REVISITING DOMAIN RANDOMIZATION VIA RELAXED STATE-ADVERSARIAL POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Domain randomization (DR) is widely used in reinforcement learning (RL) to bridge the gap between simulation and reality through maximizing its *average returns* under the perturbation of environmental parameters. Although effective, the methods have two limitations: (1) Even the most complex simulators cannot capture all details in reality due to finite domain parameters and simplified physical models. (2) Previous methods often assume that the distribution of domain parameters is a specific family of probability functions, such as a normal or a uniform distribution, which may not be correct. To enable robust RL via DR without the aforementioned limitations, we rethink DR from the perspective of *adversarial state perturbation*, without the need for re-configuring the simulator or relying on prior knowledge about the environment. We point out that perturbing agents to the worst states during training is naïve and could make the agents *over-conservative*. Hence, we present a *Relaxed State-Adversarial Algorithm* to tackle the over-conservatism issue by simultaneously maximizing the average-case and worst-case performance of policies. We compared our method to the state-of-the-art methods for evaluation. Experimental results and theoretical proofs verified the effectiveness of our method.

1 INTRODUCTION

Most reinforcement learning (RL) agents are trained in simulated environments due to the difficulties of collecting data in real environments. However, the domain shift, where the simulated and real environments are different, could significantly reduce the agents' performance. To bridge this *"reality gap"*, domain randomization (DR) methods perturb environmental parameters (Tobin et al., 2017; Rajeswaran et al., 2016; Jiang et al., 2021), such as the mass or the friction coefficient, to simulate the uncertainty in state transition probabilities and expect the agents to maximize the return over the perturbed environments. Despite its wide applicability, DR suffers from two practical limitations: (i) DR requires direct access to the underlying parameters of the simulator, and this could be infeasible if only off-the-shelf simulation platforms are available. (ii) To enable sampling of environmental parameters, DR requires a prior distribution over the feasible environmental parameters. However, the design of such a prior typically relies on domain knowledge and could significantly affect the performance in real environments.

To enable robust RL via DR without the above limitations, we rethink DR from the perspective of *adversarial state perturbation*, without the need for re-configuring the simulator or relying on prior knowledge about the environment. The idea is that perturbing the transition probabilities can be equivalently achieved by imposing perturbations upon the states after nominal state transitions. To substantiate the idea of state perturbations, a simple and generic approach from the robust optimization literature (Ben-Tal & Nemirovski, 1998) is taking a *worst-case* viewpoint and perturbing the states to nearby states that have the lowest long-term expected return under the current policy (Kuang et al., 2021). While being a natural solution, such a worst-case strategy could suffer from severe *over-conservatism*. We identify that the over-conservative behavior results from the tight coupling between the need for temporal difference (TD) learning in robust RL and the worst-case operation of state perturbation. Specifically: (1) In robust RL, the value functions are learned with the help of 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) Under the worst-case state perturbations, TD methods would update the value function based on the local minimum within a neighborhood of the nominal next state and is, therefore, completely unaware of the value of the

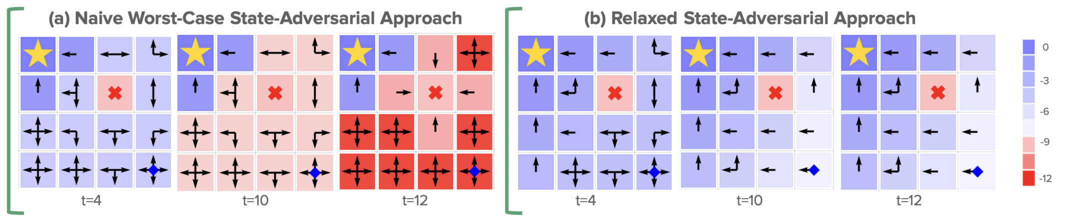

Figure 1: We illustrate the over-conservative issue of the naive worst-case state-adversarial policy optimization using a 4×4 shortest-path grid world environment. The star, cross, and dot represent the goal, the trap, and the initial state, respectively. The terminal rewards of the trap and the goal are -10 and 0. We use an arrow to represent the action that has the highest value at each state. Multiple arrows in a state indicate that the actions have equal Q-values. We also use the color to indicate the value of the best action at each state. In (a), the agent trained by the naïve worst-case state-adversarial approach fails to learn how to reach the goal. What's worse, under TD updates, the worst-case state perturbation makes the trap state indistinguishable from other states. As a result, the agent ultimately learns to move toward the trap state after 12 training iterations. In (b), our relaxed state-adversarial approach avoids the over-conservatism issue by considering both the average-case and worst-case environments. We refer readers to Appendix A.1 for the step-by-step evolution of the value functions.

