Boosting Reinforcement Learning with Strongly Delayed Feedback Through Auxiliary Short Delays

Qingyuan Wu [1] **Simon Sinong Zhan** [2] **Yixuan Wang** [2] **Yuhui Wang** [3]
Chung-Wei Lin [4] **Chen Lv** [5] **Qi Zhu** [2] **Jürgen Schmidhuber** [3 6] **Chao Huang** [1 7]

Abstract

Reinforcement learning (RL) is challenging in the common case of delays between events and their sensory perceptions. State-of-the-art (SOTA) state augmentation techniques either suffer from state space explosion or performance degeneration in stochastic environments. To address these challenges, we present a novel *Auxiliary-Delayed Reinforcement Learning (AD-RL)* method that leverages auxiliary tasks involving short delays to accelerate RL with long delays, without compromising performance in stochastic environments. Specifically, AD-RL learns a value function for short delays and uses bootstrapping and policy improvement techniques to adjust it for long delays. We theoretically show that this can greatly reduce the sample complexity. On deterministic and stochastic benchmarks, our method significantly outperforms the SOTAs in both sample efficiency and policy performance. Code is available at https://github.com/QingyuanWuNothing/AD-RL.

1. Introduction

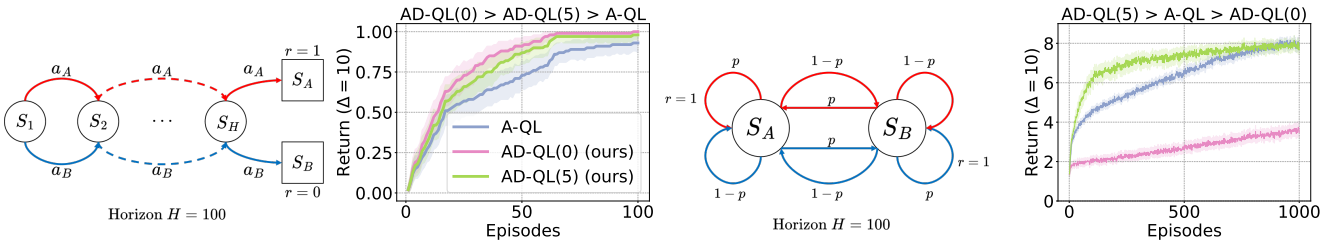
Reinforcement learning (RL) has already proved its mettle in complex tasks such as Backgammon (Tesauro, 1994), Go (Silver et al., 2018), MOBA Game (Berner et al., 2019), building control (Xu et al., 2021; 2022), and various cyber-physical systems (Wang et al., 2023a;b; Zhan et al., 2024). Most of the above RL settings assume that the agent's interaction with the environment is instantaneous, which means that the agent can always execute commands without delay and gather feedback from the environment right away. However, the persistent presence of delays in real-world applications significantly hampers agents' efficiency, performance, and safety if not handled properly (e.g., introducing estimation error (Hwangbo et al., 2017) and losing reproducibility (Mahmood et al., 2018) in practical robotic tasks). Delay also needs to be considered in many stochastic settings such as financial markets (Hasbrouck & Saar, 2013) and weather forecasting (Fathi et al., 2022). Thus, addressing delays in RL algorithms is crucial for their deployment in real-world timing-sensitive tasks.

Delays in RL can be primarily divided into three categories: observation delay, action delay, and reward delay (Firoiu et al., 2018), depending on where the delay occurs. Among them, observation delay receives considerable attention due to the application-wise generality and the technique-wise challenge: it has been proved to be a superset of action delay (Katsikopoulos & Engelbrecht, 2003; Nath et al., 2021), and unlike well-studied reward delay (Han et al., 2022; Kim & Lee, 2020), it disrupts the Markovian property of systems (i.e., the underlying dynamics depend on an unobserved state and the sequence of actions). In this work, we focus on *non-anonymous and constant observation delay* under finite Markov Decision Process (MDP) settings, where the delay is known to the agent and always a constant number of time steps (details in Section 3), as in most existing works (Schuitema et al., 2010; Chen et al., 2021).

Promising augmentation-based approaches (Altman & Nain, 1992; Katsikopoulos & Engelbrecht, 2003) transform the delayed RL problem into an MDP by augmenting the latest observed state with a sequence of actions related to the delay, also known as the information state (Bertsekas, 2012). After retrieving the Markovian property, the augmentation-based methods adopt classical RL methods to solve the delayed RL problem properly, such as augmented Q-learning (A-QL) (Nath et al., 2021). However, existing augmentation-based methods are plagued by the curse of dimensionality, shown by our toy examples in Fig. 1. Under a deterministic MDP setting (Fig. 1(a)), the original augmented state space grows exponentially with the delays, causing

[1]The University of Liverpool [2]Northwestern University [3]AI Initiative, King Abdullah University of Science and Technology [4]National Taiwan University [5]Nanyang Technological University [6]The Swiss AI Lab IDSIA/USI/SUPSI [7]The University of Southampton. Correspondence to: Chao Huang <chao.huang@soton.ac.uk>.

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



(a) Deterministic MDP. (b) Shorter Δ^τ is better. (c) Stochastic MDP ($p = 0.1$). (d) Shorter Δ^τ is not better.

Figure 1. Our AD-RL method introduces an adjoint task with short delays, enhancing the original augmentation-based method (A-QL) in deterministic MDP (Fig. 1(a)) with delay $\Delta = 10$, shown in Fig. 1(b). Whereas, in stochastic MDP (Fig. 1(c)) with delay $\Delta = 10$, a short auxiliary delays may lead to performance improvement (AD-QL(5)) or drop (AD-QL(0)) as shown in Fig. 1(d). BPQL always uses a fixed 0 auxiliary delays, equivalent to AD-QL(0) in these examples. Notably, the optimal auxiliary delays is irregular and task-specific, which we discussed in subsequent experiments in Section 6.

learning inefficiency. The variant of augmentation-based methods BPQL (Kim et al., 2023) approximates the value function based on the delay-free MDP to tackle the inefficiency, which unexpectedly results in excessive information loss. Consequently, it cannot properly handle stochastic tasks (Fig. 1(c)).

To address the aforementioned challenges, we propose a novel technique named *Auxiliary-Delayed RL (AD-RL)*. Our AD-RL is inspired by a key observation that an elaborate auxiliary task with short delays carries much more accurate information than the delay-free case about the original task with long delays, and is still easy to learn. By introducing the notion of delayed belief to bridge an auxiliary task with short delays and the original task with long delays, we can learn the auxiliary-delayed value function and map it to the original one. The changeable auxiliary delays in our AD-RL has the ability to flexibly address the trade-off between the learning efficiency and approximation accuracy error in various MDPs. In toy examples (Fig. 1(a) and Fig. 1(c)) with 10 delays, we compare the performance of A-QL and AD-RL with 0 and 5 auxiliary delays respectively (AD-QL(0) and AD-QL(5)). Our AD-RL not only remarkably enhances the learning efficiency (Fig. 1(b)) but also possesses the flexibility to capture more information under the stochastic setting (Fig. 1(d)). Notably, BPQL is a special variant of our AD-RL with fixed 0 auxiliary delays, resulting in poor performance under the stochastic setting. In Section 4, we develop AD-DQN and AD-SAC, extending from Deep Q-Network and Soft Actor-Critic with our AD-RL framework respectively. Besides, we provide an in-depth theoretical analysis of learning efficiency, performance gap, and convergence in Section 5. In Section 6, we show superior efficacy of our method over the SOTA approaches on the different benchmarks. Our contributions can be summarized as:

- We address the sample inefficiency of the original augmentation-based approaches (denoted as A-RL) and excessive approximation error of the belief-based approaches by introducing AD-RL, which is more efficient with a short auxiliary-delayed task and achieves a theoretical similar performance with A-RL.
- Adapting the AD-RL framework, we devise AD-DQN and AD-SAC to handle discrete and continuous control tasks, respectively.
- We analyze the superior sampling efficiency of AD-RL, the performance gap bound between AD-RL and A-RL, and provide the convergence guarantee of AD-RL.
- We show notable improvements of AD-RL over existing SOTA methods in policy performance and sampling efficiency for deterministic and stochastic benchmarks.

