Bring Your Own (Non-Robust) Algorithm to Solve Robust MDPs by Estimating The Worst Kernel

Uri Gadot [*1] **Kaixin Wang** [*1] **Navdeep Kumar** [1] **Kfir Y. Levy** [1] **Shie Mannor** [1 2]

Abstract

Robust Markov Decision Processes (RMDPs) provide a framework for sequential decision-making that is robust to perturbations on the transition kernel. However, current RMDP methods are often limited to small-scale problems, hindering their use in high-dimensional domains. To bridge this gap, we present **EWoK**, a novel online approach to solve RMDP that **E**stimates the **Wo**rst transition **K**ernel to learn robust policies. Unlike previous works that regularize the policy or value updates, EWoK achieves robustness by simulating the worst scenarios for the agent while retaining complete flexibility in the learning process. Notably, EWoK can be applied on top of any off-the-shelf *non-robust* RL algorithm, enabling easy scaling to high-dimensional domains. Our experiments, spanning from simple Cartpole to high-dimensional DeepMind Control Suite environments, demonstrate the effectiveness and applicability of the EWoK paradigm as a practical method for learning robust policies.

1. Introduction

In reinforcement learning (RL), we are concerned with learning good policies for sequential decision-making problems modeled as Markov Decision Processes (MDPs) (Puterman, 1994; Sutton & Barto, 2018). MDPs assume that the transition model of the environment is fixed across training and testing, but this is often violated in practical applications. For example, when deploying a simulator-trained robot in reality, a notable challenge is the substantial disparity between the simulated environment and the intricate complexities of the real world, leading to potential subpar performance upon deployment. Such a mismatch may significantly degrade the performance of the trained policy (in testing). To deal with this issue, the robust MDP (RMDP) framework has been introduced in (Iyengar, 2005; Nilim & El Ghaoui, 2005; Wiesemann et al., 2013), aiming to learn policies that are robust to any perturbation of the transition model provided it lies within an uncertainty set.

Existing works on learning robust policies in RMDPs often suffer from poor scalability (to high-dimensional domains). Specifically, model-based methods that solve RMDPs (Wiesemann et al., 2013; Ho et al., 2021; Behzadian et al., 2021; Derman et al., 2021; Grand-Clément & Kroer, 2021; Kumar et al., 2022) require access to the nominal transition probability, making it difficult to scale beyond tabular settings. While some recent works (Wang et al., 2022; Wang & Zou, 2022; Kumar et al., 2022; 2023) introduce model-free methods that add regularization to the learning process, the effectiveness of their methods is not validated in high-dimensional environments. In addition, these methods are based on particular RL algorithms (*e.g.*, policy gradient, Q learning), limiting their general applicability. We defer a more detailed discussion on related works to Section 5.

In this work, we tackle the problem of learning robust policies in RMDPs from an alternative direction. As shown in Figure 1, unlike previous works that explicitly regularize the learning process, we propose to approximately sample next states from an **E**stimated **Wo**rst transition **K**ernel (EWoK) while leaving the RL part untouched. In RMDPs, a worst transition kernel is one within the uncertainty set that leads to the minimal possible return (see Definition 3.1). Intuitively, EWoK aims to situate the agent in the worst scenarios for learning policies robust to perturbations. It can be applied on top of any (deep) RL algorithm, offering good scalability to high-dimensional domains.

Specifically, EWoK builds upon our theoretical insights into the relationship between a worst transition kernel and the nominal one. Our characterization of the worst kernel for a KL-regularized uncertainty set concludes that it essentially modifies the next-state transition probability of the nominal kernel, discouraging the transitions to states with higher values while encouraging transitions to lower-value states. Using this connection, we are able to sample the next states

Code is available at link. [*]Equal contribution . [1]Technion [2]NVIDIA Research. Correspondence to: Uri Gadot <uri.gad@campus.technion.ac.il>, Kaixin Wang <kaixin96.wang@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

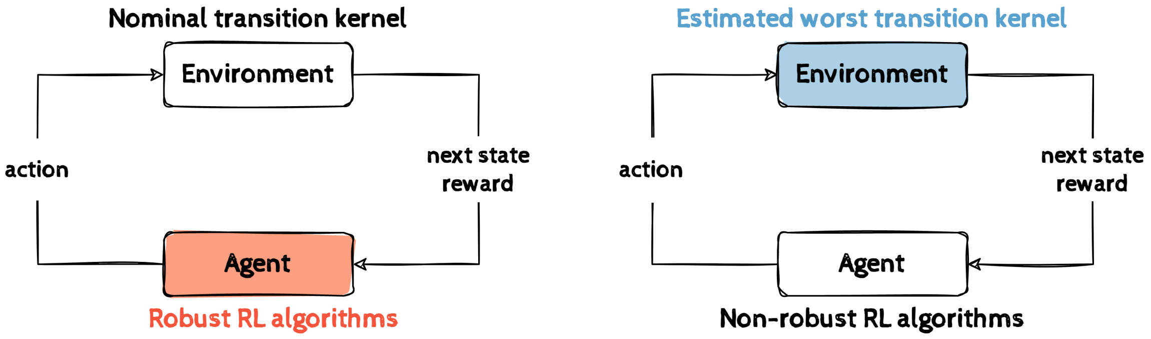


Figure 1. The agent-environment interaction loop during training. **Left**: Existing methods typically regularize how an agent updates its policy to improve robustness. **Right**: Our work estimates a worst transition kernel, so the agent essentially learns its policy under the worst scenarios and can use any non-robust RL algorithm.

such that they are approximately distributed according to the worst transition probability. We establish convergence of the estimated worst kernel to the true worst kernel and present a practical algorithm suitable for high-dimensional domains.

To verify the effectiveness of our method, we conduct experiments on multiple environments ranging from small-scale classic control tasks to high-dimensional DeepMind Control tasks (Tunyasuvunakool et al., 2020). The agent is trained on the nominal environment and tested in environments with perturbed transitions. Since our method is agnostic to the underlying RL algorithm, we showcase the applicability of our method on top of different non-robust algorithms. Experiment results demonstrate that with our method, the learned policy suffers from less performance degradation when the transition kernel is perturbed, even when the perturbation is situated within an uncertainty set that is either coupled or non-KL based.

In summary, our paper makes the following contributions:

- To learn robust policies in RMDPs, we propose to approximately simulate the “worst” transition kernel, instead of regularizing the learning process. This opens up a new paradigm for learning robust policies in RMDPs.
- We theoretically characterize the “worst” kernel for KL uncertainty sets, which is amenable to approximate simulation for environments with large state spaces.
- Our method is not tied to a particular RL algorithm and can be easily integrated with any deep RL method. This flexibility translates to the good scalability of our method in complex high-dimensional domains. To the best of our knowledge, our work is the first that enjoys such flexibility among related works in RMDPs.

2. Preliminaries

Notations. For a finite set $\mathcal{Z}$, we write the probability simplex over it as $\Delta_{\mathcal{Z}}$. Given two real functions $f, g : \mathcal{Z} \to \mathbb{R}$, their inner product is $\langle f, g \rangle = \sum_{z\in\mathcal{Z}} f(z)g(z)$. For distributions P, Q, we denote the Kullback–Leibler (KL) divergence of P from Q by $D_{\mathrm{KL}}(P \,\|\, Q)$. For a vector, we use $\succeq$ and $\preceq$ to denote entry-wise comparison.

2.1. Markov Decision Processes

A Markov decision process (MDP) (Sutton & Barto, 2018; Puterman, 1994) is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \mu)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space respectively, $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition kernel, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\mu \in \Delta_{\mathcal{S}}$ is the initial state distribution. A stationary policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ maps a state to a probability distribution over $\mathcal{A}$. We use $P(\cdot|s, a) \in \Delta_{\mathcal{S}}$ to denote the probabilities of transiting to the next state when the agent takes action a at state s. For a policy π, we denote the expected reward $R^\pi(s) \in \mathbb{R}^{|\mathcal{S}|}$ and transition $P^\pi(s'|s) \in \Delta_{\mathcal{S}}^{|\mathcal{S}|}$ by:

$$R^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)R(s, a),$$
$$P^\pi(s'|s) = \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s, a), \quad \forall s, s' \in \mathcal{S}$$

The value function $v^\pi : \mathcal{S} \to \mathbb{R}$ maps a state to the expected cumulative reward when the agent starts from that state and follows policy π, *i.e.*,

$$v^\pi(s) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right].$$

It is known that v^π is the unique fixed point of the Bellman operator $T_P^\pi v := R^\pi + \gamma P^\pi v$ (Puterman, 1994) ($T_P^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$). The agent's objective is to obtain a policy π^* that maximizes the discounted return

$$J^\pi = \mathbb{E}_{\mu,\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right] = \langle \mu, v^\pi \rangle.$$

2.2. Robust Markov Decision Processes

In MDPs, the system dynamics P is usually assumed to be constant over time. However, in real-life scenarios, it is subject to perturbations, which may significantly impact the performance in deployment (Mannor et al., 2007). Robust MDPs (RMDPs) provide a theoretical framework for taking such uncertainty into consideration, by taking P as not fixed but chosen adversarially from an uncertainty set $\mathscr{P}$ (Iyengar, 2005; Nilim & El Ghaoui, 2005). Since we may consider different dynamics P in the RMDPs context, in the following, we will use subscript P to make the dependency explicit. The objective in RMDPs is to obtain a policy $\pi^*_{\mathscr{P}}$ that maximizes the robust return

$$J^\pi_{\mathscr{P}} = \min_{P\in\mathscr{P}} J^\pi_P.$$

However, this problem is NP-hard for general uncertainty sets while an optimal policy can be non-stationary (Wiesemann et al., 2013). To make RMDPs tractable, we need to make some assumptions about the uncertainty set.

2.3. Rectangular uncertainty set

One commonly used assumption to enable tractability for RMDPs is rectangularity. Specifically, we assume that the uncertainty set $\mathscr{P}$ can be factorized over states-actions:

$$\mathscr{P} = \mathop{\times}_{(s,a)\in(\mathcal{S}\times\mathcal{A})} \mathscr{P}_{sa}, \qquad (\mathtt{sa}\text{-rectangularity})$$

where $\mathscr{P}_{sa} \subseteq \Delta_{\mathcal{S}}$. In other words, the uncertainty in one state-action pair is independent of that in another state-action pair.

