturbation. As a result, the bound can be vacuous unless the worst-case MDP P_ϵ^π is very close to the average case. (2) *The dependency of Equation 8 on the relaxation parameter α is unclear*: As Equation 8 only captures the dependency on $\mathcal{D}$ through an expectation over $\mathcal{D}$ (i.e., $\mathbb{E}_{P\sim\mathcal{D}}[\cdot]$), the dependency on α remains implicit and unclear. Given Equation 8, it remains unknown how to operate with the relaxation parameter to reconcile the worst case and the average case.

To address the above issues, we consider the smoothness of the reward function and transition property to build a tighter connection between the average-case and the worst-case returns. Specifically, Lipschitz continuity in reward function has been widely used in the theory of RL (Fehr et al., 2018; Asadi et al., 2018; Ling et al., 2016). The smoothness of the transition kernel also holds in most of the environments (Shen et al., 2020; Lakshmanan et al., 2015). For example, in grid-world, the next state must be adjacent to the current state; and in MuJoCo, the poses of consecutive periods are similar. We formulate the two properties as follows:

Definition 4 (δ-Smooth Transition Kernel in State)**.** Let P be a transition kernel and δ be a positive constant. P is a δ-*smooth transition kernel in state* if $\|s-s'\|\leq\delta$, for all a and for all s,s' with $P(s'|s,a)>0$.

Definition 5 (L_r-Lipschitz Continuous Reward Function)**.** Let $R:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ be the reward function of an MDP and L_r be a positive constant. R is L_r-*Lipschitz continuous* in state if for any pair $s,s'\in\mathcal{S}$ and any action $a\in\mathcal{A}$,

$$|R(s,a)-R(s',a)|\leq L_r\|s-s'\|. \quad (9)$$

The property of L_r-Lipschitz continuous reward function ensures that the reward function does not change dramatically as the state changes slightly. With the assumption of Lipschitz continuity in reward function and smoothness of transition kernel, we arrive at the following bound:

Theorem 2 (A Sharper Characterization of the Connection Between Worst-Case and Average-Case Returns)**.** Consider a nominal MDP with a δ-smooth transition kernel and an L_r-Lipschitz reward function (cf. Definitions 4-5). Let $\mathcal{U}_\epsilon^\pi$ be the state-adversarial uncertainty set. For any $\alpha\in[0,1]$, upon an update from the current policy π to a new policy $\tilde{\pi}$, the following bound holds:

$$J(\tilde{\pi}|P_\epsilon^\pi) \geq J(\tilde{\pi}|P_\epsilon^{\pi,\alpha}) - \frac{4\gamma(\epsilon+\delta)L_r\alpha}{(1-\gamma)^3} - \frac{4(\gamma(\epsilon+\delta)L_r+(1-\gamma)^2R_{\max})d_{\mathrm{TV}}(\pi\|\tilde{\pi})}{(1-\gamma)^3}, \quad (10)$$

where $d_{\mathrm{TV}}(\pi\|\tilde{\pi})$ is the total variation divergence between π and $\tilde{\pi}$, $P_\epsilon^{\pi,\alpha}$ and P_ϵ^π are the relaxed and worst-case state-adversarial transition kernels within the uncertainty set $\mathcal{U}_\epsilon^\pi$, respectively.

The proof for Theorem 2 is in Appendix F. Notably, this theorem holds for any value of the relaxation parameter α within the range of [0,1]. The main technical challenges in the proof include: (1) *Propagation of state perturbations over time*: The difference of trajectories under different MDPs would increase in a nonlinear and complex manner as time evolves. (2) *Quantifying the difference in rewards among trajectories generated under different transition kernels:* To assess the variations in rewards across different MDPs, it is necessary to consider not only the probability difference at a given time, but also the variations in rewards among different states. Despite the above challenges, our proof uses the finding that the difference of initial probability of state under two MDPs P_ϵ^π and $P_\epsilon^{\pi,\alpha}$ at time step t can be quantified as $\alpha\Delta_t$, where $0\leq\Delta_t\leq 1$. Then under the smoothness conditions of the reward function and the transition matrix, we can characterize a tight bound between the average-case and the worst-case performance.

