Monotonic Robust Policy Optimization with Model Discrepancy

Yuankun Jiang [1] **Chenglin Li** [2] **Wenrui Dai** [1] **Junni Zou** [1] **Hongkai Xiong** [2]

Abstract

State-of-the-art deep reinforcement learning (DRL) algorithms tend to overfit due to the model discrepancy between source and target environments. Though applying domain randomization during training can improve the average performance by randomly generating a sufficient diversity of environments in simulator, the worst-case environment is still neglected without any performance guarantee. Since the average and worst-case performance are both important for generalization in RL, in this paper, we propose a policy optimization approach for concurrently improving the policy's performance in the average and worst-case environment. We theoretically derive a lower bound for the worst-case performance of a given policy by relating it to the expected performance. Guided by this lower bound, we formulate an optimization problem to jointly optimize the policy and sampling distribution, and prove that by iteratively solving it the worst-case performance is monotonically improved. We then develop a practical algorithm, named monotonic robust policy optimization (MRPO). Experimental evaluations in several robot control tasks demonstrate that MRPO can generally improve both the average and worst-case performance in the source environments for training, and facilitate in all cases the learned policy with a better generalization capability in some unseen testing environments.

1. Introduction

With deep neural network approximation, deep reinforcement learning (DRL) has extended classical reinforcement learning (RL) algorithms to successfully solving complex control tasks, e.g., playing computer games with human-level performance (Mnih et al., 2013; Silver et al., 2018) and traffic signal control (Chen et al., 2020). DRL often requires tremendous amounts of data to train a reliable policy, especially with random exploration. It thus becomes infeasible for some practical tasks, such as robotic control and autonomous driving, as training in the real world is not only expensive and time-consuming, but also dangerous. Therefore, a very limited set of real samples is available for training these tasks, resulting in overfitting and poor generalization (Kang et al., 2019). One alternative solution is to learn the policy in a simulator (i.e., source environment for training) and then transfer it to the real world (i.e., target environment for testing). However, it is currently impossible to model the exact environment and physics of the real world. For instance, physical effects like nonrigidity and fluid dynamics are quite difficult to accurately model in simulation. How to mitigate the model discrepancy between source and target environments remains challenging for generalization in RL.

Since one can manipulate simulator settings during training, the policy can be trained on various environments, and thus be able to resist to model discrepancy when it is deployed on the test environment. Based on this, a simple yet effective method is proposed by Peng et al. (2018) and referred to as domain randomization (DR). It randomizes the simulator (e.g., by randomizing the distribution of environment parameters) to generate a variety of environments for training the policy in the source domain. Compared with training in a single environment, recent researches have shown that policies learned through an ensemble of environment dynamics obtained by DR achieve a better generalization performance in terms of expected return. The expected return is referred to as the average performance across all the trajectories sampled from different environments. Since these trajectories, regardless of their performance, are sampled according to a fixed distribution, the trajectories with the worst performance would severely degrade the overall performance.

In contrast, another line of research on the generalization in RL is from the perspective of control theory, i.e., learning policies that are robust to environment perturbations. Robust RL algorithms learn policies, also using model ensembles produced by perturbing parameters of the nominal model. EPOpt (Rajeswaran et al., 2017), a representative of them,

[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China [2]Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. Correspondence to: Yuankun Jiang <yuankun-jiang@sjtu.edu.cn>, Chenglin Li <lcl1985@sjtu.edu.cn>, Junni Zou <zou-jn@cs.sjtu.edu.cn>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

trains policy solely on the worst performing subset, i.e., trajectories with the worst α-percentile of returns, while discarding all the better performing trajectories. In other words, it seeks a better worst-case performance at the cost of degradation on the average performance. In general, robust RL algorithms may produce policies that perform overly conservative, such that the policy learned will not behave very badly in a previously unseen environment.

In this paper, we focus on the generalization issue in RL, and aim to mitigate the *model discrepancy* of the transition dynamics between source and target environments. Considering that both the average and worst-case performance are equally important for evaluating the generalization capability of the policy, we propose a policy optimization approach in which the distribution of sampled trajectories are specifically designed for concurrently improving both the average and worst-case performance. Our main contributions are summarized as follows.

- **Theoretical Lower Bound Relating Wost-case with Expected Performance.** For a given policy and a wide range of environments, we theoretically derive a lower bound for the worst-case expected return of that policy over all the environments, which is related to the expected performance. Therefore, maximizing this lower bound can be equivalently achieved by solving an expected performance maximization problem, under constraints that bound the update step in policy optimization and statistical distance between the worst and average case environments. To the best of our knowledge, this theoretical analysis of relationship between the worst-case and average performance is reported for the first time, which provides a practical guidance for updating policies towards both the worst-case and average performance maximization.

- **Criterion for Sampling Trajectory Selection.** Environments available for training may contribute differently to the generalization capacity of policy. Therefore, in face of all possible environments, the problem that which types of environments are likely to mostly affect the generalization performance should be considered. Unlike traditional uniform sampling without the worst-case performance guarantee, and different from the worst α-percentile sampling in which parameter α is empirically preset, we propose a criterion for the sampling trajectory selection based on the proposed worst-case and average performance maximization, with which both the environment diversity and the worst-case environments are taken into account.

- **MRPO for Monotonic Worst-case Performance Improvement.** Based on the proposed theoretical analysis, we develop a monotonic robust policy optimization (MRPO) algorithm to learn the optimal policy that intends to concurrently optimize the worst-case and average performance. Specifically, MRPO carries out a two-step optimization to update the policy and the distribution of sampled trajectories, respectively. We further prove that the policy optimization problem can be transformed to trust region policy optimization (TRPO) (Schulman et al., 2015) on all the possible environments, such that the policy update can be implemented by the commonly used proximal policy optimization (PPO) algorithm (Schulman et al., 2017). Finally, we prove that by updating the policy with the MRPO, the worst-case expected return can be monotonically increased.

