Bridging the Sim-to-Real Gap with Deep Robust Asymmetrical Q-Learning

Xutao Ma[1] **Junhao Zhao**[1] **Ruirong Chen**[1] **Zuming Liu**[2] **Chao Ning**[1*]

Abstract

The sim-to-real gap presents a significant challenge for implementing deep reinforcement learning (RL) in real-world applications, and robust RL frameworks offer a promising approach to bridge this gap. However, existing deep robust RL methods often struggle to accurately model the kernel uncertainty and to train robust agents efficiently. To overcome these limitations, we first propose a novel R-τ-contamination uncertainty set, which leverages the capabilities of the simulator to model kernel uncertainty with greater precision, accounting for parameter mismatches and noise. Additionally, to efficiently train the robust RL agent, we propose a fully sample-based **D**eep **R**obust **As**ymmetrical **Q**-**L**earning (**DRASQL**) method that employs an asymmetrical penalty to learn a robust Q-function, without requiring adversarial training or worst kernel estimation. Theoretically, we prove the convergence of the corresponding DRASQL-TD algorithm in the tabular RL case. Furthermore, for deep function approximation, our method can be seamlessly integrated into off-the-shelf *non-robust* deep RL algorithms, transforming them into robust versions.

1 Introduction

Reinforcement learning (RL) is a popular paradigm for sequential decision-making in a dynamic environment (Sutton, 2018) and has seen a lot of applications, including games, autonomous driving, and robotics control. For real-world applications, the RL agent is commonly trained in a simulator since the real-world training is extremely expensive, or even dangerous. However, the sim-to-real gap, referring to the mismatch between the environment of the simulator and the real-world application, is unavoidable, and it can cause a dramatic performance degeneration when we deploy the simulator-trained RL agent to the real world.

To overcome this performance degeneration, robust RL is proposed as a promising framework to bridge the sim-to-real gap (Iyengar, 2005). In robust RL, the transition kernel is assumed to reside in a user-defined uncertainty set, and the target is to find the best policy that maximizes the accumulative reward under the worst kernel in this uncertainty set. Therefore, a robust RL framework includes two aspects, the design of the uncertainty set and the associated algorithm to train the agent under this uncertainty set.

In works on robust RL, many kinds of uncertainty sets are proposed to characterize the kernel uncertainty, including f-divergence uncertainty set (Zhou et al., 2021; Panaganti & Kalathil, 2022; Shi et al., 2024; Gadot et al., 2024), Wasserstein uncertainty set (Abdullah et al., 2019; Clement & Kroer, 2021; Yu et al., 2024), L_p uncertainty set (Behzadian et al., 2021; Badrinath & Kalathil, 2021; Kumar et al., 2022), R-contamination uncertainty set (Wang & Zou, 2021, 2022; Hwang & Hong, 2024), etc. However, these uncertainty set constructions only leverage the information of the nominal transition kernel, typically a set centered around the nominal transition kernel, failing to fully take advantage of the simulator's power. As a result, these uncertainty sets can not accurately characterize the kernel uncertainty, such as the parameter mismatch.

[1]Department of Automation, Shanghai Jiao Tong University, [2]College of Smart Energy, Shanghai Jiao Tong University, [*]Corresponding Author

As for the training algorithms for robust RL, although a great bunch of works have settled the theoretical foundation of some robust RL algorithms under the aforementioned uncertainty set, like the convergence property and sample complexity, these algorithms typically involve computing the robust value under the worst transition kernel, which is intractable for large state space or deep neural approximation. Some work attempted to overcome this hardness (Zhou et al., 2024; Hwang & Hong, 2024; Gadot et al., 2024), but their methods either require adversarial training (Hwang & Hong, 2024) or worst kernel estimation (Gadot et al., 2024), so they are not efficient when the state space is large. Therefore, to efficiently train a robust RL agent, a fully sample-based method is desirable to circumvent the challenges posed by the robust value computation under the worst transition kernel.

To address the aforementioned two research limitations, we propose a novel **D**eep **R**obust **As**ymmetrical **Q**-**L**earning (**DRASQL**) method to bridge the sim-to-real gap. Specifically, we first construct a novel R-τ-contamination uncertainty set, which takes advantage of the power of the simulator, to better characterize the kernel uncertainty, accounting for the parameter mismatches and noise contamination. Then, to efficiently train the robust RL agent under the R-τ-contamination uncertainty set, we leverage an asymmetrical penalty scheme to develop the DRASQL method to circumvent the computational hardness posed by the worst transition kernel. In theory, we prove that the corresponding DRASQL-TD algorithm converges in the tabular RL case. Moreover, we also apply our DRASQL method to popular non-robust RL algorithms for deep neural approximation cases, transforming them into robust versions.

The major contributions of this paper are summarized as follows.

- A novel R-τ-contamination uncertainty set is proposed to accurately characterize the transition kernel uncertainty.
- We develop a fully sample-based DRASQL method to train the robust agent under the proposed R-τ-contamination uncertainty set, and the convergence of the corresponding DRASQL-TD algorithm is proved for the tabular RL case.
- For deep RL scenarios, we adapt our DRASQL method to popular non-robust deep RL algorithms, transforming them into robust versions.

2 Preliminary

2.1 Markov Decision Process

A Markov decision process (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition kernel, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, and γ is the discount factor. A stationary policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ maps a state to a distribution over action space $\mathcal{A}$. At each time step t, in the current state $s_t \in \mathcal{S}$, the agent takes an action a_t according to the policy $\pi(a_t|s_t)$, and then the agent receives a reward $r(s_t, a_t)$ and the state transit to a new state s_{t+1} according to the transition kernel $P_{s_t,a_t}(s_{t+1})$.

The value function of a policy π is defined as the expected discounted accumulative reward

$$\mathbb{E}_{a_t\sim\pi, s_t\sim P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s\right], \tag{1}$$

and the objective of an MDP is to find an optimal policy which maximizes the value.

2.2 Robust Markov Decision Process

The robust MDP considers a case where the transition kernel of an MDP at each time step is unknown. In this case, we denote the transition kernel as $\kappa = (P_0, P_1, \cdots, P_t, \cdots)$, where P_t is the transition kernel at time t and each of them lies in an uncertainty set $\mathcal{P}$, *i.e.*, $P_t \in \mathcal{P}$. The robust MDP evaluate a policy π by the following robust value function

$$V^\pi(s) = \min_{\kappa\in\otimes_{t=0}^{\infty}\mathcal{P}} \mathbb{E}_{a_t\sim\pi, s_t\sim P_t}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s\right], \tag{2}$$

and the robust Q-function is defined as

$$Q^\pi(s,a) = \min_{\kappa \in \otimes_{t=0}^{\infty} \mathcal{P}} \mathbb{E}_{a_t \sim \pi, s_t \sim P_t} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right], \tag{3}$$

and it holds that $V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^\pi(s,a)$ Iyengar (2005). Accordingly, the optimal robust policy is defined as

$$\pi^* = \arg\max_{\pi} V^\pi(s), \forall s \in \mathcal{S} \tag{4}$$

In this work, we focus on the (s,a)-rectangular uncertainty set $\mathcal{P} = \otimes_{s,a} \mathcal{P}_{s,a}$, where $\mathcal{P}_{s,a}$ model the uncertainty in the transition kernel $P_{s,a}$. In this case, the robust Bellman operator has a concise formulation

$$\mathcal{T}Q(s,a) = r(s,a) + \gamma \min_{p_{s,a} \in \mathcal{P}_{s,a}} \sum_{s' \in \mathcal{S}} p_{s,a}(s') V(s'), \tag{5}$$

where

$$V(s') = \max_{a' \in \mathcal{A}} Q(s', a') \tag{6}$$

Therefore, there are two core problems in the robust MDP framework:

- How to construct a good uncertainty set $\mathcal{P}$ to catch the kernel uncertainty?
- How to leverage the robust Bellman operator (5) to train the robust agent efficiently?

We will give our solution to these two problems in the next section.

3 Methodology

In this section, we first introduce our novel modeling of kernel uncertainty, *i.e.*, the R-τ-contamination uncertainty set. Then we present the DRASQL algorithm to train the robust agent under the proposed uncertainty set, for both the tabular and deep RL cases. Further, we present an advantage implementation technique to facilitate the training in the deep RL case. Finally, we compare our method with related robust RL frameworks.