Under this assumption, RMDPs admit a deterministic optimal policy as in standard MDPs (Iyengar, 2005; Nilim & El Ghaoui, 2005). The rectangularity assumption also allows the robust value function to be defined as below:

$$v^\pi_{\mathscr{P}} = \min_{P\in\mathscr{P}} v^\pi_P, \quad \text{and} \quad v^*_{\mathscr{P}} = \max_\pi v^\pi_{\mathscr{P}}.$$

Here, since there exists a single P that achieves the minimum simultaneously for all states (Iyengar, 2005), we can safely abuse $\min$ a little bit. In addition, $v^\pi_{\mathscr{P}}$ and $v^*_{\mathscr{P}}$ are the unique fixed points of the robust Bellman operator $T^\pi_{\mathscr{P}}$ and the optimal robust Bellman operator $T^*_{\mathscr{P}}$ respectively, (which are) defined as

$$T^\pi_{\mathscr{P}} v(s) = \min_{P\in\mathscr{P}} T^\pi_P v(s) \text{ and } T^*_{\mathscr{P}} v(s) = \max_\pi T^\pi_{\mathscr{P}} v(s).$$

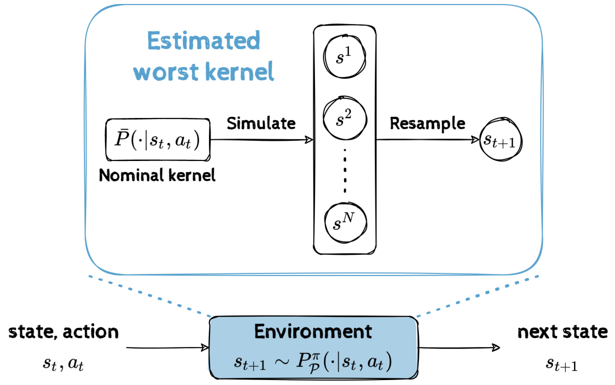


Figure 2. An illustration of how next states are sampled in the estimated worst kernel.

To model perturbations on the environment dynamics, the (rectangular) uncertainty set is often constructed (to be centered) around a nominal kernel $\bar{P}$. Since we want to measure the divergence between probability distributions, it is natural to use KL divergence (Panaganti & Kalathil, 2022; Xu et al., 2023; Shi & Chi, 2022), *i.e.*,

$$\mathscr{P}_{sa} = \{P_{sa} \in \Delta_{\mathcal{S}} \mid D_{\mathrm{KL}}(P_{sa} \,\|\, \bar{P}_{sa}) \le \beta_{sa}\}.$$

Here P_{sa} is a shorthand for $P(\cdot|s,a)$ and β_{sa} is the uncertainty radius that controls the level of perturbation.

3. Method

As introduced earlier, our work proposes to learn robust policies by approximately simulating a *worst transition kernel*, (which is) defined as the one within the uncertainty set that achieves minimal robust return. We formalize it below.

Definition 3.1. For an uncertainty set $\mathscr{P}$ and a policy π, a worst kernel is defined as

$$P^\pi_{\mathscr{P}} \in \arg\min_{P\in\mathscr{P}} J^\pi_P.$$

Training policies under this worst kernel will give us a robust policy with respect to the uncertainty set. Note that $P^\pi_{\mathscr{P}}$ itself is nothing more than a regular transition kernel. Learning a policy under $P^\pi_{\mathscr{P}}$ is no different from the standard MDP setting and we can adopt any non-robust RL algorithms to solve it. The challenge is how to approximately simulate this worst kernel $P^\pi_{\mathscr{P}}$. For a general uncertainty set $\mathscr{P}$, it requires an additional minimization process to find a worst kernel and it is also unclear how we can parameterize and learn $P^\pi_{\mathscr{P}}$ effectively.

To tackle this challenge, we characterize the connection between the nominal transition kernel and a worst one. With such a connection, we are able to obtain the next states that are approximately distributed according to $P^\pi_{\mathscr{P}}(\cdot|s,a)$, by properly resampling the next states from the nominal kernel (Figure 2). Formally, the following theorem describes this connection. All proofs are deferred to Appendix A.

Theorem 3.2. *For a KL uncertainty set $\mathscr{P}$ and a policy π, a worst kernel is related to the nominal kernel through:*

$$P^\pi_{\mathscr{P}}(s'|s,a) = \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')},$$

where δ^π is of the form

$$\delta^\pi(s') = \frac{v^\pi_{\mathscr{P}}(s') - \omega_{sa}}{\kappa_{sa}}, \tag{1}$$

and satisfies

$$\begin{aligned} &\sum_{s'\in\mathcal{S}} \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')} = 1, \\ &\sum_{s'\in\mathcal{S}} \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')}(-\delta^\pi(s')) = \beta_{sa}. \end{aligned} \tag{2}$$

Here, ω_{sa} and κ_{sa} are implicitly defined by Eqn. (2). While they do not have closed forms, we can view ω_{sa} as a threshold, encouraging transitions to states with robust values lower than ω_{sa} (*i.e.*, $\delta^\pi(s') < 0$), and discouraging transitions to states with higher robust values. κ_{sa} works as a temperature parameter to control how much we discourage/encourage transitions to states with high/low robust value. More specifically, the following proposition explicates the relationship between ω_{sa} and κ_{sa} and the uncertainty radius β_{sa}.

Proposition 3.3. *ω_{sa}, κ_{sa} and β_{sa} satisfy*

$$\omega_{sa} = \langle P^\pi_{\mathscr{P}}(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle + \beta_{sa}\kappa_{sa},$$

Based on theoretical results, we arrive at a method to approximately simulate a worst kernel. As illustrated in Figure 2, we first draw a batch of states from the nominal kernel $\bar{P}(\cdot|s,a)$, then resample the next state with probability proportional to $e^{-\delta^\pi(s')}$. This way, next states will be approximately distributed according to $P^\pi_{\mathscr{P}}(\cdot|s,a)$. In practice, we approximate $\delta^\pi(s')$ by

$$\hat{\delta}^\pi(s') = \frac{v(s') - \frac{1}{N}\sum_{i=1}^N v(s^i)}{\kappa},$$

where v is the robust value function approximated with neural networks, and κ is a hyperparameter controlling the robustness level (for all s, a, we assume $\kappa_{sa} = \kappa$). We implement the threshold ω as the average value, a choice supported by the following proposition.

Proposition 3.4. *ω_{sa} is bounded as follows,*

$$\langle P^\pi_{\mathscr{P}}(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle \le \omega_{sa} \le \langle \bar{P}^\pi(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle.$$

Algorithm 1 EWoK - Learning robust policy by Estimating Worst Kernel

```
Input: sample size $N$, robustness parameter $\kappa$
Initialize: initial state $s_0$, policy $\pi$ and value function $v$, data buffer
for $t = 0, 1, 2, \cdots$ do
    Play action $a_t \sim \pi(\cdot|s_t)$.
    Simulate next state $s^i \sim \bar{P}(\cdot|s_t, a_t), i = 1, \cdots, N$, with the nominal environment dynamic.
    Choose $s_{t+1} = s^i$ with probability proportional to $e^{-\frac{v(s^i) - \frac{1}{N}\sum_{i=1}^N v(s^i)}{\kappa}}$.
    Add $(s_t, a_t, s_{t+1})$ to the data buffer.
    Train $\pi$ and $v$ with data from the buffer using any non-robust RL method.
end for
```

Since the N next states are sampled from the nominal kernel, we can approximate an upper bound of ω_{sa} and use it as a proxy to compute $\hat{\delta}^\pi$. Putting it together, we summarize our method in Algorithm 1.

Convergence. The core of our method is the estimation of a worst transition kernel. In practice, however, we do not have the true robust value function as in Eqn. (1). We start with a randomly initialized value function and expect it to gradually converge to the robust value over training. Here, we give some theoretical analysis on the convergence of this process. Let P_n denote the estimated worst transition kernel at iteration n and $v^\pi_{P_n}$ denote the (non-robust) value function for the transition kernel P_n. We are interested in the convergence of the following updates:

$$P_{n+1}(s'|s,a) = \bar{P}(s'|s,a)e^{-\frac{v^\pi_{P_n}(s') - \omega_n}{\kappa_n}}. \tag{3}$$

ω_n and κ_n are associated with the worst-case transition kernel corresponding to the value function $v^\pi_{P_n}$. For clarity, we omit their subscript sa (even though they depend on β_{sa}). The following theorem shows that the value converges to the robust value and the estimated kernel converges to a worst kernel $P^\pi_{\mathscr{P}}$.

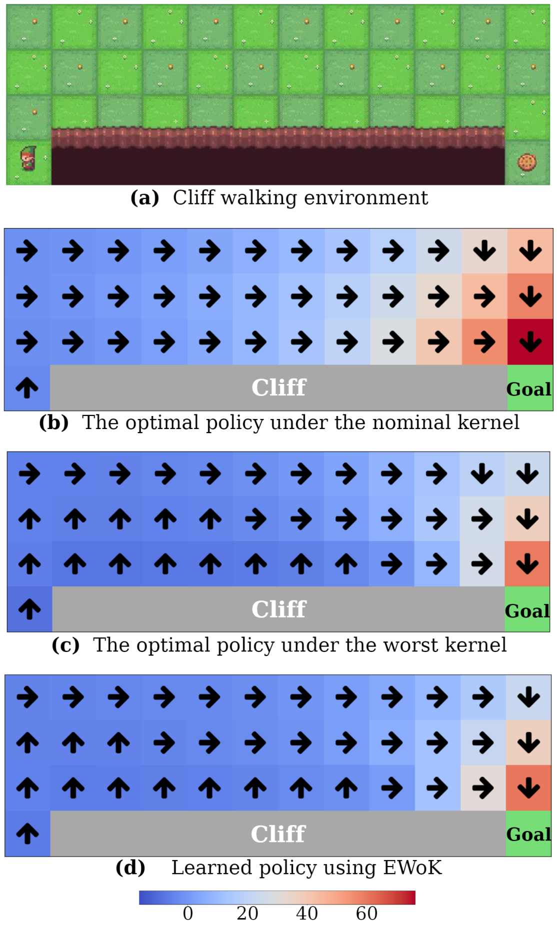
Theorem 3.5. *For the updating process in Eqn. (3), we have*

$$\|v^\pi_{P_n} - v^\pi_{\mathscr{P}}\|_\infty \le \gamma^n \|v^\pi_{\bar{P}} - v^\pi_{\mathscr{P}}\|_\infty.$$

Note that using the robust value function, a worst kernel $P^\pi_{\mathscr{P}}$ can be computed as $P^\pi(\cdot|s,a) = \bar{P}_{\mathscr{P}}(\cdot|s,a)e^{-\frac{v^\pi_{\mathscr{P}} - \omega_{sa}}{\kappa_{sa}}}$ as in Theorem 3.2. This worst kernel (or the samples from it) can be used with any non-robust RL method for policy improvement as described in Figure 2.

3.1. A Cliff Walking Example

To check if our algorithm can reliably learn the robust value function, we test it on a toy task based on OpenAI's Cliff



(a) Cliff walking environment

(b) The optimal policy under the nominal kernel

(c) The optimal policy under the worst kernel

(d) Learned policy using EWoK

Figure 3. Cliff-Walking environment and experiment results. In the bottom 3 plots, the color indicates the learned value and the arrows indicate the actions under the policy.

Walking environment (Brockman et al., 2016). As shown in Figure 3(a), the agent must reach the goal state as quickly as possible by moving in 4 cardinal directions on a grid. If the agent falls off a cliff then it will suffer a penalty and be teleported to the initial state. The nominal transition kernel is modified to incorporate stochasticity; specifically, with a small probability the agent may go to other adjacent states instead of the state indicated by its action. The uncertainty is implemented by varying such probabilities within some range. Please refer to Appendix B.2 for details.