nominal next state. As a result, the learner could fail to identify or explore those states with potentially high returns. To further illustrate this phenomenon, we consider a toy grid world example of finding the shortest path toward the goal, as shown in Figure 1(a). Although the goal state has a high value, the TD updates cannot propagate the value to other states since all nominal state transitions toward the goal state are perturbed away under the worst-case state-adversarial method. What's even worse, the agent ultimately learns to move toward the trap state due to the compounding effect of TD updates and worst-case state-adversarial perturbations. Notably, in addition to the grid world environment, such trap terminal states also commonly exist in various RL problems, such as the locomotion tasks in MuJoCo. As a result, there remains one critical unanswered 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. Specifically: (1) Instead of taking a pure worst-case perspective, we simultaneously consider both the average-case and worst-case scenarios during training. By incorporating the average-case scenarios, the TD updates can successfully propagate the values of those potentially high-return states to other states and thereby prevent the over-conservative behavior (Figure 1(b)). (2) To substantiate the above idea, we introduce a relaxed state-adversarial transition kernel, where the average-case environment can be easily represented by the interpolation of the nominal and the worst-case environments. Under this new formulation of DR, each interpolation coefficient corresponds to a distribution of state adversaries. (3) Besides, based on this formulation, we theoretically quantify the performance gap between the average-case and the worst-case environments; and prove that maximizing the average-case performance can also benefit the worst-case performance. (4) Accordingly, we present Relaxed state-adversarial policy optimization, a bi-level framework that optimizes the rewards of the two cases alternatively and iteratively. One level updates the policy to maximize the average-case performance, and the other updates the interpolation coefficient of the relaxed state-adversarial transition kernel to increase the lower bound of the return of the worst-case environment.

2 Related Work

Robust Markov Decision Process (MDP) and Robust RL. Robust MDP aims to maximize rewards in the worst situations if the testing environment deviates from the training environment (Nilim & El Ghaoui, 2005; Iyengar, 2005; Wiesemann et al., 2013). Due to the large searching space, the complexity of robust MDP grows rapidly when the dimensionality increases. Therefore, Tamar et al. (2014) developed an approximated dynamic programming to scale up the robust MDPs paradigm. Roy et al. (2017) extended the method to nonlinear estimation and guaranteed the convergence to a regional minimum. Afterward, the works of (Wang & Zou, 2021; Badrinath & Kalathil, 2021) study the convergence rate when applying function approximations under assumptions. Derman et al. (2021) showed that the regularized MDPs are a particular instance of robust MDPs with uncertain rewards. They solved regularized MDPs rather than robust MDPs to reduce computation complexity. Grand-Clément & Kroer (2020) developed efficient proximal updates to solve the distributionally robust MDP via gradient descent and improved the convergence rate. However, although several approximations were presented, such model environments are still too restrictive, and they cannot be used to solve real-world problems.

Adversary in Observations. Even a small perturbation to observations may significantly degrade agents' performance because deep neural networks are vulnerable to inputs constructed by adversaries (Huang et al., 2017). Therefore, methods were presented to train agents under environments with adversarial attacks to improve their robustness (Kos & Song, 2017; Pattanaik et al., 2018). To guarantee a lower-bound performance, the works of (Lütjens et al., 2020; Wang et al., 2019) adopted the idea of certified defense used in classification problems. When making discrete actions, agents are certifiably robust to adversaries in observation within the ϵ distance (L_p-norm). Since most real-world problems are continuous, there were also methods (Weng et al., 2019; Zhang et al., 2020; Oikarinen et al., 2021; Zhang et al., 2021) presented to improve agents' robustness for continuous actions.