3.1 R-τ-Contamination Uncertainty Set

To accurately model the kernel uncertainty, we propose the following adaptive R-τ-contamination uncertainty set.

Definition 3.1 (R-τ-Contamination Uncertainty Set). *The R-τ-contamination uncertainty set centered at a nominal transition kernel P is defined as $\mathcal{P} = \otimes_{s,a} \mathcal{P}_{s,a}$ with*

$$\mathcal{P}_{s,a} = \left\{ (1-R) P_{s,a} + Rq \middle| q \in \mathcal{Q}_{s,a} \right\}, \tag{7}$$

where

$$\mathcal{Q}_{s,a} = \left\{ q \middle| q << \widetilde{p}_{s,a}, \frac{\text{esssup } dq/d\widetilde{p}_{s,a}}{\text{essinf } dq/d\widetilde{p}_{s,a}} \leq \frac{1-\tau}{\tau} \right\}, \tag{8}$$

where $R \in [0,1]$ is the contamination fraction, $\tau \in (0, \frac{1}{2}]$ is the contamination level, $\widetilde{p}_{s,a}$ is a user-specified contaminated kernel, and $dq/d\widetilde{p}_{s,a}$ is the Radon-Nikodym derivative between q and $\widetilde{p}_{s,a}$.

By Definition 3.1, the transition kernel is contaminated with probability R and when $R = 0$, we recover the non-robust RL, while with higher R we can get more robustness. The parameter τ controls the level of the contaminated part $\mathcal{Q}_{s,a}$ and when $\tau = \frac{1}{2}$, $\mathcal{Q}_{s,a}$ contains only $\widetilde{p}_{s,a}$. When $\tau \to 0$, $\mathcal{Q}_{s,a}$ will contain all the kernels absolutely continuous with respect to contaminated kernel $\widetilde{p}_{s,a}$.

Therefore, R and τ are two scalar parameter relatively easy to pick, while the contaminated kernel $\widetilde{p}_{s,a}$ needs careful construction and at the same time offers great flexibility in modeling the kernel uncertainty. Below we give two possible construction of $\widetilde{p}_{s,a}$.

I **Noise Contamination:** Many systems suffer from random noise contamination, and in this case, we can set for example $\widetilde{p}_{s,a} = P_{s,a} + \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, where $\mathcal{N}(0, \sigma^2 \boldsymbol{I})$ is the Gaussian noise.

II **Parameter Mismatch:** In setting the simulator, there are uncertain some parameter and they can mismatch with the real environment. In this case, we denote these uncertain parameters as $\boldsymbol{\lambda}$ and the contaminated kernel can be set to $\widetilde{p}_{s,a}(s') = \int P_{s,a}(s'|\boldsymbol{\lambda}) g(\boldsymbol{\lambda}) d\boldsymbol{\lambda}$, where $g(\boldsymbol{\lambda})$ is a prior distribution over parameter $\boldsymbol{\lambda}$.

In our DRASQL method, we do not need to compute the probability of this contaminated kernel $\widetilde{p}_{s,a}$, and instead we only need to *sample* from it, and this sampling process is very easy to implement in practice. For example, in the parameter mismatches case, we can firstly sample $\boldsymbol{\lambda}$ from $g(\boldsymbol{\lambda})$, and then the contaminated state s' is sampled from $P_{s,a}(s'|\boldsymbol{\lambda})$, very similar to the domain randomization approach.

Based on the proposed R-τ-contamination uncertainty set, the corresponding robust Bellman operator has a close relation to expectile, which we state as Proposition 3.2.

Proposition 3.2. *The robust Bellman operator under policy π and the R-τ-contamination uncertainty set is*

$$\mathcal{T}Q(s,a) = r(s,a) + \gamma(1-R) \sum_{s' \in \mathcal{S}} P_{s,a}(s')V(s') + \gamma R e_\tau(V|\widetilde{p}_{s,a}), \tag{9}$$

where $e_\tau(V|\widetilde{p}_{s,a})$ is the τ-expectile of random variable V under distribution $\widetilde{p}_{s,a}$.

Proof. By Equation (5), the robust Bellman operator under the R-τ-contamination uncertainty set is

$$\mathcal{T}Q(s,a) = r(s,a) + \gamma(1-R) \sum_{s' \in \mathcal{S}} P_{s,a}(s')V(s') + \gamma R \min_{q \in \mathcal{Q}_{s,a}} \sum_{s' \in \mathcal{S}} q(s')V(s'). \tag{10}$$

Further, by Proposition 8 in Bellini et al. (2014),

$$\min_{q \in \mathcal{Q}_{s,a}} \sum_{s' \in \mathcal{S}} q(s')V(s') = e_\tau(V; \widetilde{p}_{s,a}), \tag{11}$$

so Equation (9) is derived. ☐

3.2 Deep Robust Asymmetrical Q-Learning

When it comes to the training of robust RL, the hardness lies in computing the robust value update, or more specifically, the right-hand-side of robust Bellman Equation (5), where the minimization $\min_{p_{s,a} \in \mathcal{P}_{s,a}}$ searches over all the transition kernels in the uncertainty set. Therefore, it can be extremely difficult to compute the robust target value or even become impossible when we have a large state space. Previous methods either train an adversarial agent simultaneously or estimate the worst kernel in the uncertainty set. However, these methods are not efficient when it comes to large space & action spaces and deep neural approximation.

Fortunately, under the proposed R-τ-contamination uncertainty set, the robust Bellman operator is closely linked to expectile, so we can resort to expectile regression to compute the robust target value (right-hand-side of Equation (9)). Specifically, the expectile can be formulated as the minimizer of an asymmetrical loss, *i.e.*,

$$e_\tau(V; \widetilde{p}_{s,a}) = \arg\min_{e \in \mathbb{R}} \mathbb{E}_{s' \sim \widetilde{p}_{s,a}} [\rho_\tau(V(s') - e)], \tag{12}$$

where

$$\rho_\tau(x) = \begin{cases} \tau x^2 & \text{, if } x \geq 0 \\ (1-\tau)x^2 & \text{, if } x < 0. \end{cases} \tag{13}$$

Or equally, by exploring the optimal condition of Equation (12), we have

$$\mathbb{E}_{s' \sim \widetilde{p}_{s,a}} u(V(s') - e_\tau(V; \widetilde{p}_{s,a})) = 0, \tag{14}$$

where $u(x) = \min\{\tau x, (1-\tau)x\}$ is the derivative of loss ρ_τ.

For simplicity, we denote $e_\tau(V^\pi; \widetilde{p}_{s,a})$ by $\widetilde{Q}(s,a)$, an asymmetrical Q-function, and then the robust Bellman operator becomes

$$\mathcal{T}\begin{bmatrix} Q(s,a) \\ \widetilde{Q}(s,a) \end{bmatrix} = \begin{bmatrix} r(s,a) + \gamma(1-R)\sum_{s'\in\mathcal{S}} p_{s,a}V(s') + \gamma R\widetilde{Q}(s,a) \\ \sum_{s'\in\mathcal{S}} \widetilde{p}_{s,a}(s')u(\widetilde{\gamma}V(s') - \widetilde{Q}(s,a)) + \widetilde{Q}(s,a) \end{bmatrix}. \tag{15}$$

In Equation (15), we introduce a new parameter, the shrinking coefficient $\widetilde{\gamma} \in (0,1)$. This parameter is used to ensure that the operator $\mathcal{T}$ is a contraction as stated in the following theorem.

Theorem 3.3. *If $\widetilde{\gamma} \in (0,1)$, the operator $\mathcal{T}$ defined in Equation (15) is a contraction, and the equilibrium point of $\widetilde{Q}(s,a)$ is $e_\tau(\widetilde{\gamma}\max_a Q^*(s,a); \widetilde{p}_{s,a})$, where $Q^*(s,a)$ is the equilibrium point of $Q(s,a)$ in (15).*

Proof. See Appendix A. □

By Theorem 3.3, the shrinking coefficient will affect the fixed point of $\widetilde{Q}(s,a)$ to be $e_\tau(\widetilde{\gamma}V^*; \widetilde{p}_{s,a})$. That is, the optimal robust value V^* is shrunk by $\widetilde{\gamma}$, and then we take its τ-expectile. Therefore, when $\widetilde{\gamma} \to 1$, it will recover the original robust Bellman operator (9).