Since this environment is tabular, we can obtain the ground-truth optimal robust value and policy by solving an optimization problem (see Appendix B.1.2). Then, for both the nominal and the worst transition kernels, we compute their optimal value functions and policies using value iteration. The results are plotted in Figure 3(b) and (c). We can see that even though there is a small chance that the agent would fall off the cliff, the optimal policy of the nominal kernel still advises the agent to move right. However, under the worst kernel, the optimal policy tends to avoid walking adjacent to the cliff.

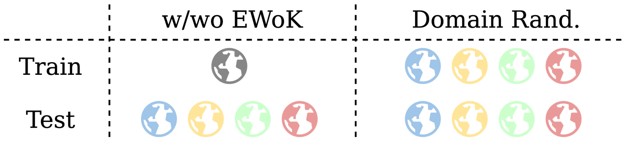


Figure 4. An illustration of the experimental setting. Grey earth denotes the unperturbed (nominal) environment while colored earths denote perturbed environments.

Next, we apply EWoK on top of Q-learning (Watkins & Dayan, 1992) to learn the optimal robust value and policy, only using samples from the nominal kernel. As Figure 3(d) shows, EWoK learns a policy closely resembling the optimal robust one, advising the agent to stay away from the cliff. This preliminary experiment acts as a proof-of-concept, demonstrating the efficacy of the proposed method. In the subsequent section, we will conduct a more thorough evaluation of EWoK in environments of greater complexity.

4. Experiments

4.1. Setting

To evaluate the effectiveness of our method in learning robust policies, we conduct experiments that train the agent online under nominal dynamics and test its performance under perturbed dynamics. We consider two high-dimensional domains including both discrete and continuous control tasks, to demonstrate that our algorithm can be "plugged and played" with any RL method. Specifically, we experiment on Cartpole - a classic control environment from OpenAI's Gym (Brockman et al., 2016) and 3 continuous control tasks (Walker-run, Walker-stand, Walker-walk) from DeepMind Control Suite (Tunyasuvunakool et al., 2020). For baseline RL algorithms, we use Double DQN (van Hasselt et al., 2016) and PPO (Schulman et al., 2017) for Cartpole, and SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018) for continuous control environments. It is worth mentioning that these realistic perturbations do not adhere to the KL uncertainty assumptions. Experimenting with realistic uncertainties (*e.g.*, coupled or non-KL-based) would demonstrate the general applicability of EWoK, beyond the scope of its theoretical motivation.

As existing methods in RMDPs literature do not scale well (see discussions in Section 5), we can not clearly compare "apples-to-apples". Therefore, we consider another commonly used robust RL approach as a reference: domain randomization (DR) (Tobin et al., 2017), and conduct the same set of experiments. DR trains the agent under diverse scenarios by perturbing the parameter of interest during training, such that the trained agent can be robust to similar perturbations during testing. Figure 4 highlights the fact that both the

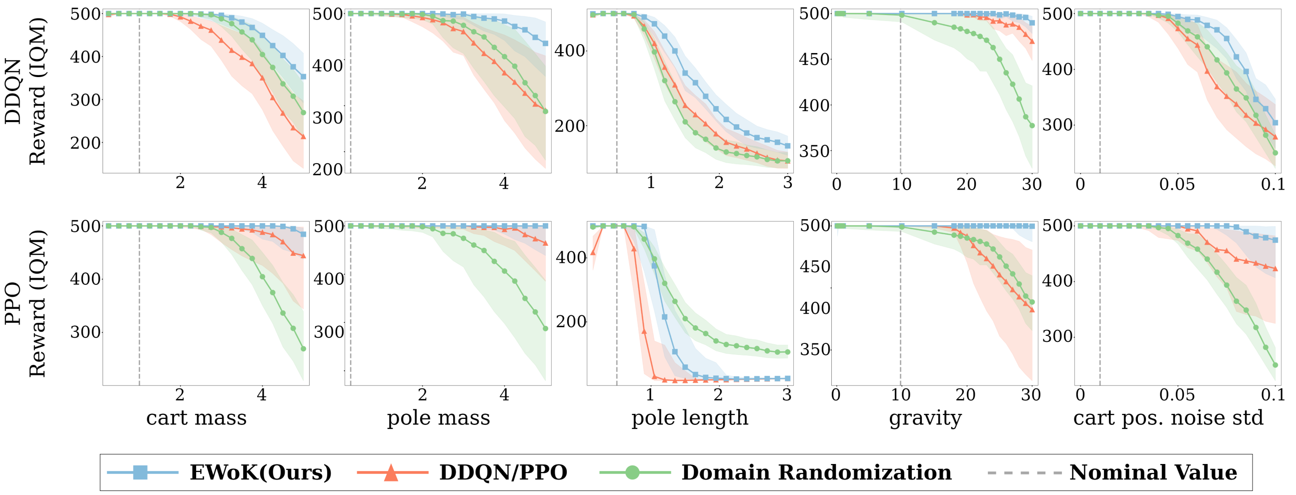


Figure 5. Evaluation results on Cartpole with noise and environment parameters perturbations for both DDQN and PPO algorithms.

non-robust RL methods and their EWoK-augmented counterparts solely interact with the (unperturbed) nominal environment dynamics (e.g., a particular value of pole-length) during training. After training completion, we fix the policy and subsequently evaluate it across various perturbed environments. In contrast, the DR baseline benefits from exposure to diverse perturbed environments during training, giving it a privileged edge over our method.

To obtain stable results, we run each experiment with multiple random seeds, and report the interquartile mean (IQM) and 95% stratified bootstrap confidence intervals (CIs) as recommended by (Agarwal et al., 2021). More details about environments, implementations, training, and evaluation can be found in Appendix B.

4.2. Noise perturbation

In this subsection, we evaluate our method in scenarios where perturbations on the transition dynamics are implemented as noise perturbations. Specifically, we consider stochastic nominal kernels in which the stochasticity is controlled by some (observation or action) noise. The agent is trained under a fixed noise (*i.e.*, the nominal kernel) and tested with varying noises (*i.e.*, perturbed kernels).

On Cartpole, we implement the stochasticity by adding Gaussian noise to the state after applying the original deterministic dynamics of the environment, *i.e.*, $\tilde{s}_{t+1} = s_{t+1} + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma)$. Then, $\tilde{s}_{t+1}$ is considered as the next state output from the stochastic nominal kernel. The noise scale σ is fixed during training and varies during testing. The agent's test performance across different perturbed values is depicted in the rightmost plot in Figure 5. When the noise scale deviates from the nominal value, EWoK achieves better performance than the baseline non-robust RL algorithm and the domain randomization mechanism.

Next, we evaluate our method on continuous control tasks in the DeepMind Control Suite. The stochasticity is implemented by adding Gaussian noise to the action since directly adding noise to the state might lead to an invalid physical state. During testing, we perturb the mean of the Gaussian noise. Figure 6 shows the agent's performance across different perturbed values. We can see that EWoK suffers from less performance degradation as the noise mean deviates from zero (the nominal value), clearly outperforming the baseline non-robust RL algorithm. In the walker-run task, EWoK achieves lower reward under the nominal dynamic but performs better under perturbed ones, which indicates a trade-off between the performance under the nominal kernel and robustness under perturbations.

4.3. Perturbing environment parameters

To further validate the effectiveness of our method, we consider a more realistic scenario where some physical/logical parameters in the environment (*e.g.*, pole length in Cartpole) are perturbed. Similarly, the agent is trained with a fixed parameter, and tested under perturbed parameters.

For Cartpole, we perturb cart mass, pole mass, pole length, and gravity. Figure 5 summarizes the testing results of the agents trained under the nominal dynamics. Again, EWoK achieves better performance than the baseline non-robust RL algorithm and the domain randomization technique when the environment parameters deviate from the nominal value. It is noteworthy that for every environmental parameter (*e.g.*, pole mass), we train a separate domain randomization agent that undergoes perturbations only on that parameter during training. In comparison, EWoK is trained only once and then tested on different parameters.

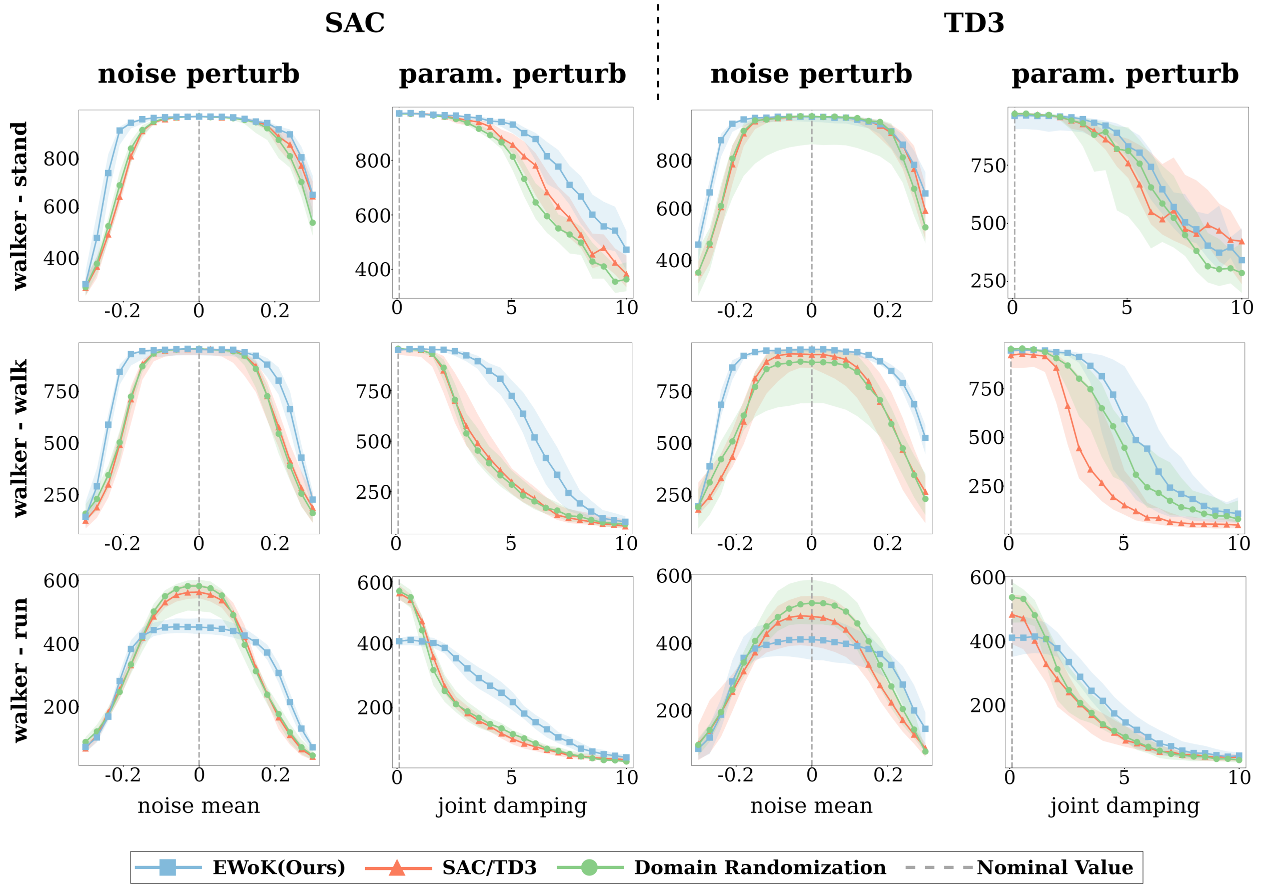


Figure 6. Evaluation results on DeepMind Control environments with noise and environment parameter (joint damping) perturbations for both SAC and TD3 algorithms.