Why does Theorem 2 provide a tighter lower bound? Theorem 2 offers a tighter lower bound than Theorem 1 because of the two reasons: (1) The main difference between the second term of Equation 8 and that of Equation 10 lies in $R_{\max}$ and $L_r(\epsilon+\delta)$ (given that α and $E_{P\sim\mathcal{D}}\,[d_{\mathrm{TV}}(P_\epsilon^\pi\|P)]$ both capture the transition kernel shifts and are comparable). (2) Recall that $R_{\max}$ can be very large in many benchmark RL environments (e.g., MuJoCo), and it results in a fairly loose bound. By contrast, $L_r(\epsilon+\delta)$ explicitly characterizes the effect of environment perturbation on the return and thereby can offer a tighter bound. To further illustrate this, we provide an example on the Reacher task in Appendix H.

4.3. Online Adaptation of the Relaxation Parameter

We leverage Theorem 2 to address both the average-case and worst-case performance. Specifically, we present a *bi-level* approach to maximize the lower-bound of the worst-case performance (i.e., RHS of Theorem 2). Since α and π are correlated, the two unknowns should be optimized simultaneously. Details are as follows:

- **Lower-level task for average-case return**: On the lower level, we improve the policy by optimizing the objective $J(\tilde{\pi}|P_{\epsilon}^{\pi,\alpha})$ under a fixed relaxation parameter α. This can be done by using any off-the-shelf RL algorithm (e.g., proximal policy optimization (PPO) (Schulman et al., 2017) with a clipped objective).
- **Upper-level task for worst-case return**: On the upper level, we design a meta objective $J_{\text{meta}}(\alpha)$ to represent the lower bound of the worst case performance (i.e., RHS of Equation 10). In other words, $J(\tilde{\pi}|P_{\epsilon}^{\pi}) \geq J_{\text{meta}}(\alpha)$. The task aims to find a relaxation parameter α that can maximize $J_{\text{meta}}(\alpha)$ so as to increase the worst-case performance $J(\tilde{\pi}|P_{\epsilon}^{\pi})$. On one hand, increasing α improves the average performance $J(\tilde{\pi}|P_{\epsilon}^{\pi,\alpha})$ since the average-case moves toward a nominal environment, yet the price is increasing the MDP shift (i.e., the second term of $J_{\text{meta}}(\alpha)$). On the other hand, decreasing α changes the performance and the penalty oppositely. To enable a stable training, we iteratively update α by applying the online cross-validation algorithm (Sutton, 1992).

Both the lower and upper level tasks aim to increase the lower bound of the worst-case performance $J(\tilde{\pi}|P_{\epsilon}^{\pi})$. On the lower level, maximizing the average-case performance $J(\tilde{\pi}|P_{\epsilon}^{\pi,\alpha})$ also increases the lower bound of the worst-case performance $J(\tilde{\pi}|P_{\epsilon}^{\pi})$ because the first term of $J_{\text{meta}}(\alpha)$ increases. On the upper level, the optimization adjusts α to maximize this lower bound directly.

Algorithm 1 outlines the steps of our approach. At each iteration step t, we use PPO to improve the policy π_{θ_t} by maximizing the average-case return $J(\pi_{\theta_t}|P^{\pi_{\theta_{t-1}},\alpha_t})$. Following this, we adjust the relaxation parameter α_t in order to increase the lower bound of the worst-case return, as specified in Equation 10. Note that the samples used in the two steps are different (Lines 3 and 6 of Algorithm 1) because the meta-objective optimization is an online method. In addition, we choose PPO as a base algorithm since it prevents the model from being updated significantly in a single step, which helps control the penalty term $d_{\text{TV}}(\pi\|\tilde{\pi})$ in Theorem 2. Further implementation details are in Appendix I.

5. Experimental Results and Evaluations

We performed two experiments on the MuJoCo platform (Todorov et al., 2012) to assess the performance of our relaxed state adversarial policy optimization (RAPPO) against various adversaries. The baselines and our method were implemented using the PPO algorithm (Schulman et al., 2017), and the default parameters were used.