By leveraging the new robust Bellman operator (15), we can develop the deep robust asymmetrical Q-learning algorithm (DRASQL) to find its equilibrium points.

3.2.1 Tabular RL Case

We first apply the DRASQL method to tabular RL case and propose the DRASQL-TD algorithm (Algorithm 1). In Algorithm 1, we observe s_{t+1} from the nominal transition kernel P_{s_t,a_t}, and use it to update the robust Q-function $Q(s,a)$ as in the standard TD algorithm. For updating the asymmetrical Q-function $\widetilde{Q}(s,a)$, we sample a contaminated state s'_{t+1} from the contaminated kernel $\widetilde{p}_{s_t,a_t}$, and use the sample $u(\widetilde{\gamma}V_t(s'_{t+1}) - \widetilde{Q}_t(s_t,a_t))$ to approximate the expectation $\sum_{s'\in\mathcal{S}} \widetilde{p}_{s,a}(s')u(\widetilde{\gamma}V(s') - \widetilde{Q}(s,a))$ in (15).

Algorithm 1 DRASQL-TD

```
Input: Robust Q-function $Q(s,a)$, Asymmetrical Q-function $\widetilde{Q}(s,a)$, behavior policy $\pi_b$, step size
  $\alpha_t$, shrinking coefficient $\widetilde{\gamma} \in (0,1)$
  for $t = 0, 1, 2, \cdots, T$ do
      Sample action $a_t$ from $\pi_b(\cdot|s_t)$
      Observe $r(s_t,a_t)$ and $s_{t+1}$, sample $s'_{t+1}$ from $\widetilde{p}_{s_t,a_t}$
      $V_t(s) \leftarrow \max_{a\in\mathcal{A}} Q_t(s,a), \forall s \in \mathcal{S}$
      $Q_{t+1}(s_t,a_t) \leftarrow Q_t(s_t,a_t) + \alpha_t[r(s_t,a_t) + \gamma(1-R)V_t(s_{t+1}) + \gamma R\widetilde{Q}_t(s_t,a_t) - Q_t(s_t,a_t)]$
      $\widetilde{Q}_{t+1}(s_t,a_t) \leftarrow \widetilde{Q}_t(s_t,a_t) + \alpha_t u(\widetilde{\gamma}V_t(s'_{t+1}) - \widetilde{Q}_t(s_t,a_t))$, where $u(x) = \min\{\tau x, (1-\tau)x\}$
  end for
```

To analyze Algorithm 1, we need the following ergodic assumption which is common in the vanilla TD literature.

Assumption 3.4. *The Markov chain induced by the behavior policy π_b and the nominal transition kernel P is uniformly ergodic.*

Then we can state the convergence theorem.

Theorem 3.5 (Convergence Theorem of DRASQL-TD)**.** *If the step size $\alpha_t \in (0,1]$ satisfies $\sum_{t=1}^\infty \alpha_t = \infty$ and $\sum_{t=1}^\infty \alpha_t^2 \le \infty$, then $Q \to Q^*$ and $\widetilde{Q} \to \widetilde{Q}^*$ as $t \to \infty$ with probability 1, where Q^* and $\widetilde{Q}^*$ are equilibrium points of the robust Bellman operator (15).*

Proof. See Appendix B. □

3.2.2 Deep RL Case

For the deep RL case, where the value function and/or the policy are parameterized by deep neural networks, the proposed DRASQL method can be seamlessly integrated into off-the-shelf non-robust deep RL algorithms.

For discrete action problems, we propose the DRASQL-DQN algorithm, which applies our DRASQL method to the vanilla DQN algorithm (Mnih, 2013; Van Hasselt et al., 2016). In the same spirit as the DRASQL-TD algorithm, we regress the robust Q-network by the MSE loss and train the asymmetrical Q-network by the asymmetrical loss ρ_τ.

Algorithm 2 DRASQL-DQN

```
Input: Robust Q-network $Q(s,a;\phi)$, Target robust Q-network $Q^{\text{Tar}}(s,a,\overline{\phi})$, Asymmetrical Q-network $\widetilde{Q}(s,a;\psi)$, Target asymmetrical Q-network $\widetilde{Q}^{\text{Tar}}(s,a;\overline{\psi})$, Learning rate $\lambda$, Exploration factor $\epsilon$, Target update rate $\sigma \in (0,1)$
  for each step do
      With probability $\epsilon$ select a random action, otherwise select $a_t = \arg\max_a Q(s_t,a)$.
      Observe $s_{t+1}$ and $r(s_t,a_t)$, and store $(s_t,a_t,s_{t+1},r(s_t,a_t))$ into the replay buffer.
      if it is update step then
          Sample batch $\mathcal{D} = \{(s,a,s',r)\}$ from the replay buffer.
          Sample $s'' \sim \widetilde{p}_{s,a}$.
          Compute target value:
              $y_Q(s,a,r,s') = r + \gamma(1-R)\max_{a\in\mathcal{A}} Q(s',a;\phi) + \gamma R\widetilde{Q}(s,a;\psi)$
              $y_{\widetilde{Q}}(s'') = \max_{a\in\mathcal{A}} Q(s'',a;\phi)$
          Loss Computation:
              $\mathcal{L}_Q = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s',r)\in\mathcal{D}}(y_Q - Q(s,a;\phi))^2$
              $\mathcal{L}_{\widetilde{Q}} = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s',r)\in\mathcal{D}}\rho_\tau(\widetilde{\gamma} y_{\widetilde{Q}} - \widetilde{Q}(s,a;\psi))$
          Q-network update: $\phi \leftarrow \phi - \lambda\nabla_\phi\mathcal{L}_Q, \psi \leftarrow \psi - \lambda\nabla_\psi\mathcal{L}_{\widetilde{Q}}$.
          Target network update: $\overline{\phi} = \sigma\phi + (1-\sigma)\overline{\phi}$, $\overline{\psi} = \sigma\psi + (1-\sigma)\overline{\psi}$.
      end if
  end for
```

Further, for continuous action problems, we propose the DRASQL-DDPG, DRASQL-SAC, and DRASQL-PPO algorithms, which are adapted from DDPG (Lillicrap, 2015), SAC (Haarnoja et al., 2018), and PPO (Schulman et al., 2017), respectively. These algorithms are presented in Appendix C.1. Moreover, for the special case when $R = 1$, these algorithms can be further simplified as presented in Appendix C.2.

3.3 Advantage Implementation

The interdependence of the robust Q function and the asymmetrical Q function makes it particularly hard to train these two networks in the deep RL case, especially when R is large, and to relieve this training instability, we propose the advantage implementation of the DRASQL method.

To begin with, the first equality in the robust Bellman Equation (15) can be rewritten as

$$Q(s,a) = r + \gamma(1-R)\sum_{s'\in\mathcal{S}} p_{s,a}(s')V(s') + \gamma R\widetilde{Q}(s,a;\psi) \tag{16}$$

$$= r + \gamma\sum_{s'\in\mathcal{S}} p_{s,a}(s')V(s') + \gamma R\Big(\widetilde{Q}(s,a;\psi) - \sum_{s'\in\mathcal{S}} p_{s,a}(s')V(s')\Big). \tag{17}$$

Therefore, instead of learning the $\widetilde{Q}$, we can resort to learning the 'advantage' term $\widetilde{Q} - \mathbb{E}_{p_{s,a}}V(s')$, and when the nominal environment is deterministic, the advantage term is $\widetilde{Q} - V(s')$. So, in the deterministic nominal environment case, we can let

$$\widetilde{A}(s,a) = \widetilde{Q}(s,a) - V(s'), \tag{18}$$

where $\widetilde{A}(s,a)$ can be regarded as the advantage term.

Notice that $\widetilde{A}(s,a)$ satisfies

$$\mathbb{E}_{\widetilde{p}_{s,a}} u(V(s'') - V(s') - \widetilde{A}(s,a)) = 0, \tag{19}$$

so $\widetilde{A}(s,a)$ can also be learned by the asymmetrical penalty ρ_τ.

Based on the above observation, we present the Advantage implementation of the DRASQL-DQN algorithm in Algorithm 3.