For DeepMind control tasks, we implement the perturbations on the environment parameters using the Real-World Reinforcement Learning Suite (Dulac-Arnold et al., 2020). Specifically, we perturb joint damping, thigh length, and torso length. For all of the results, please refer to Appendix D. As shown in Figure 6, EWoK generally works better than the baseline under model mismatch, improving the robustness of the learned policy. Similar to our observations in the previous section, the walker-run task emphasizes the inherent trade-off of solving RMDPs: optimizing the worst-case scenario can lead to suboptimal performance under the nominal model.

4.4. Ablation studies

In this subsection, we conduct ablation experiments to investigate the effects of our hyperparameters on the performance. Recall that κ controls the skewness of the distribution for resampling, while N controls the number of next-state samples. Intuitively, when we decrease κ, we are essentially considering a higher level of robustness. If κ is very small, then with a high probability the environment dynamic will transit to the “worst” state (*i.e.*, one with the lowest value). In addition, by increasing N we effectively improve our empirical estimation of the nominal kernel’s next state distribution, which should improve the worst kernel estimation.

We experiment on the DeepMind Control tasks under noise perturbation setting, using different κ and N when we train the agent. For clarity, we plot the performance difference between our method and the baseline instead of the absolute performance and defer the original results with CIs (shaded areas) to Appendix D. Figure 7 shows the results of changing the values of κ. In the walker domain, decreasing κ makes our algorithm perform better in perturbed environments, which aligns with our expectations. Figure 8 shows the results of changing the values of N. We can see that a small sample size will result in limited performance gain compared to the baseline, but increasing the sample size may not bring monotonic improvements. In addition, more samples will incur longer simulation time in each environment step. In our experiments, we observed minimal impact on walk-clock time, due to fast simulation. In practical scenarios where sampling next states could be slow, however,

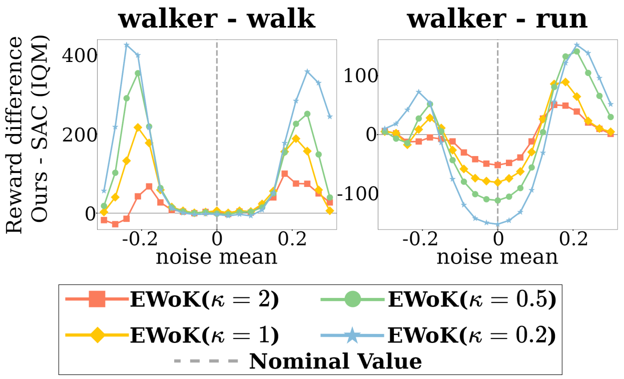


Figure 7. Evaluation results on DeepMind Control tasks with noise perturbations for different κ. The y-axis represents the performance difference between our method and the baseline

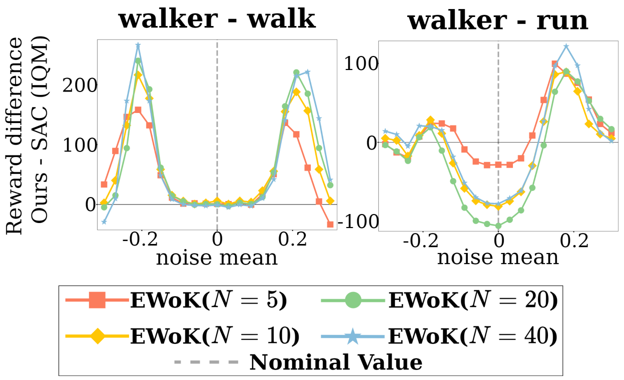


Figure 8. Evaluation results on DeepMind Control tasks with noise perturbations for different N. The y-axis represents the performance difference between our method and the baseline

we need to take this factor into consideration. Nonetheless, we believe should not significantly increase simulation time to a prohibitive extent.

It is worth mentioning that the influence of κ depends on the environment. Decreasing it too much can lead to too conservative policies and may not always work well. In addition, we observe the robustness-performance trade-off in the walker-run task once again. While large κ achieves high performance under the nominal kernel, it significantly underperforms when the kernel is perturbed.

5. Related works

Early works in RMDPs lay the theoretical foundations for solving RMDPs with robust dynamic programming (Wiesemann et al., 2013; Iyengar, 2005; Nilim & El Ghaoui, 2005; Kaufman & Schaefer, 2013; Bagnell et al., 2001). Recent works attempt to reduce the time complexity for certain uncertainty sets, such as L_1 uncertainty (Ho et al., 2018; 2021) and KL uncertainty (Grand-Clément & Kroer, 2021). However, they require full knowledge of the nominal model.

One line of work aims to design methods that can be applied in the online robust RL setting where we do not have full knowledge about the transition model. Derman et al. (2021) define new regularized robust Bellman operators that suggest a possible online sample-based method. However, the contraction of the Bellman operators implicitly assumes that the state space can not be very large. On regularizing the learning process, Kumar et al. (2022; 2023) introduce Q-learning and policy gradient methods for L_p uncertainty sets, but do not experimentally evaluate their methods with experiments. Another type of uncertainty is the R-contamination, for which previous works have derived a robust Q-learning algorithm (Wang & Zou, 2021) and a regularized policy gradient algorithm (Wang & Zou, 2022). R-contamination uncertainty assumes that the adversary can take the agent to any state, which is too conservative in practice. In addition, all of those methods are tied to a particular type of RL algorithm. Our work, however, aims to tackle the problem from a different perspective by approximating a worst kernel and can adopt any non-robust RL algorithm to learn an optimal robust policy. A recent work (Wang et al., 2023) has shown that the worst kernel can be computed using gradient descent, but their method takes more iterations to converge.

Outside of RMDP literature, perturbing the training environment was previously discussed in unsupervised environment design (Dennis et al., 2020; Jiang et al., 2021), domain randomization (Peng et al., 2018; Tobin et al., 2017), robust adversarial RL (Pinto et al., 2017; Rigter et al., 2022) and risk aversion (Greenberg et al., 2022; Pan et al., 2019). However, their focus on robustness differs from our perspective. They either assume access to environment parameters or aim for better generalization. Our method is theoretically driven as a solution to RMDPs.

Our work is also closely related to (Kumar et al., 2023), which characterizes the worst kernel for L_p uncertainty set. Different from their work, we propose to approximately simulate this worst kernel, opening a new paradigm for learning robust policies in RMDPs. The work of (Zhou et al., 2023) ran parallel to ours. They employ a sample method to establish a new manageable uncertainty set, enabling the computation of a robust Bellman operator through both value-based and policy-based methods. In contrast, our approach involves estimating the worst kernel through sampling from the nominal one to address the problem.

6. Conclusions and discussion

In this paper we introduce an approach that tackles the RMDP problem from a new perspective, by approximately simulating a worst transition kernel while leaving the RL part untouched. The highlight of our method is that it can be applied on top of any existing non-robust deep RL algorithms to learn robust policies, exhibiting attractive scalability to high-dimensional domains. We believe this new perspective will offer some insights for future works on RMDPs.

One limitation of our work is that we require the ability to sample next states from the transition model multiple times. In future work, we will study how to combine our method with a learned transition model (*i.e.*, world models (Hafner et al., 2020; Ha & Schmidhuber, 2018)) where the challenge of next-state sampling is mitigated. We also believe that using EWoK for model-based or offline setups might lower the effect of compounding error issue (Asadi et al., 2018). Another future direction might involve using EWoK as a robust solution to the Safe RL framework.

Impact statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

This Project has received funding from the European Union's Horizon Europe Programme under grant agreement No.101070568 and partially supported by Israel PBC-VATAT, by the Technion Artificial Intelligent Hub (Tech.AI) and by the Israel Science Foundation (grant No. 447/20).

Navdeep Kumar developed the initial idea and formulated the theories. Kaixin Wang and Uri Gadot refined the idea into a practical algorithm, conducted all experiments, and wrote the paper. Kfir Y. Levy and Shie Mannor provided supervision for this project.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

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

Bagnell, J. A., Ng, A. Y., and Schneider, J. G. Solving Uncertain Markov Decision Processes. *Technical Report*, 1 2001. doi: 10.1184/R1/6560927.v1. URL https://kilthub.cmu.edu/articles/journal_contribution/Solving_Uncertain_Markov_Decision_Processes/6560927.

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

Borkar, V. S. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Hindustan Book Agency, 2022. doi: 10.1007/978-81-951961-1-1. URL https://doi.org/10.1007%2F978-81-951961-1-1.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *ArXiv preprint*, abs/1606.01540, 2016. URL https://arxiv.org/abs/1606.01540.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., and Levine, S. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33: 13049–13061, 2020.

Derman, E., Geist, M., and Mannor, S. Twice regularized mdps and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34:22274–22287, 2021.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. An empirical investigation of the challenges of real-world reinforcement learning. *CoRR*, abs/2003.11881, 2020. URL https://arxiv.org/abs/2003.11881.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Grand-Clément, J. and Kroer, C. Scalable first-order methods for robust mdps. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 12086–12094. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17435.

Greenberg, I., Chow, Y., Ghavamzadeh, M., and Mannor, S. Efficient risk-averse reinforcement learning. *arXiv preprint arXiv:2205.05138*, 2022.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018,* *December 3-8, 2018, Montréal, Canada*, pp. 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Ho, C. P., Petrik, M., and Wiesemann, W. Fast bellman updates for robust mdps. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1984–1993. PMLR, 2018. URL http://proceedings.mlr.press/v80/ho18a.html.

Ho, C. P., Petrik, M., and Wiesemann, W. Partial policy iteration for l1-robust markov decision processes. *The Journal of Machine Learning Research*, 22(1):12612–12657, 2021.

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

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., and Rocktäschel, T. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

Kaufman, D. L. and Schaefer, A. J. Robust modified policy iteration. *INFORMS J. Comput.*, 25:396–410, 2013.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kumar, N., Levy, K., Wang, K., and Mannor, S. Efficient policy iteration for robust Markov decision processes via regularization. *ArXiv preprint*, abs/2205.14327, 2022. URL https://arxiv.org/abs/2205.14327.

Kumar, N., Derman, E., Geist, M., Levy, K., and Mannor, S. Policy gradient for s-rectangular robust markov decision processes. *ArXiv preprint*, abs/2301.13589, 2023. URL https://arxiv.org/abs/2301.13589.

Mannor, S., Simester, D., Sun, P., and Tsitsiklis, J. N. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

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

Pan, X., Seita, D., Gao, Y., and Canny, J. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8522–8528. IEEE, 2019.

Panaganti, K. and Kalathil, D. Sample complexity of robust reinforcement learning with a generative model. In *International Conference on Artificial Intelligence and Statistics*, pp. 9582–9602. PMLR, 2022.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826. PMLR, 2017. URL http://proceedings.mlr.press/v70/pinto17a.html.

Puterman, M. L. Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*, 1994.