2. Preliminaries

2.1. Delay-free RL

The delay-free RL problem is usually modelled as a Markov Decision Process (MDP), defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho \rangle$. An MDP consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a probabilistic transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a discount factor $\gamma \in (0, 1)$ and an initial state distribution ρ. At each time step t, based on the input state $s_t \in \mathcal{S}$ and the policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the agent has an action $a_t \sim \pi(\cdot|s_t)$ where $a_t \in \mathcal{A}$, and then the MDP evolves to a new state $s_{t+1} \in \mathcal{S}$ based on the probabilistic transition function $\mathcal{P}$ and the agent receive a reward signal r_t from reward function $\mathcal{R}(s_t, a_t)$. We use $d^\pi_{s_0}$ to denote the visited state distribution starting from s_0 based on policy π. The objective of the agent in an MDP is to find a policy that maximizes return over the horizon H. Given a state s, the value function of policy π is defined as

$$V^\pi(s) = \mathop{\mathbb{E}}_{\substack{s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t) \\ a_t \sim \pi(\cdot|s_t)}} \left[\sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t) \middle| s_0 = s \right].$$

Similarly, given a state-action pair (s, a), the Q-function of policy π can be defined as

$$Q^\pi(s, a) = \mathop{\mathbb{E}}_{\substack{s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t) \\ a_t \sim \pi(\cdot|s_t)}} \left[\sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t) \middle| \begin{matrix} s_0 = s \\ a_0 = a \end{matrix} \right].$$

2.2. Deep Q-Network and Soft Actor-Critic

A widely used off-policy RL method is the Deep Q-Network (DQN) (Mnih et al., 2015) with the Q-function $Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by θ. It conducts temporal-difference (TD) learning based on the Bellman optimality equation. Given the transition data (s_t, a_t, r_t, s_{t+1}), DQN updates the Q-function via minimizing TD error.

$$\nabla_\theta \left[\frac{1}{2} (Q_\theta(s_t, a_t) - \mathbb{Y})^2 \right]$$

where $\mathbb{Y} = r_t + \gamma \max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1})$ is the TD target.

Based on the maximum entropy principle, Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) provides a more stable actor-critic method by introducing a soft value function. Given transition data (s_t, a_t, r_t, s_{t+1}), SAC conducts TD update for the critic using the soft TD target $\mathbb{Y}^{soft}$.

$$\mathbb{Y}^{soft} = r_t + \gamma \mathop{\mathbb{E}}_{a_{t+1} \sim \pi_\psi(\cdot|s_{t+1})} [Q_\theta(s_{t+1}, a_{t+1}) - \log \pi_\psi(a_{t+1}|s_{t+1})]$$

where π_ψ is the policy function parameterized by ψ. For the policy π_ψ, it can be optimized by the gradient update:

$$\nabla_\psi \mathop{\mathbb{E}}_{\hat{a} \sim \pi_\psi(\cdot|s_t)} [\log \pi_\psi(\hat{a}|s_t) - Q_\theta(s_t, \hat{a})]$$

3. Problem Setting

We assume that delay-free MDP is endowed with a constant delay variable $\Delta \in \mathbb{N}$. In this setting, the state of environment s_t is only observed by the agent at a later timestep $t + \Delta$. In other words, the real state of the environment is s_t, but the agent's observation is $s_{t-\Delta}$. To retrieve the Markov property in this Delayed MDP (DMDP) (Altman & Nain, 1992; Katsikopoulos & Engelbrecht, 2003), we need to augment the state space $\mathcal{X} = \mathcal{S} \times \mathcal{A}^\Delta$, where $\mathcal{A}^\Delta$ stands for actions in delay time steps. An augmented state $x_t = (s_{t-\Delta}, a_{t-\Delta}, \ldots, a_{t-1}) \in \mathcal{X}$ is composed with the latest observed state $s_{t-\Delta}$ and actions taken in last Δ time steps $(a_{t-\Delta}, \ldots, a_{t-1})$. Thus, with consideration of the delay into the dynamics, we can formulate a new MDP dynamic called *Constant Delayed MDP* (CD-MDP), $\langle \mathcal{X}, \mathcal{A}, \mathcal{P}_\Delta, \mathcal{R}_\Delta, \gamma, \rho_\Delta \rangle$, where $\mathcal{X}$ is defined above, $\mathcal{A}$ stands for the action-space. $\mathcal{P}_\Delta$ is the delayed probabilistic transition function defined below.

$$\mathcal{P}_\Delta(x_{t+1}|x_t, a_t) = \mathcal{P}(s_{t-\Delta+1}|s_{t-\Delta}, a_{t-\Delta}) \delta_{a_t}(a'_t) \Pi_{i=1}^{\Delta-1} \delta_{a_{t-i}}(a'_{t-i})$$

where δ is the *Dirac distribution*. We also have a new delayed reward function $\mathcal{R}_\Delta$ defined as follows.

$$\mathcal{R}_\Delta(x_t, a_t) = \mathop{\mathbb{E}}_{s_t \sim b(\cdot|x_t)} [\mathcal{R}(s_t, a_t)]$$

Correspondingly, the initial state distribution is represented as $\rho_\Delta = \rho \Pi_{i=1}^{\Delta} \delta_{a_{-i}}$, where $b(s_t|x_t)$ is called belief defined as follows.

$$b(s_t|x_t) = \int_{\mathcal{S}^\Delta} \Pi_{i=0}^{\Delta-1} \mathcal{P}(s_{t-\Delta+i+1}|s_{t-\Delta+i}, a_{t-\Delta+i}) \mathrm{d}s_{t-\Delta+i+1} \tag{1}$$

The idea is to infuse delayed state information $s_{t-\Delta}$ to s_t into the augmented state x_t (Gangwani et al., 2020).

In this work, we assume the MDPs, policies and Q-functions satisfy the following Lipschitz Continuity (LC) property, where *Euclidean distance* is adopted in a deterministic space (e.g., $d_\mathcal{S}$ for state space $\mathcal{S}$, $d_\mathcal{A}$ for action space $\mathcal{A}$ and $d_\mathcal{R}$ for reward space $\mathcal{R}$), and $L1$-*Wasserstein distance* (Villani et al., 2009), denoted as W_1, is used in a probabilistic space (e.g., transition space $\mathcal{P}$ and policy space Π) respectively.

Definition 3.1 (**Lipschitz Continuous MDP** (Rachelson & Lagoudakis, 2010))**.** An MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho \rangle$ is $(L_\mathcal{P}, L_\mathcal{R})$-LC, if $\forall (s_1, a_1), (s_2, a_2) \in \mathcal{S} \times \mathcal{A}$, we have

$$W_1(\mathcal{P}(\cdot|s_1, a_1) || \mathcal{P}(\cdot|s_2, a_2)) \leq L_\mathcal{P}(d_\mathcal{S}(s_1, s_2) + d_\mathcal{A}(a_1, a_2))$$
$$d_\mathcal{R}(\mathcal{R}(s_1, a_1) - \mathcal{R}(s_2, a_2)) \leq L_\mathcal{R}(d_\mathcal{S}(s_1, s_2) + d_\mathcal{A}(a_1, a_2))$$

Definition 3.2 (**Lipschitz Continuous Policy** (Rachelson & Lagoudakis, 2010))**.** A stationary markovian policy π is L_π-LC, if $\forall s_1, s_2 \in \mathcal{S}$, we have

$$W_1(\pi(\cdot|s_1) || \pi(\cdot|s_2)) \leq L_\pi d_\mathcal{S}(s_1, s_2)$$

Definition 3.3 (**Lipschitz Continuous Q-function** (Rachelson & Lagoudakis, 2010))**.** Given $(L_\mathcal{P}, L_\mathcal{R})$-LC MDP and L_π-LC policy π, such that $\gamma L_P(1 + L_\pi) < 1$ where γ is the discount factor of MDP, then Q-function Q^π is L_Q-LC with $L_Q = \frac{L_R}{1 - \gamma L_P(1 + L_\pi)}$.

Note that is a mild assumption commonly used in RL literature (Rachelson & Lagoudakis, 2010; Liotet et al., 2022).

4. Our Approach: Auxiliary-Delayed RL

In this section, we introduce our AD-RL framework to address the sample inefficiency of the original augmentation-based approach and illustrate the underlying relation between learning the original delayed task and the auxiliary one in Section 4.1. Then we extend the AD-RL framework to the practical algorithms in Section 4.2 and Section 4.3.