Domain Randomization. Environments can induce the uncertainty of transition probabilities. To simulate this circumstance, 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. Given that a significant perturbation may fail the training, Cobbe et al. (2019) increased the level of difficulty step by step when training agents to improve their average rewards. Jiang et al. (2021) further considered the expected return in the optimal case and introduced monotonic robust policy optimization to maximize the average-case and worst-case returns simultaneously. Since perturbing transition probabilities through environmental parameters demands prior knowledge, Kuang et al. (2021) transferred states to the nearby local minimum 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 the training environment. The 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. (2021) 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 Markov decision process (robust MDP) is characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \mu, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is action space, $\mathcal{P}$ 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 $P_0 \in \mathcal{P}$ denote 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 $P \in \mathcal{P}$ 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 $V_P^\pi(s) := \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t,a_t)} \left[\sum_{a_t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s \right]$. To learn a policy in a robust MDP, the DR approaches are built on two major design principles: (1) *Construction of uncertainty set*: DR presumes that one could have access to the environment parameters of the simulator. The uncertainty set $\mathcal{P}$ 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 $\mathcal{P}$, i.e., $\mathbb{E}_{P \sim \mathcal{D}}[J(\pi|P)]$.

4 Domain Randomization via Relaxed State-Adversary

4.1 Connecting Domain Randomization and State Perturbation

Conventional DR methods enforce attacks on state transitions by perturbing the environment parameters of a simulator. This goal can be achieved by perturbing the state after each nominal transition (Kuang et al., 2021): Let (s, a) be some state-action pair, and $\Gamma : \mathcal{S} \to \mathcal{S}$ be a state perturbation function. In a nominal environment, the probability of the transition to some 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 state adversarial attack is too effective since a value function considers the expected future return, and a perturbation to an early state may significantly influence the later states. The over-conservatism problem therefore occurs. We present a relaxed state-adversarial policy optimization to overcome the problem. We also prove that the relaxed MDP enjoys two main properties under relaxation: (1) it stands for the average performance of the uncertainty set; (2) it guarantees the improvement the performance of the worst-case MDP. Further, we prove that a specific average-case MDP corresponds to a relaxation parameter. Hence, we propose an algorithm for adapting the relaxation parameters during training.

4.2 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.

Definition 1 (State Perturbation Matrix). *Given 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 $i, j \in \mathcal{S}$,*

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

The justifications for choosing the above surrogate perturbation model are two-fold: (1) The model can be interpreted as constructing adversarial examples for the true states. (2) The perturbation model is closely related to the perturbation of environment parameters, which serve as the standard machinery in the canonical DR formulation, as described in (Kuang et al., 2021).

Remark 1. *In continuous state space, the* arg min *in Equation 2 can be computed by adapting the fast gradient sign method (FGSM) (Goodfellow et al., 2014). 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 $\Gamma(s) = s - \epsilon \cdot sign(\nabla_s V(\phi, s))$ that has the minimum value, where $||s - \Gamma(s)||_\infty \le \epsilon$, and the gradient at s is computed using back-propagation.*

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

$$P^\pi_\sigma(\cdot|s, a) := [Z^\pi_\sigma]^\top P_0(\cdot|s, a), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}. \tag{3}$$

Recall that P_0 is the nominal transition kernel. We use the notation $P^\pi_\sigma = [Z^\pi_\sigma]^\top P_0$ in the later paragraphs for simplicity. Note that the state adversarial transition matrix Z^π_σ depends on the strength of perturbation σ. Each perturbation radius σ results in a unique state-adversarial MDP P^π_σ.

Remark 2. The state-adversarial MDP defined in Definition 2 involves perturbation of the true states, which is fundamentally different from the perturbation of observations (Zhang et al., 2020).

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

$$\mathcal{P}^\pi_\epsilon := \{P^\pi_\sigma : P^\pi_\sigma = [Z^\pi_\sigma]^\top P_0 \text{ and } \sigma \le \epsilon\}. \tag{4}$$

The adversarial attack transits agents toward low-value states. Agents trained using this state adversarial MDP would prevent themselves from falling into the worst situation (Kuang et al., 2021). However, a large ϵ will make agents too conservative and fail to reach any goal state because its value cannot be propagated to neighboring states by the TD updates (Figure 1). Although using a small ϵ can ease the problem, agents would completely omit the risks outside the bounding area. Besides, this strategy is unachievable in a discrete environment due to the lower-bound value of ϵ. For example, the agent's movement in the grid world is one hop and cannot be 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. The reward function J satisfies*

$$J(\pi|P^{\pi}_{\sigma}) \geq J(\pi|P^{\pi}_{\sigma+x}), \quad \forall\pi. \tag{5}$$

The proof is in Appendix A.3. Notably, Lemma 1 indicates that among the transition kernels in the uncertainty set $\mathcal{P}^{\pi}_{\epsilon}$, the worst-case occurs when $\sigma = \epsilon$.