- **Practical Implementation and Evaluation of MRPO.** To greatly reduce the computational complexity, we impose Lipschitz continuity assumptions on the transition dynamics and propose a practical implementation of MRPO. We then conduct experiments on six robot control tasks with variable transition dynamics of environments, and show that MRPO can improve both the average and worst-case performance in the training environments compared to DR and Robust RL baselines, and significantly facilitate the learned policy with a better generalization capability in previously unseen environments used for testing.

2. Background

Under the standard RL setting, the environment is modeled as a Markov decision process (MDP) defined by a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, R >$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. For the convenience of derivation, we assume they are finite. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition dynamics determined by the environment parameter $p \in \mathcal{P}$, where $\mathcal{P}$ denotes the environment parameter space. For example, in robot control, environment parameter could be physical coefficients that directly affect the control, such as friction of joints and torso mass. Throughout this paper, by environment p, we mean that an environment has the transition dynamics determined by parameter p. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. At each time step t, the agent observes the state $s_t \in \mathcal{S}$ and takes an action $a_t \in \mathcal{A}$ guided by policy $\pi(\mathrm{a}_t|\mathrm{s}_t)$. Then, the agent will receive a reward $r_t = R(s_t, a_t)$ and the environment shifts from current state s_t to the next state s_{t+1} with probability $\mathcal{T}(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t, p)$. The goal of RL is to search for a policy π that maximizes the expected cumulative discounted reward $\eta(\pi|p) = \mathbb{E}_{\boldsymbol{\tau}}\left[G(\boldsymbol{\tau}|p)\right]$, $G(\boldsymbol{\tau}|p) = \sum_{t=0}^{\infty} \gamma^t r_t$, where $\boldsymbol{\tau} = \{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{\infty}$ denotes the trajectory generated by policy π in environment p, $\gamma \in [0, 1]$ is the discount factor, and $\eta(\pi|p)$ measures the performance in return of π in environment p. We can then define the state value function as $V_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|\mathrm{s}_t = s\right]$, the action value function as $Q_\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|\mathrm{s}_t = s, \mathrm{a}_t = a\right]$, and the advantage function as $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$. We denote the state distribution under environment p and policy

π as $P_\pi(\mathrm{s}|p)$ and that at time step t as $P_\pi^t(\mathrm{s}|p)$. During policy optimization in RL, by updating the current policy π to a new policy $\tilde{\pi}$, Schulman et al. (2015) prove that:

$$\eta(\tilde{\pi}|p) \geq L_\pi(\tilde{\pi}|p) - \frac{2\lambda\gamma}{(1-\gamma)^2}\beta^2, \tag{1}$$

$$L_\pi(\tilde{\pi}|p) = \eta(\pi|p) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim P_\pi, a\sim\pi}\left[\frac{\tilde{\pi}(a|s)}{\pi(a|s)}A_\pi(s,a)\right],$$

where $\lambda = \max_s |\mathbb{E}_{a\sim\pi(a|s)}[A_\pi(s,a)]|$ is the maximum expected advantage following current policy π, and $\beta = \max_s D_{TV}(\pi(\cdot|s)\|\tilde{\pi}(\cdot|s))$ is the maximum total variation (TV) distance between π and $\tilde{\pi}$. The policy's expected return after updating can be monotonically improved by maximizing the lower bound in (1) w.r.t. $\tilde{\pi}$. Based on this and with a certain approximation, Schulman et al. (2015) then propose a algorithm named trust region policy optimization (TRPO) that optimizes $\tilde{\pi}$ towards direction of maximizing $L_\pi(\tilde{\pi})$, subject to the trust region constraint $\beta \leq \delta$.

In standard RL, environment parameter p is fixed without any model discrepancy. While under the domain randomization (DR) settings, because of existence of the model discrepancy, environment parameter should actually be a random variable p following a probability distribution P over $\mathcal{P}$. By introducing DR, the goal of policy optimization is to maximize the expected performance over all the possible environment parameters, i.e., $\max_\pi \mathbb{E}_{\mathrm{p}\sim P}[\eta(\pi|p)]$.

3. Policy Optimization for Monotonic Worst-case Improvement

3.1. Relating Worst-case with Expected Performance

In face of model discrepancy, our goal is to provide a performance improvement guarantee for the worst-case environment, and meanwhile to improve the average performance over all environments. To this end, we relate the worst-case and expected performance through the following inequality.

Lemma 1. *Guided by a certain policy π, there exists a non-negative constant $C \geq 0$, such that:*

$$\mathbb{E}_{\mathrm{p}\sim P}[\eta(\pi|p)] \geq \eta(\pi|p_w) \geq \mathbb{E}_{\mathrm{p}\sim P}[\eta(\pi|p)] - C, \tag{2}$$

where p_w denotes the environment that corresponds to the worst-case performance.

Proof. See Appendix A.1 for details. □

Lemma 1 states that the worst-case performance $\eta(\pi|p_w)$ is lower bounded by the expected performance $\mathbb{E}_{\mathrm{p}\sim P}[\eta(\pi|p)]$ subtracted with C, where C is related to π and p_w. Specifically, this lower bound is derived in the following theorem, with further consideration of the policy update.

