Learning Robust Policy against Disturbance in Transition Dynamics via State-Conservative Policy Optimization

Yufei Kuang[1], Miao Lu[1], Jie Wang[1,2]*, Qi Zhou[1], Bin Li[1], Houqiang Li[1,2]

[1]CAS Key Laboratory of Technology in GIPAS, University of Science and Technology of China
[2]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
{yfkuang, lumiao, zhouqida}@mail.ustc.edu.cn
{jiewangx, binli, lihq}@ustc.edu.cn

Abstract

Deep reinforcement learning algorithms can perform poorly in real-world tasks due to the discrepancy between source and target environments. This discrepancy is commonly viewed as the disturbance in transition dynamics. Many existing algorithms learn robust policies by modeling the disturbance and applying it to source environments during training, which usually requires prior knowledge about the disturbance and control of simulators. However, these algorithms can fail in scenarios where the disturbance from target environments is unknown or is intractable to model in simulators. To tackle this problem, we propose a novel model-free actor-critic algorithm—namely, **S**tate-**C**onservative **P**olicy **O**ptimization (SCPO)—to learn robust policies *without modeling the disturbance in advance*. Specifically, SCPO reduces the disturbance in transition dynamics to that in state space and then approximates it by a simple gradient-based regularizer. The appealing features of SCPO include that it is simple to implement and does not require additional knowledge about the disturbance or specially designed simulators. Experiments in several robot control tasks demonstrate that SCPO learns robust policies against the disturbance in transition dynamics.

1 Introduction

Deep reinforcement learning (DRL) has achieved remarkable success in many complex control tasks (Mnih et al. 2015; Lillicrap et al. 2016; Tobin et al. 2017; OpenAI et al. 2019). However, existing DRL algorithms tend to perform poorly in real-world tasks due to the environment discrepancy (Rajeswaran et al. 2017; Jiang et al. 2021). For example, policies trained to control robots in source environments can fail to generalize in target environments with slightly changed physical parameters (e.g., mass and friction). This discrepancy usually comes from test-generalization and simulation-transfer in real-world tasks, and it is commonly viewed as the disturbance in transition dynamics (Pinto et al. 2017; Tessler, Efroni, and Mannor 2019). Therefore, research on learning robust policies against the disturbance in transition dynamics is receiving increasing attention.

In order to learn robust policies, many previous algorithms model the disturbance and apply it to source environments during training. One simple and effective algorithm is

---

*Corresponding author.

domain randomization (DR) (Tobin et al. 2017), which randomizes the parameters in source environments to generate an ensemble of similar transition dynamics. DR uses these dynamics to approximate the potential disturbance in target environments. Another line of research is based on robust Markov decision process (RMDP) theory (Iyengar 2005), which considers the worst-case disturbance in transition dynamics at each step during training. For example, robust adversarial reinforcement learning (RARL) algorithm (Pinto et al. 2017) models the disturbance as trainable forces in source environments based on the insight that environment discrepancy can be viewed as extra forces, and then RARL learns the worst-case disturbance by adversarial training.

Learning robust policies by modeling the disturbance in advance, though effective, has two limitations in practice. First, these algorithms usually require task-specific prior knowledge to model the disturbance (Mankowitz et al. 2020). For example, to randomize the environmental parameters during training, DR requires a pre-given uncertainty set that contains specific parameters and their corresponding variation ranges; to approximate the disturbance in target environments, RARL employs extra forces that are manually designed for each task. However, when we train policies for new tasks or test in unseen environments, we may lack enough knowledge about the disturbance (Tessler, Efroni, and Mannor 2019). Second, these algorithms usually assume control of specially designed simulators (Packer et al. 2018), as disturbing the agents during training in the real world can be dangerous. However, sometimes building new simulators can be expensive, and sometimes the disturbance is even intractable to model in simulators (e.g., disturbance caused by wear-and-tear errors and aerodynamic effects in robot control tasks) (Abdullah et al. 2019). In general, learning robust policies by modeling the disturbance and applying it to source environments can be impractical in many scenarios.

In this paper, we propose a novel algorithm to learn robust policies without modeling the disturbance in advance. First, we reduce the disturbance in transition dynamics to that in state space based on the intuition that any disturbance in transition dynamics eventually influences the process via the change of future states. Then, we propose the **S**tate-**C**onservative **M**arkov **D**ecision **P**rocess (SC-MDP), which considers the worst-case disturbance in state space at each step. Compared with the RMDP, SC-MDP reduces the origi-

nal constrained optimization problem in transition dynamics to a simpler one in state space, whose solving process does not require taking infimum over a subset of the infinite dimensional transition probability space. Finally, we propose a model-free actor-critic algorithm, called **S**tate-**C**onservative **P**olicy **O**ptimization (SCPO), which approximates the disturbance in state space by a simple gradient-based regularizer and learns robust policies by maximizing the objective of SC-MDP. The appealing features of SCPO include that it is simple to implement and does not require task-specific prior knowledge and specially designed simulators to model the disturbance. Experiments in MuJoCo benchmarks show that SCPO consistently learns robust policies against different types of disturbance in transition dynamics.

2 Related Work

In this section, we discuss related work that aims to learn robust policies from different perspectives. Note that we mainly focus on learning robust policies against the disturbance in transition dynamics in this paper.

Robustness against the Disturbance in Transition Dynamics The disturbance in transition dynamics mainly comes from test-generalization and simulation-transfer (Pinto et al. 2017; Rajeswaran et al. 2017). Many previous model-free algorithms learn robust policies by modeling the disturbance and applying it to source environments during training. Tobin et al. (2017) propose domain randomization (DR) to generate environments with different transition dynamics. Iyengar (2005) propose robust Markov decision process (RMDP), which maximizes cumulative rewards under the worst-case disturbance. Based on RMDP, Mankowitz et al. (2020) propose robust MPO (R-MPO) to learn robust policies with pre-given uncertainty sets of environmental parameters; Jiang et al. (2021) propose monotonic robust policy optimization (MRPO) with a theoretical performance lower bound; Abdullah et al. (2019) propose the Wasserstein robust reinforcement learning (WR^{2}L) to train the environment parameters with protagonists jointly. Compared with these algorithms, SCPO does not require specific priors and control of simulators to model the disturbance in advance.