4.3 Relaxed State-Adversarial MDPs

We present a relaxation framework to address the over-conservatism issue. To begin with, we consider a relaxation on the state-adversarial transition kernel 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.,

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

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

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

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

The proof is in Appendix A.4. 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 $J(\tilde{\pi}|P^{\pi}_{\epsilon})$ when updating the current policy π to a new policy $\tilde{\pi}$, and then apply an optimization technique to maximize both of them. One naïve bound is as follows.

Theorem 1 (A naïve connection between the average-case and the worst-case returns). *Given a nominal MDP with state adversaries, when updating the current policy π to a new policy $\tilde{\pi}$, the following bound holds (Jiang et al., 2021):*

$$J(\tilde{\pi}|P^{\pi}_{\epsilon}) \geq \mathbb{E}_{P\sim\mathcal{D}}[J(\tilde{\pi}|P)] - 2R_{max}\frac{\gamma\mathbb{E}_{P\sim\mathcal{D}}[d_{TV}(P^{\pi}_{\epsilon}\|P)]}{(1-\gamma)^2} - 4R_{max}\frac{d_{TV}(\pi,\tilde{\pi})}{(1-\gamma)^2}, \tag{7}$$

where R_{max} is the maximum reward, $d_{TV}(\pi,\tilde{\pi})$ indicates the total variation divergence between π and $\tilde{\pi}$, and P^{π}_{ϵ} is the worst state-adversarial transition kernels.

Theorem 1 indicates that the gap between the *average*- and the *worst*- case performance can be expressed using the MDP shift $\mathbb{E}_{P\sim\mathcal{D}}[d_{\mathrm{TV}}(P^{\pi}_{\epsilon}\|P)]$ and the policy evolution $d_{TV}(\pi,\tilde{\pi})$. The proof is in Appendix A.5. Note that the bound in Theorem 1 is loose because the value on the right hand side (RHS) of Equation 7 can be tiny. Specifically, the transition kernel probability shift $\mathbb{E}_{P\sim\mathcal{D}}[d_{\mathrm{TV}}(P^{\pi}_{\epsilon}\|P)]$ is multiplied by the total maximum return $\frac{R_{\max}}{1-\gamma}$, and the additional denominator $1-\gamma$ makes the value even smaller since γ is usually set to 0.99 in RL applications. As a result, the bound can be meaningless unless the worst-case MDP P^{π}_{ϵ} is very close to the average MDP.

Since state perturbation only perturbs states to nearby states, we consider the smoothness of the reward function and transition property to build a tight 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, no matter what the state-action pairs are considered. Formally, we define this smoothness property of transition kernels as:

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*

$$\|s - s'\| \leq \delta, \tag{8}$$

for all a and for all s, s' with $P(s'|s,a) > 0$.

With the assumption of Lipschitz continuity in reward function and smoothness of transition kernel, we arrive at the following bound:

Theorem 2 (Connecting Worst-Case and Average-Case Returns)**.** *Given a nominal MDP with two properties: (1) Reward function of corresponding Markov Reward Process (MRP) with respect to any policy is an L_r-Lipschitz function. (2) Nominal transition kernel P_0 has the smooth transition property δ, where $\|s - s'\|_2 \leq \delta, \forall a$ and $\forall P_0(s'|s,a) > 0$. Then, after updating the current policy π to a new policy $\tilde{\pi}$, the following bound holds:*

$$J(\tilde{\pi}|P_\epsilon^\pi) \geq J(\tilde{\pi}|P_\epsilon^{\pi,\alpha}) - \frac{4\gamma(\epsilon+\delta)L_r\alpha}{(1-\gamma)^3} - \frac{4(\gamma(\epsilon+\delta)L_r + (1-\gamma)^2 R_{max})d_{TV}(\pi,\tilde{\pi})}{(1-\gamma)^3}, \tag{9}$$

where $d_{TV}(\pi,\tilde{\pi})$ is total variation divergence between π and $\tilde{\pi}$, $P_\epsilon^{\pi,\alpha}$ is a relaxed state-adversarial transition kernel, and P_ϵ^π is a worst-case state-adversarial transition kernel.