Algorithm 1 Relaxed State-Adversarial Policy Optimization

Input : MDP $(\mathcal{S}, \mathcal{A}, P_0, R, \gamma)$, Objective function J, step size parameter η, number of iterations T, number of update samples T_{upd}, P_0 is the nominal transition kernel, uncertainty set radius ϵ

1 Initialize the policy π_{θ_0}
2 **for** $t = 0, \cdots, T-1$ **do**
3 Sample the tuple $\{s_i, a_i, r_i, s_i'\}_{i=1}^{T_{\text{upd}}}$, where $a_i \sim \pi_{\theta_t}(\cdot|s_i)$, and $s_i' \sim P_0(\cdot|s_i, a_i)$
4 Evaluate $J(\pi_{\theta_t}|P_{\epsilon}^{\pi_{\theta_{t-1}},\alpha_t})$
5 Update the policy to $\pi_{\theta_{t+1}}$ by applying multi-step SGD to the objective function as PPO
6 Sample the tuple $\{s_i, a_i, r_i, s_i'\}_{i=1}^{T_{\text{upd}}}$, where $a_i \sim \pi_{\theta_{t+1}}(\cdot|s_i)$, and $s_i' \sim P_0(\cdot|s_i, a_i)$
7 Update the relaxation parameter to α_{t+1} via one SGD
8 update with respect to the meta-objective
9 **end**

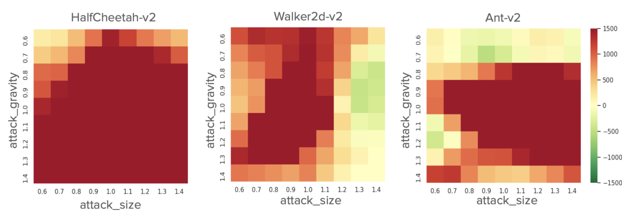

Figure 2. We perturbed the size and gravity of the environments and measured the mean rewards achieved by the agents trained using a DR method, MRPO, and our RAPPO. The heatmaps show the subtractions of MRPO's reward from RAPPO's reward. The higher value (red) indicates that RAPPO outperformed MRPO.

Robustness Against Environmental Adversaries. We compared RAPPO to the state-of-the-art DR method, MRPO (Jiang et al., 2021), to evaluate its robustness against uncertainty in environmental parameters[1]. The agents trained using the two methods were evaluated in environments where the size and gravity were varied between 0.6 and 1.4. To simulate the scenario where domain knowledge is unavailable, MRPO perturbed mass and friction in the range of $[0.8, 1.2]$ during training, while RAPPO perturbed the states using its value function. Figure 2 shows the difference in rewards between the two methods. RAPPO outperformed MRPO because state adversaries are more general than environmental adversaries. An agent trained by MRPO experiences only a specific type of perturbation during training and lacks the

[1]We used the official implementation and the default setting of MRPO from https://proceedings.mlr.press/v139/jiang21c.html.

Table 1. We evaluated the performance of agents trained using PPO, SCPPO, and RAPPO in various Mujoco environments, with varying levels of state perturbation. The results were obtained from 5 different seeds and 50 initial states. We present the mean and standard deviation of the rewards, and the average ranks, with higher rewards and lower ranks indicating better performance.