Robustness against the Disturbance in State Observations The disturbance in state observations can originate from sensor errors and equipment inaccuracy (Zhang et al. 2020). Mandlekar et al. (2017) learn robust policies by adversarial training with FGSM based attacks, and Zhang et al. (2020) derive a robust policy regularizer that relates to total variation distance or KL-divergence on perturbed policies. Note that the disturbance in state observations does not affect the ground-truth states and the underlying transition dynamics. Thus, there are gaps betweem the algorithms designed for these two different tasks (Zhang et al. 2020).

Robustness against the Disturbance in Action Space The disturbance in action space can come from controller inaccuracy and environment noise. OpenAI et al. (2019) propose the random network adversary (RNA) algorithm, which injects noise in action space during training using a neural network with randomly sampled weights; Tessler, Efroni, and Mannor (2019) propose probabilistic action robust MDP (PR-MDP) and noisy action robust MDP (NR-MDP) to model two types of disturbance in action space and then solve them by adversarial training. Sometimes we also use the disturbance in action space to approximate the unmodeled disturbance in transition dynamics, even though the disturbance in transition dynamics is not necessarily originated from the disturbance in action space. We compare SCPO with the action robust algorithm in our experiments.

3 Preliminaries

In this section, we briefly introduce the preliminaries.

Markov Decision Process (MDP) An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. Here $\mathcal{S} \subset \mathbb{R}^m$ is the state space, $\mathcal{A} \subset \mathbb{R}^n$ is the action space, and $p : \mathcal{S} \times \mathcal{A} \to \Delta_\mathcal{S}$ is the transition dynamic, where $\Delta_\mathcal{S}$ is the set of Borel probability measures on $\mathcal{S}$. The reward function $r : \mathcal{S} \times \mathcal{A} \to [R_{\min}, R_{\max}]$ is bounded, and $\gamma \in (0, 1)$ is the discount factor. Let $\pi : \mathcal{S} \to \Delta_\mathcal{A}$ be a stationary policy, where $\Delta_\mathcal{A}$ is the set of Borel probability measures on $\mathcal{A}$. For convenience, we overload the notation to let $\pi(\cdot|s)$ and $p(\cdot|s, a)$ also denote the probability density function (PDF) without ambiguity.

Robust Markov decision process (RMDP) The RMDP considers the disturbance in transition dynamics at each step. It is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, r, γ are defined as above and $\mathcal{P}$ is an uncertainty set of possible transitions. Specifically, for each $(s, a) \in \mathcal{S} \times \mathcal{A}$, $\mathcal{P}(s, a)$ is a set of transitions $p(s'|s, a) : \mathcal{S} \times \mathcal{A} \mapsto \Delta_\mathcal{S}$. When the agent is at state s and takes action a, the next state s' is determined by one of the transitions in $\mathcal{P}(s, a)$. By definition, the uncertainty set $\mathcal{P}$ described all possible disturbance in transition dynamics. RMDP aims to learn a policy that maximizes the expected return under the worst-case disturbance (Nilim and Ghaoui 2004). The objective of RMDP is defined as

$$J_\mathcal{P}(\pi) \triangleq \inf_{p \in \mathcal{P}} \mathbb{E}_{p,\pi}[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t)]. \tag{1}$$

That is, taking minimum over all possible probability measures induced by π and $\mathcal{P}$ on the trajectory space (Iyengar 2005). Based on the objective, the robust value functions are defined as $V_\mathcal{P}^\pi(s) \triangleq \inf_{p \in \mathcal{P}} \mathbb{E}_{p,\pi}[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t)|s_0 = s]$ and $Q_\mathcal{P}^\pi(s, a) \triangleq \inf_{p \in \mathcal{P}} \mathbb{E}_{p,\pi}[\sum_{t=0}^{+\infty} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a]$. Note that $Q_\mathcal{P}^\pi$ is the fixed point of the following robust Bellman operator (which is a contraction on $\mathbb{R}^{|\mathcal{S} \times \mathcal{A}|}$)

$$\mathcal{T}_\mathcal{P}^\pi Q \triangleq r(s, a)$$
$$+ \gamma \inf_{p(\cdot|s,a) \in \mathcal{P}(s,a)} \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')}[Q(s', a')]. \tag{2}$$

Correspondingly, the optimal policy and the optimal action-value function is defined to be $Q_\mathcal{P}^{\pi^*} \triangleq \sup_\pi Q_\mathcal{P}^\pi$.

Wasserstein Uncertainty Set A bounded uncertainty set makes the problem well-defined, as optimizing over arbitrary dynamics only results in non-performing system. The uncertainty set $\mathcal{P}(s, a)$ has recently been described using the Wasserstein distance by many researchers (Abdullah et al. 2019; Hou et al. 2020). Let $p_0(s'|s, a)$ denote the transition probability in the source environment, which is assumed to be fixed. Then, the Wasserstein uncertainty set is defined as

$$\mathcal{P}_\epsilon \triangleq \left\{ p : \mathbb{W}^{(p)}\big(p(\cdot|s, a), p_0(\cdot|s, a)\big) < \epsilon, \ \forall(s, a) \right\}, \tag{3}$$

where $\mathbb{W}^{(p)}$ is the pth-order Wasserstein metric on $\Delta_{\mathcal{S}}$ given by $\mathbb{W}^p(\mu, \nu) \triangleq (\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{S} \times \mathcal{S}} d(x,y)^p \, d\gamma)^{1/p}$, d is a distance metric on $\mathcal{S}$, and $\Gamma(\mu, \nu)$ refers to all the couplings of μ and ν. By recent progresses in optimal transport (Blanchet and Murthy 2019), the robust Bellman operator (2), when described by Wasserstein uncertainty set (3), can be rewrite without explicitly taking infimum over a set of probability measures. We show the result in the following proposition, which serves as a motivation to our method.