Raffin, A. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Rigter, M., Lacerda, B., and Hawes, N. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *arXiv preprint arXiv:2204.12581*, 2022.

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

Shi, L. and Chi, Y. Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity. *arXiv preprint arXiv:2208.05767*, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

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

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In Schuurmans, D. and Wellman, M. P. (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 2094–2100. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389.

Wang, Q., Ho, C. P., and Petrik, M. Policy gradient in robust mdps with global convergence guarantee. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35763–35797. PMLR, 2023. URL https://proceedings.mlr.press/v202/wang23i.html.

Wang, Y. and Zou, S. Online robust reinforcement learning with model uncertainty. *ArXiv preprint*, abs/2109.14523, 2021. URL https://arxiv.org/abs/2109.14523.

Wang, Y. and Zou, S. Policy gradient method for robust reinforcement learning. *International Conference on Machine Learning*, 162:23484–23526, 2022.

Wang, Y., Miao, F., and Zou, S. Robust constrained reinforcement learning. *ArXiv preprint*, abs/2209.06866, 2022. URL https://arxiv.org/abs/2209.06866.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8:279–292, 1992.

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

Xu, Z., Panaganti, K., and Kalathil, D. Improved sample complexity bounds for distributionally robust reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 9728–9754. PMLR, 2023.

Yarats, D. and Kostrikov, I. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

Zhou, R., Liu, T., Cheng, M., Kalathil, D., Kumar, P., and Tian, C. Natural actor-critic for robust reinforcement learning with function approximation. *arXiv preprint arXiv:2307.08875*, 2023.

A. Proof

A.1. Proof of Theorem 3.2

Recall that the worst values are defined as

$$P^\pi_{\mathscr{P}} \in \arg\min_{P\in\mathscr{P}} J^\pi_P$$

for any general uncertainty set $\mathscr{P}$. Further, for sa-rectangular uncertainty set $\mathscr{P} = \times_{s\in\mathcal{S},a\in\mathcal{A}}\mathscr{P}_{sa}$, the robust value function exists, that is, the following is well defined (Nilim & El Ghaoui, 2005; Iyengar, 2005)

$$v^\pi_{\mathscr{P}} = \min_{P\in\mathscr{P}} v^\pi_P.$$

This implies,

$$v^\pi_{\mathscr{P}} =\Big(I - \gamma(P^\pi_{\mathscr{P}})^\pi \Big)^{-1} R^\pi$$

is the fixed point of robust Bellman operator $T^\pi_{\mathscr{P}}$(Nilim & El Ghaoui, 2005; Iyengar, 2005), defined as

$$T^\pi_{\mathscr{P}} v := \min_{P\in\mathscr{P}} T^\pi_P v.$$

Proposition A.1. *The worst values can be computed from the robust value function. That is*

$$\arg\min_{P\in\mathscr{P}} T^\pi_P v^\pi_{\mathscr{P}} \subseteq \arg\min_{P\in\mathscr{P}} v^\pi_P \subseteq \arg\min_{P\in\mathscr{P}} J^\pi_P.$$

Proof. Let

$$P^* \in \arg\min_{P\in\mathscr{P}} T^\pi_P v^\pi_{\mathscr{P}}.$$

Now, from the fixed point of robust Bellman operator, we have

$$\begin{aligned} v^\pi_{\mathscr{P}} =& T^\pi_{\mathscr{P}} v^\pi_{\mathscr{P}} = \min_{P\in\mathscr{P}} T^\pi_P v^\pi_{\mathscr{P}}, \\ &= T^\pi_{P^*} v^\pi_{\mathscr{P}}, \qquad \text{(by construction)}, \\ &= R^\pi + \gamma(P^*)^\pi v^\pi_{\mathscr{P}}, \qquad \text{(by definition)}. \end{aligned}$$

The above implies,

$$v^\pi_{\mathscr{P}} =\Big(I - \gamma(P^*)^\pi \Big)^{-1} R^\pi.$$

This implies,

$$P^* \in \arg\min_{P\in\mathscr{P}} v^\pi_P.$$

The last inclusion is trivial, that is, every minimizer of value function is a minimizer of robust return. □

Theorem 3.2. *For a KL uncertainty set $\mathscr{P}$ and a policy π, a worst kernel is related to the nominal kernel through:*

$$P^\pi_{\mathscr{P}}(s'|s,a) = \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')},$$

where δ^π is of the form

$$\delta^\pi(s') = \frac{v^\pi_{\mathscr{P}}(s') - \omega_{sa}}{\kappa_{sa}}, \tag{1}$$

and satisfies

$$\begin{aligned} &\sum_{s'\in\mathcal{S}} \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')} = 1, \\ &\sum_{s'\in\mathcal{S}} \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')}(-\delta^\pi(s')) = \beta_{sa}. \end{aligned} \tag{2}$$

Proof. Recall Definition 3.1

$$P^\pi_{\mathscr{P}} \in \arg\min_{P\in\mathscr{P}} J^\pi_P.$$

From Proposition A.1, for `sa`-rectangular uncertainty set $\mathscr{P}$, a worst kernel can be computed using robust value function as

$$P^\pi_{\mathscr{P}} \in \arg\min_{P\in\mathscr{P}} T^\pi_P v^\pi_{\mathscr{P}}.$$

Recall, our KL-constrained uncertainty $\mathscr{P}$ is defined as

$$\mathscr{P} := \{P \mid P \in (\Delta_{\mathcal{S}})^{\mathcal{S}\times\mathcal{A}}, D_{KL}(\bar{P}_{s,a}, P_{sa}) \leq \beta_{sa}, \forall s, a\}.$$

where D_{KL} is KL norm that is defined as

$$D_{KL}(P, Q) = \sum_s P(s) \log\left(\frac{P(s)}{Q(s)}\right).$$

Using Proposition A.1 and definition of uncertainty set $\mathscr{P}$, the worst kernel can be extracted as

$$P^\pi_{\mathscr{P}}(\cdot|s,a) \in \arg\min_{D_{KL}(p, P_0(\cdot|s,a))\leq\beta_{sa}, \sum_s p(s)=1, p\succeq 0} \langle p, v^\pi_{\mathscr{P}}\rangle.$$

Using the Lemma A.2, we get the desired solution. □

In the following lemma, we use $\frac{p}{q}$ as a shorthand for the vector containing entry-wise ratios $\frac{p(i)}{q(i)}$. For example, $p\ln(\frac{p}{q}) \leq \beta$ represents $\sum_i p(i)\ln(\frac{p(i)}{q(i)}) \leq \beta$.

Lemma A.2. *For $q \in \Delta_{\mathcal{S}}, v \in \mathbb{R}^{\mathcal{S}}, \beta \geq 0$, a solution to*

$$\min_{p\ln(\frac{p}{q})\leq\beta, 1^T p=1, p\succeq 0} \langle p, v\rangle.$$

is given by

$$p = qe^{-\frac{v-\omega}{\lambda}},$$

where

$$p\log(\frac{p}{q}) = \Big\langle\ qe^{-\frac{v-\omega}{\lambda}}, \frac{v-\omega}{\lambda}\ \Big\rangle = -\beta$$

and

$$\sum_s q(s)e^{-\frac{v(s)-\omega}{\lambda}} = 1.$$

Proof. We have the following optimization problem,

$$\min_{p\ln(\frac{p}{q})\leq\beta, 1^T p=1, p\succeq 0} \langle p, v\rangle.$$

We ignore the constraint $p \succeq 0$ for the moment (as we see later, this constrained is automatically satisfied), and focus on

$$\min_{p\ln(\frac{p}{q})\leq\beta, 1^T p=1} \langle p, v\rangle.$$

We define Lagrange multiplier as

$$L(p, \lambda, \mu) = \langle p, v\rangle + \lambda\ \Big(\ p\ln(\frac{p}{q}) - \beta\ \Big) + \mu\ \Big(\ 1^T p - 1\ \Big).$$

We now put the stationarity condition:

$$\begin{aligned} &\frac{\partial L}{\partial p} = v + \lambda\Big(\ln(\frac{p}{q}) + 1\Big) + \mu 1 = 0 \\ \implies & p = qe^{-1}e^{-\frac{v+\mu}{\lambda}}. \end{aligned}$$

With appropriate change of variable $\mu \to \omega$, we have

$$p = qe^{-\frac{v-\omega}{\lambda}}.$$

We have to find the constants ω and λ, using the constraints

$$p\log(\frac{p}{q}) = \Big\langle qe^{-\frac{v-\omega}{\lambda}}, \frac{v-\omega}{\lambda}\Big\rangle = -\beta$$

and

$$\sum_s p(s) = \sum_s q(s)e^{-\frac{v(s)-\omega}{\lambda}} = 1.$$

We further note that the constraint $1 \geq p(s) \geq 0$ is automatically satisfied as

$$p(s) = q(s)e^{-\frac{v(s)-\omega}{\lambda}} \geq 0$$

and $\sum_s p(s) = 1$, ensures $p(s) \leq 1 \quad \forall s$. □

A.2. Proof of Proposition 3.3 and 3.4

Proposition A.3. *ω_{sa} can be upper-bounded as follows,*

$$\omega_{sa} \leq \langle \bar{P}(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle, \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Proof. From the constraint in Theorem 3.2, we have

$$\begin{aligned} &\sum_{s'} \bar{P}(s'|s,a)e^{-\frac{v^\pi_{\mathscr{P}}(s')-\omega_{sa}}{\kappa_{sa}}} = 1 \\ \implies & e^{-\sum_{s'}\bar{P}(s'|s,a)\frac{v^\pi_{\mathscr{P}}(s')-\omega_{sa}}{\kappa_{sa}}} \leq 1 \qquad \text{(using Jenson's inequality)} \\ \implies & e^{-\sum_{s'}\bar{P}(s'|s,a)\frac{v^\pi_{\mathscr{P}}(s')}{\kappa_{sa}}} e^{\frac{\omega_{sa}}{\kappa_{sa}}} \leq 1 \\ \implies & \frac{\omega_{sa}}{\kappa_{sa}} \leq \sum_{s'} \bar{P}(s'|s,a)\frac{v^\pi_{\mathscr{P}}(s')}{\kappa_{sa}} \\ \implies & \omega_{sa} \leq \sum_{s'} \bar{P}(s'|s,a)v^\pi_{\mathscr{P}}(s'). \end{aligned}$$ □

Proposition 3.3. *ω_{sa}, κ_{sa} and β_{sa} satisfy*

$$\omega_{sa} = \langle P^\pi_{\mathscr{P}}(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle + \beta_{sa}\kappa_{sa},$$

Proof. From the constraint in Theorem 3.2, we have

$$\begin{aligned} &\sum_{s'} \bar{P}^\pi(s'|s,a)e^{-\delta^\pi(s')}(-\delta^\pi(s')) = \beta_{sa} \\ \implies & \sum_{s'} P^\pi_{\mathscr{P}}(s'|s,a)\frac{v^\pi_{\mathscr{P}}(s') - \omega_{sa}}{\kappa_{sa}} = \beta_{sa} \\ \implies & \sum_{s'} P^\pi_{\mathscr{P}}(s'|s,a)v^\pi_{\mathscr{P}}(s') = -\beta_{sa}\kappa_{sa} + \omega_{sa}. \end{aligned}$$ □

Proposition 3.4. *ω_{sa} is bounded as follows,*

$$\langle P^\pi_{\mathscr{P}}(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle \leq \omega_{sa} \leq \langle \bar{P}^\pi(\cdot|s,a), v^\pi_{\mathscr{P}}\rangle.$$

Proof. The lower bound is direct from Proposition 3.3, as β and κ are positive quantities by definition. The upper bound comes from Proposition A.3. □

A.3. Proof of Theorem 3.5

Given a policy π, let P_{n+1} be the updated kernel:

$$P_{n+1} = \arg\min_{P\in\mathscr{P}} T^\pi_P v^\pi_{P_n}.$$

We continue to prove the following lemmas.