Theorem 1. *In MDPs where the reward function is bounded, and for any distribution P over $\mathcal{P}$, by updating the current policy π to a new policy $\tilde{\pi}$, the following bound holds:*

$$\eta(\tilde{\pi}|p_w) \geq \mathbb{E}_{\mathrm{p}\sim P}[\eta(\tilde{\pi}|p)] - 2|r|_{\max}\frac{\gamma\mathbb{E}_{\mathrm{p}\sim P}[\epsilon(p_w\|p)]}{(1-\gamma)^2} - \frac{4|r|_{\max}d(\pi,\tilde{\pi})}{(1-\gamma)^2}, \tag{3}$$

where p_w denotes the environment that corresponds to the worst-case performance under policy π, and we define

$$\epsilon(p_w\|p) \triangleq \max_t \mathbb{E}_{\mathrm{s}',\mathrm{a}} D_{TV}\left(\mathcal{T}(\mathrm{s}|\mathrm{s}',\mathrm{a},p_w)\|\mathcal{T}(\mathrm{s}|\mathrm{s}',\mathrm{a},p)\right),$$
$$d(\pi,\tilde{\pi}) \triangleq \max_t \mathbb{E}_{\mathrm{s}'} D_{TV}\left(\pi(\mathrm{a}|\mathrm{s}')\|\tilde{\pi}(\mathrm{a}|\mathrm{s}')\right),$$

with s$'$ *sampled from state distribution $P_\pi^t(\cdot|p_w)$ and* a *sampled according to current policy $\pi(\cdot|\mathrm{s}')$.*

Proof. See Appendix A.2 for details, and Appendix A.6 for validation of bounded reward function condition. □

In (3), $\epsilon(p_w\|p)$ specifies the model discrepancy between two environments p_w and p, in terms of the maximum expected TV distance of their transition dynamics of all time steps in trajectory sampled in environment p_w using policy π. And $d(\pi,\tilde{\pi})$ denotes the maximum expected TV distance of the two policies, π and $\tilde{\pi}$, along trajectory sampled in environment p_w using policy π. In general, the RHS of (3) provides a lower-bound for the expected return achieved in the worst-case environment p_w, where the first term denotes the expected performance over all environments following the sampling distribution P, while the other two terms can be considered as penalization on a large TV distance between the worst-case environment p_w and the average case, and a large update step from the current policy π to the new policy $\tilde{\pi}$, respectively.

3.2. Provably Monotonic Worst-case Improvement

Referring to Theorem 1, we expect to improve the worst-case performance by maximizing the RHS of (3), which is equivalent to the following constrained optimization problem with two constraints:

$$\max_{\tilde{\pi},P} \mathbb{E}_{\mathrm{p}\sim P}[\eta(\tilde{\pi}|p)]$$
$$\text{s.t.} \quad d(\pi,\tilde{\pi}) \leq \delta_1, \quad \mathbb{E}_{\mathrm{p}\sim P}[\epsilon(p_w\|p)] \leq \delta_2. \tag{4}$$

The optimization objective is to maximize the expected performance over all the possible environments, by updating not only the policy $\tilde{\pi}$, but the environment parameter's sampling distribution P. The first constraint imposes a similar trust region to TRPO (Schulman et al., 2017) that constrains the update step in policy optimization. In addition, we further propose a new trust region constraint on the sampling distribution P. It specifies that the TV distance between

Algorithm 1 Monotonic Robust Policy Optimization

```
1: Input: Initial policy $\pi_0$
2: for $k = 0$ to $N - 1$ do
3:     Determine the worst-case environment under policy
       $\pi_k$, with $p_w^k = \arg\min_{p \in \mathcal{P}} \eta(\pi_k|p)$.
4:     Compute $\mathcal{E}(p, \pi_k)$ for $p \in \mathcal{P}$.
5:     Use PPO for one-step policy optimization for environ-
       ments in the set $\{p|\mathcal{E}(p, \pi_k) \geq \mathcal{E}(p_w^k, \pi_k), p \in \mathcal{P}\}$.
6: end for
```

the worst-case environment p_w and average case over P is bounded, such that by achieving the optimization objective in (4), the worst-case performance is also improved.

To solve the constrained optimization problem in (4), we need to seek for the optimal policy $\tilde{\pi}$ and the distribution P of the sampled trajectories at the same time, which requires a large amount of computation. In practice, we carry out a two-step optimization procedure to reduce the computational complexity. First, we fix the policy by letting $\tilde{\pi} = \pi$, and optimize the objective in (4) w.r.t. the distribution P. In this case, we no longer need to consider the first constraint on the policy update, and thus can convert the second constraint on the sampling distribution into the objective with the guidance of Theorem 1, formulating the following unconstrained optimization problem:

$$\max_{P} \mathbb{E}_{\mathrm{p} \sim P} \left[\mathcal{E}(p, \tilde{\pi}) \right], \tag{5}$$

where for notation simplicity we denote $\mathcal{E}(p, \tilde{\pi}) \triangleq \eta(\tilde{\pi}|p) - \frac{2|r|_{max}\gamma\epsilon(p\|p_w)}{(1-\gamma)^2}$. The first term in $\mathcal{E}(p, \tilde{\pi})$ indicates policy $\tilde{\pi}$'s performance in environment p, while the second term measures the model discrepancy between environments p and p_w. Since the objective function in (5) is linear to P, we can update P by assigning a higher probability to environment p with higher $\mathcal{E}(p, \tilde{\pi})$. As a consequence, sampling according to $\mathcal{E}(p, \tilde{\pi})$ would increase the sampling probability of environments with both poor and good-enough performance, and avoid being trapped in the worst-case environment. Specifically, we propose to select samples from environment p that meets $\mathcal{E}(p, \tilde{\pi}) \geq \mathcal{E}(p_w, \tilde{\pi})$ for the training of policy $\tilde{\pi}$, which is equivalent to assigning a zero probability to other samples.

In the second step, we target at optimizing the policy $\tilde{\pi}$ with the updated distribution P fixed, i.e., the following optimization problem:

$$\max_{\tilde{\pi}} \mathbb{E}_{\mathrm{p} \sim P} \left[\eta(\tilde{\pi}|p) \right] \quad \text{s.t.} \quad d(\pi, \tilde{\pi}) \leq \delta_1 \tag{6}$$

Optimization in (6) can be transformed to a trust region robust policy optimization similar to TRPO and solved practically with PPO (refer to Appendix A.3 and Schulman et al. (2017) for more information). To summarize, we propose a monotonic robust policy optimization (MRPO) in Algorithm 1. At each iteration k, we obtain the worst-case environment p_w according to the performance of policy π_k in each environment. We then perform one-step PPO optimization for environments that satisfy $\mathcal{E}(p, \pi_k) \geq \mathcal{E}(p_w^k, \pi_k)$.