Environment	Method	Reward	Reward	Rank	Reward	Rank	Reward	Rank	Reward	Rank	Reward	Rank
		Nominal	$\sigma = 0.005$		$\sigma = 0.01$		$\sigma = 0.015$		$\sigma = 0.02$		$\sigma = 0.025$	
HalfCheetah-v2	PPO	5286 ± 1004	4280 ± 1552	2.52	3186 ± 1875	2.24	1996 ± 1743	2.00	1256 ± 1251	1.50	819 ± 1003	**1.20**
	SCPPO	6157 ± 709	5046 ± 1533	2.42	3367 ± 2090	2.74	1795 ± 1758	2.92	875 ± 1259	3.00	60 ± 791	3.00
	RAPPO	6146 ± 742	**5519 ± 774**	**1.06**	**4353 ± 1510**	**1.02**	**3087 ± 1568**	**1.08**	**1878 ± 1287**	**1.50**	**846 ± 951**	1.80
		Nominal	$\sigma = 0.0008$		$\sigma = 0.0016$		$\sigma = 0.002$		$\sigma = 0.0024$		$\sigma = 0.003$	
Hopper-v2	PPO	3330 ± 619	1357 ± 787	2.52	615 ± 194	3.00	494 ± 151	3.00	462 ± 141	3.00	417 ± 131	3.00
	SCPPO	2644 ± 951	1369 ± 620	2.48	876 ± 347	2.00	773 ± 357	2.00	782 ± 437	1.98	732 ± 412	2.00
	RAPPO	3301 ± 520	**2198 ± 859**	**1.00**	**1457 ± 537**	**1.00**	**1244 ± 584**	**1.00**	**1067 ± 605**	**1.02**	**1014 ± 779**	**1.00**
		Nominal	$\sigma = 0.001$		$\sigma = 0.0015$		$\sigma = 0.002$		$\sigma = 0.0025$		$\sigma = 0.003$	
Walker2d-v2	PPO	3781 ± 1165	1564 ± 1285	2.60	903 ± 521	2.72	763 ± 353	2.60	628 ± 241	2.66	575 ± 222	2.54
	SCPPO	4313 ± 979	2647 ± 1584	2.28	1604 ± 1082	2.20	985 ± 704	2.28	772 ± 492	2.14	666 ± 412	1.88
	RAPPO	4608 ± 962	**3998 ± 1487**	**1.12**	**3298 ± 1478**	**1.08**	**2160 ± 1408**	**1.12**	**1470 ± 1013**	**1.20**	**1173 ± 783**	**1.58**
		Nominal	$\sigma = 0.01$		$\sigma = 0.02$		$\sigma = 0.03$		$\sigma = 0.04$		$\sigma = 0.05$	
Ant-v2	PPO	6075 ± 889	4489 ± 1342	**1.82**	2071 ± 1156	1.74	1016 ± 523	**1.76**	703 ± 283	1.92	615 ± 248	1.94
	SCPPO	5915 ± 728	4203 ± 1441	2.28	1661 ± 951	2.64	831 ± 398	2.46	609 ± 320	2.42	489 ± 273	2.64
	RAPPO	6022 ± 698	**4381 ± 1357**	1.90	**2284 ± 1225**	**1.62**	**1038 ± 553**	1.78	**733 ± 255**	**1.66**	**672 ± 219**	**1.42**
		Nominal	$\sigma = 0.003$		$\sigma = 0.004$		$\sigma = 0.005$		$\sigma = 0.006$		$\sigma = 0.007$	
Humanoid-v2	PPO	5357 ± 1618	3033 ± 1834	2.48	2373 ± 1742	2.14	1802 ± 1446	2.20	1287 ± 1068	2.42	939 ± 750	2.46
	SCPPO	5410 ± 1340	3196 ± 1781	2.32	2387 ± 1472	2.62	1783 ± 1256	2.66	1271 ± 838	2.40	1060 ± 678	2.24
	RAPPO	5355 ± 1491	**3768 ± 1972**	**1.20**	**3227 ± 1883**	**1.24**	**2537 ± 1698**	**1.14**	**1747 ± 1274**	**1.18**	**1350 ± 1133**	**1.30**