Proposition 1. *When the uncertainty set is described by Wasserstein distance of order p as in Equation (3), the robust Bellman operator (2) is equivalent to*

$$\mathcal{T}_{\mathcal{P}_{\epsilon}}^{\pi} Q(s,a) = r(s,a) + \gamma \sup_{\lambda \geq 0} \mathbb{E}_{\tilde{s} \sim p_0(\cdot|s,a)}$$
$$\left\{ \inf_{s' \in \mathcal{S}} \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')] + \lambda \big(d(s', \tilde{s})^p - \epsilon \big) \right\}. \quad (4)$$

4 From RMDP to State-Conservative MDP

In this section, we propose a new state-conservative objective and the corresponding state-conservative Markov decision process (SC-MDP). SC-MDP reduces the disturbance in transition dynamics to that in state space, allowing us to learn robust policies without full prior knowledge about the disturbance. Before we formally propose the new objective, we first explain the motivation from state disturbance.

4.1 A View from State Disturbance

Consider the robust Bellman operator (2) and its Wasserstein variant (4). We can write the empirical version of (4) as

$$\widehat{\mathcal{T}}_{\mathcal{P}_{\epsilon}}^{\pi} Q(s_t, a_t) = r(s_t, a_t) + \gamma \sup_{\lambda \geq 0} \quad (5)$$
$$\left\{ \inf_{s' \in \mathcal{S}} \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')] + \lambda \left(d(s', s_{t+1})^p - \epsilon \right) \right\},$$

where $s_{t+1} \sim p_0(\cdot|s_t, a_t)$. Now assume that both the value function $V(s') \triangleq \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]$ and the pth-order distance metric $d(s', s_{t+1})^p$ are convex functions of the variable s' in Equation (4). By the duality theory (Boyd and Vandenberghe 2004), Equation (5) is equivalent to the following form (detailed proof in Appendix A.1):

$$\widehat{\mathcal{T}}_{\mathcal{P}_{\epsilon}}^{\pi} Q(s_t, a_t) = r(s_t, a_t) + $$
$$\gamma \inf_{s' \in B_{\epsilon}(s_{t+1})} \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q(s',a')]. \quad (6)$$

Such a form involves taking infimum of the future state-value $V^{\pi}(s')$ over all s' in the ϵ-ball $B_{\epsilon}(s_{t+1}) \triangleq \{s' \in \mathcal{S} : d(s', s_{t+1})^p \leq \epsilon\}$. Note that the convexity of $V(s')$ can hold under linear and deterministic assumptions on rewards, transitions and policies when the MDP is of finite horizon.

In general, the state-value function $V^{\pi}(s')$ is not necessarily convex. However, this observation gives us a different view on promoting robustness against the disturbance in transition dynamics. Intuitively, any disturbance in transition p finally influences the process via the change of future states. Thus, instead of considering the worst-case perturbed transition as in the robust Bellman operator (2), we

can directly consider the worst-case perturbed next states as in Equation (6). In other words, we can learn robust policies by considering the worst state-value over *all possible states* close to $s_{t+1} \sim p_0(\cdot|s_t, a_t)$. We refer to this method as *state disturbance*, which is more compatible in practice since it only involves finding a constraint minimum in the finite-dimensional state space and thus does not require priors to model the disturbance in advance. Starting from this view, we propose a new objective in Section 4.2 and further extend it to a continuous control RL algorithm in Section 5.

4.2 State-Conservative Markov Decision Process

Now we formally propose the *state-conservative objective* based on the state disturbance method in Section 4.1. Different from the objective of RMDP in (1), we define it as

$$J_{\epsilon\text{-}\mathcal{S}}(\pi) \triangleq \mathbb{E}_{s_0 \sim d_0} \big[\inf_{s_0' \in B_{\epsilon}(s_0)} \mathbb{E}_{a_0' \sim \pi(\cdot|s_0')}[r(s_0', a_0')$$
$$+ \gamma \mathbb{E}_{s_1 \sim p_0(\cdot|s_0', a_0')} \big[\inf_{s_1' \in B_{\epsilon}(s_1)} \mathbb{E}_{a_1' \sim \pi(\cdot|s_1')}[r(s_1', a_1')$$
$$+ \gamma \mathbb{E}_{s_2 \sim p_0(\cdot|s_1', a_1')} \big[\inf_{s_2' \in B_{\epsilon}(s_2)} \mathbb{E}_{a_2' \sim \pi(\cdot|s_2')}[r(s_2', a_2')$$
$$+ \cdots \big] \cdots \big] \quad (7)$$

where d_0 and p_0 represent the initial distribution and the transition dynamic in source environments, ϵ is a non-negative real controlling the degree of state disturbance, and $B_{\epsilon}(s) \triangleq \{s' \in \mathcal{S} : d(s', s) \leq \epsilon\}$ is the ϵ-ball in $\mathcal{S} \subseteq \mathbb{R}^m$ induced by a distance metric $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$. Note that by setting ϵ to 0, our objective recovers the original reinforcement learning objective. We call the Markov decision process aiming to maximize the objective (7) as state-conservative Markov decision process (SC-MDP).

Understand SC-MDP We illustrate the intuition of SC-MDP in Figure 1. Consider an MDP with one-dimensional state space. Suppose its rewards only depend on s. Then, the original objective (i.e., the cumulative rewards) in Figure 1(a) encourages the stationary state distribution $\rho(s)$ to concentrate on states with higher rewards, while the state-conservative objective in Figure 1(b) encourages $\rho(s)$ to concentrate on states that are more stable under disturbance.

Relation to RMDP Our newly proposed SC-MDP coincides with RMDP in specific settings. If we only consider deterministic transitions and let the uncertainty set of RMDP bounded by Wasserstein distance, then one can easily check that our state-conservative objective (7) equals to the objective (1) of RMDP. Though this equivalence does not necessarily hold in general (discussions in Appendix B.1), the new objective helps us design algorithms to learn robust policies.