We now formally show that by applying Algorithm 1, the worst-case performance within all the environments can be monotonically improved by MRPO.

Theorem 2. *Under the assumption that the expected returns of worst-case environment between two iterations are similar, which stems from the trust region constraint we impose on the update step between current and new policies, the sequence of policy $\{\pi_1, \pi_2, \ldots, \pi_N\}$ generated by Algorithm 1 is guaranteed with the monotonic worst-case performance improvement, i.e.,*

$$\eta(\pi_1|p_w^1) \leq \eta(\pi_2|p_w^2) \leq \cdots \leq \eta(\pi_N|p_w^N), \tag{7}$$

where p_w^k denotes the parameter of environment with the worst-case performance guided by the current policy π_k at iteration k.

Proof. See Appendix A.4 for details. □

3.3. Practical Implementation Using Simulator

Motivated by Theorem 1, we propose Algorithm 1 that provably promotes monotonic improvement for the policy's performance in the worst-case environment according to Theorem 2. However, Theorem 1 imposes calculation of $\epsilon(p_w\|p)$ that requires the estimation of the expected total variation distance between the worst-case environment and every other sampled environment at each time step. Estimation by sampling takes exponential complexity. Besides, in the model-free setting, we unfortunately have no access to analytical equation of the environment's transition dynamics. Hence, computation for the total variation distance between two environments is unavailable. Under the deterministic state transition, an environment will shift its state with a probability of one. Hence, we can set the state of one environment's simulator to the state along the trajectory from the other environment, taking the same action. We then can compare the next state and compute the total variation distance step by step. Though feasible, this method requires large computational consumption. Here, we propose instead a practical implementation for the MRPO in Algorithm 1.

According to Appendix A.8, we first make an assumption that the transition dynamics model is L_p-Lipschitz in terms of the environment parameter p:

$$\|\mathcal{T}(s|s', a, p) - \mathcal{T}(s|s', a, p_w)\| \leq L_p\|p - p_w\|. \tag{8}$$

Then, we can simplify the calculation of $\epsilon(p_w\|p)$ via:

$$\begin{aligned}\epsilon(p_w\|p) &\leq \max_t \mathbb{E}_{s'\sim P_\pi^t(\cdot|p_w)}\mathbb{E}_{a\sim\pi(\cdot|s')}\frac{1}{2}\sum_s L_p\|p-p_w\| \\ &= \frac{1}{2}\sum_s L_p\|p-p_w\|. \end{aligned} \tag{9}$$

It can be seen from the expression of $\epsilon(p_w\|p)$ that it measures the transition dynamics distance between the worst-case environment p_w and a specific environment p. In simulator, the environment's transition dynamics would vary with the environment parameters, such as the friction and mass in robot simulation. Hence the difference between environment parameters can reflect the distance between the transition dynamics. In addition, if we use in practice the penalty coefficient $\frac{2|r|_{\max}\gamma}{(1-\gamma)^2}$ of $\epsilon(p_w\|p)$ as recommended by Theorem 1, the trajectory subset $\mathbb{T}$ on which we conduct policy optimization would be very small. Therefore, we integrate it with the Lipschitz constant L_p to form a tunable hyperparameter κ and get a practical estimation of $\mathcal{E}$ as $\hat{\mathcal{E}}$. To summarize, we propose a practical version of MRPO in Algorithm 2. At each iteration k, we uniformly sample M environments. For each environment p_i, we sample L trajectories $\{\boldsymbol{\tau}_{i,j}\}_{j=1}^{L}$, calculate the cumulative discounted reward $G(\boldsymbol{\tau}_{i,j}|p_i)$ for each trajectory, approximate $\eta(\pi_k|p_i)$ with $\hat{\eta}(\pi_k|p_i) = \sum_{j=0}^{L-1} G(\boldsymbol{\tau}_{i,j}|p_i)/L$, and determine the worst-case environment p_w^k based on $\hat{\eta}(\pi_k|p_i)$ of a given set of environments $\{p_i\}_{i=0}^{M-1}$. For each environment p_i, we compute $\hat{\mathcal{E}}(p_i,\pi_k) = \hat{\eta}(\pi_k|p_i) - \kappa\|p_i - p_w^k\|$. We then optimize the policy with PPO on the selected trajectory subset $\mathbb{T}$ according to $\hat{\mathcal{E}}(p_i,\pi_k)$ and $\hat{\mathcal{E}}(p_w^k,\pi_k)$.

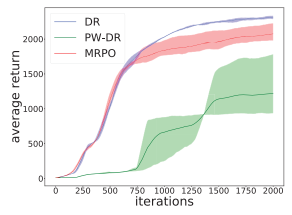
Algorithm 2 Practical Implementation of MRPO

```
1:  Initialize policy $\pi_0$, uniform distribution of environment parameters $U$, number of environment parameters sampled per iteration $M$, maximum number of iterations N, and maximum episode length $T$.
2:  for $k = 0$ to $N-1$ do
3:      Sample a set of environment parameters $\{p_i\}_{i=0}^{M-1}$ according to $U$.
4:      for $i = 0$ to $M-1$ do
5:          Sample $L$ trajectories $\boldsymbol{\tau}_i = \{\boldsymbol{\tau}_{i,j}\}_{j=0}^{L-1}$ in environment $p_i$ using $\pi_k$ and compute $\hat{\eta}(\pi_k|p_i) = \sum_{j=0}^{L-1} G(\boldsymbol{\tau}_{i,j}|p_i)/L$.
6:      end for
7:      Find $p_w^k = \arg\min_{p_i\in\{p_i\}_{i=0}^{M-1}} \hat{\eta}(\pi_k|p_i)$.
8:      for $i = 0$ to $M-1$ do
9:          Compute $\hat{\mathcal{E}}(p_i,\pi_k) = \hat{\eta}(\pi_k|p_i) - \kappa\|p_i - p_w^k\|$ for environment $p_i$.
10:     end for
11:     Select trajectory subset $\mathbb{T} = \{\boldsymbol{\tau}_i : \hat{\mathcal{E}}(p_i,\pi_k) \geq \hat{\mathcal{E}}(p_w^k,\pi_k)\}$.
12:     Use PPO for policy optimization on $\mathbb{T}$ to get the updated policy $\pi_{k+1}$.
13: end for
```