The proof is provided in Appendix A.6. Notably, Theorem 2 holds for any relaxation parameter $\alpha \in [0,1]$. We now briefly discuss the technical challenges in the proof: (1) *Propagation of state perturbations across time*: The main difficulty lies in the fact that the difference of trajectories under different MDPs would increase in a rather nonlinear and complex manner as time evolves. (2) *Quantifying the difference in rewards among trajectories generated under different transition kernels:* To measure the difference in rewards under different MDPs, it is necessary to consider not only the probability difference at time t but also the difference in rewards at 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 are able to characterize a tight bound between the average-case and the worst-case performance.

The intuition of Theorem 2 can be expressed using the terms on the RHS of Equation 9. The first term is the average performance of all MDPs in the uncertainty set. The second term penalizes the large value of α because it implies that the relaxed MDP is close to the nominal environment. In other words, we expect the average case performance to be high while pushing the uncertainty set close to the worst-case MDP. Finally, the third term prevents a significant update in a single step by reducing the total variation divergence $d_{TV}(\pi,\tilde{\pi})$.

4.4 Online Adaptation of the Relaxation Parameter

We leverage Theorem 2 to address both the average-case and the 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 the unknowns α and π are correlated. The two tasks are optimized alternatively and iteratively. Details are as follows:

- **Lower-level task for average-case return**: On the lower level, we improve the policy by optimizing the objective $J(\pi_{\theta_t}|P_\epsilon^{\pi_{\theta_{t-1}},\alpha_t})$ under a fixed relaxation parameter α_t. This can be done by using any off-the-shelf RL algorithm (e.g., PPO with a clipped objective).
- **Upper-level task for worst-case return**: On the upper level, we design a meta objective $J_{\text{meta}}(\alpha_t)$ to represent the lower bound of the worst case performance (RHS of Equation 9). Hence, the task aims to find a relaxation parameter α_t that can maximize $J_{\text{meta}}(\alpha_t)$. To enable a stable training, we iteratively update α_t 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 $J(\pi_{\theta_t}|P_\epsilon^{\pi_{\theta_{t-1}}})$ (Equation 9). In the lower-level, a constant relaxation parameter α_t represents a specific distribution $\mathcal{D}$. It seeks to maximize the average return over all environments in the uncertainty set following distribution $\mathcal{D}$. In the upper-level, the optimization adjusts α to maximize this lower bound. On one hand, increasing α_t improves the average performance $J(\pi_{\theta_t}|P_\epsilon^{\pi_{\theta_{t-1}},\alpha_t})$ since the average-case moves toward a nominal environment, yet the price is increasing the MDP shift (i.e., the second term of RHS in Equation 9). On the other hand, decreasing α changes the performance and the penalty oppositely. Since π is weak initially and its performance gradually improves, the meta objective optimization tends to decrease and then increase α during training.

Algorithm 1 illustrates our implementation. We first update the policy π_{θ_t} to maximize the average-case return $J(\pi_{\theta_t}|P_\epsilon^{\pi_{\theta_{t-1}},\alpha_t})$ using the proximal policy optimization (PPO). Afterward, we update the relaxation parameter α to ensure that the worst-case return is higher than a specific bound (Equation 9). Note that 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 chose PPO as a base algorithm since it prevents the model from being updated significantly in a single step. It helps to control the penalty term $d_{TV}(\pi, \tilde{\pi})$ in Theorem 2. The implementation details are provided in Appendix A.7.

Algorithm 1: Relaxed State-Adversarial Policy Optimization

```
Input : MDP $(\mathcal{S}, \mathcal{A}, P_0, r, \gamma)$, Objective function $L$, step size parameter $\eta$, number of iterations
        $T$, $P_0$ is the nominal transition kernel, $\epsilon$-Neighborhood
1 Initialize the policy $\pi_{\theta_0}$ for $t = 0, \dots, T-1$ do
2     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 $s'_i \sim P_0(\cdot|s_i, a_i)$
3     Evaluate $J(\pi_{\theta_t}|P_\epsilon^{\pi_{\theta_{t-1}},\alpha_t})$
4     Update the policy to $\pi_{\theta_{t+1}}$ by applying multi-step SGD to the objective function as PPO
5     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 $s'_i \sim P_0(\cdot|s_i, a_i)$
6     Update the relaxation parameter to $\alpha_{t+1}$ via one SGD update with respect to the
        meta-objective
7 end
```

5 Experimental Results and Evaluations

We conducted two experiments on Mujoco (Todorov et al., 2012) to evaluate the performance of our relaxed state adversarial policy optimization (RAPPO). All the baselines and our method were implemented on the PPO (Schulman et al., 2017), and the default training parameters were used. In addition, the results were averaged from five different runs/seeds.