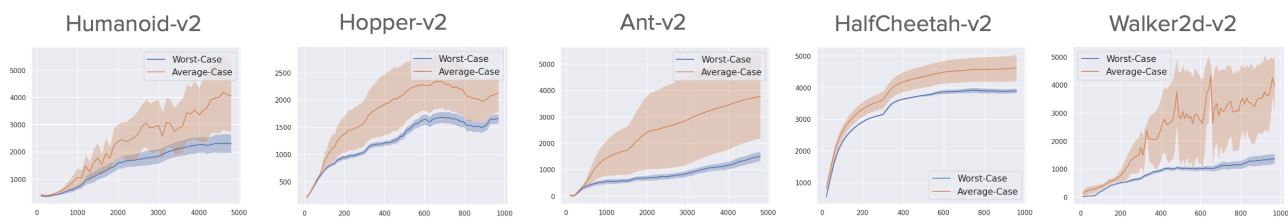

Figure 3. Our RAPPO can steadily improve the average-case and worst-case rewards during training. The solid lines and shaded areas indicate the mean and standard deviation of the rewards, respectively. Note that the variance of the average-case rewards results from the inherent large variability in the adversarial strength in the average-case scenario.

ability to adapt to another disturbances during testing.

Robustness Against States Adversaries. To assess the robustness of RAPPO against state adversaries, we compared our RAPPO to SCPPO (Kuang et al., 2022), the state-of-the-art method for robust RL via state perturbations from a worst-case perspective. We also included the original PPO algorithm in the experiment for comparison, as it forms the foundation for both RAPPO and SCPPO. To ensure a fair comparison, we used the same parameters for RAPPO and SCPPO. Specifically, we set ϵ to 0.015, 0.002, 0.002, 0.03, and 0.005 for the HalfCheetah-v2, Hopper-v2, Ant-v2, Walker2d-v2, and Humanoid-v2 environments, respectively. These values were chosen based on the mean magnitude of actions taken in each environment.

Table 1 displays the test results. We evaluated the agents' performance under multiple strengths of attack, using their respective value functions. Since we repeated each experiment 250 times (i.e., across 5 different seeds and 50 initial states) for evaluation, the mean and standard deviation of the rewards are reported. The results clearly show that the agents' performance decreased as the strength of the attack increased, in accordance with Lemma 1. Additionally, RAPPO performed comparably to PPO and SCPPO in nominal environments and its performance decreased at a slower rate as the attack strength increased. It is worth noting that the attacks in the last two columns of Table 1 were stronger than the worst-case scenario (i.e., σ is larger than the radius of the uncertainty set ϵ used in training), and RAPPO still performed the best in these environments.

The variances of the total rewards in Table 1 are large because we attacked the agents in the direction that would decrease their value the most at each step. An episode could terminate shortly after a critical attack, resulting in a considerably low total expected reward for the trajectory. As a result, the mean-variance ratios were highest in the nominal environment and decreased as the attack strength increased. To further confirm that RAPPO outperformed PPO and SCPPO, we followed the guidelines in (Chan et al., 2020) and computed the *rank* of each method under the same seed and the same initial state. The best performing method was assigned a rank of 1, the second best was 2, and so on. We reported the average ranks of each method across 5 different seeds and 50 initial states in Table 1. As shown, higher average rewards often correspond to better ranks (low values), and RAPPO had the best average ranks in most of the experiments.

Notably, we perturbed policies every step, and each attack aimed to push the policy to the worst neighboring state in

Table 2. We evaluated the performance of agents trained using RAPPO and simple schedulers of α. The Scheduler=, Scheduler+ and Scheduler- indicate that the value of α was fixed at 0.5 (i.e., the middle of the nominal and worst-case environments), gradually increased from 0 to 1, and gradually decreased from 1 to 0, respectively.