4.1. Auxiliary-Delayed Reinforcement Learning

Instead of learning on the original augmented state space $\mathcal{X}$ with delays Δ, we introduce a corresponding auxiliary-delayed task with the shorter delays $\Delta^\tau (\Delta^\tau < \Delta)$ and, accordingly, a much smaller augmented state space $\mathcal{X}^\tau$. Sharing a similar idea as belief function b in Eq. (1), $\mathcal{X}$ and $\mathcal{X}^\tau$

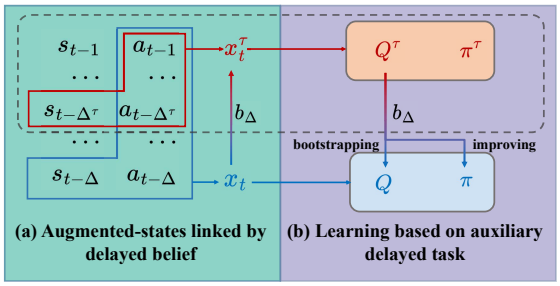


Figure 2. The overview of AD-RL framework. Compared with the conventional augmentation-based method, AD-RL additionally introduces the auxiliary-delayed task shown in the dashline box.

can be bridged by a delayed belief function b_Δ as:

$$b_\Delta(x_t^\tau|x_t) = \int_{\mathcal{S}^\Delta} \Pi_{i=0}^{\Delta-\Delta^\tau-1}\mathcal{P}(s_{t-\Delta+i+1}|s_{t-\Delta+i}, a_{t-\Delta+i})\mathrm{d}s_{t-\Delta+i+1} \tag{2}$$

where $x_t = (s_{t-\Delta}, a_{t-\Delta}, \ldots, a_{t-1}) \in \mathcal{X}$ and $x_t^\tau = (s_{t-\Delta^\tau}, a_{t-\Delta^\tau}, \ldots, a_{t-1}) \in \mathcal{X}^\tau$. As shown in the Fig. 2 (a), both x_t and x_t^τ share the sub-sequence $(a_{t-\Delta^\tau}, \ldots, a_{t-1})$ of $\mathcal{A}^\Delta$. Besides, in the original MDP setting, transitioning from the state $s_{t-\Delta}$ to the state $s_{t-\Delta^\tau}$ can be accomplished by applying the action sub-sequence $(a_{t-\Delta}, \ldots, a_{t-\Delta^\tau+1})$.

Remark 4.1 (Implicit Delayed Belief). Practically, we do not need to learn the delayed belief b_Δ explicitly. As in the CDMDP, every state will be observed by the agent eventually. In other words, given an entire trajectory collected by the agent, we can create the synthetic augmented state for any required delay.

With b_Δ we can transform learning the original Δ-delayed task into learning the auxiliary Δ^τ-delayed task which is much easier to learn for a much smaller augmented state space. As shown in Fig. 2 (b), we can use the easier-to-learn auxiliary Q-function Q^τ to help bootstrapping the Q-function Q or improving the policy π. The specific algorithms will be proposed in the next sections. In this way, we can significantly improve the learning efficiency of the Δ-delayed task, and a more rigorous proof will be presented in Section 5.

As a highly flexible delayed RL framework (Algorithm 1), our AD-RL can be naturally embedded in most of the existing RL methods to serve different task specifications. In this paper, we specifically develop two practical algorithms AD-DQN and AD-SAC based on DQN (Mnih et al., 2015) and SAC (Haarnoja et al., 2018a) to tackle discrete and continuous control tasks respectively.

Remark 4.2. BPQL (Kim et al., 2023) can be seen as a special case of our AD-RL via setting the auxiliary delays to fixed zero (i.e., $\Delta^\tau = 0$). However, the excessive loss of information might lead to poor performance in stochastic MDP in Fig. 1(d). We provide more experimental results about this in Section 6.

Algorithm 1 Auxiliary-Delayed RL Framework

Input: Q, π for Δ delays; Q^τ, π^τ for Δ^τ delays
for each update step **do**
  # Learning Δ^τ-delayed task
  Updating Q^τ, π^τ by a given delayed RL method
  # Learning Δ-delayed task based on Q^τ
  Bootstrapping Q based on Q^τ via Eq. (3) # discrete
  Improving π based on Q^τ via Eq. (5)# continuous
end for
Output: Q, π

4.2. Discrete Control: from AD-VI to AD-DQN

Before developing the practical algorithm: AD-DQN, we first have to derive AD-VI, the AD-RL version of value iteration (VI) (Sutton & Barto, 2018). In the tabular setting, AD-VI maintains two Q-functions Q and Q^τ for the original delayed task(Δ) and the auxiliary-delayed task(Δ^τ), respectively. Different from Q^τ updated by the original Bellman operator, we update Q by applying the auxiliary-delayed Bellman operator $\mathcal{T}$ as follow:

$$\mathcal{T}Q(x_t, a_t) \triangleq \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau \sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)}} \left[Q^\tau(x_{t+1}^\tau, \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1}))\right] \tag{3}$$

Then AD-DQN can be extended from AD-VI naturally via approximating the Q-functions by the parameterized functions (e.g., neural networks). The implementation details of AD-DQN are presented in Appendix A.

4.3. Continuous Control: from AD-SPI to AD-SAC

Similarly, before AD-SAC, we begin with deriving AD-SPI, soft policy iteration (SPI) (Haarnoja et al., 2018a) in the context of our AD-RL. AD-SPI also alternates between two steps: policy evaluation and policy improvement. In policy evaluation, we evaluate the policy π via iteratively applying the auxiliary-delayed soft Bellman operator $\mathcal{T}^\pi$ as follow:

$$\mathcal{T}^\pi Q(x_t, a_t) \triangleq \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{a_{t+1}\sim\pi(\cdot|x_{t+1}) \\ x_{t+1}^\tau \sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)}} \left[Q^\tau(x_{t+1}^\tau, a_{t+1}) - \log\pi(a_{t+1}|x_{t+1})\right] \tag{4}$$

where Q^τ is updated by the original soft Bellman operator (Haarnoja et al., 2018a). In policy improvement, we will

update the policy π based on Q^τ instead of Q as follow:

$$\pi_{new} = \arg\min_{\pi' \in \Pi} \mathrm{KL}\left(\pi'(\cdot|x_t) \middle|\middle| \frac{\exp(\mathbb{E}_{x_t^\tau \sim b_\Delta(\cdot|x_t)}[Q^\tau(x_t^\tau, \cdot)])}{Z(x_t^\tau, \cdot)}\right) \tag{5}$$

where Π is the set of policies for tractable learning, KL is the Kullback-Leibler divergence and Z is the term for normalizing the distribution. Functions approximations of the Q-functions (Q and Q^τ) and policies (π and π^τ) bring us AD-SAC, the practical algorithm for continuous control task. In AD-SAC, the policy π_ψ parameterized by ψ is updated by gradient with

$$\nabla_\psi \mathop{\mathbb{E}}_{\substack{\hat{a}\sim\pi_\psi(\cdot|x_t)\\ x_t^\tau \sim b_\Delta(\cdot|x_t)}} [\log \pi_\psi(\hat{a}|x_t) - Q^\tau(x_t^\tau, \hat{a})] \tag{6}$$

In addition, we further improve AD-SAC with multi-step value estimation (Sutton & Barto, 2018; Bouteiller et al., 2020) for accelerating learning. The implementation details of AD-SAC are presented in Appendix A.

5. Theoretical Analysis

In this section, we first discuss why our AD-RL has better **sample efficiency** in Section 5.1, then analyse the **performance gap** between optimal auxiliary-delayed value function and optimal delayed value function in Section 5.2, and finally derive the **convergence guarantee** of our AD-RL in Section 5.3.

5.1. Sample Efficiency Analysis

Though it is hard to directly derive a formal conclusion on the sample efficiency of AD-RL, as the learning process is correlated to two different learning tasks at the same time, some existing works (Even-Dar et al., 2003; Azar et al., 2011) related to sample complexity provide insight into why our method has better sample efficiency. The sample complexity of the optimized Q-learning is $\mathcal{O}\left(\frac{\log(|\mathcal{S}||\mathcal{A}|)}{\epsilon^{2.5}(1-\gamma)^5}\right)$ (Azar et al., 2011), which shows the amount of samples are required for Q-learning to guarantee an ϵ-optimal Q-function with high confidence. We can conclude that the sample complexity of augmented Q-learning in the augmented state space with delay Δ is $\mathcal{O}\left(\frac{\log(|\mathcal{S}||\mathcal{A}|^{\Delta+1})}{\epsilon^{2.5}(1-\gamma)^5}\right)$. Then our AD-RL makes bootstrapping in the auxiliary Δ^τ-augmented state-space instead of the original Δ-augmented state-space, the sample efficiency is improved by $\mathcal{O}(|\mathcal{A}|^{\Delta-\Delta_\tau})$.