Lemma A.4. *The kernel update process produces monotonically decreasing value functions:*

$$v^\pi_{P_n} \succeq v^\pi_{P_{n+1}}, \quad \forall n = 1, 2, \cdots .$$

Where $\succeq$ refers to an entry-wise comparison operator (i.e. $\forall s \in \mathcal{S} : v^\pi_{P_n}(s) \geq v^\pi_{P_{n+1}}(s)$).

Proof. Recall that $v^\pi_{P_n} = T^\pi_{P_n} v^\pi_{P_n} = R^\pi + \gamma P^\pi_n v^\pi_{P_n}$. Since we have

$$P_{n+1} = \arg\min_{P\in\mathscr{P}}[R^\pi + \gamma P^\pi v^\pi_{P_n}],$$

we can obtain

$$\begin{aligned}
 & R^\pi + \gamma P^\pi_n v^\pi_{P_n} \geq \min_{P\in\mathscr{P}}[R^\pi + \gamma P^\pi v^\pi_{P_n}] \\
\Rightarrow\quad & v^\pi_{P_n} \geq R^\pi + \gamma P^\pi_{n+1} v^\pi_{P_n} \\
\Rightarrow\quad & (I - \gamma P^\pi_{n+1}) v^\pi_{P_n} \geq R^\pi \\
\Rightarrow\quad & v^\pi_{P_n} \geq (I - \gamma P^\pi_{n+1})^{-1} R^\pi = v^\pi_{P_{n+1}}.
\end{aligned}$$
□

Lemma A.5. *The robust bellman operators are monotonic functions, that is:*

$$v \leq u \quad\Longrightarrow\quad T^\pi_{\mathscr{P}} v \leq T^\pi_{\mathscr{P}} u$$

Proof. Since $v \leq u$, and the fact that P has only non-negative entries, we know that:

$$\begin{aligned}
 & R^\pi + \gamma P^\pi v \leq R^\pi + \gamma P^\pi u, \quad \forall P \in \mathscr{P} \\
\Rightarrow\quad & \min_{P\in\mathscr{P}}(R^\pi + \gamma P^\pi v) \leq \min_{P\in\mathscr{P}}(R^\pi + \gamma P^\pi u) \\
\Rightarrow\quad & T^\pi_{\mathscr{P}} v \leq T^\pi_{\mathscr{P}} u
\end{aligned}$$

□

Theorem 3.5. *For the updating process in Eqn. (3), we have*

$$\|v^\pi_{P_n} - v^\pi_{\mathscr{P}}\|_\infty \leq \gamma^n \|v^\pi_P - v^\pi_{\mathscr{P}}\|_\infty.$$

Proof. We prove it by showing that:

$$\|v^\pi_{P_{n+1}} - v^\pi_{\mathscr{P}}\|_\infty \leq \gamma \|v^\pi_{P_n} - v^\pi_{\mathscr{P}}\|_\infty, \quad \forall n.$$

First, by optimality, we have

$$v^\pi_{P_{n+1}} - v^\pi_{\mathscr{P}} \geq 0.$$

Now we can focus only on the upper bound:

$$\begin{aligned}
v^\pi_{P_{n+1}} - v^\pi_{\mathscr{P}} &= T^\pi_{P_{n+1}} v^\pi_{P_{n+1}} - T^\pi_{\mathscr{P}} v^\pi_{\mathscr{P}} \\
&\leq T^\pi_{P_{n+1}} v^\pi_{P_n} - T^\pi_{\mathscr{P}} v^\pi_{\mathscr{P}} && \text{(Lemma A.4 and A.5)} \\
&= \min_{P \in \mathscr{P}} T^\pi_P v^\pi_{P_n} - T^\pi_{\mathscr{P}} v^\pi_{\mathscr{P}} \\
&= T^\pi_{\mathscr{P}} v^\pi_{P_n} - T^\pi_{\mathscr{P}} v^\pi_{\mathscr{P}} \\
&\leq \gamma \| v^\pi_{P_n} - v^\pi_{\mathscr{P}} \|_\infty && (T^\pi_{\mathscr{P}} \text{ is a } \gamma\text{-contraction operator}).
\end{aligned}$$

Putting it together, we have

$$\| v^\pi_{P_{n+1}} - v^\pi_{\mathscr{P}} \|_\infty \leq \gamma \| v^\pi_{P_n} - v^\pi_{\mathscr{P}} \|_\infty .$$

The desired result is proved by applying the above result iteratively. □

A.4. Simplification of Algorithm 1

Algorithm 1 is a *practical* algorithm that updates the policy, value, and estimates worst transition dynamic simultaneously. It also involves various approximations (e.g., sampling finite N next states, using sample-based updates), making it difficult to analyze. To facilitate theoretical investigation on the convergence, we isolate the policy improvement part and focus on the core of our EWoK method: estimating a worst transition kernel $P^\pi_{\mathcal{P}}$ for a fixed policy π.

Specifically, in each iteration, we compute the non-robust value function for an estimated worst transition dynamic P_n and then update the estimated worst dynamic to P_{n+1} according to Eqn. (3). For better clarity, we summarize this iterative process in the following multi-time-scale algorithm:

```
**Input:** Fixed policy $\pi$, nominal transition dynamic $\bar{P}$, reward function $R$
**Output:** Robust value function $v^\pi_{\mathscr{P}}$, worst transition dynamic $P^\pi_{\mathcal{P}}$ for policy $\pi$

Initialize $P_1 \leftarrow \bar{P}$
**For** $n = 1, 2, \ldots$ until convergence:
    If $n = 1$, randomly initialize $v_1$, otherwise $v_1 \leftarrow v^\pi_{P_n}$
    **For** $m = 1, 2, \ldots$ until convergence:                    [Policy evaluation]
        $v_{m+1} = R^\pi + \gamma P^\pi_n v_m$
    $v^\pi_{P_n} \leftarrow v_m$
    $P_{n+1} = \bar{P} e^{-\frac{v^\pi_{P_n} - \omega_n}{\kappa_n}}$                          [ Eqn. ( 3)]
$v^\pi_{\mathcal{P}} \leftarrow v^\pi_{P_n}$, $P^\pi_{\mathcal{P}} \leftarrow P_n$
```

Theorem 3.5 establishes the convergence of the above algorithm. In the practical Algorithm 1, the update of v_m and P_n are intertwined. With appropriate conditions (Borkar, 2022), Theorem 3.5 suggests the convergence of the above mentioned algorithm.

B. Experiment details

B.1. Cliff Walking

B.1.1. Environment description

Cliff Walking [1] is a tabular environment from OpenAI's Gym (Brockman et al., 2016). Usually, this environment transition kernel is deterministic. To create a distributional uncertainty set around the nominal kernel, we We introduced stochasticity to the transition kernel. Specifically, we decreased the probability of moving in the intended direction from 1 to 0.9. Additionally, we introduced a 0.02 probability of moving in the opposite direction (e.g., Up instead of Down), and the remaining 0.08 probability is evenly distributed between moving sideways (e.g., Left or Right instead of Up). Consequently, there is a 0.04 probability of moving in any of these lateral directions.

In terms of reward - the agent receives a -1 penalty every time step before reaching the goal state. When it does, it encounters a reward of 100, and each time the agent falls off the cliff, it suffers from a penalty of -10 and will be teleported to the initial state.

B.1.2. Worst Environment

To determine the worst transition kernel, we computed, for each (s, a) pair, the updated worst transition. This involved encouraging actions that would move the agent closer to the cliff. For instance, if the agent is positioned adjacent to the cliff and moves upward, we would try to find a transition kernel such that the probability of moving downward is maximal (while staying within the bounds of the uncertainty set).

Specifically Given $p \in \Delta_{\mathcal{S}}$ we want to find q by solving the following optimization problem:

$$\begin{aligned} &\max \; q[i] \\ \text{s.t.} &\sum_{s \in \mathcal{S}} q[s] = 1 \\ &q[s] \geq 0, \quad \forall s \in \mathcal{S} \\ &D_{KL}(q||p) \leq \beta \end{aligned}$$

where i is the outcome of getting closer to the cliff.

B.1.3. Q-Learning Hyperparameters

To obtain the optimal policy we used Value iteration (Puterman, 1994), using the access to the true kernel (either nominal or worst). To learn the robust policy we used a simple Q-learning algorithm (Watkins & Dayan, 1992) using samples from the nominal kernel (while allowing multiple samples for any timesteps). The detailed configurations for hyperparameters used in this experiment are summarized in Table 1.

B.2. Environments

B.2.1. Classic control tasks

Cartpole [2] is one of the classic control tasks in OpenAI Gym (Brockman et al., 2016). The task is to balance a pendulum on a moving cart, by moving the cart either left or right. The state consists of the location and velocity of the cart, as well as the angle and angular velocity of the pendulum. To make the transition dynamic stochastic, we add Gaussian noises to the cart position. The detailed configurations for the nominal values and the perturbation ranges are summarized in Table 2.

B.2.2. DeepMind Control Suite

The DeepMind Control Suite (Tunyasuvunakool et al., 2020) is a set of continuous control tasks powered by the MuJoCo physics engine (Todorov et al., 2012). It is widely used to benchmark reinforcement learning agents. As mentioned in the main text, we consider 3 tasks in our paper: `walker-stand`, `walker-walk`, `walker-run`. For those tasks, the observations are 24-dimensional vectors and the actions are 6-dimensional vectors. For noise perturbation, we fix the

[1] https://gymnasium.farama.org/environments/toy_text/cliff_walking/

[2] https://gymnasium.farama.org/environments/classic_control/cart_pole/

Table 1. Hyperparameters used in Cliff Walking environment

PARAMETER	VALUE
Q-Learning Hyperparameters	
Number of episodes	20000
Learning rate	0.01
Discount factor (γ)	0.8
Exploration factor ϵ	0.2
Value Error stopping condition	$1e^{-6}$
Uncertainty set parameters	
β	0.4
EWoK Hyperparameters	
Number of samples (N)	5
κ	0.4

Table 2. Perturbation configurations for Cartpole environment.

	PARAMETER	NOMINAL VALUE	PERTURBATION RANGE
NOISE PERTUBRATION	Cart position noise (std)	0.01	$[0, 0.1]$
ENV. PARAM. PERTUBRATION	Pole mass	0.1	$[0.15, 3.0]$
	Pole length	0.5	$[0.25, 5.0]$
	Cart mass	1	$[0.25, 5.0]$
	Gravity	9.8	$[0.1, 30]$

standard deviation of the Gaussian noise to 0.2. The nominal value and the perturbation range are summarized in Table 3.