Environment		Nominal	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.015$	$\sigma = 0.02$	$\sigma = 0.025$
HalfCheetah-v2	Scheduler +	5744 ± 883	4591 ± 1395	3542 ± 1730	2001 ± 1910	1096 ± 1859	408. ± 1644
	Scheduler -	5865 ± 770	5176 ± 829	3456 ± 1903	1970 ± 2033	941 ± 1483	-158 ± 922
	Scheduler =	5830 ± 779	5185 ± 782	4084 ± 1266	2743 ± 1644	1459 ± 1439	406 ± 912
	RAPPO	**6146 ± 742**	**5519 ± 774**	**4353 ± 1510**	**3087 ± 1568**	**1878 ± 1287**	**846 ± 951**
		Nominal	$\sigma = 0.0008$	$\sigma = 0.0016$	$\sigma = 0.002$	$\sigma = 0.0024$	$\sigma = 0.003$
Hopper-v2	Scheduler +	3032 ± 763	1156 ± 614	829 ± 353	691 ± 250	591 ± 210	479 ± 238
	Scheduler -	3032 ± 822	1276 ± 592	789 ± 428	758 ± 456	617 ± 332	486 ± 225
	Scheduler =	2497 ± 1041	1626 ± 1064	1122 ± 817	795 ± 426	704 ± 393	521 ± 201.27
	RAPPO	**3301 ± 520**	**2198 ± 859**	**1457 ± 537**	**1244 ± 584**	**1067 ± 605**	**1014 ± 779**
		Nominal	$\sigma = 0.001$	$\sigma = 0.0015$	$\sigma = 0.002$	$\sigma = 0.0025$	$\sigma = 0.003$
Walker2d-v2	Scheduler +	4051 ± 1130	1548 ± 1068	994 ± 600	736 ± 314	597 ± 237	618 ± 287
	Scheduler -	4043 ± 1238	2410 ± 1581	1601 ± 1206	1601 ± 1206	771 ± 768	771 ± 768.81
	Scheduler =	4113 ± 899	2394 ± 1471	1881 ± 1398	1520 ± 1387	1249 ± 1282	888 ± 970
	RAPPO	**4608 ± 962**	**3998 ± 1487**	**3298 ± 1478**	**2160 ± 1408**	**1470 ± 1013**	**1173 ± 783**
		Nominal	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.03$	$\sigma = 0.04$	$\sigma = 0.05$
Humanoid-v2	Scheduler +	5162 ± 1714	2903 ± 2094	2365 ± 1840	1689 ± 1580	1347 ± 1282	915 ± 694
	Scheduler -	5039 ± 1798	3316 ± 2153	2655 ± 2043	1847 ± 1527	1322 ± 958	1022 ± 676
	Scheduler =	5169 ± 1468	3031 ± 1810	1941 ± 1336	1550 ± 1165	1035 ± 551	874 ± 458
	RAPPO	**5355 ± 1491**	**3768 ± 1972**	**3227 ± 1883**	**2537 ± 1698**	**1747 ± 1274**	**1350 ± 1133**

the experiment. This strategy would inevitably introduce high variance to the total returns. This phenomenon did not appear only in our results but also in all the baselines, such as PPO and SCPPO. Take the performance of PPO in the HalfCheetah environment as an example: the mean-variance ratio was 5.26 (5286/1004) when the environment was nominal, and the ratio decreased to 0.81 (819/1003) when the magnitude σ of attack became 0.025. The performance of the SCPPO in the HalfCheetah environment also had a similar phenomenon. The mean-variance ratio decreased from 8.68 (6157/709) to 0.08 (60/791).

Aggregate Levels Results. Given the low mean-variance ratio, we calculated the interquartile mean (IQM) (Agarwal et al., 2021) using the rliable library [2] to evaluate the results at the aggregate levels. Specifically, we aggregated the results across the strengths of perturbation both in each environment and in all environments. Due to the various reward ranges in different environments, we normalized the rewards to the range of [0, 1] to facilitate comparison across the tasks. Figure 4 shows the aggregate metrics with 95% Confidence Intervals (CIs) of aggregated scores for the five Mujoco environments. The CIs are estimated using the percentile bootstrap with stratified sampling. Higher IQM scores are better. In addition, we conducted a test of significance of the total expected rewards to verify that the results are statistically significant. The results are in Appendix J.