Algorithm 3 DRASQL-DQN (Advantage Implementation)

```
Input: Robust Q-network $Q(s,a;\phi)$, Target robust Q-network $Q^{\text{Tar}}(s,a,\overline{\phi})$, Advantage network
  $\widetilde{A}(s,a;\psi)$, Target advantage network $\widetilde{A}^{\text{Tar}}(s,a;\overline{\psi})$, Learning rate $\lambda$, Exploration factor $\epsilon$, Target
  update rate $\sigma \in (0,1)$
  for each step do
      With probability $\epsilon$ select a random action, otherwise select $a_t = \arg\max_a Q(s_t,a)$.
      Observe $s_{t+1}$ and $r(s_t,a_t)$, and store $(s_t,a_t,s_{t+1},r(s_t,a_t))$ into the replay buffer.
      if it is update step then
          Sample batch $\mathcal{D} = \{(s,a,s',r)\}$ from the replay buffer.
          Sample $s'' \sim \widetilde{p}_{s,a}$.
          Compute target value:
             $y_Q = r + \gamma Q(s',a;\overline{\phi}) + \gamma R\widetilde{A}(s,a;\overline{\psi})$
             $y_{\widetilde{A}} = \min\{\widetilde{\gamma}\max_{a\in\mathcal{A}} Q(s'',a;\overline{\phi}) - \max_{a\in\mathcal{A}} Q(s',a;\overline{\phi}), 0\}$
          Loss Computation:
             $\mathcal{L}_Q = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s',r)\in\mathcal{D}}(y_Q - Q(s,a;\phi))^2$
             $\mathcal{L}_{\widetilde{A}} = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s',r)\in\mathcal{D}}\rho_\tau(y_{\widetilde{A}} - \widetilde{A}(s,a;\psi))$
          Network update: $\phi \leftarrow \phi - \lambda\nabla_\phi\mathcal{L}_Q, \psi \leftarrow \psi - \lambda\nabla_\psi\mathcal{L}_{\widetilde{A}}$.
          Target network update: $\overline{\phi} = \sigma\phi + (1-\sigma)\overline{\phi}$, $\overline{\psi} = \sigma\psi + (1-\sigma)\overline{\psi}$.
      end if
  end for
```

For a non-deterministic nominal environment, we can also use the advantage implementation, but in this case, $\widetilde{A}(s,a;\psi)$ will converge to $e_\tau(V(s'') - V(s')|s'' \sim \widetilde{p}_{s,a}, s' \sim P_{s,a})$, which takes the randomness of P into account, so it will be smaller than $e_\tau(V(s'') - \mathbb{E}_{s'\sim P_{s,a}}V(s')|s'' \sim \widetilde{p}_{s,a})$ when τ is set close to 0. Therefore, we shall set a larger τ in this case.

3.4 Comparison with Related Robust RL Frameworks

The proposed robust RL modeling with R-τ-contamination uncertainty set is a quite general framework, and by taking specific R, τ, and $\widetilde{p}_{s,a}$, it can recover many existing robust RL frameworks as follows.

- **RL with Domain Randomization:** By taking $R = 1$, $\tau = 1/2$, and $\tilde{p}_{s,a}$ as in parameter mismatch (case Item II), the proposed framework will be reduced to vanilla RL with domain randomization.
- **Robust RL with R-contamination Uncertainty Set (Wang & Zou, 2021, 2022):** By taking $\tau \to 0$ and $\tilde{p}_{s,a} = U(\mathcal{S})$ to be the uniform distribution on state space $\mathcal{S}$, our framework can recover the R-contamination RL proposed in Wang & Zou (2021, 2022).
- **Robust RL with adjacent R-contamination Uncertainty Set (Hwang & Hong, 2024):** By taking $\tau \to 0$ and $\tilde{p}_{s,a}(s') = \int P_{s,a'}(s')u(a')d(a')$, where $u(a')$ is the uniform distribution over action space $\mathcal{A}$, we can recover the R-contamination RL proposed in Hwang & Hong (2024).

It can be noticed that the R-contamination type uncertainty set is of particular interest in the research community, and it corresponds to our R-τ uncertainty set when $\tau \to 0$. However, fundamental differences exist between the R-contamination type uncertainty set and the proposed R-τ uncertainty set. Firstly, previous research (Wang & Zou, 2022; Hwang & Hong, 2024) does not allow for a user-specified contaminated kernel $\widetilde{p}_{s,a}$, limiting their capability of modeling the kernel uncertainty.

On the contrary, our method features a flexible way of uncertainty modeling and naturally fits with the domain randomization technique. Second, these researches failed to provide an efficient way of training the robust agent. In contrast, our method provides τ for adjusting the robust level and at the same time develops a fully sample-based DRASQL algorithm for training. Therefore, our method also provides an efficient way for optimizing the robust RL under the R-contamination type uncertainty set, which we state as the following theorem.

Theorem 3.6. *The optimal robust Q-function under the R-τ-contamination uncertainty set converges to the optimal robust Q-function under the corresponding R-contamination uncertainty set, i.e., $\mathcal{P}_{s,a} = \{(1-R)P_{s,a} + Rq | q \in \Delta_{\text{supp}(\widetilde{p}_{s,a})}\}$, where $\text{supp}(\widetilde{p}_{s,a})$ is the support set of the contaminated kernel.*

Proof. See Appendix D. ☐

4 Experiment

To demonstrate the effectiveness of the proposed DRASQL method, we conduct experiments on the classic pendulum control problem (Towers et al., 2024) and compare it with the vanilla method and domain randomization method. We choose DDPG (Lillicrap, 2015) and SAC (Haarnoja et al., 2018) as our baseline methods, and we apply our DRASQL to transform them into a robust version.

It is worth noting that all of our DRASQL methods adopt the advantage implementation. For stable training, we initialize the R to 0 and gradually increase it to the target value during training. In this way, at the beginning of the training, the robust Q network updates just as under the nominal kernel since the contamination R is close to 0, so it can be regarded as a warm-up approach.

In the experiments, after each method is trained, we manually perturb the simulation parameter and run the policy 100 times, and we report the mean as well as the minimum and maximum accumulative reward.

4.1 Comparison under Length Perturbation

We compare our DRASQL-DDPG and DRASQL-SAC methods with the vanilla DDPG and SAC, and the domain randomization DDPG methods under the length perturbation. The vanilla DDPG and SAC methods are trained in an unperturbed environment where the length is 1.0. The domain randomization DDPG and SAC methods are trained by resetting the length randomly from a uniform distribution over 0.9-2.1 at the beginning of each epoch.

For our DRASQL-DDPG and DRASQL-SAC methods, we set the nominal kernel just as the domain-randomization approach to obtain a diverse state-action distribution. For the hyperparameters, we set $R = 1$ and $\tau = 0.2$.

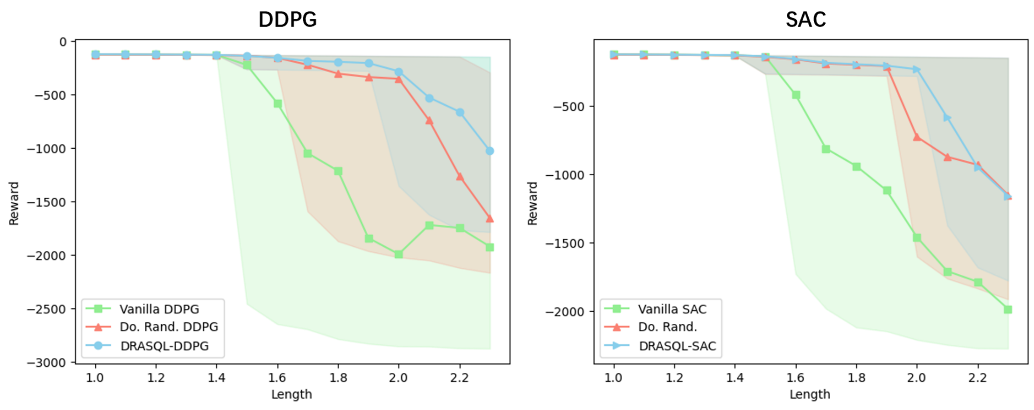

Figure 1: Comparison under length perturbation. (Do. Rand. represents the domain randomization method.)

Figure 1 shows the experiment results. We observe that our DRASQL-DDPG and DRASQL-SAC methods achieve significant improvement compared with the vanilla and domain randomization methods in the mean reward and also have a smaller variance.