Remark 5.1 (Sample Inefficiency Issue). AD-RL provides an effective framework to alleviate the sample inefficiency issue. However, as shown in Fig. 2 (d), in the stochastic environment with longer delays Δ, we need to set relatively longer auxiliary delays Δ^τ to achieve better performance while somewhat compromising the sample efficiency.

5.2. Performance Gap

While acknowledging that bootstrapping in a smaller augmented state space combined with delayed belief might lead to sub-optimal performance, we demonstrate in this section that this degradation can be effectively bounded. We start by deriving the Lemma 5.2, unifying performance gap between policies (π and π^τ) on the same auxiliary-delayed state space $\mathcal{X}^\tau$. Then, we derive the bounds on the performance gap through the difference of policies in Theorem 5.3. Next, we extend this bound to get the bound on Q-functions of different state spaces in Theorem 5.4. Finally, we show that the bound on optimal Q-functions will become nominal under the deterministic MDP setting.

Following the similar proof sketch with (Kakade & Langford, 2002; Liotet et al., 2022), we give the general delayed policies performance difference lemma as below.

Lemma 5.2 (General Delayed Performance Difference, see Appendix B.2 for proof)**.** *For policies π^τ and π, with delays $\Delta^\tau < \Delta$. Given any $x_t \in \mathcal{X}$, the performance difference is denoted as $I(x_t)$*

$$\begin{aligned} I(x_t) &= \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} [V^\tau(x_t^\tau)] - V(x_t) \\ &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{\hat{x}^\tau \sim b_\Delta(\cdot|\hat{x})\\ a\sim\pi(\cdot|\hat{x})\\ \hat{x}\sim d^\pi_{x_t}}} [V^\tau(\hat{x}^\tau) - Q^\tau(\hat{x}^\tau, a)] \end{aligned}$$

Lemma 5.2 tells us that the performance difference I between policies (π and π^τ) can be measured by the corresponding value functions (V and V^τ) sitting on different augmented state spaces ($\mathcal{X}$ and $\mathcal{X}^\tau$). Since the connection of $\mathcal{X}$ and $\mathcal{X}^\tau$ can be specified by the delayed belief b_Δ, we can unify the expressions on $\mathcal{X}^\tau$ to measure the performance gap, using the auxiliary value functions (V^τ and Q^τ). Assuming Q^τ is L_Q-LC (Definition 3.3), we can show that this performance difference between policies (π and π^τ) can be bounded by the L_1-Wassestein distance between them as followed Theorem 5.3.

Theorem 5.3 (Delayed Performance Difference Bound, proof in Appendix B.3)**.** *For policies π^τ and π, with $\Delta^\tau < \Delta$. Given any $x_t \in \mathcal{X}$, if Q^τ is L_Q-LC, the performance difference between policies can be bounded as follow*

$$\begin{aligned} &\mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t)\\ a_t\sim\pi(\cdot|x_t)}} [V^\tau(x_t^\tau) - Q^\tau(x_t^\tau, a_t)] \\ &\leq L_Q \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} [\mathcal{W}_1(\pi^\tau(\cdot|x_t^\tau)||\pi(\cdot|x_t))] \end{aligned}$$

Combining Lemma 5.2 and Theorem 5.3, we can extend the bound on state-values (V and V^τ) to the bound on Q-values (Q and Q^τ) and optimal Q-values ($Q_{(*)}$ and $Q^\tau_{(*)}$) by the L_1-Wassestein distance of the corresponding policies (π and π^τ) and optimal policies ($\pi_{(*)}$ and $\pi^\tau_{(*)}$), respectively.

Theorem 5.4 (Delayed Q-value Difference Bound, proof in Appendix B.4)**.** *For policies π and π^τ, with $\Delta^\tau < \Delta$. Given any $x_t \in \mathcal{X}$, if Q^τ is L_Q-LC, the corresponding Q-value difference can be bounded as follow*

$$\mathop{\mathbb{E}}_{\substack{a_t\sim\pi(\cdot|x_t)\\ x_t^\tau\sim b_\Delta(\cdot|x_t)}}[Q^\tau(x_t^\tau,a_t)-Q(x_t,a_t)] \le \frac{\gamma L_Q}{1-\gamma}\mathop{\mathbb{E}}_{\substack{\hat{x}^\tau\sim b_\Delta(\cdot|\hat{x})\\ \hat{x}\sim d^\pi_{x_{t+1}}\\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)\\ a_t\sim\pi(\cdot|x_t)}}[\mathcal{W}_1(\pi^\tau(\cdot|\hat{x}^\tau)||\pi(\cdot|\hat{x}))]$$

Specially, for optimal policies $\pi_{(*)}$ and $\pi^\tau_{(*)}$, if $Q^\tau_{(*)}$ is L_Q-LC, the corresponding optimal Q-value difference can be bounded as follow

$$\left|\left|\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{(*)}(x_t^\tau,a_t)\right]-Q_{(*)}(x_t,a_t)\right|\right|_\infty \le \frac{\gamma^2 L_Q}{(1-\gamma)^2}\mathop{\mathbb{E}}_{\substack{\hat{x}^\tau\sim b_\Delta(\cdot|\hat{x})\\ \hat{x}\sim d^\pi_{x_{t+1}}\\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)\\ a_t\sim\pi_{(*)}(\cdot|x_t)}}\left[\mathcal{W}_1\left(\pi^\tau_{(*)}(\cdot|\hat{x}^\tau)\middle|\middle|\pi_{(*)}(\cdot|\hat{x})\right)\right]$$

The results provided in Theorem 5.4, however, is hard to derive the unifying insight further. For instance, it is difficult to calculate $\mathcal{W}_1(\pi^\tau_{(*)}(\cdot|\hat{x}^\tau)||\pi_{(*)}(\cdot|\hat{x}))$, as the optimal policies ($\pi_{(*)}$ and $\pi^\tau_{(*)}$) indeed depend on the property of the underlying MDP. In the case of deterministic MDP where the optimal policies ($\pi_{(*)}$ and $\pi^\tau_{(*)}$) are the same, we can conclude that the optimal delayed Q-value difference becomes nominal in the following remark.

Remark 5.5 (Deterministic MDP Case). For deterministic MDP, b_Δ is also deterministic and becomes injection function meaning that given the x, the x^τ is determined. And then due to $\pi_{(*)}(\cdot|x) = \mathbb{E}_{x^\tau\sim b_\Delta(\cdot|x)}\left[\pi^\tau_{(*)}(\cdot|x^\tau)\right]$, we have

$$\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{(*)}(x_t^\tau,a_t)\right] = Q_{(*)}(x_t,a_t)$$

We also discuss stochastic MDPs as summarized in the following remark.

Remark 5.6 (Stochastic MDP Case). In the case of stochastic MDP, the performance gap might become larger as the difference between Δ and Δ^τ increases. Using a moderate auxiliary delays Δ^τ could trade-off the sample efficiency (closer to 0) and performance consistency (closer to Δ). We also provide experimental results to investigate this in Section 6. Additionally, in Appendix C, we give a stochastic MDP case to exemplify the above conclusion.

5.3. Convergence Analysis

We show in this section that our AD-RL does not sacrifice the convergence. Before presenting the final result, we assume action space $\mathcal{A}$ is finite, which means $|\mathcal{A}| < \infty$. Then, for any $x \in \mathcal{X}$, the L_1-Wassestein distance between policies π and π^τ becomes the bounded l_1 distance, and then the following holds

$$\mathop{\mathbb{E}}_{x^\tau\sim b_\Delta(\cdot|x)}[\mathcal{W}_1(\pi^\tau(\cdot|x^\tau)||\pi(\cdot|x))] < \infty$$

Furthermore, the entropy of the policy π is also bounded as $\mathcal{H}(\pi(\cdot|x)) < \infty$. Then, we show the convergence guarantee of AD-VI and AD-SPI in Section 5.3.1 and Section 5.3.2 respectively.