Combinations of Adversarial Attacks. To evaluate the agents' performance under the combined attacks (against environmental adversaries and states adversaries), we modified an environmental parameter – size by offsetting the value from 1.0 to 1.2 and perturbed the states to nearby positions (using the value function) after each state transition. The results indicate that RAPPO outperformed the other two methods under the combined attacks. Particularly, RAPPO outperformed the methods by a clear margin in the environments of HalfCheetah and Humanoid. The results are in Table 3

Steady Improvements of Average and Worst Case Environments. We employed a *bi-level* approach to optimize both the average and worst-case environments. To demonstrate the feasibility of this approach, we evaluated the agents' performance under these two cases during training. To determine the worst-case result, we generated 50 trajectories from different initial states, perturbed the states with the same strength as the training ϵ, and then averaged the rewards. In contrast, the average-case result was determined from 50 initial states and 10 different perturbation strengths, which were evenly distributed between 0 and ϵ. In total, the rewards of 500 trajectories were averaged. Figure 3 illustrates that RAPPO can steadily improve the average-case performance without sacrificing the worst-case performance. Note that the high variance of the average-case rewards is expected due to the varying adversarial strengths. Figure 6 in Appendix provides a comparison of the learning performances of PPO, SCPPO, and RAPPO.

The Value of Relaxation Parameter α. Theorem 2 suggests that the policy π and the relaxation parameter α are interdependent and should be optimized together. To verify this claim, we also conducted experiments under various popular schedulers, where α was held constant (Scheduler=), increased linearly (Scheduler+), or decreased linearly (Scheduler-) during training. The results, shown in Table 2, reveal that RAPPO performed better than these schedulers in most environments. This is not surprising,

[2] https://github.com/google-research/rliable

Figure 4. The interquartile mean (IQM) reveals the aggregated results across the strengths of perturbation in each environment and across strengths and environments.

Table 3. Evaluated the agents' performance under the combined attacks.

	HalfCheetah-v2	Walker-v2	Ant-v2	Hopper-v2	Humanoid-v2
PPO	1568±980	286±74	814±579	450±9	891±321
SCPPO	1083±1293	134±67	842±461	474±2	1035±400
RAPPO	3124±307	311±156	969±870	478±2	2644±1481

as the schedulers did not take into account the correlation between α and π, making it difficult to find a good solution.

Extending SAPPO Using Relaxed State Adversaries. While RAPPO effectively improves the robustness of agents against state adversaries, a classical method, SAPPO (Zhang et al., 2020), can help agents against perturbations of observations. We thus extended SAPPO by incorporating our relaxed state adversarial attacks and evaluated its effectiveness. The results are in Table 4 in Appendix G. As indicated, the extended RA-SAPPO performed better than SAPPO in most of the environments, particularly under strong attacks.

6. Concluding Remarks

We have presented a relaxed state adversarial policy optimization approach to enhance the robustness of agents against uncertain environments. In contrast to the conventional DR methods, we used adversarial attacks to perturb states, allowing us to decouple randomization from simulators and eliminating the need for prior knowledge of selecting environmental parameters or assumptions about parameter distributions. Furthermore, we implemented a relaxation strategy to address the over-conservatism problem caused by state adversarial attacks. Our policy optimization simultaneously maximizes rewards in average-case environments while maintaining lower-bound rewards in worst-case environments. Experimental results and theoretical proofs validate the effectiveness of our method.

Limitations and Future Work. Our relaxation method is state-independent, meaning that the value of α is adjusted based on the overall performance of the policy. Given that the level of difficulty varies between states, it would be interesting to investigate state-dependent relaxation methods. Additionally, we currently assume that each dimension of states is of equal importance, which may not always be the case. We plan to further study this issue in the future.

Acknowledgments

We thank the reviewers for their constructive comments. This material is based upon work partially supported by the National Science and Technology Council (NSTC), Taiwan under Contract No. 110-2628-E-A49-014 and Contract No. 111-2628-E-A49-019 and Contract No. 111-2221-E-A49 -129 -MY3 and Contract No. 111-2634-F-A49 -013, and based upon work partially supported by the Higher Education Sprout Project of the National Yang Ming Chiao Tung University and Ministry of Education (MOE), Taiwan.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Asadi, K., Misra, D., and Littman, M. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273. PMLR, 2018.

Badrinath, K. P. and Kalathil, D. Robust reinforcement learning using least squares policy iteration with provable performance guarantees. In *International Conference on Machine Learning*, pp. 511–520. PMLR, 2021.

Behzadian, B., Petrik, M., and Ho, C. P. Fast algorithms for L_∞-constrained S-rectangular robust MDPs. *Advances in Neural Information Processing Systems*, 34:25982–25992, 2021.