4.2 Comparison with SAC

We investigate the performance under the mass perturbation. The vanilla DDPG and SAC methods are trained in an unperturbed environment where the mass is 1.0. The domain randomization DDPG and SAC methods are trained by resetting the length randomly from a uniform distribution over 0.8-4.0 at the beginning of each epoch.

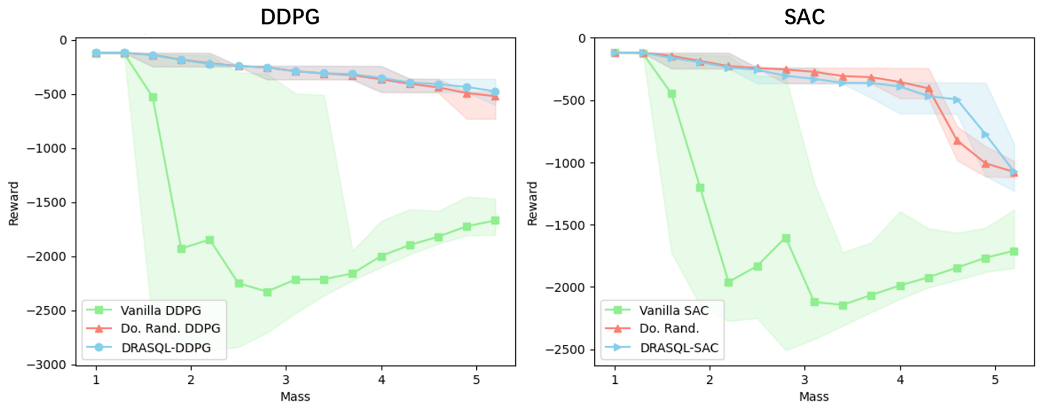

Figure 2: Comparison under Mass perturbation. (Do. Rand. represents the domain randomization method.)

Figure 2 shows the experiment results. From Figure 2, our DRASQL-DDPG method performs as well as the domain randomization approach but achieves a smaller variance. For the SAC-based algorithms, the performance of the domain randomization method degenerates sharply outside the training zone while our DRASQL-SAC method achieves a slower degenerate rate.

5 Conclusion

In this paper, we developed a new robust RL framework for bridging the sim-to-real gap. We first propose a novel R-τ-contamination uncertainty set, which leverages the capabilities of the simulator to model kernel uncertainty with greater precision, accounting for parameter mismatches and noise. Then, to efficiently train the robust RL agent, we propose a fully sample-based DRASQL method that employs an asymmetrical penalty to learn a robust Q-function, without requiring adversarial training or worst kernel estimation. Theoretically, we prove the convergence of the corresponding DRASQL-TD algorithm in the tabular RL case. Furthermore, for deep function approximation, our method can be seamlessly integrated into off-the-shelf non-robust deep RL algorithms, transforming them into robust versions. In the experiment, we apply our DRASQL method to DDPG and SAC algorithms and compare them with the vanilla and domain randomization approaches, and the results demonstrate the effectiveness of our method.

References

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

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

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

Bellini, F., Klar, B., Müller, A., and Gianin, E. R. Generalized quantiles as risk measures. *Insurance: Mathematics and Economics*, 54:41–48, 2014.

Bonnans, J. F. and Shapiro, A. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

Borkar, V. S. and Meyn, S. P. The ode method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469, 2000.

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

Gadot, U., Wang, K., Kumar, N., Levy, K. Y., and Mannor, S. Bring your own (non-robust) algorithm to solve robust mdps by estimating the worst kernel. In *Forty-first International Conference on Machine Learning*, 2024.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hwang, U. and Hong, S. On practical robust reinforcement learning: Adjacent uncertainty set and double-agent algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

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

Kumar, N., Levy, K., Wang, K., and Mannor, S. Efficient policy iteration for robust markov decision processes via regularization. *arXiv preprint arXiv:2205.14327*, 2022.

Lillicrap, T. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Marzban, S., Delage, E., and Li, J. Y.-M. Deep reinforcement learning for option pricing and hedging under dynamic expectile risk measures. *Quantitative Finance*, 23(10):1411–1430, 2023.

Mnih, V. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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.

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

Shen, Y., Tobia, M. J., Sommer, T., and Obermayer, K. Risk-sensitive reinforcement learning. *Neural computation*, 26(7):1298–1328, 2014.

Shi, L., Li, G., Wei, Y., Chen, Y., Geist, M., and Chi, Y. The curious price of distributional robustness in reinforcement learning with a generative model. *Advances in Neural Information Processing Systems*, 36, 2024.

Sutton, R. S. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulao, M., Kallinteris, A., Krimmel, M., KG, A., et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wang, Y. and Zou, S. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34:7193–7206, 2021.

Wang, Y. and Zou, S. Policy gradient method for robust reinforcement learning. In *International conference on machine learning*, pp. 23484–23526. PMLR, 2022.

Yu, Z., Dai, L., Xu, S., Gao, S., and Ho, C. P. Fast bellman updates for wasserstein distributionally robust mdps. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhou, R., Liu, T., Cheng, M., Kalathil, D., Kumar, P., and Tian, C. Natural actor-critic for robust reinforcement learning with function approximation. *Advances in neural information processing systems*, 36, 2024.

Zhou, Z., Zhou, Z., Bai, Q., Qiu, L., Blanchet, J., and Glynn, P. Finite-sample regret bound for distributionally robust offline tabular reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3331–3339. PMLR, 2021.

A Proof of Theorem 3.3

Proof. For $[Q(s,a), \widetilde{Q}(s,a)]$ and $[Q'(s,a), \widetilde{Q}'(s,a)]$, we have

$$\left\| \mathcal{T} \begin{bmatrix} Q_t(s,a) \\ \widetilde{Q}_t(s,a) \end{bmatrix} - \mathcal{T} \begin{bmatrix} Q'_t(s,a) \\ \widetilde{Q}'_t(s,a) \end{bmatrix} \right\|_\infty \tag{20}$$

$$= \left\| \begin{bmatrix} \gamma(1-R) \sum_{s' \in \mathcal{S}} p_{s,a}[V(s') - V'(s')] + \gamma R[\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)] \\ \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s')[u(\widetilde{\gamma} V(s') - \widetilde{Q}_t(s,a)) - u(\widetilde{\gamma} V'(s') - \widetilde{Q}'_t(s,a))] + \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \end{bmatrix} \right\|_\infty \tag{21}$$

For the first line in Equation (21), we have

$$\|\gamma(1-R) \sum_{s' \in \mathcal{S}} p_{s,a}[V(s') - V'(s')] + \gamma R[\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)]\|_\infty \tag{22}$$

$$\leq \gamma \max\{\|Q_t(s,a) - Q'_t(s,a)\|_\infty, \|\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)\|_\infty\} = \gamma \left\| \begin{bmatrix} Q_t(s,a) - Q'_t(s,a) \\ \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \end{bmatrix} \right\|_\infty \tag{23}$$

For the second line in Equation (21), by definition of u, we have $\tau \leq \frac{u(x)-u(y)}{x-y} \leq 1-\tau$. Then, for each $s' \in \mathcal{S}$, we set

$$u(\widetilde{\gamma} V(s') - \widetilde{Q}_t(s,a)) - u(\widetilde{\gamma} V'(s') - \widetilde{Q}'_t(s,a)) = \xi(s,a,s')(\widetilde{\gamma}[V(s') - V'(s')] - [\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)]), \tag{24}$$

where $\tau \leq \xi(s,a,s') \leq 1-\tau$. Therefore, we have

$$\left\| \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s')[u(\widetilde{\gamma} V(s') - \widetilde{Q}_t(s,a)) - u(\widetilde{\gamma} V'(s') - \widetilde{Q}'_t(s,a))] + \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \right\|_\infty \tag{25}$$

$$= \left\| \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s')(\widetilde{\gamma}[V(s') - V'(s')] - [\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)]) + \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \right\|_\infty \tag{26}$$

$$= \left\| \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s') \widetilde{\gamma}[V(s') - V'(s')] + \left(1 - \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s')\right) [\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)] \right\|_\infty \tag{27}$$

$$\leq \left\| \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s') \widetilde{\gamma}[V(s') - V'(s')] \right\|_\infty + \left\| \left(1 - \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s')\right) [\widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a)] \right\|_\infty \tag{28}$$

$$\leq \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s') \widetilde{\gamma} \|Q - Q'\|_\infty + \left(1 - \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s')\right) \left\| \widetilde{Q} - \widetilde{Q}' \right\|_\infty \tag{29}$$

$$\leq \left(1 - (1-\widetilde{\gamma}) \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \xi(s,a,s')\right) \left\| \begin{bmatrix} Q_t(s,a) - Q'_t(s,a) \\ \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \end{bmatrix} \right\|_\infty \tag{30}$$

$$\leq \left(1 - (1-\widetilde{\gamma}) \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') \tau\right) \left\| \begin{bmatrix} Q_t(s,a) - Q'_t(s,a) \\ \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \end{bmatrix} \right\|_\infty \tag{31}$$

$$= \left(1 - (1-\widetilde{\gamma}) \tau\right) \left\| \begin{bmatrix} Q_t(s,a) - Q'_t(s,a) \\ \widetilde{Q}_t(s,a) - \widetilde{Q}'_t(s,a) \end{bmatrix} \right\|_\infty \tag{32}$$