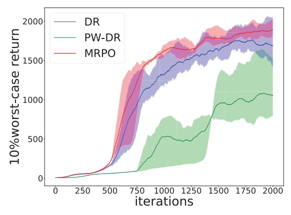
4. Experiments

We now evaluate the proposed MRPO in six robot control benchmarks designed for evaluation of generalization under changeable dynamics. These six environments are modified from the open-source generalization benchmarks (Packer et al., 2018), which are implemented based on the open-source robot control simulation environment, Roboschool (Schulman & Klimov, 2017). We compare MRPO with two baselines, PPO-DR and PW-DR, respectively. In PPO-DR, PPO is applied for the policy optimization in DR. For DR, we utilize uniform distribution on the sampling of environment parameters. In PW-DR, we use purely trajectories from the α-percentile worst-case environments for training and still apply PPO for the policy optimization. Note that in our experiments, PW-DR is an implementation of EPOpt algorithm (Rajeswaran et al., 2017) on Roboschool. We utilize two 64-unit hidden layers to construct the policy network and value function in PPO. For MRPO, we use the practical implementation as described in Algorithm 2. Further note that during the experiments, we find that environments generating poor performance would be far away from each other in terms of the TV distance between their transition dynamics with the dimension of changeable parameters increasing. In other words, a single worst-case environment usually may not represent all the environments in which the current policy performs very poorly. Hence, we choose the 10% worst-case environments to replace the calculation of $\hat{\mathcal{E}}(p_w^k,\pi_k)$ in Algorithm 2. That is, the trajectories we add to the subset $\mathbb{T}$ should have the $\hat{\mathcal{E}}(p_i,\pi_k)$ greater than and equal to those of all the 10% worst-case environments. From the expression of $\hat{\mathcal{E}}(p_i,\pi_k) = \sum_{j=0}^{L-1} G(\boldsymbol{\tau}_{i,j}|p_i)/L - \kappa\|p_i - p_w^k\|$ for environment p_i, it can be seen that the trajectory selection is based on a trade-off between the performance and the distance to the worst-case environment, as we described in detail in the paragraph under (5). Our experiments are designed to investigate the following two classes of questions:

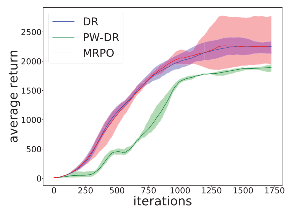
- Can MRPO effectively improve the policy's worst-case and average performance over the whole range of environments during training. Compared to DR, will the policy's average performance degrade by using MRPO? And will MRPO outperform PW-DR in the worst-case environments?
- How does the performance of robust policy trained using MRPO degrade when employed in environments with unseen dynamics? And what determines the generalization performance to unseen environments during training?



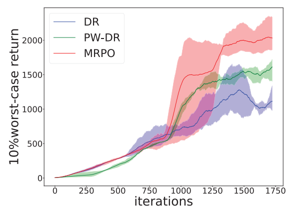
(a) Walker2d A



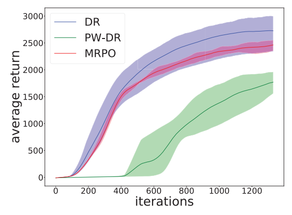
(b) Walker2d W



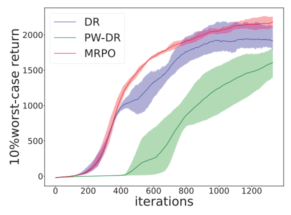
(c) Hopper A



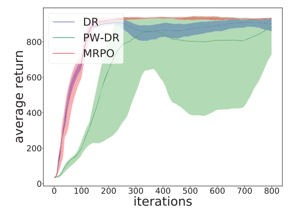
(d) Hopper W



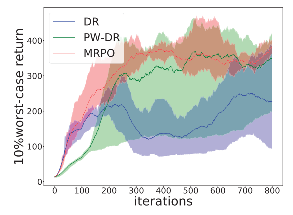
(e) Halfcheetah A



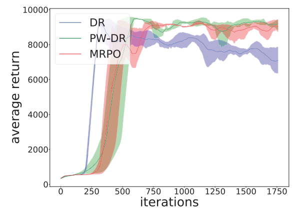
(f) Halfcheetah W



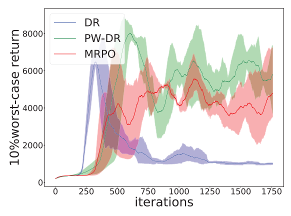
(g) InvertedPendulum A



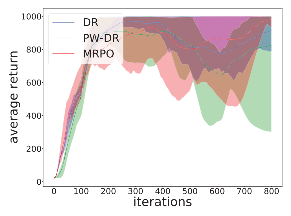
(h) InvertedPendulum W



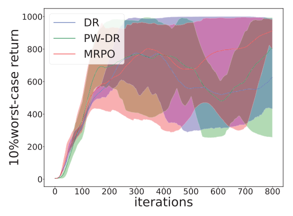
(i) InvertedDoublePendulum A



(j) InvertedDoublePendulum W



(k) Cartpole A



(l) Cartpole W

Figure 1. Training curves of average return (A) and 10% worst-case return (W).

Table 1. Range of environment parameter (EP) for different tasks.