Table 3. Perturbation configurations for DeepMind Control Suite tasks.

	PARAMETER	NOMINAL VALUE	PERTURBATION RANGE
NOISE PERTUBRATION	action noise (mean)	0.0	$[-0.3, 0.3]$
ENV. PARAM. PERTUBRATION	thigh length	0.225	$[0.1, 0.5]$
	torso length	0.3	$[0.1, 0.7]$
	joint damping	0.1	$[0.1, 10]$

B.3. Training and evaluation

For our method and the baseline, we first train the agent under the nominal environment, and then for each perturbed environment during testing, we calculate the average reward from 30 episodes. We repeat this process with 40 random seeds for DDQN experiments and 20 random seeds for PPO experiments in the classic control environments. For DeepMind Control environments we used 10 different seeds for SAC experiments and 5 different seeds for TD3 experiments. Following the recommended practice in (Agarwal et al., 2021), we report the Interquartile Mean (IQM) and the 95% stratified bootstrap confidence intervals (CIs), using The IQM metric is measured by discarding the top and bottom 25% of the results, and averaging across the remaining middle 50%. IQM has the benefit of being more robust to outliers than a regular mean, and being a better estimator of the overall performance than the median. We use the rliable library[3] to calculate IQM and CIs.

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

As mentioned earlier, we use Double-DQN (van Hasselt et al., 2016) and PPO (Schulman et al., 2017) as the vanilla non-robust RL algorithms for environments with discrete action spaces. Specifically, we follow the implementation in Stable-Baselines3 (Raffin et al., 2021). For the Cartpole environment, we use the hyperparameters suggested in RL Baselines3 Zoo (Raffin, 2020). The detailed configurations are summarized in Table 4 and Table 5.

Table 4. Hyperparameters for training cartpole with DDQN used in the experiments.

PARAMETER	VALUE
batch size	64
buffer size	100000
exploration final epsilon	0.04
exploration fraction	0.16
gamma	0.99
gradient steps	128
learning rate	0.0023
learning starts	1000
target update interval	10
train frequency	256
total time-steps	50000

Table 5. Hyperparameters for training cartpole with PPO used in the experiments.

PARAMETER	VALUE
number of parallel environments	1
batch size	32
Number of steps	32
gae lambda	0.98
gamma	0.98
Number of epochs	20
entropy coefficient	0
clip range	0.2
learning rate	0.001
total time-steps	100000

For environments with continuous action spaces, we choose the SAC algorithm (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018) as the vanilla non-robust RL algorithms. For SAC, we follow the implementations and hyperparameters choices in (Yarats & Kostrikov, 2020). Both the actor and critic use a two-layer MLP neural network with 1024 hidden units per layer. Table 6 lists the hyperparameters. For TD3, we follow the implementations and hyperparameters choices in (Raffin et al., 2021). Table 7 lists the hyperparameters.

The configurations for the sample size N and the robustness parameter κ used in our experiments are summarized in Table 8.

B.4. Computational resources and costs

We used the following resources in our experiments:

- **CPU:** AMD EPYC 7742 64-Core Processor
- **GPU:** NVIDIA GeForce RTX 2080 Ti

Table 9 lists the training time.

Table 6. Hyperparameters for SAC used in the experiments.

PARAMETER	VALUE
Total steps	1e6
Warmup steps	5000
Replay size	1e6
Batch size	1024
Discount factor γ	0.99
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	1e-4
Target smoothing coefficient	0.005
Target update interval	2
Actor update interval	1
Initial temperature	0.1
Learnable temperature	Yes

Table 7. Hyperparameters for TD3 used in the experiments.

PARAMETER	VALUE
Total steps	1e6
Warmup steps	5000
Replay size	1e6
Batch size	1024
Discount factor γ	0.99
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	1e-4
Target smoothing coefficient	0.005
Target update interval	2
Actor update interval	2
Target policy noise std	0.2
Target policy noise clip	0.5

Table 8. Hyperparameters specific to our method used in the experiments.

	ENVIRONMENT	SAMPLE SIZE N	ROBUSTNESS PARAMETER κ
CLASSIC CONTRO	Cartpole (DDQN)	15	0.1
	Cartpole (PPO)	15	0.2
DEEPMIND CONTROL SUITE	walker-stand (SAC & TD3)	10	0.2
	walker-walk (SAC & TD3)	10	0.2
	walker-run (SAC & TD3)	10	0.2

Table 9. Training time per run of our experiments on a single GPU.

	ENVIRONMENT	BASELINE	EWOK(OURS)
CLASSIC CONTROL (DDQN)	Cartpole	$\sim$ 4 minutes	$\sim$ 5 minutes
CLASSIC CONTROL (PPO)	Cartpole	$\sim$ 60 minutes	$\sim$ 60 minutes
DM CONTROL (SAC)	all tasks	$\sim$ 5 hours	$\sim$ 6 hours
DM CONTROL (TD3)	all tasks	$\sim$ 6 hours	$\sim$ 7 hours

C. Comparison of EWoK to existing Dynamic programming methods

Even though existing methods do not scale well to high-dimensional environments. To test how EWoK compares to existing robust model-based methods, we chose the Robust Dynamic Programming (Robust DP) approach in (Nilim & El Ghaoui, 2005). Their solution for the KL-based rectangular uncertainty set involves a bisection algorithm to solve the inner optimization problem of the robust Bellman operator. To test how our algorithm compared to the Robust DP approach, we considered the simple MDP depicted in Figure 9.

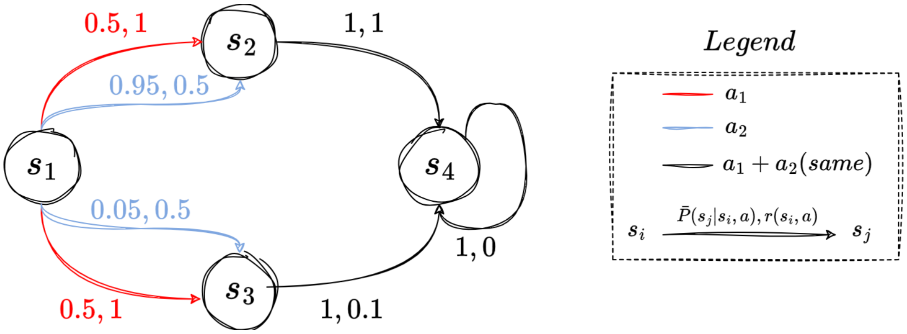


Figure 9. Simple MDP used in comparison experiments.

For this simple MDP, under the nominal transition kernel $\bar{P}$ the optimal action at state s_1 is a_1. But allowing KL-perturbation (e.g. $D_{\mathrm{KL}}(P(\cdot|s,a) \parallel \bar{P}(\cdot|s,a)) \leq 0.2, \quad \forall(s,a)$) can cause the optimal robust action (under worst possible dynamics) to change into a_2.

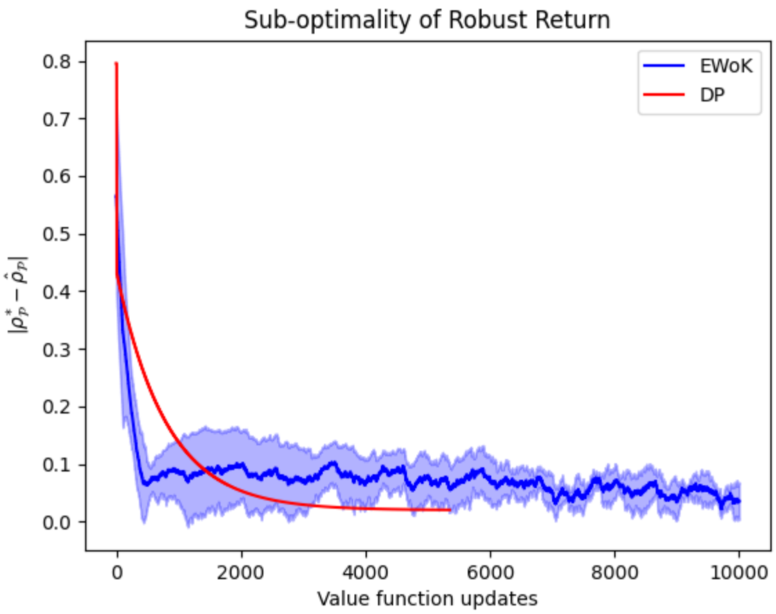
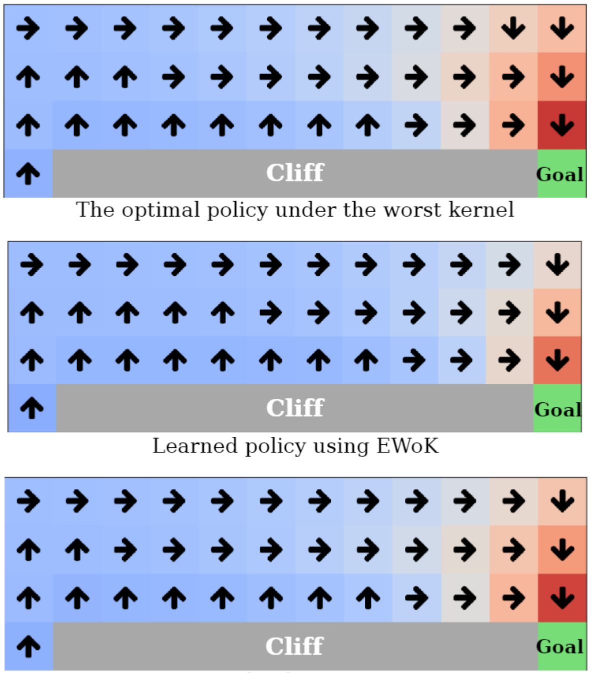


Figure 10. Error of estimated robust return, Q-Learning algorithm on the simple MDP.

We applied both EWoK with model-free Q learning and Robust DP with bisection algorithm, we plotted the error of the estimated robust return w.r.t. the optimal robust return for both methods in Figure 10. The x-axis represents the number of Q-function updates. (notice that the comparison is not ”apples to apples” since EWoK is a sample-based approach while Robust DP utilizes the knowledge of the transition kernel) Since the sampling mechanism in EWoK introduces more noise to the process, we ran this experiment over 5 different seeds and presented the mean and standard deviation. We can see that EWoK converges into a very similar robust return while achieving the same optimal robust policy as the DP approach.

In addition, we tried the Robust DP approach on the cliff-walking toy environment (with some reward scaling changes for



The optimal policy under the worst kernel

Learned policy using EWoK

Learned policy using DP

Figure 11. Experiments on a modified Cliff-Walking environment to compare EWoK with DP method.

numerical considerations). The learned optimal robust policy turned out pretty similar to the one we achieved using EWoK (as can be seen in Figure 11.

D. Additional results

In section 4.3 we show the result of testing EWoK on different parameters perturbation. Here we show the results of all of the experimented parameters we have perturbed during test time. The results for SAC algorithm can be seen in Figure 12 and the results for TD3 algorithm can be seen in Figure 13.