By combining Equation (21), Equation (23), and Equation (32), we can conclude

$$\left\|\mathcal{T}\begin{bmatrix}Q_t(s,a)\\ \widetilde{Q}_t(s,a)\end{bmatrix}-\mathcal{T}\begin{bmatrix}Q'_t(s,a)\\ \widetilde{Q}'_t(s,a)\end{bmatrix}\right\|_\infty \leq \max\{\gamma, 1-(1-\widetilde{\gamma})\tau\}\left\|\begin{bmatrix}Q_t(s,a)-Q'_t(s,a)\\ \widetilde{Q}_t(s,a)-\widetilde{Q}'_t(s,a)\end{bmatrix}\right\|_\infty \tag{33}$$

Therefore, $\mathcal{T}$ is a contraction. □

B Proof of Theorem 3.5

Proof. The TD update in Algorithm 1 can be rewritten as:

$$\begin{bmatrix}Q_{t+1}(s_t,a_t)\\ \widetilde{Q}_{t+1}(s_t,a_t)\end{bmatrix} = (1-\alpha_t)\begin{bmatrix}Q_t(s_t,a_t)\\ \widetilde{Q}_t(s_t,a_t)\end{bmatrix} + \alpha_t\left(\mathcal{T}\begin{bmatrix}Q_t(s_t,a_t)\\ \widetilde{Q}_t(s_t,a_t)\end{bmatrix} + \begin{bmatrix}\eta_t(s_t,a_t,s_{t+1})\\ \widetilde{\eta}_t(s_t,a_t,s'_{t+1})\end{bmatrix}\right) \tag{34}$$

where

$$\mathcal{T}\begin{bmatrix}Q_t(s_t,a_t)\\ \widetilde{Q}_t(s_t,a_t)\end{bmatrix} = \begin{bmatrix}r(s_t,a_t)+\gamma(1-R)\sum_{s'\in\mathcal{S}} p_{s_t,a_t}V_t(s')+\gamma R\widetilde{Q}_t(s_t,a_t)\\ \sum_{s'\in\mathcal{S}}\widetilde{p}_{s_t,a_t}(s')u(\widetilde{\gamma}V_t(s')-\widetilde{Q}_t(s_t,a_t))+\widetilde{Q}_t(s_t,a_t)\end{bmatrix} \tag{35}$$

and

$$\begin{bmatrix}\eta_t(s_t,a_t,s_{t+1})\\ \widetilde{\eta}_t(s_t,a_t,s'_{t+1})\end{bmatrix} = \begin{bmatrix}\gamma(1-R)V(s_{t+1})-\gamma(1-R)\sum_{s'\in\mathcal{S}}p_{s,a}V_t(s')\\ u(\widetilde{\gamma}V(s'_{t+1})-\widetilde{Q}_t(s,a))-\sum_{s'\in\mathcal{S}}\widetilde{p}_{s,a}(s')u(\widetilde{\gamma}V(s')-\widetilde{Q}_t(s,a))\end{bmatrix} \tag{36}$$

For the noise term η_t and $\widetilde{\eta}_t$, we have

$$\mathbb{E}_{p_{s_t,a_t}}\eta_t(s_t,a_t,s_{t+1}) = 0 \tag{37}$$

$$\mathbb{E}_{\widetilde{p}_{s_t,a_t}}\widetilde{\eta}_t(s_t,a_t,s'_{t+1}) = 0 \tag{38}$$

For the variance, we have

$$\mathbb{E}_{p_{s_t,a_t}}[\eta_t(s_t,a_t,s_{t+1})]^2 \leq \gamma^2(1-R)^2\max_{s,a}Q(s,a)^2 \tag{39}$$

$$\begin{aligned}\mathbb{E}_{\widetilde{p}_{s_t,a_t}}[\widetilde{\eta}_t(s_t,a_t,s'_{t+1})]^2 &\leq \max_{s'}u(\widetilde{\gamma}V(s')-\widetilde{Q}_t(s_t,a_t))^2\\ &\leq 4(1-\tau)^2\max\{\max_{s,a}Q(s,a)^2,\max_{s,a}\widetilde{Q}(s,a)^2\}.\end{aligned} \tag{40}$$

Therefore, the noise has bounded variance. Further, by Theorem 3.3, $\mathcal{T}$ is a contraction, so according to Theorem 2.2 in Borkar & Meyn (2000), Algorithm 1 converges to the fixed point of the robust Bellman operator (15). □

C DRASQL for Continuous Action Space

C.1 Adapted algorithms

In this subsection, we present the DRASQL-DDPG algorithm in Algorithm 4, DRASQL-SAC algorithm in Algorithm 5, and DRASQL-PPO algorithm in Algorithm 6.

In the DRASQL-DDPG algorithm, we learn the robust Q-network and the asymmetrical Q-network by MSE and asymmetrical loss, respectively.

In the DRASQL-SAC algorithm, two robust Q-networks and one asymmetrical Q-network are trained. In constructing the target of asymmetrical Q-network, we use the smaller value of two Q-networks to align with the approach in the standard SAC method.

Algorithm 4 DRASQL-DDPG

```
Input: Robust Q-network $Q(s, a; \phi)$, Asymmetrical Q-network $\widetilde{Q}(s, a; \psi)$, Policy network $\mu(s; \theta)$,
    Learning rate $\lambda_Q, \lambda_\theta$, Target update rate $\sigma \in (0, 1)$
  for each step do
      Selection action $a_t = \mu(s_t; \theta)$
      Observe $s_{t+1}$ and $r(s_t, a_t)$, and store $(s_t, a_t, s_{t+1}, r(s_t, a_t))$ into the replay buffer.
      if it is update step then
          Sample batch $\mathcal{D} = \{(s, a, s', r)\}$ from the replay buffer.
          Sample $s'' \sim \widetilde{p}_{s,a}$.
          Compute target value:
            $y_Q(s, a, r, s') = r + \gamma(1 - R)Q(s', \mu(s'; \theta); \phi) + \gamma R\widetilde{Q}(s, a; \psi)$
            $y_{\widetilde{Q}}(s'') = Q(s'', \mu(s''; \theta); \phi)$
          Loss Computation:
            $\mathcal{L}_Q = \frac{1}{|\mathcal{D}|} \sum_{(s,a,s',r)\in\mathcal{D}} (y_Q - Q(s, a; \phi))^2$
            $\mathcal{L}_{\widetilde{Q}} = \frac{1}{|\mathcal{D}|} \sum_{(s,a,s',r)\in\mathcal{D}} \rho_\tau(\widetilde{\gamma} y_{\widetilde{Q}} - \widetilde{Q}(s, a; \psi))$
          Q-network update:
            $\phi \leftarrow \phi - \lambda_Q \nabla_\phi \mathcal{L}_Q, \psi \leftarrow \psi - \lambda_Q \nabla_\psi \mathcal{L}_{\widetilde{Q}}$.
          Policy network update:
            $\theta \leftarrow \theta + \frac{1}{|\mathcal{D}|} \sum_{(s,a,s',r)\in\mathcal{D}} \lambda_\theta \nabla_\theta Q(s, \mu(s_t; \theta))$
          Target network update: $\overline{\phi} = \theta\phi + (1 - \sigma)\overline{\phi}$, $\overline{\psi} = \sigma\psi + (1 - \sigma)\overline{\psi}$, $\overline{\theta} = \sigma\theta + (1 - \sigma)\overline{\theta}$.
      end if
  end for
```