5.3.1. Convergence of AD-VI

We assume that auxiliary Q-function Q^τ converges to the fixed point $Q^\tau_{(*)}$ and the Q-function Q is updated based on $Q^\tau_{(*)}$, since we only care about the final converged point of Q. Then, we can update an initial Q-function $Q_{(0)}$ by repeatedly applying the Bellman operator $\mathcal{T}$ given by Eq. (3) to get the fixed point $Q_{(\approx)} = \mathbb{E}_{x^\tau\sim b_\Delta(\cdot|x)}\left[Q^\tau_{(*)}(x^\tau,a)\right]$ (Theorem 5.7).

Theorem 5.7 (AD-VI Convergence Guarantee, proof in Appendix B.5)**.** *Consider the bellman operator $\mathcal{T}$ in Eq. (3) and the initial Q-function $Q_{(0)}$: $\mathcal{X}\times\mathcal{A}\to\mathbb{R}$ with $|\mathcal{A}| < \infty$, and define a sequence $\{Q_{(k)}\}_{k=0}^\infty$ where $Q_{(k+1)} = \mathcal{T}Q_{(k)}$. As $k\to\infty$, $Q_{(k)}$ will converge to the fixed point $Q_{(\approx)}$ with Q^τ converges to $Q^\tau_{(*)}$. And for any $(x_t,a_t)\in\mathcal{X}\times\mathcal{A}$, we have*

$$Q_{(\approx)}(x_t,a_t) = \mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{(*)}(x_t^\tau,a_t)\right]$$

Theorem 5.7 guarantees the convergence of the Bellman operator $\mathcal{T}$ and ensures the stability and effectiveness of the learning process in the context of the corresponding practical method, AD-DQN.

5.3.2. Convergence of AD-SPI

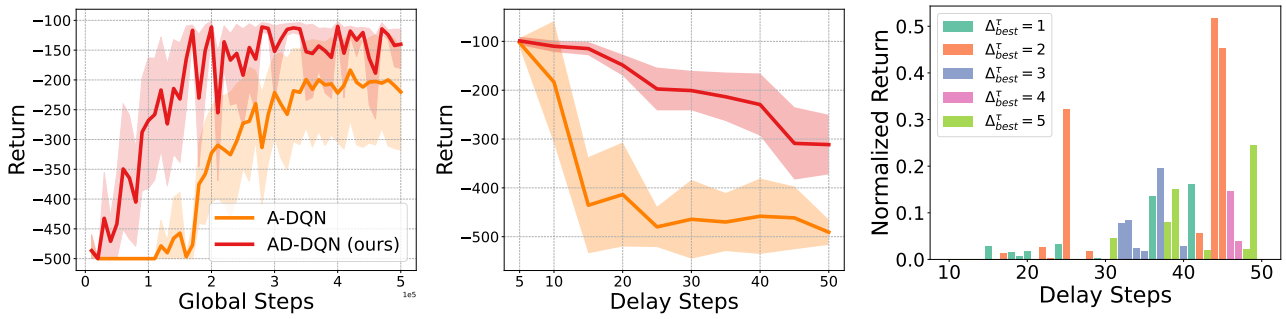
Next, we derive the convergence guarantee of AD-SPI which consists of policy evaluation (Eq. (4)) and policy improvement (Eq. (5)). Similar to AD-VI, we assume that Q^τ has converged to the soft Q-value Q^τ_{soft} (Haarnoja et al., 2017; 2018a) in the context of AD-SPI. For the policy evaluation, Q-function Q can converge to a fixed point via iteratively applying the soft Bellman operator $\mathcal{T}^\pi$ defined in Eq. (4). Lemma 5.8 shows this convergence guarantee.

Lemma 5.8 (Policy Evaluation Convergence Guarantee, proof in Appendix B.6)**.** *Consider the soft bellman operator $\mathcal{T}^\pi$ in Eq. (4) and the initial Q-value function $Q_{(0)}$: $\mathcal{X}\times\mathcal{A}\to\mathbb{R}$ with $|\mathcal{A}| < \infty$, and define a sequence $\{Q_{(k)}\}_{k=0}^\infty$ where $Q_{(k+1)} = \mathcal{T}^\pi Q_{(k)}$. Then for any $(x_t,a_t)\in\mathcal{X}\times\mathcal{A}$, as $k\to\infty$, $Q_{(k)}(x_t,a_t)$ will converge to the fixed point*

$$\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{soft}(x_t^\tau,a_t)\right] - \log\pi(a_t|x_t)$$

Table 1. Results of MuJoCo tasks with 25 delays for 1M global time-steps. Each method was evaluated with 10 trials and is shown with the standard deviation denoted by $\pm$. The best performance is in blue.

Delays=25	Ant-v4	HalfCheetah-v4	Hopper-v4	Humanoid-v4	HumanoidStandup-v4	Pusher-v4	Reacher-v4	Swimmer-v4	Walker2d-v4
A-SAC	$0.07_{\pm 0.07}$	$0.04_{\pm 0.01}$	$0.13_{\pm 0.04}$	$0.05_{\pm 0.01}$	$0.97_{\pm 0.09}$	$0.49_{\pm 0.32}$	$0.96_{\pm 0.02}$	$0.72_{\pm 0.02}$	$0.12_{\pm 0.02}$
DC/AC	$0.19_{\pm 0.02}$	$0.16_{\pm 0.07}$	$0.19_{\pm 0.04}$	$0.04_{\pm 0.01}$	$1.03_{\pm 0.03}$	$1.12_{\pm 0.02}$	$1.00_{\pm 0.00}$	$0.78_{\pm 0.12}$	$0.26_{\pm 0.08}$
DIDA	$0.29_{\pm 0.07}$	$0.12_{\pm 0.03}$	$0.27_{\pm 0.08}$	$0.07_{\pm 0.00}$	$0.97_{\pm 0.02}$	$1.04_{\pm 0.01}$	$0.98_{\pm 0.01}$	$0.93_{\pm 0.09}$	$0.10_{\pm 0.02}$
BPQL	$0.57_{\pm 0.11}$	$0.87_{\pm 0.04}$	$1.21_{\pm 0.18}$	$0.12_{\pm 0.01}$	$1.09_{\pm 0.05}$	$1.07_{\pm 0.06}$	$0.87_{\pm 0.05}$	$1.36_{\pm 0.56}$	$0.59_{\pm 0.30}$
AD-SAC (ours)	$0.66_{\pm 0.04}$	$0.71_{\pm 0.12}$	$0.86_{\pm 0.25}$	$0.25_{\pm 0.16}$	$1.15_{\pm 0.08}$	$1.29_{\pm 0.03}$	$0.98_{\pm 0.02}$	$2.52_{\pm 0.40}$	$0.72_{\pm 0.11}$



(a) (deterministic) 10 delays (b) (deterministic) varying delays (c) (stochastic) normalized return of Δ^{τ}_{best}

Figure 3. Results of deterministic Acrobot for (a) learning curves with 10 delays and (b) final performance with varying delays (5-50). The shaded area is the standard deviation. Results of stochastic Acrobot for (c) the normalized return of Δ^{τ}_{best} with different delays (10-50). Different colors stand for different best returns achieved by different optimal auxiliary delays Δ^{τ}_{best}.

For the soft policy improvement, we improve the old policy π_{old} to the new one π_{new} via applying the update rule in Eq. (5). And formalizing in Lemma 5.9, we can show that the improved policy π_{new} is better than π_{old}.

Lemma 5.9 (Policy Improvement Guarantee, proof in Appendix B.7)**.** *Consider the policy update rule in Eq. (5), and let π_{old}, π_{new} be the old policy and new policy improved from old one respectively. Then for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$, we have $Q_{old}(x_t, a_t) \leq Q_{new}(x_t, a_t)$.*

Alternating between the policy evaluation and policy improvement, the policy learned by the AD-SPI will converge to a policy $\pi_{(\approx)}$ having the highest value among the policies in Π. This result formalized in Theorem 5.10 establishes the theoretical fundamental for us to develop the AD-SAC.