Ben-Tal, A. and Nemirovski, A. Robust convex optimization. *Mathematics of operations research*, 23(4):769–805, 1998.

Chan, S. C., Fishman, S., Canny, J., Korattikara, A., and Guadarrama, S. Measuring the reliability of reinforcement learning algorithms. *International Conference on Learning Representations*, 2020.

Chow, Y., Tamar, A., Mannor, S., and Pavone, M. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems*, 28, 2015.

Clement, J. G. and Kroer, C. First-order methods for wasserstein distributionally robust mdp. In *International Conference on Machine Learning*, pp. 2010–2019. PMLR, 2021.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

Derman, E., Geist, M., and Mannor, S. Twice regularized MDPs and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34, 2021.

Fehr, M., Buffet, O., Thomas, V., and Dibangoye, J. ρ-POMDPs have Lipschitz-continuous ϵ-optimal value functions. *Advances in neural information processing systems*, 31, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

Ho, C. P., Petrik, M., and Wiesemann, W. Fast bellman updates for robust MDPs. In *International Conference on Machine Learning*, pp. 1979–1988. PMLR, 2018.

Huang, J., Guan, D., Xiao, A., and Lu, S. Fsdr: Frequency space domain randomization for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6891–6902, 2021.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv:1702.02284*, 2017.

Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., and Hofmann, K. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Jiang, Y., Li, C., Dai, W., Zou, J., and Xiong, H. Monotonic robust policy optimization with model discrepancy. In *International Conference on Machine Learning*, pp. 4951–4960. PMLR, 2021.

Kos, J. and Song, D. Delving into adversarial attacks on deep policies. *5th International Conference on Learning Representations, Workshop*, 2017.

Kuang, Y., Lu, M., Wang, J., Zhou, Q., Li, B., and Li, H. Learning robust policy against disturbance in transition dynamics via state-conservative policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7247–7254, 2022.

Lakshmanan, K., Ortner, R., and Ryabko, D. Improved regret bounds for undiscounted continuous reinforcement learning. In *International Conference on Machine Learning*, pp. 524–532. PMLR, 2015.

Ling, C. K., Low, K. H., and Jaillet, P. Gaussian process planning with lipschitz continuous reward functions: Towards unifying bayesian optimization, active learning, and beyond. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Lütjens, B., Everett, M., and How, J. P. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*, pp. 1328–1337. PMLR, 2020.

Nilim, A. and El Ghaoui, L. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Oikarinen, T., Zhang, W., Megretski, A., Daniel, L., and Weng, T.-W. Robust deep reinforcement learning through adversarial loss. *Advances in Neural Information Processing Systems*, 34, 2021.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2040–2042, 2018.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations*, 2017.

Roy, A., Xu, H., and Pokutta, S. Reinforcement learning under model mismatch. *Advances in neural information processing systems*, 30, 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shen, Q., Li, Y., Jiang, H., Wang, Z., and Zhao, T. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pp. 8707–8718. PMLR, 2020.

Sutton, R. S. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI*, pp. 171–176. San Jose, CA, 1992.

Tamar, A., Mannor, S., and Xu, H. Scaling up robust MDPs using function approximation. In *International conference on machine learning*, pp. 181–189. PMLR, 2014.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Wang, Y. and Zou, S. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34, 2021.

Wang, Y.-S., Weng, T.-W., and Daniel, L. Verification of neural network control policy under persistent adversarial perturbation. *arXiv preprint arXiv:1908.06353*, 2019.

Weng, T.-W., Dvijotham, K. D., Uesato, J., Xiao, K., Gowal, S., Stanforth, R., and Kohli, P. Toward evaluating robustness of deep reinforcement learning with continuous control. In *International Conference on Learning Representations*, 2019.

Wiesemann, W., Kuhn, D., and Rustem, B. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., and Hsieh, C.-J. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33: 21024–21037, 2020.

Zhang, H., Chen, H., Boning, D., and Hsieh, C.-J. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021.