Robustness against Environmental Adversaries. We compared our RAPPO with the latest DR method, MRPO (Jiang et al., 2021), to evaluate its robustness against the uncertainty of environmental parameters[1]. Agents trained using the two methods were evaluated in the environments, in which the size and gravity were drifted in the range of 0.6 - 1.4. To simulate the situation that domain knowledge is unavailable, during training, MRPO perturbed mass and friction in the range of 0.8 - 1.2, and our RAPPO attacked the states by its value function. Figure 2 shows the subtractions of the rewards of the two methods. As can be seen, our RAPPO outperformed MRPO since state adversaries were more general than environmental adversaries. Agents trained by MRPO could perform poorly when the perturbations in the training and testing environments were different.

Robustness Against States Adversaries. We compared our RAPPO with SCPPO (Kuang et al., 2021) to evaluate its robustness against state adversaries. Both of the methods perturb states to improve agents' robustness. We also included vanilla PPO in the experiment because it is the base algorithm of RAPPO and SCPPO. To achieve a fair comparison, the parameters used in RAPPO and SCPPO were the same. Specifically, we set ϵ to 0.015, 0.002, 0.03, 0.001, and 0.005 to the environments of HalfCheetah-v2, Hopper-v2, Ant-v2, Walker-v2, and Humanoid2d-v2, respectively. The parameters were chosen according to the variance of actions in the environments.

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

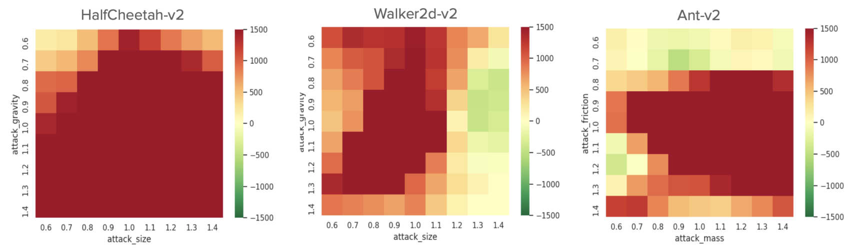

Figure 2: We perturbed the size and gravity of the environments and measured the mean rewards achieved by the agents trained using a DR method, MRPO, and our RAPPO. The heatmaps show the subtractions of MRPO's reward from RAPPO's reward. The higher value (red) indicates that RAPPO outperformed MRPO.

Environment		Nominal	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.015$	$\sigma = 0.02$	$\sigma = 0.025$
HalfCheetah	PPO	5286 ± 1004	4280 ± 1552	3186 ± 1875	1996 ± 1743	1256 ± 1251	819 ± 1003
	SCPPO	6157 ± 709	5046 ± 1533	3367 ± 2090	1795 ± 1758	875 6± 1259	60 ± 791
	RAPPO-C	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	PPO	3330 ± 619	1357 ±787	615 ± 194	494 ± 151	462 ± 141	417 ± 131
	SCPPO	2644 ± 951	1369 ± 620	876 ± 347	773 ± 357	782 ± 437	732 ± 412
	RAPPO-C	2497 ± 1041	1626 ± 1064	1122 ± 817	795 ± 426	704 ± 393	521 ± 201
	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	PPO	3781 ± 1165	1564 ± 1285	903 ±52	763 ±353	628 ±241	575 ± 222
	SCPPO	4313 ±979	2647 ±1584	1604 ±1082	985 ±704	772 ±492	666 ± 412
	RAPPO-C	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	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.03$	$\sigma = 0.04$	$\sigma = 0.05$
Ant	PPO	6075 ± 889	**4489 ± 1342**	2071 ± 1156	1016 ± 523	703 ± 283	615 ± 248
	SCPPO	5915 ± 728	4203 ±1 441	1661 ± 951	831 ± 398	609 ± 320	489 ± 273
	RAPPO-C	5954 ± 746	4380 ± 1299	2077 ± 1117	1010 ± 568	668 ± 282	537 ± 225
	RAPPO	6022 ± 698	4381 ± 1357	**2284 ± 1225**	**1038 ± 553**	**733 ± 255**	**672 ± 219**
		Nominal	$\sigma = 0.003$	$\sigma = 0.004$	$\sigma = 0.005$	$\sigma = 0.006$	$\sigma = 0.007$
Humanoid	PPO	5357 ± 1618	3033 ± 1834	2373 ± 1742	1802 ± 1446	1287 ± 1068	939 ± 750
	SCPPO	5410 ± 1340	3196 ± 1781	2387 ± 1472	1783 ± 1256	1271 ± 838	1060 ± 678
	RAPPO-C	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**

Table 1: We compared the performance of agents trained using PPO, SCPPO, our RAPPO-C (i.e., constant relaxation parameter α) and RAPPO in Mujoco environments under multiple degrees of state perturbation. Mean and Standard deviations are reported.