Theorem 5.10 (AD-SPI Convergence Guarantee)**.** *Applying policy evaluation in Eq. (4) and policy improvement in Eq. (5) repeatedly to any given policy $\pi \in \Pi$, it converges to $\pi_{(\approx)}$ such that $Q^{\pi}(x_t, a_t) \leq Q^{\pi}_{(\approx)}(x_t, a_t)$ for any $\pi \in \Pi$, $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.*

6. Experimental Results

6.1. Benchmarks and Baselines

Benchmarks. We choose the Acrobot (Sutton, 1995) and the MuJoCo control suite (Todorov et al., 2012) for discrete and continuous control tasks, respectively. Especially, to investigate the effectiveness of AD-RL in the stochastic MDPs, we adopt the stochastic Acrobot where the input from the agent is disturbed by the noise from a uniform distribution with the probability of 0.1. In this way, the output for the same inputted action from the agent becomes stochastic.

Baselines. We select a wide range of techniques as baselines to examine the performance of our AD-RL under different environments. In Acrobot, we mainly compare AD-DQN against the augmented DQN (A-DQN) (Mnih et al., 2015). For the continuous control task, we compared our AD-SAC against existing SOTAs: augmented SAC (A-SAC) (Haarnoja et al., 2018a), DC/AC (Bouteiller et al., 2020), DIDA (Liotet et al., 2022) and BPQL (Kim et al., 2023). In the deterministic Acrobot and Mujoco, the auxiliary delays Δ^{τ} of our AD-RL is set to 0. For computationally fairness, we keep all methods training with the same amount of gradient descent. In other words, the training times of π in our AD-RL is **half of that in all baselines**, as AD-RL trains π^{τ} additional to the original task at the same time. The result of each method was obtained using 10 random seeds and its hyper-parameters are in Appendix A.

6.2. Empirical Results Analysis

Deterministic Acrobot. We first compare A-DQN and our AD-DQN in the deterministic Acrobot under fixed 10 delays. From Fig. 3(a), we can tell that within the 500k time steps, AD-DQN learns much faster than the A-DQN. In addition,

Table 2. Results of MuJoCo tasks with stochastic delays for 1M global time-steps. Each method was evaluated with 10 trials and is shown with the standard deviation denoted by $\pm$. The best performance is in blue.

Task	A-SAC	DC/AC	DIDA	BPQL	AD-SAC
Ant-v4	$0.18_{\pm 0.01}$	$0.27_{\pm 0.02}$	$0.55_{\pm 0.08}$	$0.58_{\pm 0.12}$	$0.69_{\pm 0.17}$
HalfCheetah-v4	$0.36_{\pm 0.12}$	$0.36_{\pm 0.18}$	$0.75_{\pm 0.02}$	$0.76_{\pm 0.16}$	$1.03_{\pm 0.06}$
Hopper-v4	$0.85_{\pm 0.22}$	$0.94_{\pm 0.29}$	$0.31_{\pm 0.08}$	$0.68_{\pm 0.34}$	$1.05_{\pm 0.22}$
Humanoid-v4	$0.15_{\pm 0.06}$	$0.67_{\pm 0.18}$	$0.07_{\pm 0.01}$	$0.40_{\pm 0.42}$	$0.97_{\pm 0.07}$
Standup-v4	$1.03_{\pm 0.05}$	$1.20_{\pm 0.08}$	$1.00_{\pm 0.00}$	$1.10_{\pm 0.07}$	$1.26_{\pm 0.07}$
Pusher-v4	$1.11_{\pm 0.02}$	$1.17_{\pm 0.02}$	$1.02_{\pm 0.01}$	$1.07_{\pm 0.05}$	$1.22_{\pm 0.01}$
Reacher-v4	$0.98_{\pm 0.01}$	$1.02_{\pm 0.01}$	$1.02_{\pm 0.00}$	$0.85_{\pm 0.11}$	$1.05_{\pm 0.01}$
Swimmer-v4	$0.82_{\pm 0.10}$	$1.47_{\pm 0.58}$	$1.03_{\pm 0.02}$	$1.53_{\pm 0.52}$	$2.36_{\pm 0.64}$
Walker2d-v4	$0.68_{\pm 0.28}$	$0.89_{\pm 0.08}$	$0.54_{\pm 0.09}$	$0.63_{\pm 0.39}$	$1.19_{\pm 0.14}$

as delays change from 5 to 50, from Fig. 3(b), A-DQN is not able to learn any useful policy after 20 delays. However, our AD-DQN shows robust performance even under 50 delays.

MuJoCo. The results for MuJoCo are presented in Table 1. Our AD-SAC and BPQL provide leading performance in most MuJoCo tasks. We report the delay-free normalized scores $Ret_{nor} = (Ret_{alg} - Ret_{rand})/(Ret_{df} - Ret_{rand})$, where Ret_{df} is the return of a delay-free agent trained by the soft actor-critic and Ret_{rand} is the return of a random policy. The experiment results support our argument that learning the auxiliary-delayed task facilitates agents to learn the original delayed task. In deterministic scenarios, the delayed belief estimation degenerates to a deterministic function for any auxiliary delays Δ^τ, and a smaller Δ^τ benefits the sampling efficiency. BPQL (a special variant of our approach when $\Delta^\tau = 0$) and AD-SAC(0) thus provide comparable results. The results for MuJoCo tasks with 5 and 50 delays, presented in Appendix D, also validate this conclusion.

MuJoCo with Stochastic Delays. We evaluate the performance and robustness of AD-RL in MuJoCo which has the delay $\Delta = 5$ with a probability of 0.9, and the delay $\Delta \in [1, 5]$ with a probability of 0.1. We adopt a similar stochastic delay setting as BPQL (Kim et al., 2023) wherein the evaluation environment has the delay $\Delta = 5$ with a probability of 0.9, and the delay $\Delta \in [1, 5]$ with a probability of 0.1. As shown in Table 2, AD-RL outperforms all the other baselines including BPQL in all tasks with stochastic delays. Compared with the result of AD-RL in the constant delay setting, we can see that AD-RL is more robust than others.

Stochastic Acrobot. Then we investigate the influence of auxiliary delays for AD-RL on the stochastic Acrobot. We conduct a series of experiments for various combinations of delays (ranging from 10 to 50) and auxiliary delays (ranging from 0 to 5). The optimal auxiliary delays, denoted as Δ^τ_{best}, which yield the best performance under different delays, are recorded. Furthermore, the relative return is defined as $Ret_{rela} = (Ret_{\Delta^\tau_{best}} - Ret_0)/(Ret_0 - Ret_{rand})$ where $Ret_{\Delta^\tau_{best}}$ is the return of setting $\Delta^\tau = \Delta^\tau_{best}$, Ret_0 is the return of setting $\Delta^\tau = 0$ and Ret_{rand} is the return of the random policy. It measures the comparatively improved performance in return by $\Delta^\tau = \Delta^\tau_{best}$ instead of setting $\Delta^\tau = 0$. As illustrated in Fig. 3(c), we can observe that $\Delta^\tau = 0$ (as in BPQL) may not always be the optimal choice and the selection of best auxiliary delays appears irregular in the stochastic MDP. It is evident to confirm our aforementioned conclusion: the shorter auxiliary delays Δ^τ could improve the learning efficiency but also potentially result in performance degradation caused by a more significant information loss in stochastic environments. So there exists a trade-off between learning efficiency (Δ^τ closes to 0) and performance consistency (Δ^τ closes to Δ).

Table 3. The best auxiliary delays and corresponding normalized score of AD-RL in Stochastic MuJoCo with 50 delays.

Delays=50	Normalized score (Δ^τ_{best})
Ant-v4	$0.16 \pm_{0.04}$ (4)
HalfCheetah-v4	$0.05 \pm_{0.28}$ (2)
Hopper-v4	$0.05 \pm_{0.17}$ (5)
Humanoid-v4	$0.00 \pm_{0.00}$ (0)
HumanoidStandup-v4	$0.03 \pm_{0.06}$ (1)
Pusher-v4	$0.02 \pm_{0.01}$ (3)
Reacher-v4	$0.03 \pm_{0.02}$ (1)
Swimmer-v4	$0.00 \pm_{0.00}$ (0)
Walker2d-v4	$0.00 \pm_{0.00}$ (0)

Stochastic MuJoCo. We adopt a similar stochastic MuJoCo setting as BPQL (Kim et al., 2023) wherein an agent-unaware action noise is added into the environment with a probability of 0.1, making the MuJoCo stochastic (different outputs for the same input action). In stochastic MuJoCo with 50 delays, we train AD-RL with different auxiliary delays Δ^τ ranging from 0 to 5, and we report the best auxiliary delays Δ^τ_{best} and corresponding normalized scores $(Ret_{best} - Ret_0)/(Ret_0 - Ret_{rand})$. As shown in Table 3, the best choice is not always 0, and it is sensitive to the specific task setting and shows strong irregularity.