4.3 State-Conservative Policy Iteration

Given a fixed policy π, we aim to evaluate its value iteratively. Based on the newly proposed objective (7), we define the corresponding action-value function by

$$Q_{\epsilon\text{-}\mathcal{S}}^{\pi}(s,a) \triangleq r(s,a) + \gamma \mathbb{E}_{s_1 \sim p_0(\cdot|s,a)} \big[$$
$$\inf_{s_1' \in B_{\epsilon}(s_1)} \mathbb{E}_{a_1' \sim \pi(\cdot|s_1')}[r(s_1', a_1') + \gamma \mathbb{E}_{s_2 \sim p_0(\cdot|s_1', a_1')} \big[$$
$$\inf_{s_2' \in B_{\epsilon}(s_2)} \mathbb{E}_{a_2' \sim \pi(\cdot|s_2')}[r(s_2', a_2') + \cdots] \cdots \big]. \quad (8)$$
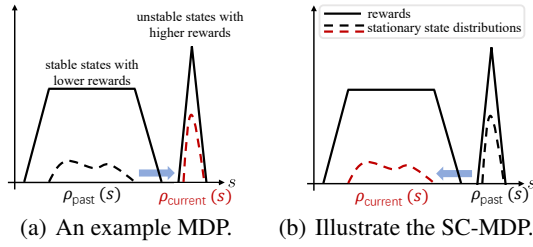
(a) An example MDP. (b) Illustrate the SC-MDP.

Figure 1: An illustration of the state-conservative MDP.

We call $Q^\pi_{\epsilon\text{-}\mathcal{S}}$ the state-conservative action-value function of policy π. It satisfies the following Bellman equation

$$Q^\pi_{\epsilon\text{-}\mathcal{S}}(s,a) = r(s,a)+$$
$$\gamma \mathbb{E}_{\tilde{s}\sim p_0(\cdot|s,a)} \left[\inf_{s'\in B_\epsilon(\tilde{s})} \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q^\pi_{\epsilon\text{-}\mathcal{S}}(s',a')] \right]. \quad (9)$$

Thus, $Q^\pi_{\epsilon\text{-}\mathcal{S}}$ is the fixed point of the following operator

$$\mathbb{T}^\pi_\epsilon Q(s,a) = r(s,a)+$$
$$\gamma \mathbb{E}_{\tilde{s}\sim p_0(\cdot|s,a)} \left[\inf_{s'\in B_\epsilon(\tilde{s})} \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q(s',a')] \right]. \quad (10)$$

We refer to $\mathbb{T}^\pi_\epsilon$ as the state-conservative Bellman operator. Now we show that $\mathbb{T}^\pi_\epsilon$ is a contraction mapping.

Proposition 2 (Contraction Mapping). *For any $\epsilon \geq 0$ and any fixed policy π, the state-conservative Bellman operator $\mathbb{T}^\pi_\epsilon$ in (10) is a contraction mapping on $(\mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}, \|\cdot\|_\infty)$. Thus, $\mathbb{T}^\pi_\epsilon$ has a unique fixed point, which is just the $Q^\pi_{\epsilon\text{-}\mathcal{S}}$.*

The proof of Proposition 2 is in Appendix A.2. Based on it, we can derive a policy evaluation procedure, which starts from arbitrary $Q^0 \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ and obtain $Q^\pi_{\epsilon\text{-}\mathcal{S}}$ by iteratively calculating $Q^{k+1} = \mathbb{T}^\pi_\epsilon Q^k$. We call this procedure *state-conservative policy evaluation* (SC-PE). After evaluating the old policy, we introduce the policy improvement step, which greedily chooses policies state-wisely to maximize the state-conservative objective starting from given states.

Proposition 3 (Policy Improvement). *Suppose $Q^{\pi_{old}}_{\epsilon\text{-}\mathcal{S}}$ is the action-value function of π_{old} and π_{new} is chosen greedily by:*

$$\pi_{new}(\cdot|s) \triangleq \arg\max_{\pi\in\Delta_\mathcal{A}} \inf_{s'\in B_\epsilon(s)} \mathbb{E}_{a'\sim\pi}[Q^{\pi_{old}}_{\epsilon\text{-}\mathcal{S}}(s',a')] \quad (11)$$

Then $Q^{\pi_{new}}(s,a) \geq Q^{\pi_{old}}(s,a)$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$.

The proof of Proposition 3 is in Appendix A.3. Now we can derive a policy iteration algorithm that alternates between the state-conservative policy evaluation and improvement steps introduced above. We call it *state-conservative policy iteration* (SC-PI) and show it in Algorithm 1 in Appendix B.2. By setting ϵ to 0, SC-PI recovers the original policy iteration algorithm (Sutton and Barto 2018).

5 State-Conservative Policy Optimization

In this section, we extend the SC-PI algorithm to a model-free actor-critic algorithm, called state-conservative policy optimization (SCPO), in continuous action space. First, we introduce the main components of SCPO. Then, we propose a simple and efficient gradient-based implementation of SCPO. Finally, we apply SCPO to the soft actor-critic (SAC) algorithm (Haarnoja et al. 2018) and present the state-conservative soft actor-critic (SC-SAC) as an instance.

5.1 State-Conservative Policy Optimization

Let Q_θ denote the Q-function parameterized by θ, π_ϕ denote the policy parameterized by ϕ, and $\mathcal{D}$ denote the collected data. Then, the state-conservative policy optimization (SCPO) algorithm mainly includes a state-conservative policy evaluation step to *train the critic* and a state-conservative policy improvement step to *train the actor*. We conclude the pseudo code of SCPO in Algorithm 2 in Appendix B.3.