Table 1 shows the testing results. We attacked the agents using their respective value functions under multiple strengths. Specifically, we repeated the experiments from 5 different seeds and generated 50 trajectories for each seed from different initial states for evaluation. The means and standard deviations of the rewards were reported. Clearly, the results fulfilled Lemma 1, where agents' performance decreased as the strength of attack increased. In addition, our RAPPO was competitive to PPO and SCPPO in nominal environments, and its performance decreased the slowest as the strength of attack increased. It deserves noting that the attacks in the last two columns of Table 1 were stronger than that of the worst-case. Our RAPPO performed the best in the environments.

Extending SAPPO Using Relaxed State Adversaries. While our RAPPO successfully improves the robustness of agents against state adversaries, a classical method, SAPPO (Zhang et al., 2020), can help agents against the perturbation of state observations. We thus extended SAPPO by adopting our relaxed state adversarial attacks during training and evaluated its effectiveness. Similarly, we compared the methods on the trajectories of 5 seeds and 50 initial states. Table 2 shows the results. As indicated, the extended RA_SAPPO outperformed SAPPO in most of the environments, particularly under strong attacks.

Steady Improvements of the Average and Worst Case Environments. We apply a *bi-level* approach to optimize the average and worst-case environments during training. To verify 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,

Environment		Nominal	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.015$	$\sigma = 0.02$	$\sigma = 0.025$
Halfcheetah	SAPPO	4928 ± 370	4765 ± 359	4485 ±394	4036 ± 582	3282 ± 1175	2533 ± 1495
	RA_SAPPO	5784 ± 1081	**5371 ± 1323**	**4874 ± 1311**	**4775 ± 933**	**4106 ± 1273**	**3152 ± 1750**
		Nominal	$\sigma = 0.001$	$\sigma = 0.0015$	$\sigma = 0.002$	$\sigma = 0.0025$	$\sigma = 0.003$
Walker	SAPPO	4135 ± 962	2211 ± 1322	940 ± 405	673 ± 318	667 ± 326	614 ± 311
	RA_SAPPO	4539 ± 1014	**3229 ± 1590**	**1564 ± 1410**	**921 ± 789**	**832 ± 806**	**746 ± 772**
		Nominal	$\sigma = 0.003$	$\sigma = 0.004$	$\sigma = 0.005$	$\sigma = 0.006$	$\sigma = 0.007$
Humanoid	SAPPO	5736 ± 1194	3690 ± 2068	2926 ± 1956	1944 ± 1438	1409 ± 1098	1156 ± 789
	RA_SAPPO	5320 ± 1164	**3960 ± 2082**	**3335 ± 2117**	**2882 ± 2066**	**2129 ± 1776**	**1567 ± 1474**

Table 2: We extended the SAPPO by adopting the relaxed state adversarial strategy and evaluated whether the extension (i.e., RA_SAPPO) can improve the agents' robustness against state perturbation. Mean and Standard deviations are reported.

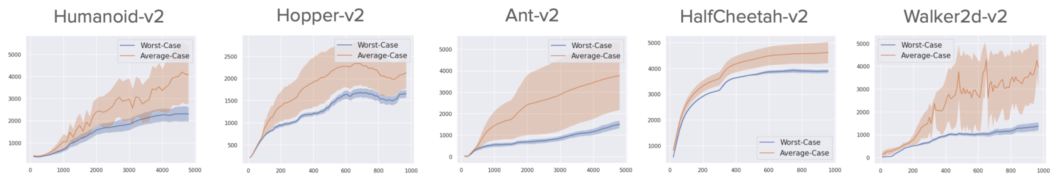

Figure 3: Our RAPPO can steadily improve the average-case and worst-case rewards during training. The solid lines and shaded areas indicate the mean and standard deviation of the rewards, respectively. Note that the variance of the average-case rewards is caused by different adversarial strengths.

perturbed 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 uniformly distributed between 0 and ϵ. In total, the rewards of 50×10 trajectories were averaged. Figure 3 shows that our 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 reasonable because of different adversarial strengths.