Task	EP	Training Range	Testing Range
W2D	Density	$[750, 1250]$	$[1, 750]$
	Friction	$[0.2, 2.50]$	$[0.05, 0.2]$
HP	Density	$[750, 1250]$	$[1, 750]$
	Friction	$[0.5, 1.1]$	$[0.2, 0.5]$
HC	Density	$[750, 1250]$	$[1, 750]$
	Friction	$[0.2, 2.25]$	$[0.05, 0.2]$
IP	Cart size	$[0.05, 0.25]$	$[0.025, 0.05]$
	Pole length	$[0.50, 2.00]$	$[0.25, 0.50]$
IDP	Pole 1 length	$[0.50, 1.00]$	$[1.00, 1.10]$
	Pole 2 length	$[0.50, 1.00]$	$[1.00, 1.10]$
CP	Force magnitude	$[1.00, 20.00]$	10
	Pole length	$[0.05, 1.00]$	$[1.00, 6.00]$
	Pole mass	$[0.01, 1.00]$	$[1.00, 6.00]$

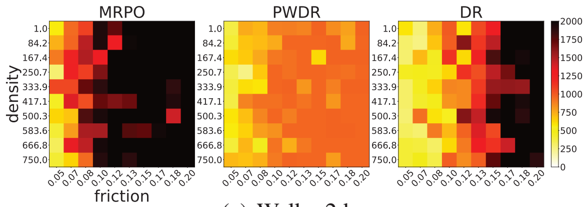
4.1. Training Performance with Different Dynamics

The six robot control tasks are as follows. (1) **Walker2d (W2D)**: control a 2D bipedal robot to walk; (2) **Hopper (HP)**: control a 2D one-legged robot to hop as fast as possible; (3) **HalfCheetah (HC)**: control a 2D cheetah robot to run (Brockman et al., 2016); (4) **InvertedPendulum (IP)**: control a cart (attached to a pendulum system by a joint) by applying a force to prevent the pendulum from falling over; (5) **InvertedDoublePendulum (IDP)**: control a cart (attached to a two-link pendulum system by a joint) by applying a force to prevent the two-link pendulum from falling over; and (6) **Cartpole (CP)**: control a cart (attached to a pole by a joint) by applying a force to prevent the pole from falling over. In robot control, the environment dynamics are directly related to the values of some physical coefficients. For example, if the friction in Walker2d is too small, it is more likely for the robot to slip down, and thus will be more difficult for the agent to manipulate the robot compared to a larger friction. Hence, a policy that performs well for a large friction may not be generalized to the environment with a small friction due to the change of dynamics. In the simulator, by randomizing certain environment parameters, we can obtain a set of environments with the same goal but different dynamics (Packer et al., 2018). The range of parameters that we preset for training of each environment is shown in Table 1.

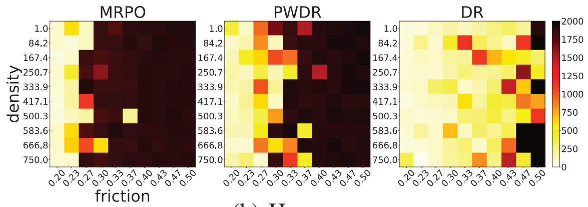
We run the training process for the same number of iterations N on environments sampled from preset range of environment parameters. At each iteration k, we generate

Table 2. Average and worst-case performance in training.

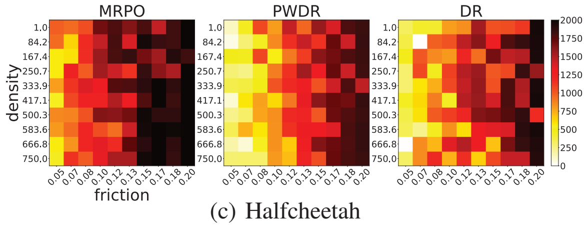
ALGORITHM	AVERAGE W2D	WORST W2D	AVERAGE HP	WORST HP	AVERAGE HC	WORST HC
MRPO	2064.8 ± 106.2	$\mathbf{1878.2 \pm 83.3}$	$\mathbf{2257.4 \pm 423.5}$	$\mathbf{2048.0 \pm 246.8}$	2537.0 ± 98.3	$\mathbf{2166.7 \pm 70.4}$
DR	$\mathbf{2304.7 \pm 16.7}$	$1704.9.5 \pm 234.8$	2250.0 ± 80.4	1254.0 ± 348.5	$\mathbf{2970.0 \pm 30.0}$	1883.3 ± 103.7
PW-DR	1213.1 ± 386.0	1069.3 ± 370.7	1903.3 ± 49.2	1440.0 ± 243.9	1810.0 ± 41.2	1278.7 ± 215.9
ALGORITHM	AVERAGE IP	WORST IP	AVERAGE IDP	WORST IDP	AVERAGE CP	WORST CP
MRPO	$\mathbf{929.2 \pm 2.0}$	$\mathbf{350.5 \pm 19.9}$	9185.2 ± 165.6	5137.0 ± 1421.1	$\mathbf{998.7 \pm 1.7}$	$\mathbf{988.4 \pm 15.4}$
DR	907.2 ± 19.0	239.2 ± 86.3	8252.6 ± 439.8	1354.5 ± 381.2	995.5 ± 5.6	961.6 ± 48.3
PW-DR	843.4 ± 153.4	332.3 ± 90.0	$\mathbf{9222.2 \pm 173.3}$	$\mathbf{5450.4 \pm 1687.3}$	895.4 ± 147.2	791.4 ± 288.9



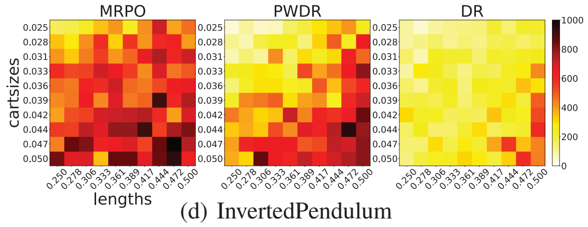
(a) Walker2d



(b) Hopper



(c) Halfcheetah



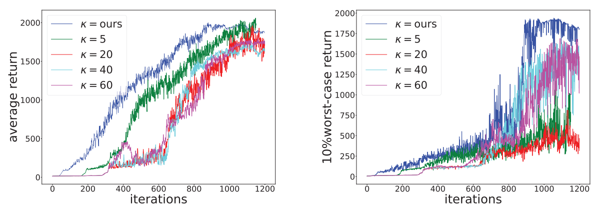
(d) InvertedPendulum

Figure 2. Heatmap of return in unseen environments on Walker2d, Hopper, Halfcheetah and InvertedPendulum, with policies trained by MRPO, PW-DR and DR in the training environments, where a darker color represents a higher return.