State-Conservative Policy Evaluation In this step, we train the critic Q_θ by minimizing the Bellman residual

$$J_{\epsilon\text{-}Q}(\theta) \triangleq \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}} \left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \widehat{Q}(s_t,a_t)\right)^2 \right], \quad (12)$$

where the learning target $\widehat{Q}$ is defined as

$$\widehat{Q}(s_t,a_t) \triangleq r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p_0}\left[\widehat{V}(s_{t+1})\right], \quad (13)$$
$$\widehat{V}(s_{t+1}) \triangleq \inf_{s'\in B_\epsilon(s_{t+1})} \mathbb{E}_{a'\sim\pi_\phi(\cdot|s')}\left[Q_\theta(s',a')\right]. \quad (14)$$

State-Conservative Policy Improvement In this step, we train the actor π_ϕ by maximizing the objective of SC-MDP

$$J_{\epsilon\text{-}\pi}(\phi) \triangleq \mathbb{E}_{s_t\sim D} \left[\inf_{s\in B_\epsilon(s_t)} \mathbb{E}_{a\sim\pi_\phi(\cdot|s)}\left[Q_\theta(s,a)\right] \right]. \quad (15)$$

The ∞-Norm Based ϵ-Ball. We choose the ∞-norm as the distance metric on $\mathcal{S}$ to specify $B_\epsilon(s)$ in SCPO, i.e.,

$$B_\epsilon(s) \triangleq \{s' \in \mathcal{S} : \|s'-s\|_\infty \leq \epsilon\}. \quad (16)$$

Note that the ∞-norm is commonly used to measure noise and disturbance as it indicates that *the intensity of the disturbance in each dimension of the state space $\mathcal{S}$ is independent*. Besides, ∞-norm based ϵ-ball allows us to propose a simple and efficient implementation of SCPO in the next section.

5.2 Gradient Based Implementation

To implement the SCPO algorithm proposed in Section 5.1, we need to solve the following constrained optimization problem which appears in Equation (12) and Equation (15),

$$\inf_{s'\in B_\epsilon(s)} \mathbb{E}_{a'\sim\pi_\phi(\cdot|s')}[Q_\theta(s',a')]. \quad (17)$$

A natural idea to solve Equation (17) is to use the stochastic gradient descent (SGD) method in $B_\epsilon(s)$. However, both performing SGD repeatedly for each batch and estimating $\mathbb{E}_{a'\sim\pi(\cdot|s')}$ by sampling actions from $\pi(\cdot|s')$ are computationally expensive in practice. To tackle this problem, we propose a gradient based method that *approximately* solves the constrained optimization problem (17) *efficiently*.

Gradient Based Regularizer For simplicity, we define $U_{\theta,\phi}(s) \triangleq \mathbb{E}_{a\sim\pi_\phi(\cdot|s)}[Q_\theta(s,a)]$. By Taylor's Theorem, for given $s \in \mathcal{S}$, we can expand $U_{\theta,\phi}(s')$ in $B_\epsilon(s)$ as

$$U_{\theta,\phi}(s) + \langle\nabla_s U_{\theta,\phi}(s), s'-s\rangle + o(\|s'-s\|_2). \quad (18)$$

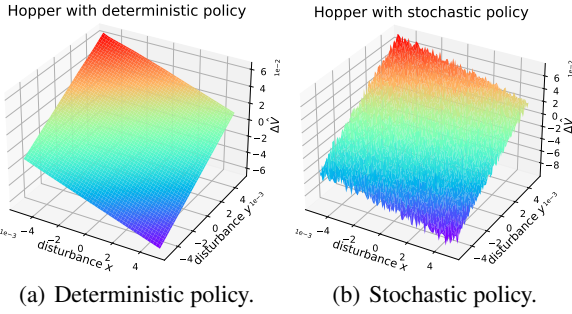(a) Deterministic policy. (b) Stochastic policy.

Figure 2: Visualize the local linear property of critics in SAC (notations refer to Section 5.3). We randomly choose a state s and two mutually perpendicular directions x, y in the normalized state space, and visualize the value change $\Delta \hat{V} \triangleq \mathbb{E}_\delta \left[\Delta Q_\theta^{\min} \left(s', f_\phi \left(\delta; s \right) \right) \right]$ in span(x, y). In Figure 2(a), we make the trained policy in SAC to be deterministic (i.e., let $\delta = 0$). In Figure 2(b), we sample $\delta \sim \mathcal{N}(0, 1)$ 1000 times to estimate the expectation for each point.

Note that when the disturbance ϵ is small enough, we can use $\bar{U}_{\theta,\phi,s}(s') = U_{\theta,\phi}(s) + \langle \nabla_s U_{\theta,\phi}(s), s' - s \rangle$ to approximate $U_{\theta,\phi}$ without losing too much information. Then, rather than solving problem (17) directly, we can solve $\inf_{s' \in B_\epsilon(s)} \bar{U}_{\theta,\phi,s}(s')$ instead, which enjoys the following closed form solution under a small ∞-norm ball $B_\epsilon(s)$,

$$\inf_{s' \in B_\epsilon(s)} \bar{U}_{\theta,\phi,s}(s') = U_{\theta,\phi}(s) - \epsilon \|\nabla_s U_{\theta,\phi}(s)\|_1. \quad (19)$$

We refer to Equation (19) as a **G**radient **B**ased **R**egularizer (GBR), which approximates the solution to Equation (17) via simply subtracting a gradient norm term.

Does the Approximation Make Sense in Practice? In our experiments, we find that the local linear property of $U_{\theta,\phi}(s)$ for small ϵ is well satisfied for existing actor-critic algorithms (we use the popular off-policy algorithm SAC as an instance to illustrate in Figure 2), and this approximation achieves satisfactory performance as reported in Section 6.

5.3 State-Conservative SAC: An Instance

Now we apply SCPO proposed in Section 5.1 to existing actor-critic algorithms to learn robust policies. Specifically, we present the SCPO-based algorithm **S**tate-**C**onservative **S**oft **A**ctor-**C**ritic (SC-SAC) as an instance (Haarnoja et al. 2018), where we use the GBR proposed in Section 5.2 for seek of efficient implementation.

Soft Actor-Critic (SAC) (Haarnoja et al. 2018) is an off-policy actor-critic algorithm based on the maximum entropy RL framework, where the actor aims to maximize both the expected return and the entropy. According to SAC, we reparameterize the policy as $a_t = f_\phi \left(\delta_t; s_t \right)$, where δ_t is standard Gaussian noise; we use the target Q-value network $Q_{\bar{\theta}}$ whose parameters $\bar{\theta}$ is updated in a moving average fashion; we use the automatic tuning for the temperature parameter α and the clipped double Q-learning $Q_\theta^{\min}(s, a) \triangleq \min_{i=1,2} Q_{\theta_i}(s, a)$ (Fujimoto, van Hoof, and Meger 2018).