In section 4.4, we show the relative performance for the ablation study on parameter κ and N for two environments only. In Figures 14 and 16 we depict the full results for all of the 3 environments. In Figures 15 and 17 we also include the absolute results (with the reference baseline algorithm).

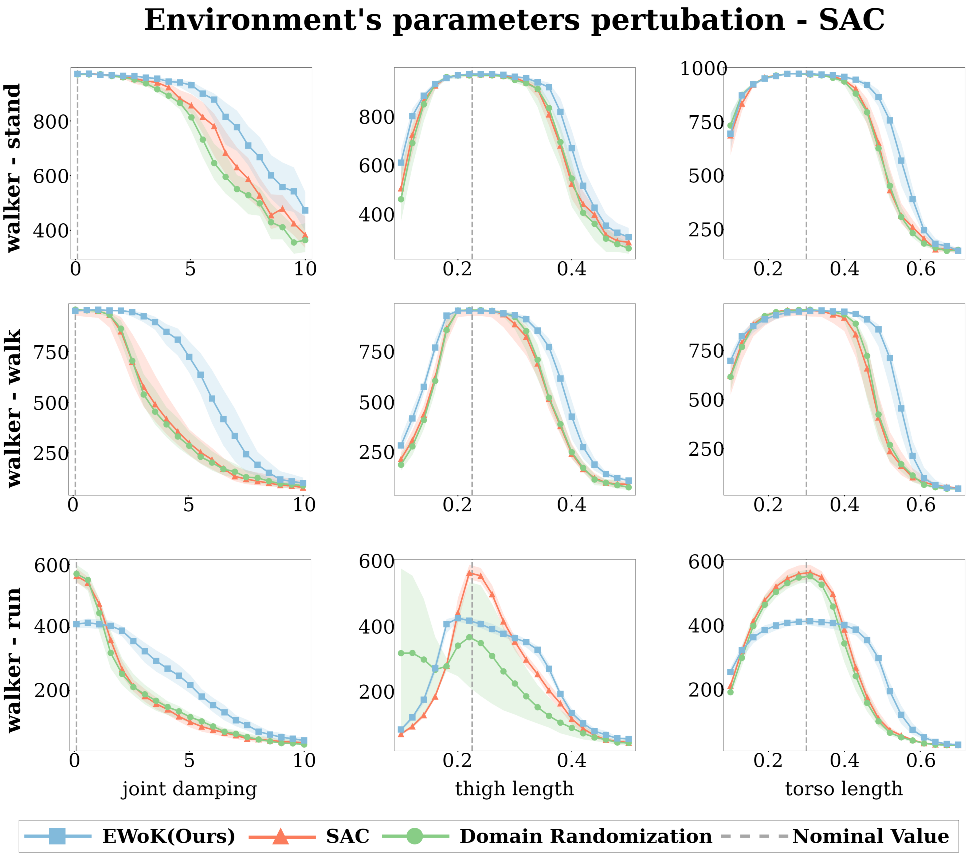


Figure 12. Evaluation results on DeepMind Control environments with environment's parameters perturbations for SAC algorithm.

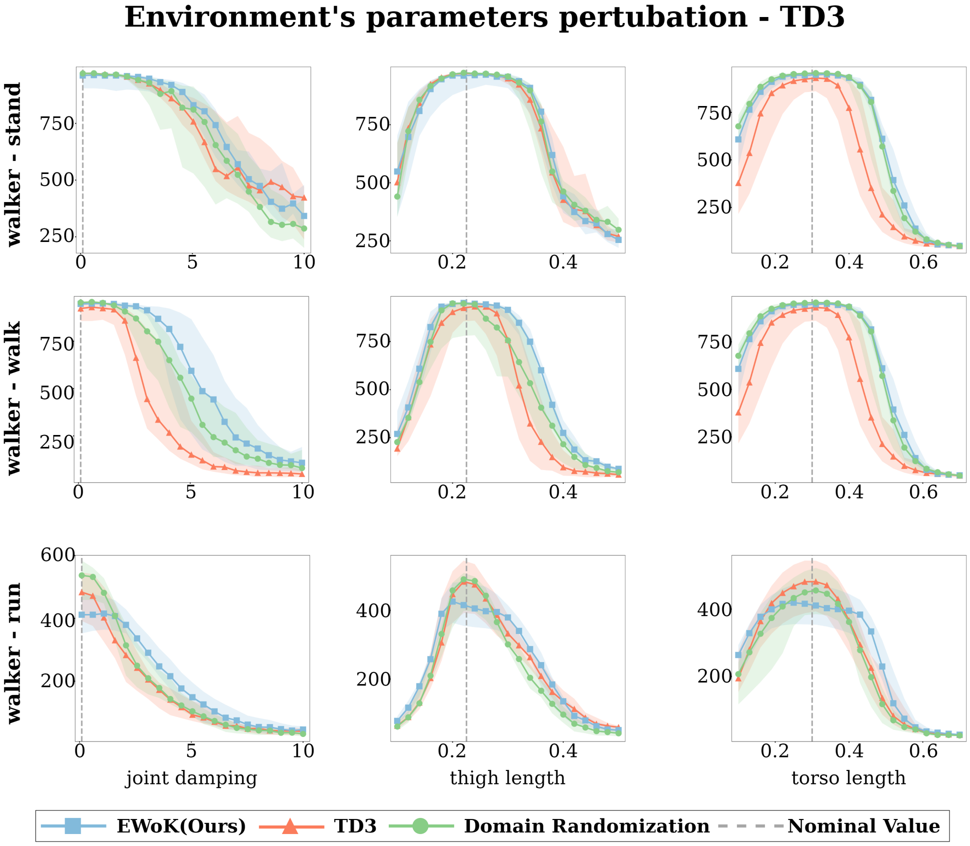


Figure 13. Evaluation results on DeepMind Control environments with environment's parameters perturbations for TD3 algorithm.

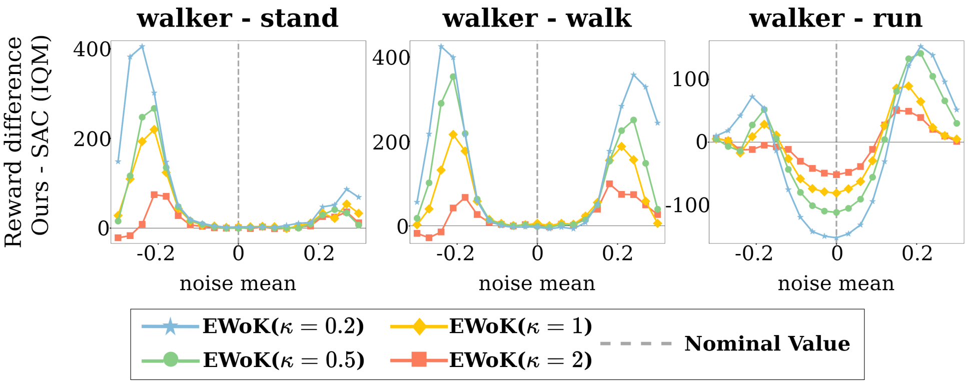


Figure 14. Evaluation results on DeepMind Control tasks with noise perturbations for different κ.

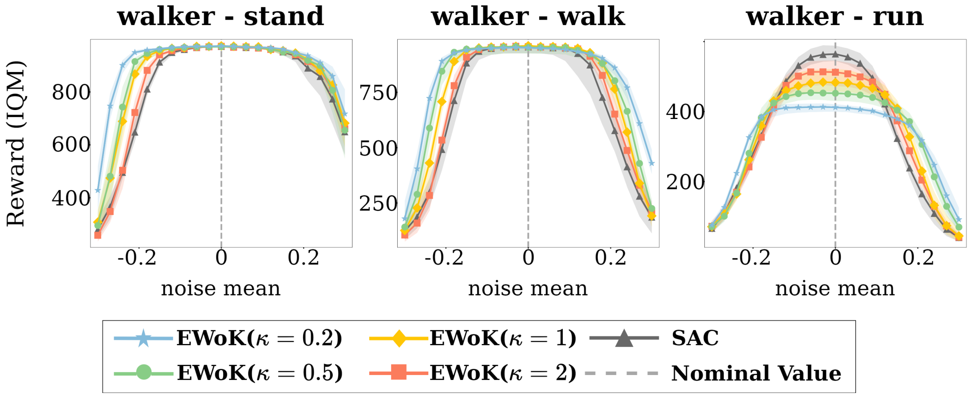


Figure 15. Evaluation results (absolute) on DeepMind Control tasks with noise perturbations for different κ.

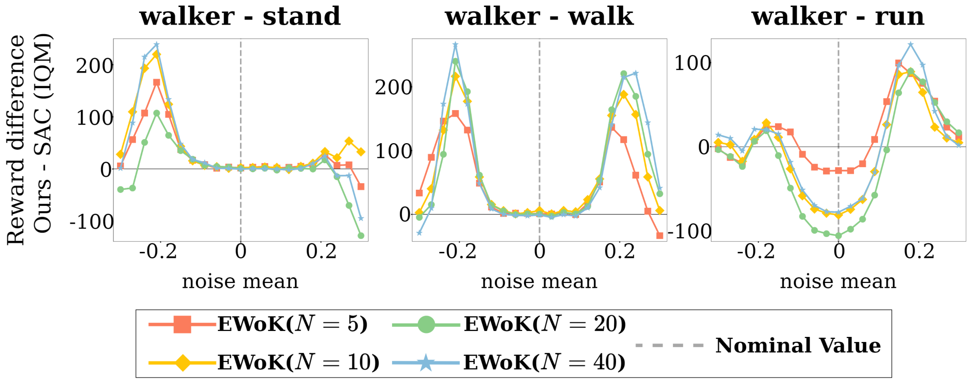


Figure 16. Evaluation results on DeepMind Control tasks with noise perturbations for different N.

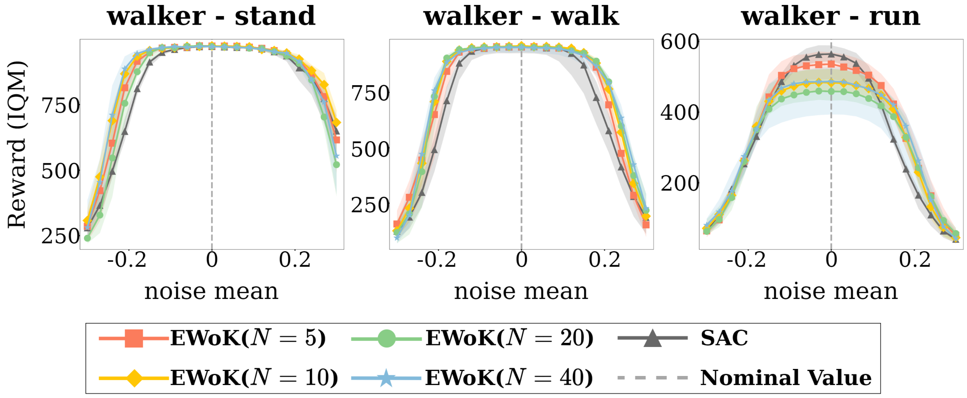


Figure 17. Evaluation results (absolute) on DeepMind Control tasks with noise perturbations for different N.