trajectories from $M = 100$ environments sampled according to a uniform distribution U. Referring to Appendix A.7, we sample $L = 1$ trajectory for each environment to strike a trade-off between the Monte Carlo estimation accuracy and training complexity. Referring to Appendix A.11, we set $\alpha = 10\%$ for PW-DR since it almost achieves the best performance on all the training environments. The results are obtained by running each algorithm with four different random seeds. The average return is computed over the returns of M sampled environments at each iteration k. We show the training curves of Walker2d, Hopper, Halfcheetah, InvertedPendulum, InvertedDoublePendulum, and Cartpole, respectively, in Figs. 1(a)-1(l). In Fig. 1, the solid curve is used to represent the mean performance of each algorithm on all the four seeds, while the shaded-area denotes the standard error. It is seen that DR can steadily improve the average performance on the whole training range as expected, while MRPO does not significantly degrade the average performance in all the six tasks. In some cases like InvertedPendulum and Cartpole, MPRO achieves an even better average performance than DR. PW-DR, on the other hand, focuses on the worst-case performance optimization, leading to a significant degradation of average performance on Walker2d, Hopper and Halfcheetah. We measure the worst-case performance by computing the worst 10% performance in all the sampled environments at iteration k and the corresponding training curves are illustrated in Figs. 1(b), 1(d), 1(f), 1(h), 1(j) and 1(l), respectively. It can be observed that MRPO presents the best worst-case performance on Walker2d, Hopper, Halfcheetah, InvertedPendulum and Cartpole, while DR neglects the optimization on its worst-case performance. PW-DR shows limited improvement on the worst-case performance compared to MRPO on Walker2d, Hopper, and Halfcheetah. The tabular comparison of the average and worst-case performance achieved during training by different algorithms in different tasks can be found in Table 2, where the performance is averaged on 50 continuous iterations after the training curve is stable. We discuss the generalization of MRPO to higher dimensional randomization of environment parameters in Appendix A.12, and show its potential to generalize to higher dimensional cases.

4.2. Generalization to Unseen Environments

MRPO has been demonstrated in theoretical analysis to optimize both average and worst-case performance during training. Here, we carry out experiments to show that MRPO can generalize to a broader range of unseen environments in testing. To this end, we compare the testing performance on some unseen environments of Walker2d, Hopper, Halfcheetah and InvertedPendulum with the best policies obtained



(a) Hopper A (b) Hopper W

Figure 3. Training curves of (a) average return and (b) 10% worst-case return of MRPO on Hopper with different κ.

by MRPO, DR and PW-DR from training, with the range of parameters set for testing as show in in Table 1. The heatmap of return on these four tasks achieved by different algorithm is shown and compared in Fig. 2, where the testing performances are all averaged on five runs. Please also refer to Appendix A.9 for the heatmap of return on the other two tasks. It is observed that policies all degrade with the decrease of friction (or cart-size in Fig. 2(d)), while the impact of unseen density (or length in Fig. 2(d)) is not that obvious as the friction. In addition, it can be seen that MRPO has better generalization ability to the unseen environments, while DR or PW-DR can hardly generalize in testing for all the four tasks. Compared to them, MRPO has a broader generalization range with a higher performance, from which we remark that both the worst-case and average performance during training are crucial for the generalization to an unseen environment.

4.3. Hyperparameter κ

In Algorithm 2, when we update the sampling distribution P for policy optimization, κ is a hyperparameter that controls the trade-off between the expected cumulative discounted reward $\eta(\pi_k|p_i)$ and distance $\|p_i - p_w^k\|$ to the worst-case environment. Theoretically, a larger κ means that the policy cares more about the poorly performing environments, while a smaller κ would par more attention to the average performance. As empirical evaluation, we conduct experiment of MRPO on Hopper with different choices of hyperparameter κ. The training curves of both average return and the 10% worst-case return are shown in Figs. 3(a) and 3(b), respectively. It can be verified that for the fixed value choice of κ, the curve of $\kappa = 5$ outperforms the curves of $\kappa = 20, 40, 60$ in terms of the average return in Fig. 3(a), while the curve of $\kappa = 60$ outperforms the curves of $\kappa = 5, 20, 40$ in terms of the 10% worst-case return in Fig. 3(b). In practical implementation, we gradually increase κ to a fixed high value. It can therefore strike a tradeoff between the average return and 10% worst-case return, demonstrating the best performance both in Figs. 3(a) and 3(b). For performance comparison of tunning κ on Waker2d and InvertedPendulum, please refer to Appendix A.10.

5. Related Work

With the success of RL in recent years, plenty of works have focused on how to improve the generalization ability for RL. Learning a policy that is robust to the worst-case environment is one strategy. Based on theory of $\mathcal{H}_\infty$ control (Zhou et al., 1996), robust RL takes into account the disturbance of environment parameters and model it as an adversary that is able to disturb transition dynamics in order to prevent the agent from achieving higher rewards (Morimoto & Doya, 2005). The policy optimization is then formulated as a zero-sum game between the adversary and the RL agent. Pinto et al. (2017) incorporate robust RL to DRL method, which improves robustness of DRL in complex robot control tasks. To solve robust RL problem, robust dynamic programming formulates a robust value function and proposes accordingly a robust Bellman operator (Iyengar, 2005; Mankowitz et al., 2020). The optimal robust policy can then be achieved by iteratively applying the robust Bellman operator in a similar way to the standard value iteration (Sutton & Barto, 2018). Besides, Rajeswaran et al. (2017) leverage data from the worst-case environments as adversarial samples to train a robust policy. However, the aforementioned robust formulations will lead to an unstable learning. What's worse, the overall improvement of the average performance over the whole range of environments will also be stumbled by their focus on the worst-case environments. In contrast, in addition to the worst-case formulation, we also aim to improve the average performance.