Algorithm 5 DRASQL-SAC

```
Input: Robust Q-network $Q(s, a; \phi_i)$, i=1,2; Asymmetrical Q-network $\widetilde{Q}(s, a; \psi)$; Policy network
    $\pi_\theta$; Learning rate $\lambda_Q, \lambda_\theta$; Target update rate $\sigma \in (0, 1)$
  for each step do
      Selection action $a_t \sim \pi_\theta(\cdot|s_t)$
      Observe $s_{t+1}$ and $r(s_t, a_t)$, and store $(s_t, a_t, s_{t+1}, r(s_t, a_t))$ into the replay buffer.
      if it is update step then
          Sample batch $\mathcal{D} = \{(s, a, s', r)\}$ from the replay buffer.
          Sample $s'' \sim \widetilde{p}_{s,a}$, $a' \sim \pi_\theta(\cdot|s')$, and $a'' \sim \pi_\theta(\cdot|s'')$
          Compute target value:
            $y_Q(s, a, r, s') = r + \gamma(1 - R)\Big(\min_{i=1,2} Q(s', a'; \phi_i) - \alpha \log\pi_\theta(a'|s')\Big) + \gamma R\widetilde{Q}(s, a; \psi)$
            $y_{\widetilde{Q}}(s'') = \min_{i=1,2} Q(s'', a''; \phi_i) - \alpha \log\pi_\theta(a''|s'')$
          Loss Computation:
            $\mathcal{L}_{Q_i} = \frac{1}{|\mathcal{D}|} \sum_{(s,a,s',r)\in\mathcal{D}} (y_Q - Q(s, a; \phi_i))^2$
            $\mathcal{L}_{\widetilde{Q}} = \frac{1}{|\mathcal{D}|} \sum_{(s,a,s',r)\in\mathcal{D}} \rho_\tau(\widetilde{\gamma} y_{\widetilde{Q}} - \widetilde{Q}(s, a; \psi))$
          Q-network update:
            $\phi_i \leftarrow \phi_i - \lambda_Q \nabla_{\phi_i} \mathcal{L}_{Q_i}, \psi \leftarrow \psi - \lambda_Q \nabla_\psi \mathcal{L}_{\widetilde{Q}}$.
          Policy network update:
            $\theta \leftarrow \theta + \frac{1}{|\mathcal{D}|} \sum_{(s,a,s',r)\in\mathcal{D}} \lambda_\theta \nabla_\theta \Big[\min_{i=1,2} Q(s, a_\theta(s); \phi_i) - \alpha \log\pi_\theta(a_\theta(s)|s)\Big]$
          Target network update: $\overline{\phi}_i = \theta\phi_i + (1 - \sigma)\overline{\phi}_i$, $\overline{\psi} = \sigma\psi + (1 - \sigma)\overline{\psi}$.
      end if
  end for
```

In the DRASQL-PPO algorithm, we learn the robust value network V and asymmetrical Q-network $\widetilde{Q}$, the regression target of V is constructed by one-step TD and the policy update is to maximize the standard PPO-Cliped objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|} \sum_{(s,a,r,s') \in \mathcal{D}_k} \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \text{clip}(\epsilon, A^{\pi_{\theta_k}}(s,a)) \right), \tag{41}$$

where the clip function is $\text{clip}(\epsilon, A) = \max((1-\epsilon)A, (1+\epsilon)A)$.

Algorithm 6 DRASQL-PPO

```
Input: Robust value network $V(s;\phi)$; Asymmetrical Q-network $\widetilde{Q}(s,a;\psi)$; Policy network $\pi_\theta$;
  Learning rate $\lambda$
  for $k = 0, 1, 2, \cdots$ do
     Collect set of trajectories $\mathcal{D}_k$ by running policy $\pi_{\theta_k}$ under the nominal kernel $P$.
     Sample $s'' \sim \widetilde{p}_{s,a}$.
     Compute target value:
        $y_V(s,a,r,s') = r + \gamma(1-R)V(s';\phi_k) + \gamma R\widetilde{Q}(s,a;\psi_k)$
        $A^{\pi_{\theta_k}}(s,a) = y_V(s,a,r,s') - V(s;\phi_k)$
        $y_{\widetilde{Q}} = V(s'';\phi)$
     Loss computation:
        $\mathcal{L}_V = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s',r)\in\mathcal{D}}(y_V - V(s;\phi))^2$
        $\mathcal{L}_{\widetilde{Q}} = \frac{1}{|\mathcal{D}|}\sum_{(s,a,s',r)\in\mathcal{D}}\rho_\tau(\widetilde{\gamma} y_{\widetilde{Q}} - \widetilde{Q}(s,a;\psi))$
     Value network update:
        $\phi \leftarrow \phi - \lambda\nabla_\phi\mathcal{L}_V, \psi \leftarrow \psi - \lambda\nabla_\psi\mathcal{L}_{\widetilde{Q}}$.
     Update the policy by maximizing the PPO-Cliped objective (41).
  end for
```

C.2 Simplified Case

For the case $R = 1$, the first equality of Equation (15) becomes

$$Q(s,a) = r(s,a) + \gamma\widetilde{Q}(s,a), \tag{42}$$

so the asymmetrical Q function $\widetilde{Q}$ can be directly represented by the robust Q function, *i.e.*,

$$\widetilde{Q}(s,a) = \frac{Q(s,a) - r(s,a)}{\gamma}. \tag{43}$$

Using Equation (43), we can further rewrite the second equality of Equation (15) as

$$\sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') u(\widetilde{\gamma} V(s') - \widetilde{Q}(s,a)) = 0 \tag{44}$$

$$\iff \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') u\left(\widetilde{\gamma} V(s') - \frac{Q(s,a) - r(s,a)}{\gamma}\right) = 0 \tag{45}$$

$$\iff \sum_{s' \in \mathcal{S}} \widetilde{p}_{s,a}(s') u(r(s,a) + \gamma\widetilde{\gamma} V(s') - Q(s,a)) = 0, \tag{46}$$

where Equation (45) is derived by using Equation (43) and Equation (46) hold for the fact that $\forall \gamma > 0, u(\gamma x) = \gamma u(x)$.

In Equation (46), we can regard $r(s,a) + \gamma\widetilde{\gamma} V(s')$ as the target value and do the asymmetrical regression directly to the robust Q-network $Q(s,a)$ without the need to train the asymmetrical Q-network $\widetilde{Q}(s,a)$. Also, it is worth noting that Equation (46) is closely related to the risk-averse RL (Shen et al., 2014; Marzban et al., 2023).

By the above observations, we can further simplify the algorithms in Appendix C.1.

For the DRASQL-DDPG algorithm (Algorithm 4), the darkblue part can be replaced by

$$\text{Compute target value: } y_Q(r, s'') = r + \gamma\widetilde{\gamma}Q(s'', \mu(s'', \theta); \phi) \tag{47}$$

$$\text{Loss computation: } \mathcal{L}_Q = \frac{1}{|\mathcal{D}|}\sum_{(s,a,r,s')\in\mathcal{D}} \rho_\tau(y_Q - Q(s, a; \phi)) \tag{48}$$

$$\text{Q-network update: } \phi \leftarrow \phi - \lambda_Q \nabla_\phi \mathcal{L}_Q. \tag{49}$$

Accordingly, for the DRASQL-SAC algorithm (Algorithm 5), the the darkblue part can be simplified by

$$\text{Compute target value: } y_Q(r, s'', a'') = r + \gamma\widetilde{\gamma}\Big(\min_{i=1,2} Q(s'', a''; \phi_i) - \alpha\log\pi_\theta(a''|s'')\Big) \tag{50}$$

$$\text{Loss computation: } \mathcal{L}_{Q_i} = \frac{1}{|\mathcal{D}|}\sum_{(s,a,r,s')\in\mathcal{D}} \rho_\tau(y_Q - Q(s, a; \phi_i)) \tag{51}$$

$$\text{Q-network update: } \phi_i \leftarrow \phi_i - \lambda_{Q_i}\nabla_\phi \mathcal{L}_{Q_i}. \tag{52}$$

For the DRASQL-PPO algorithm (Algorithm 6), we have

$$\text{Compute target value: } y_V = r + \gamma\widetilde{\gamma}V(s''; \phi_k), A^{\pi_{\theta_k}}(s, a) = y_V - V(s; \phi_k) \tag{53}$$

$$\text{Loss computation: } \mathcal{L}_V = \frac{1}{|\mathcal{D}|}\sum_{(s,a,r,s')\in\mathcal{D}} \rho_\tau(y_V - V(s; \phi)) \tag{54}$$

$$\text{Q-network update: } \phi \leftarrow \phi - \lambda\nabla_\phi \mathcal{L}_V. \tag{55}$$

D Proof of Theorem 3.6

We first introduce the concept of epi-convergence.