In the policy evaluation step, we train the critic by minimizing the Bellman residual $J_{\epsilon\text{-}Q}$ defined in Equation (12),

---

Algorithm 3: State-Conservative Soft Actor-Critic

1: **Input:** Critic $Q_{\theta_1}, Q_{\theta_2}$. Actor π_ϕ. Initial temperature parameter α. Step size $\beta_Q, \beta_\pi, \beta_\alpha$. Target smoothing coefficient τ.
2: $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2, \mathcal{D} \leftarrow \emptyset$.
3: **for** each iteration **do**
4:     **for** each environment step **do**
5:         $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim p_0(\cdot|s_t, a_t)$.
6:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, r(s_t, a_t), s_{t+1}\}$.
7:     **end for**
8:     **for** each training step **do**
9:         $\theta_i \leftarrow \theta_i - \beta_Q \nabla J_{\epsilon\text{-}Q}(\theta)$ for $i = 1, 2$.
10:       $\phi \leftarrow \phi - \beta_\pi \nabla J_{\epsilon\text{-}\pi}(\phi)$.
11:       $\alpha \leftarrow \alpha - \beta_\alpha \nabla J_\alpha(\alpha)$.
12:       $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau)\bar{\theta}_i$ for $i = 1, 2$.
13:     **end for**
14: **end for**
15: **Output:** θ_1, θ_2, ϕ

---

with the target $\widehat{V}(s_{t+1})$ in Equation (14) now defined as

$$\begin{aligned} \widehat{V}(s_{t+1}) \triangleq \mathbb{E}_{\delta_t \sim \mathcal{N}} \big[& Q_{\bar{\theta}}^{\min} \left(s_{t+1}, f_\phi \left(\delta_{t+1}; s_{t+1} \right) \right) \\ & - \epsilon \nabla_s Q_{\bar{\theta}}^{\min} \left(s_{t+1}, f_\phi \left(\delta_{t+1}; s_{t+1} \right) \right) \\ & - \alpha \log \pi_\phi \left(f_\phi \left(\delta_{t+1}; s_{t+1} \right) | s_{t+1} \right) \big]. \end{aligned} \quad (20)$$

In the policy improvement step, we train the actor by maximizing the state-conservative entropy regularized objective

$$\begin{aligned} J_{\epsilon\text{-}\pi}(\phi) \triangleq \mathbb{E}_{s_t \sim D, \delta_t \sim \mathcal{N}} \big[& Q_\theta^{\min} \left(s_t, f_\phi \left(\delta_t; s_t \right) \right) \quad (21) \\ & - \epsilon \nabla_s Q_\theta^{\min} \left(s_t, f_\phi \left(\delta_t; s_t \right) \right) - \alpha \log \pi_\phi \left(f_\phi \left(\delta_t; s_t \right) | s_t \right) \big] \end{aligned}$$

In the temperature α update step, we tune α by minimizing

$$J_\alpha(\alpha) \triangleq -\alpha \, \mathbb{E}_{s_t \sim \mathcal{D}, \delta_t \sim \mathcal{N}} \left[\log \pi_\phi \left(f_\phi \left(\delta_t; s_t \right) | s_t \right) + \mathcal{H} \right], \quad (22)$$

where $\mathcal{H}$ is the target value of the entropy term. We show the pseudo code of SC-SAC in Algorithm 3.

6 Experiments

In this section, we conduct experiments on the SCPO-based algorithm SC-SAC in several MuJoCo benchmarks (Todorov, Erez, and Tassa 2012) to evaluate its performance.

Hyperparameter Setting The hyperparameter ϵ serves as a regularization coefficient in SCPO. By the definition in Section 4.2, larger ϵ implies higher intensity of the disturbance considered in SC-MDP. However, too large ϵ can lead to suboptimal policies and thus degraded performance. Thus, we tune the hyperparameter ϵ in the Hopper-v2 task by grid search and find that it achieves the best performance when $\epsilon = 0.005$. We then *set $\epsilon = 0.005$ for all the tasks in our experiments*. See Section 6.2 for sensitivity analysis.

Implementation and Evaluation Settings We normalize the observations for both SAC and SC-SAC in all tasks. We keep all the parameters in SC-SAC the same as those in original SAC. We train policies for 200k steps (i.e., 200 epochs) in InvertedDoublePendulum-v2 and 1000k steps (i.e., 1000 epochs) in other tasks. We train policy for each task with
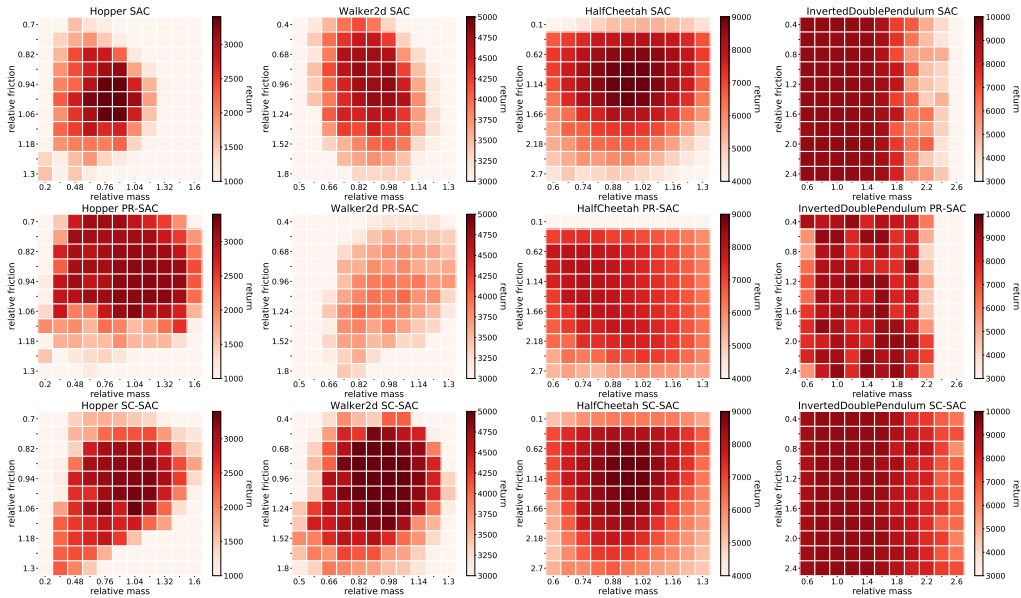
Figure 3: Compare SAC, PR-SAC and SC-SAC in target environments with perturbed parameters (mass, friction). We randomly choose 4 policies from the last 10 epochs for each seed and run 8 episodes for each policy. That is, each point here is evaluated over 160 episodes ($5 \times 4 \times 8$). More details for the implementations of PR-SAC can be found in Appendix C. The results show that SC-SAC trained policies are more robust than the original SAC and the PR-SAC in perturbed target environments.
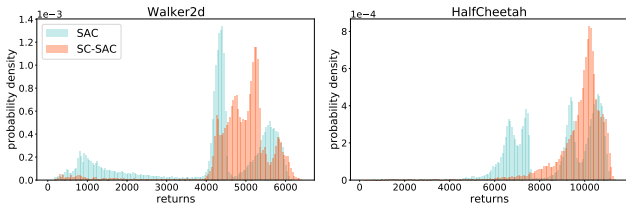