In the same vein as our work, the extension of conditional value at risk (CVaR) to DR settings can mitigate the conservative policy caused by considering only the average-case or the worst-case. Hiraoka et al. (2019) extend the CVaR-based policy gradient (Chow et al., 2017) to deal with model discrepancy under DR settings. Instead of optimizing policy towards robustness directly, they apply the extension method to learn the robust options (temporally abstract actions) (Bacon et al., 2017). Lobo et al. (2020) form a static soft-robust objective by weighed summing up the average-case objective and CVaR objective over all the possible environments to train a robust policy. Guided from our theory, our method aims to monotonically improve the worst-case performance by solving constrained average-case optimization problem.

For generalization across different state spaces, an effective way is domain adaptation, which maps different state space to a common embedding space. The policy trained on this common space can then be easily generalized to a specific environment (Higgins et al., 2017b; James et al., 2019; Ammar et al., 2015) through a learned mapping, with certain mapping methods, such as β-VAE (Higgins et al., 2017a), cGAN (Isola et al., 2017), and manifold alignment (Wang & Mahadevan, 2009).

Function approximation enables RL to solve complex tasks with high-dimensional state and action spaces, which also incurs inherent generalization issue under supervised learning. Deep neural network (DNN) suffers overfitting due to the distribution discrepancy between training and testing sets. l_2-regularization, dropout and dataset augmentation (Goodfellow et al., 2016) play an significant role for generalization in deep learning, which have also enabled improvement of policy's generalization on some specifically designed environments (Cobbe et al., 2019; Farebrother et al., 2018).

In terms of the theoretical analysis, Murphy (2005) provides a generalization error bound for Q-learning, where the generalization error is represented by the distance between expected discounted reward achieved by converged Q-learning policy and the optimal policy. Wang et al. (2019) analyze the generalization gap in reparameterizable RL limited to the Lipschitz assumptions on transition dynamics, policy and reward function. For monotonic policy optimization in RL, Schulman et al. (2015) propose to optimize a constrained surrogate objective, which can guarantee the performance improvement of updated policy. In the context of model-based RL, Janner et al. (2019); Luo et al. (2019) formulate the lower bound for a certain policy's performance on true environment in terms of the performance on the learned model. It can therefore monotonically improve the performance on true environment by maximizing this lower bound. Different from this, the proposed MRPO in this work can guarantee the robustness of the policy in terms of the monotonically increased worst-case performance, and also improve the average performance.

6. Conclusion

In this paper, we have proposed a robust policy optimization approach, named MRPO, for improving both the average and worst-case performance of policies. Specifically, we theoretically derived a lower bound for the worst-case performance of a given policy over all environments, and formulated an optimization problem to optimize the policy and sampling distribution together, subject to constraints that bounded the update step in policy optimization and statistical distance between the worst and average case environments. We proved that the worst-case performance was monotonically improved by iteratively solving this optimization problem. We have validated MRPO on several robot control tasks, demonstrating a performance improvement on both the worst and average case environments, as well as a better generalization ability to a wide range of unseen environments.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 61931023, Grant 61831018, Grant 61871267, Grant 61972256, Grant 61838303, Grant 61971285, Grant 61720106001, Grant 61932022, in part by the Program of Shanghai Science and Technology Innovation Project under Grant 20511100100, and in part by Shanghai Rising-Star Program under Grant 20QA1404600.

References

Ammar, H. B., Eaton, E., Ruvolo, P., and Taylor, M. E. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. Citeseer, 2015.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., and Li, Z. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3414–3421, 2020.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

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

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. 2017a.

Higgins, I., Pal, A., Rusu, A. A., Matthey, L., Burgess, C. P., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017b.

Hiraoka, T., Imagawa, T., Mori, T., Onishi, T., and Tsuruoka, Y. Learning robust options by conditional value at risk optimization. In *Advances in Neural Information Processing Systems*, pp. 2619–2629, 2019.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

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

James, S., Wohlhart, P., Kalakrishnan, M., Kalashnikov, D., Irpan, A., Ibarz, J., Levine, S., Hadsell, R., and Bousmalis, K. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Conference on Computer Vision and Pattern Recognition*, pp. 12619–12629, 2019.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pp. 12519–12530, 2019.

Kang, K., Belkhale, S., Kahn, G., Abbeel, P., and Levine, S. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *International Conference on Robotics and Automation*, pp. 6008–6014. IEEE, 2019.

Lobo, E. A., Ghavamzadeh, M., and Petrik, M. Soft-robust algorithms for handling model misspecification. *CoRR*, abs/2011.14495, 2020. URL https://arxiv.org/abs/2011.14495.

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2019.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Morimoto, J. and Doya, K. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Murphy, S. A. A generalization error for q-learning. *Journal of Machine Learning Research*, 6(Jul):1073–1097, 2005.

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

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE international conference on robotics and automation*, pp. 1–8. IEEE, 2018.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826, 2017.

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

Schulman, J. and Klimov, O. Roboschool. *https://openai.com/blog/roboschool/*, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

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

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

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

Wang, C. and Mahadevan, S. Manifold alignment without correspondence. In *International Joint Conferences on Artificial Intelligence*, volume 2, pp. 3. Citeseer, 2009.

Wang, H., Zheng, S., Xiong, C., and Socher, R. On the generalization gap in reparameterizable reinforcement learning. In *International Conference on Machine Learning*, pp. 6648–6658, 2019.

Zhou, K., Doyle, J. C., Glover, K., et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.