The value of the relaxation parameter α. Our meta-objective optimization determines the relaxation parameter α (Equation 6) to control the strengths of state adversaries during training. While α is unknown, an intuitive idea is to consider α a hyper-parameter and let users specify the value. However, we point out that the value of α should vary at different training stages since agents are weak initially and can perform well after training. To verify that a dynamic α is over a constant α (i.e., RAPPO-C), we evaluated the performance of agents under state perturbed environments. In the experiments, we set $\alpha = 0.5$ for RAPPO-C since it is in the middle of nominal and worst-case environments. The remaining parameters between the methods were exactly the same. As indicated in Table 1, RAPPO outperformed RAPPO-C without a doubt. We also refer readers to Appendix A.8 for the dynamics of α during training.

6 Conclusions

We have presented a relaxed state adversarial policy optimization to improve the robustness of agents against the uncertainty of environments. Compared to the methods in DR, we perturbed states using the adversarial attack so as to decouple randomization from simulators. Neither prior knowledge of selecting environmental parameters nor prior assumption of parameter distribution are needed. In addition, we introduced a relaxation strategy to tackle the over-conservative problem caused by state adversarial attacks. Our policy optimization maximizes rewards in the average-case while holding the lower-bound rewards in the worst-case environments simultaneously. Experiment results and theoretical proofs demonstrate the effectiveness of our method.

Limitations and Future Work. Our relaxation method is state-independent, in which the value of α is adjusted according to the overall performance of policy. Since the degrees of difficulty vary from states to states, it will be interesting to investigate the state-dependent relaxation method. In addition, we currently assume that each dimension of states is equally important, which may not be the case. We will also explore the weight of each dimension when perturbing states in the future.

REFERENCES

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

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

Bahram Behzadian, Marek Petrik, and Chin Pang Ho. Fast algorithms for L_∞-constrained S-rectangular robust MDPs. *Advances in Neural Information Processing Systems*, 34:25982–25992, 2021.

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

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems*, 28, 2015.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

Esther Derman, Matthieu Geist, and Shie Mannor. Twice regularized MDPs and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34, 2021.

Mathieu Fehr, Olivier Buffet, Vincent Thomas, and Jilles Dibangoye. ρ-POMDPs have Lipschitz-continuous ϵ-optimal value functions. *Advances in neural information processing systems*, 31, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Julien Grand-Clément and Christian Kroer. First-order methods for wasserstein distributionally robust MDP. *arXiv preprint arXiv:2009.06790*, 2020.

Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Fast bellman updates for robust MDPs. In *International Conference on Machine Learning*, pp. 1979–1988. PMLR, 2018.

Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. 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.

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

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

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

Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. Monotonic robust policy optimization with model discrepancy. In *International Conference on Machine Learning*, pp. 4951–4960. PMLR, 2021.

Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Yufei Kuang, Miao Lu, Jie Wang, Qi Zhou, Bin Li, and Houqiang Li. Learning robust policy against disturbance in transition dynamics via state-conservative policy optimization. *arXiv preprint arXiv:2112.10513*, 2021.

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

Chun Kai Ling, Kian Hsiang Low, and Patrick Jaillet. Gaussian process planning with lipschitz continuous reward functions: Towards unifying bayesian optimization, active learning, and beyond. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

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

Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel, and Tsui-Wei Weng. Robust deep reinforcement learning through adversarial loss. *Advances in Neural Information Processing Systems*, 34, 2021.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2040–2042, 2018.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. *Advances in neural information processing systems*, 30, 2017.

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

Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pp. 8707–8718. PMLR, 2020.

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

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

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 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.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

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

Yuh-Shyang Wang, Tsui-Wei Weng, and Luca Daniel. Verification of neural network control policy under persistent adversarial perturbation. *arXiv preprint arXiv:1908.06353*, 2019.

Tsui-Wei Weng, Krishnamurthy Dj Dvijotham, Jonathan Uesato, Kai Xiao, Sven Gowal, Robert Stanforth, and Pushmeet Kohli. Toward evaluating robustness of deep reinforcement learning with continuous control. In *International Conference on Learning Representations*, 2019.

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

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021.