7. Related Work

Early works model a continuous system with observation delay by Delay Differential Equations (DDE) (Myshkis, 1955; Cooke, 1963) which have been extensively studied in the control community in terms of reachability (Fridman & Shaked, 2003; Xue et al., 2021), stability (Feng et al., 2019), and safety (Xue et al., 2020). These techniques rely on explicit dynamics models and cannot scale well.

In recent RL approaches, the environment with observation delay is modelled under the MDP framework (Altman & Nain, 1992; Katsikopoulos & Engelbrecht, 2003). Unlike the well-studied reward delay problem which raises the credit assignment issue (Sutton, 1984; Arjona-Medina et al., 2019; Wang et al., 2024), the observation delay problem disrupts the Markovian property required for traditional RL. Existing techniques differ from each other in how to tame the delay, including memoryless-based, model-based, and augmentation-based methods. Inspired by the partially observed MDP (POMDP), memory-less approaches, e.g., dQ and dSARSA, were developed in (Schuitema et al., 2010), which learn the policy based on the observed state. However, these methods ignore the non-Markovian property of the problem and could lead to performance degeneration.

Model-based methods retrieve the Markovian property by predicting the unobserved state and then selecting an action based on it. The performance thus highly relies on state generation techniques. Walsh et al. propose a deterministic generative model which is learned via model-based simulation. Similarly, Derman et al. suggested successively applying an approximate forward model to estimate state (Derman et al., 2021). Different stochastic generative models, e.g., the ensemble of Gaussian distributions (Chen et al., 2021) and Transformers (Liotet et al., 2021), were also explored. However, the non-negligible prediction error results in sub-optimal returns (Liotet et al., 2021).

Augmentation-based methods seek to equivalently retrieve the Markov property by augmenting the delay-related information into the state-space (Altman & Nain, 1992). For instance, inspired by multi-step learning, Bouteiller et al. develops a partial trajectory resampling technique to accelerate the learning process (Bouteiller et al., 2020). Additionally, based on imitation learning and dataset aggregation technique, Liotet et al. trains an undelayed expert policy and subsequently generalizes the expert's behavior into an augmented policy (Liotet et al., 2022). Despite possessing optimality and the Markov property, augmentation-based methods are plagued by the curse of dimensionality in facing tasks with long delays, resulting in learning inefficiency. BPQL (Kim et al., 2023) evaluates the augmented policy by a non-augmented Q-function. However, BPQL suffers from performance loss in stochastic environments. Certain difficult problems can be solved by 'divide and conquer' methods such as the neural sequence chunker (Schmidhuber, 1991a) which learns to hierarchically decompose sequences into predictable subsequences, and the subgoal generator (Schmidhuber, 1991b), which decomposes task into smaller tasks with shorter solutions by learning appropriate subgoals.

8. Conclusion

In this work, we focus on RL in environments with delays between events and their observations. Existing methods exhibit learning inefficiency in the presence of long delays and performance degradation in stochastic environments. To address these issues, we propose AD-RL, which leverages auxiliary tasks with short delays to enhance learning. Under the AD-RL framework, we develop AD-DQN and AD-SAC for discrete and continuous control tasks respectively. We further provide a theoretical analysis in terms of sample efficiency, performance gap, and convergence. In both deterministic and stochastic benchmarks, we empirically show that AD-RL achieves new state-of-the-art performance, dramatically outperforming existing methods.

9. Limitations and Chanllenges

Hyperparameter Δ^{τ}_{best} It is worth noting that the performance of AD-RL is subject to the selection of the auxiliary delays Δ^{τ}_{best}. Such selection is deeply related to the specific tasks and even the delay Δ, and is highly challenging: Fig. 3(c) demonstrates that the relation between Δ^{τ}_{best} and Δ is not linear or parabolic. We will consider a systemic framework to select the best auxiliary delay by exploring the underlying relationship between the auxiliary delay, the original delay and the task specification, etc.

Implicit Belief Learning Learning delayed belief, especially in stochastic environments, proves to be challenging. AD-RL implicitly represents the belief function by sampling in two augmented state spaces (Δ and Δ^{τ}) to overcome this issue, leading to additional memory cost compared to conventional augmentation-based approaches. In the future, we will explore learning explicit belief using different neural representations.

Sample Efficiency Sample efficiency remains a critical challenge, especially in environments characterized by long delays or stochasticity. In both cases, AD-RL needs to set relatively longer auxiliary delays Δ^{τ} to carry more information of the augmented state in the auxiliary state. While a longer auxiliary delay achieves better performance, it brings back the sample inefficiency issue, which will be investigated in the next step.

Acknowledgement

We sincerely acknowledge the support by the grant EP/Y002644/1 under the EPSRC ECR International Collaboration Grants program, funded by the International Science Partnerships Fund (ISPF) and the UK Research and Innovation, and the grant 2324936 by the US National Science Foundation. This work is also supported by Taiwan NSTC under Grant Number NSTC-112-2221-E-002-168-MY3, and the European Research Council (ERC, Advanced Grant Number 742870).

Impact Statement

This paper aims to advance the field of Machine Learning. While acknowledging there are many potential societal consequences of our work, we believe that none need to be specially highlighted here.

References

Altman, E. and Nain, P. Closed-loop control with delayed information. *ACM sigmetrics performance evaluation review*, 20(1):193–204, 1992.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Azar, M. G., Munos, R., Ghavamzadeh, M., and Kappen, H. Speedy q-learning. In *Advances in neural information processing systems*, 2011.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bertsekas, D. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

Bouteiller, Y., Ramstedt, S., Beltrame, G., Pal, C., and Binas, J. Reinforcement learning with random delays. In *International conference on learning representations*, 2020.

Chen, B., Xu, M., Li, L., and Zhao, D. Delay-aware model-based reinforcement learning for continuous control. *Neurocomputing*, 450:119–128, 2021.

Cooke, K. L. Differential—difference equations. In *International symposium on nonlinear differential equations and nonlinear mechanics*, pp. 155–171. Elsevier, 1963.

Derman, E., Dalal, G., and Mannor, S. Acting in delayed environments with non-stationary markov policies. *arXiv preprint arXiv:2101.11992*, 2021.

Even-Dar, E., Mansour, Y., and Bartlett, P. Learning rates for q-learning. *Journal of machine learning Research*, 5 (1), 2003.

Fathi, M., Haghi Kashani, M., Jameii, S. M., and Mahdipour, E. Big data analytics in weather forecasting: A systematic review. *Archives of Computational Methods in Engineering*, 29(2):1247–1275, 2022.

Feng, S., Chen, M., Zhan, N., Fränzle, M., and Xue, B. Taming delays in dynamical systems: Unbounded verification of delay differential equations. In *International Conference on Computer Aided Verification*, pp. 650–669. Springer, 2019.

Firoiu, V., Ju, T., and Tenenbaum, J. At human speed: Deep reinforcement learning with action delay. *arXiv preprint arXiv:1810.07286*, 2018.

Fridman, E. and Shaked, U. On reachable sets for linear systems with delay and bounded peak inputs. *Automatica*, 39(11):2005–2010, 2003.

Gangwani, T., Lehman, J., Liu, Q., and Peng, J. Learning belief representations for imitation learning in pomdps. In *uncertainty in artificial intelligence*, pp. 1061–1071. PMLR, 2020.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

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, 2018a.

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

Han, B., Ren, Z., Wu, Z., Zhou, Y., and Peng, J. Off-policy reinforcement learning with delayed rewards. In *International Conference on Machine Learning*, pp. 8280–8303. PMLR, 2022.

Hasbrouck, J. and Saar, G. Low-latency trading. *Journal of Financial Markets*, 16(4):646–679, 2013.

Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the*

Nineteenth International Conference on Machine Learning, pp. 267–274, 2002.

Katsikopoulos, K. V. and Engelbrecht, S. E. Markov decision processes with delays and asynchronous cost collection. *IEEE transactions on automatic control*, 48(4): 568–574, 2003.

Kim, J., Kim, H., Kang, J., Baek, J., and Han, S. Belief projection-based reinforcement learning for environments with delayed feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kim, J.-G. and Lee, B. Automatic p2p energy trading model based on reinforcement learning using long short-term delayed reward. *Energies*, 13(20):5359, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Liotet, P., Venneri, E., and Restelli, M. Learning a belief representation for delayed reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Liotet, P., Maran, D., Bisi, L., and Restelli, M. Delayed reinforcement learning by imitation. In *International Conference on Machine Learning*, pp. 13528–13556. PMLR, 2022.