Definition D.1 (Bonnans & Shapiro (2013), p.41)**.** *A sequence of functions $\{\mathcal{R}_n(\boldsymbol{\phi})\}$ epi-converges to a function $\mathcal{R}(\boldsymbol{\phi})$ if and only if $\forall\boldsymbol{\phi} \in \Phi$, condition (i) and (ii) hold.*

(i) For any sequence $\{\boldsymbol{\phi}_n\}$ converges to $\boldsymbol{\phi}$, $\liminf_{n\to\infty} \mathcal{R}_n(\boldsymbol{\phi}_n) \geq \mathcal{R}(\boldsymbol{\phi})$

(ii) There exists a sequence $\{\boldsymbol{\phi}_n\}$ converging to $\boldsymbol{\phi}$ such that $\limsup_{n\to\infty} \mathcal{R}_n(\boldsymbol{\phi}_n) \leq \mathcal{R}(\boldsymbol{\phi})$

Now we can state the proof of Theorem 3.6.

Proof. We denote the Robust Bellman operator of the proposed R-τ-contamination uncertainty and the corresponding R-contamination uncertainty as $\mathcal{T}_\tau$ and $\mathcal{T}_0$, *i.e.*,

$$\mathcal{T}_\tau Q(s, a) = r(s, a) + \gamma(1 - R)\sum_{s'\in\mathcal{S}} P_{s,a}(s')V(s') + \gamma R e_\tau(V|\widetilde{p}_{s,a}) \tag{56}$$

$$\mathcal{T}_0 Q(s, a) = r(s, a) + \gamma(1 - R)\sum_{s'\in\mathcal{S}} P_{s,a}(s')V(s') + \gamma R \,\text{essinf}_{\widetilde{p}_{s,a}} V, \tag{57}$$

where $\text{essinf}_{\widetilde{p}_{s,a}} V$ is the essential infimum of V under measure $\widetilde{p}_{s,a}$.

The corresponding optimal robust Q-functions are the minimizer of $\|Q_n - \mathcal{T}_{\tau_n} Q_n\|_\infty$ and $\|Q - \mathcal{T}_0 Q\|_\infty$, respectively.

We first consider the convergence of $\mathcal{T}_\tau$ with fixed Q, *i.e.*, $\lim_{\tau\to 0} \mathcal{T}_\tau Q = \mathcal{T}_0 Q$, which entails the convergence of the expectile operator $e_\tau(V; \widetilde{p}_{s,a})$, *i.e.*,

$$\lim_{\tau\to 0} e_\tau(V; \widetilde{p}_{s,a}) = \text{essinf}_{\widetilde{p}_{s,a}} V \tag{58}$$

Let $\widetilde{Q}_\tau(s, a) = e_\tau(V; \widetilde{p}_{s,a})$, and then it satisfies

$$\tau\mathbb{E}_{s'\sim\widetilde{p}_{s,a}}[V(s') - \widetilde{Q}_\tau(s, a)]^+ = (1 - \tau)\mathbb{E}_{s'\sim\widetilde{p}_{s,a}}[V(s') - \widetilde{Q}_\tau(s, a)]^-, \tag{59}$$

where $x^+ = \max\{x, 0\}$ and $x^- = \max\{-x, 0\}$.

From Equation (59) we can note that $\mathbb{E}_{s' \sim \widetilde{p}_{s,a}}[V(s') - \widetilde{Q}_\tau(s,a)]^-$ converges to 0 as $\tau \to 0$ and that $\widetilde{Q}_\tau(s,a)$ is monotonically decreasing, we conclude that $\lim_{\tau \to 0} Q_\tau \leq \text{essinf}_{\widetilde{p}_{s,a}} V$. Further noticing that $Q_\tau \geq \text{essinf}_{\widetilde{p}_{s,a}} V$, otherwise $\mathbb{E}_{s' \sim \widetilde{p}_{s,a}}[V(s') - \widetilde{Q}_\tau(s,a)]^+ > 0$ and $\mathbb{E}_{s' \sim \widetilde{p}_{s,a}}[V(s') - \widetilde{Q}_\tau(s,a)]^- = 0$, which contradicts Equation (59), therefore $\lim_{\tau \to 0} e_\tau(V; \widetilde{p}_{s,a}) = \text{essinf}_{\widetilde{p}_{s,a}} V$.

From the above argument, we conclude that $\lim_{\tau \to 0} \mathcal{T}_\tau Q = \mathcal{T}_0 Q$, therefore $\lim_{\tau \to 0} \|\mathcal{T}_\tau Q - \mathcal{T}_0 Q\|_\infty = 0$ for any Q.

We next prove that for any sequence $\tau_n \to 0$, the sequence of functions $\{\|Q - \mathcal{T}_{\tau_n} Q\|_\infty, n = 1, \cdots, \infty\}$ epi-converges to $\|Q - \mathcal{T}_0 Q\|_\infty$. Actually, for any sequence $Q_n \to Q$, we have

$$\begin{aligned} \big|\|Q_n - \mathcal{T}_{\tau_n} Q_n\|_\infty - \|Q - \mathcal{T}_0 Q\|_\infty\big| &\leq \|Q_n - \mathcal{T}_{\tau_n} Q_n - (Q - \mathcal{T}_0 Q)\|_\infty && (60)\\ &= \|(Q_n - Q) - (\mathcal{T}_{\tau_n} Q_n - \mathcal{T}_{\tau_n} Q) - (\mathcal{T}_{\tau_n} Q - \mathcal{T}_0 Q)\|_\infty && (61)\\ &\leq \|Q_n - Q\|_\infty + \|\mathcal{T}_{\tau_n} Q_n - \mathcal{T}_{\tau_n} Q\|_\infty + \|\mathcal{T}_{\tau_n} Q - \mathcal{T}_0 Q\|_\infty && (62)\\ &\leq 2\|Q_n - Q\|_\infty + \|\mathcal{T}_{\tau_n} Q - \mathcal{T}_0 Q\|_\infty, && (63) \end{aligned}$$

where from (62) to (63) we use the contraction of operator $\mathcal{T}$, *i.e.*, $\|\mathcal{T}_{\tau_n} Q_n - \mathcal{T}_{\tau_n} Q\|_\infty \leq \|Q_n - Q\|_\infty$ by Iyengar (2005).

By the former discussion, we know that $\lim_{n \to \infty} \|\mathcal{T}_{\tau_n} Q - \mathcal{T}_0 Q\|_\infty = 0$ and $\lim_{n \to \infty} \|Q_n - Q\|_\infty = 0$. Therefore, $\lim_{n \to \infty} \big|\|Q_n - \mathcal{T}_{n \to \infty} Q_n\|_\infty - \|Q - \mathcal{T}_0 Q\|_\infty\big| = 0$, or equivalently,

$$\lim_{n \to \infty} \|Q_n - \mathcal{T}_{\tau_n} Q_n\|_\infty = \|Q - \mathcal{T}_0 Q\|_\infty. \tag{64}$$

So, $\|Q - \mathcal{T}_{\tau_n} Q\|_\infty$ epi-converges to $\|Q - \mathcal{T}_0 Q\|_\infty$. As a result, according to Proposition 4.6 in Bonnans & Shapiro (2013), the sequence $\{Q_\tau \in \arg\min_Q \|Q - \mathcal{T}_\tau Q\|_\infty, \tau \to 0\}$ converges to $Q_0 \in \arg\min_Q \|Q - \mathcal{T}_0 Q\|_\infty$. $\square$