Figure 4: Compare the return distribution between SAC and SC-SAC with truncated Gaussian mass and friction. We evaluate the return 10^5 times for each task with policy sampled from last 10 epochs and environmental parameters sampled from truncated Gaussian distributions (see Appendix C for details). The results show that the return distribution of SC-SAC is more concentrated in the high-value areas.
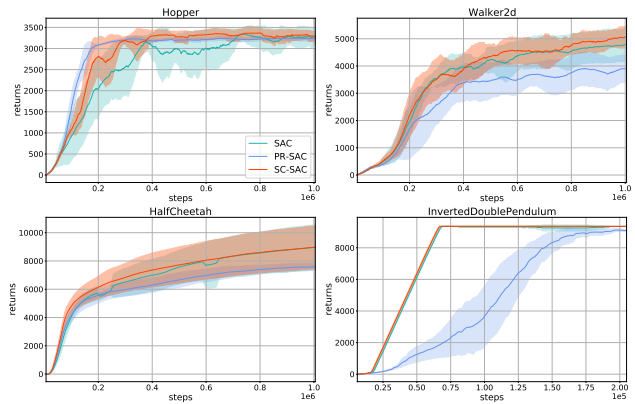

Figure 5: Training curves for different algorithms. The results show that SC-SAC achieves almost identical performance as SAC in source environments for all tasks, while PR-SAC leads to performance degradation for several tasks.

5 random seeds. We perturb the parameters (e.g., mass and friction) in target environments to generate disturbance in transition dynamics. More details for implementation and evaluation settings can be found in Appendix C.

6.1 Comparative Evaluation

We compare SC-SAC with the original SAC and PR-MDP (Tessler, Efroni, and Mannor 2019) in this part.

Comparison with Original SAC We compare SC-SAC with original SAC in target environments with perturbed environmental parameters (mass, friction). Figure 3 illustrates that *SC-SAC is more robust than SAC when generalize to target environments*. We then evaluate policies in target environments with truncated Gaussian distributed mass and friction, since the perturbation on environmental parameters

in the real world usually follows Gaussian distribution. Results in Figure 4 demonstrate that the return distributions of SC-SAC are more concentrated in high-value areas than that of SAC. Note that *SC-SAC does not use any prior knowledge* about the disturbance (e.g., mass and friction changes) during training. We further compare the computational efficiency between SC-SAC and SAC in Appendix D.1.

Comparison with the Action Robust Algorithm Action robust (AR) algorithms learn robust policies by using the disturbance in action space to approximate the unmodeled disturbance in transition dynamics. We apply the state-
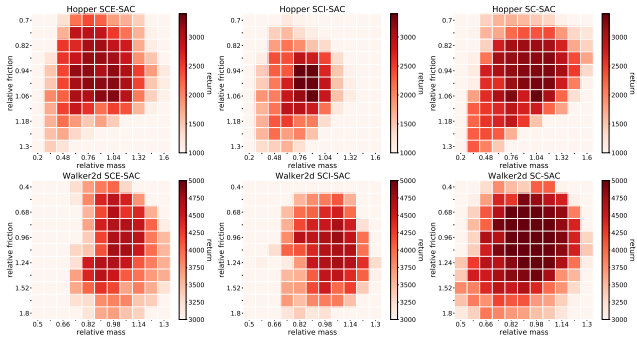
Figure 6: Evaluate polices trained with only the state-conservative policy evaluation step (SCE-SAC) or the state-conservative policy improvement step (SCI-SAC). Each point here is evaluated over 160 episodes (similar to that in Figure 3). Results show that both the SCE step and the SCI step in SCPO help to learn robust policies during training.
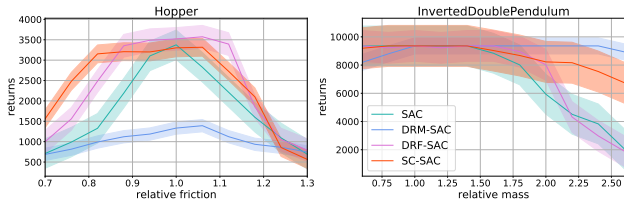

Figure 7: Compare SCPO with DR in target environments with one parameter perturbed. DR uniformly randomizes the mass (DRM-SAC) or the friction (DRF-SAC) in the range given in Figure 3 during training in source environments, while SC-SAC only uses a gradient based regularizer. We then evaluate them in target environments with perturbed mass or friction. The results show that SCPO learned policies are more robust to unmodeled disturbance than DR.

of-the-art AR algorithm PR-MDP, which uses an adversarial policy to disturb actions during training, to SAC (called PR-SAC) and compare it with SC-SAC in Figure 3. Results show that *SC-SAC achieves higher average returns than PR-SAC in most tasks*. This is mainly because that adversarial training tends to result in degraded performance, while regularizer-based SCPO maintains the original performance of SAC more easily (see Figure 5 for the training curves).

Comparison with Domain Randomization The domain randomization (DR) algorithm randomizes parameters in source environments during training to learn policies that are robust to perturbations on these parameters. However, when testing in target environments, we can suffer from unmodeled disturbance that comes from perturbations on unseen parameters. To compare the robustness against unmodeled disturbance between SCPO and DR, we train SAC with one parameter (mass or friction) uniformly randomized in the range given in Figure 3 and then test it in target environments with the other parameter perturbed. As shown in Figure 7, DR trained policies are robust to the disturbance that is modeled in advance but perform poorly when generalize to unmodeled disturbance. By contrast, *though the perturba-*
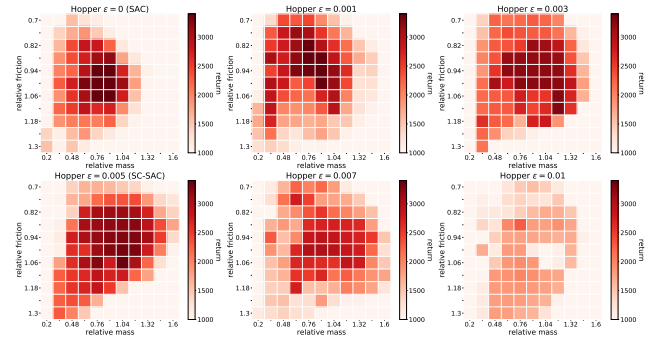

Figure 8: Sensitivity analysis on the hyperparameter ϵ. We choose $\epsilon = 0.001, 0.003, 0.005, 0.007, 0.01$ in SC-SAC and test in perturbed target environments. Each point is evaluated over 160 episodes. Results show that SC-SAC is relatively insensitive to ϵ within a specific range, while a too large ϵ hurts the performance of SAC in source environments.

tions on both mass and friction are unmodeled during training, SC-SAC trained policies are robust to them consistently.

6.2 Ablation Study

In this part, we conduct ablation study on different components of SC-SAC to analyze their effects empirically.

Policy Evaluation v.s. Policy Improvement SCPO based algorithms include a state-conservative policy evaluation step and a state-conservative policy improvement step. We conduct ablation study to analyze their effects in SCPO respectively. Specifically, we apply SCPO to SAC with only the policy evaluation step (SCE-SAC) or the policy improvement step (SCI-SAC) and then report their performance in perturbed target environments in Figure 6. Results show that *both the policy evaluation step and the policy improvement step in SCPO play an important role in the training process*.

Sensitivity Analysis on ϵ We analyze the sensitivity of SC-SAC to the hyperparameter ϵ. We evaluate policies trained with different ϵ in perturbed environments and show the results in Figure 8. Results demonstrate that *SC-SAC is relatively insensitive to ϵ within a specific range ($\epsilon \in [0.001, 0.005]$)*, while *with too large epsilon ($\epsilon \geq 0.01$) SCPO can result in suboptimal policies* that perform poorly in target environments. In practice, we find an ϵ that makes the value of GBR be around 0.5% of the original objective can usually serve as a good initial value for grid search.

7 Conclusion

In this paper, we propose the state-conservative MDP (SC-MDP) to learn robust policies against disturbance in transition dynamics without modeling the disturbance in advance. To solve continuous control tasks, we further propose the State-Conservative Policy Optimization (SCPO) algorithm and approximately solve it via a simple Gradient Based Regularizer (GBR). Experiments show that the SCPO learns robust policies against the disturbance in transition dynamics. Applying SCPO to on-policy and offline reinforcement learning algorithms is exciting avenues for future work.

References

Abdullah, M. A.; Ren, H.; Ammar, H. B.; Milenkovic, V.; Luo, R.; Zhang, M.; and Wang, J. 2019. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*.

Blanchet, J. H.; and Murthy, K. R. A. 2019. Quantifying Distributional Model Risk via Optimal Transport. *Math. Oper. Res.*, 44(2): 565–600.

Boyd, S.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1587–1596. PMLR.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hou, L.; Pang, L.; Hong, X.; Lan, Y.; Ma, Z.; and Yin, D. 2020. Robust reinforcement learning with Wasserstein constraint. *arXiv preprint arXiv:2006.00945*.

Iyengar, G. N. 2005. Robust Dynamic Programming. *Math. Oper. Res.*, 30(2): 257–280.

Jiang, Y.; Li, C.; Dai, W.; Zou, J.; and Xiong, H. 2021. Monotonic robust policy optimization with model discrepancy. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 4951–4960. PMLR.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

Mandlekar, A.; Zhu, Y.; Garg, A.; Fei-Fei, L.; and Savarese, S. 2017. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3932–3939.

Mankowitz, D. J.; Levine, N.; Jeong, R.; Abdolmaleki, A.; Springenberg, J. T.; Shi, Y.; Kay, J.; Hester, T.; Mann, T.; and Riedmiller, M. 2020. Robust reinforcement learning for continuous control with model misspecification. In *International Conference on Learning Representations*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Nilim, A.; and Ghaoui, L. 2004. Robustness in Markov Decision Problems with Uncertain Transition Matrices. In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems*, volume 16. MIT Press.

OpenAI; Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; Schneider, J.; Tezak, N.; Tworek, J.; Welinder, P.; Weng, L.; Yuan, Q.; Zaremba, W.; and Zhang, L. 2019. Solving Rubik's cube with a robot hand. *arXiv preprint*.

Packer, C.; Gao, K.; Kos, J.; Krähenbühl, P.; Koltun, V.; and Song, D. 2018. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 2817–2826. PMLR.

Rajeswaran, A.; Ghotra, S.; Ravindran, B.; and Levine, S. 2017. EPOpt: Learning Robust Neural Network Policies Using Model Ensembles. In *International Conference on Learning Representations*. OpenReview.net.

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

Tessler, C.; Efroni, Y.; and Mannor, S. 2019. Action robust reinforcement learning and applications in continuous control. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 6215–6224. PMLR.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. 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)*, 23–30.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Liu, M.; Boning, D.; and Hsieh, C.-J. 2020. Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21024–21037. Curran Associates, Inc.