Mahmood, A. R., Korenkevych, D., Komer, B. J., and Bergstra, J. Setting up a reinforcement learning task with a real-world robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4635–4640. IEEE, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Myshkis, A. D. Lineare differentialgleichungen mit nacheilendem argument. 1955.

Nath, S., Baranwal, M., and Khadilkar, H. Revisiting state augmentation methods for reinforcement learning with stochastic delays. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1346–1355, 2021.

Rachelson, E. and Lagoudakis, M. G. On the locality of action domination in sequential decision making. 2010.

Schmidhuber, J. neural sequence chunkers. 1991a.

Schmidhuber, J. Learning to generate sub-goals for action sequences. In *Artificial neural networks*, pp. 967–972, 1991b.

Schuitema, E., Buşoniu, L., Babuška, R., and Jonker, P. Control delay in reinforcement learning for real-time dynamic systems: A memoryless approach. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3226–3231. IEEE, 2010.

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

Sutton, R. S. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst, 1984.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8, 1995.

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

Tesauro, G. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6 (2):215–219, 1994.

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.

Villani, C. et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Walsh, T. J., Nouri, A., Li, L., and Littman, M. L. Learning and planning in environments with delayed feedback. *Autonomous Agents and Multi-Agent Systems*, 18:83–105, 2009.

Wang, Y., Zhan, S., Wang, Z., Huang, C., Wang, Z., Yang, Z., and Zhu, Q. Joint differentiable optimization and verification for certified reinforcement learning. In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, pp. 132–141, 2023a.

Wang, Y., Zhan, S. S., Jiao, R., Wang, Z., Jin, W., Yang, Z., Wang, Z., Huang, C., and Zhu, Q. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*, pp. 36593–36604. PMLR, 2023b.

Wang, Y., Strupl, M., Faccio, F., Wu, Q., Liu, H., Grudzień, M., Tan, X., and Schmidhuber, J. Highway reinforcement learning. *arXiv preprint arXiv:2405.18289*, 2024.

Xu, S., Fu, Y., Wang, Y., O'Neill, Z., and Zhu, Q. Learning-based framework for sensor fault-tolerant building hvac control with model-assisted learning. In *Proceedings of the 8th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pp. 1–10, 2021.

Xu, S., Fu, Y., Wang, Y., Yang, Z., O'Neill, Z., Wang, Z., and Zhu, Q. Accelerate online reinforcement learning for building hvac control with heterogeneous expert guidances. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '22, pp. 89–98, 2022.

Xue, B., Wang, Q., Feng, S., and Zhan, N. Over-and under-approximating reach sets for perturbed delay differential equations. *IEEE Transactions on Automatic Control*, 66 (1):283–290, 2020.

Xue, B., Bai, Y., Zhan, N., Liu, W., and Jiao, L. Reach-avoid analysis for delay differential equations. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 1301–1307. IEEE, 2021.

Zhan, S. S., Wang, Y., Wu, Q., Jiao, R., Huang, C., and Zhu, Q. State-wise safe reinforcement learning with pixel observations. *6th Annual Learning for Dynamics and Control Conference*, 2024.

A. Implementation Detail

The hyper-parameters setting used in this work is provided in Table 4 for Acrobot and MuJoCo benchmarks. We provide the pseudo-code of AD-DQN and AD-SAC in Algorithm 2 and Algorithm 3, respectively, along with the description of how to practically implement them in detail. The code for reproducing our results can be found in the supplementary material.

A.1. Hyper-parameters Setting

Hyper-parameter	Setting(Acrobot)	Setting(MuJoCo)
buffer size	1,000	1,000,000
batch size	128	256
total timesteps	500,000	1,000,000
discount factor	0.99	0.99
learning rate	2.5e-4	3e-4(actor), 1e-3(critic)
network layers	3	3
network neurons	[128, 64]	[256, 256]
activation	ReLU	ReLU
optimizer	Adam (Kingma & Ba, 2014)	Adam
initial ϵ for ϵ-greedy	1.0	-
final ϵ for ϵ-greedy	0.05	-
initial entropy α for SAC	-	0.2
learning rate for entropy α	-	1e-3
train frequency	5	2(actor), 1(critic)
target network update frequency	500	-
target network soft update factor	-	5e-3
n for N-steps	-	3
auxiliary delays Δ^τ for AD-RL	0	0

Table 4. Hyper-parameters Setting on Acrobot and MuJoCo benchmarks.

A.2. Discrete Control: AD-DQN

In practice, we will maintain two Q-networks (Q_ψ, Q_θ^τ) at the same time, and each of them corresponds to different delays (Δ, Δ^τ). When the agent needs to select an action, it will first enquire about two Q-networks and get the two best actions in different views of delays, separately. Then evaluate these two actions based on the auxiliary Q-network Q_θ^τ. Here, we experimentally found that the *argmin* operator, taking a more conservative action is more stable than *argmax* operator. For the auxiliary Q-network Q_θ^τ, its update rule is the same as the original DQN. And for the Q-network Q_ψ^τ, we update it in our way: bootstrapping on Q_θ^τ. To stabilize the training process, we also adopt the target networks to estimate the td-targets (Mnih et al., 2015).

A.3. Continuous Control: AD-SAC

Applying our method to soft actor-critic, we adopt some advanced modifications (Haarnoja et al., 2018b), including removing the unnecessary value network, adjusting entropy automatically. We maintain two policies π_ψ and π_ϕ^τ for delays Δ and Δ^τ respectively. To stabilize off-policy training and reduce bootstrapping error (Kumar et al., 2019) in AD-SAC, we maintain two Q-functions ($Q_{\theta_1}^\tau, Q_{\theta_2}^\tau$) for evaluating π_ψ and π_ϕ^τ respectively. Similar to the AD-DQN, selecting the conservative action from policies π_ψ and π_ϕ^τ can stabilize the learning process.

Algorithm 2 Auxiliary-Delayed Deep Q-Network (AD-DQN)

Input: Q-network Q_ψ and target network $\hat{Q}_{\hat{\psi}}$ for delays Δ; Q-network Q^τ_θ and target network $\hat{Q}^\tau_{\hat{\theta}}$ for auxiliary delays Δ^τ; discount factor γ; empty replay buffer $\mathcal{D}$;
for each episode **do**
  Initial state buffer $(s_1, \cdots, s_{\Delta^\tau}, \cdots, s_\Delta)$ and action buffer $(a_1, \cdots, a_{\Delta^\tau}, \cdots, a_\Delta)$
  for each environment step t **do**
    Observe augment states $x_t = (s_1, a_1, \cdots, a_\Delta)$ and $x^\tau_t = (s_{\Delta^\tau}, a_{\Delta^\tau}, \cdots, a_\Delta)$
    Sample and take action $a_t = \underset{\hat{a} \in \{\arg\max_a Q_\psi(x_t, a), \arg\max_a Q^\tau_\theta(x^\tau_t, a)\}}{\arg\min} Q^\tau_\theta(x^\tau_t, \hat{a})$
    Observe reward r_t and next state s_{t+1}
    Update state buffers: $(s_1, \cdots, s_\Delta) \leftarrow (s_2, \cdots, s_\Delta, s_t)$ and action buffers: $(a_1, \cdots, a_\Delta) \leftarrow (a_2, \cdots, a_\Delta, a_t)$
    Update replay buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t, x^\tau_t, a_t, r_t, x_{t+1}, x^\tau_{t+1}\}$
    for each batch $\{x_t, x^\tau_t, a_t, r_t, x_{t+1}, x^\tau_{t+1}\} \sim \mathcal{D}$ **do**
      if Uniform$(0, 1) > 0.5$ **then**
        Update Q^τ_θ via applying gradient descent

$$\nabla_\theta \left(Q^\tau_\theta(x^\tau_t, a_t) - \left[r_t + \gamma \max_{a_{t+1}} \hat{Q}^\tau_{\hat{\theta}}(x^\tau_{t+1}, a_{t+1}) \right] \right)$$

      else
        Update Q_ψ via applying gradient descent

$$\nabla_\psi \left(Q_\psi(x_t, a_t) - \left[r_t + \gamma \hat{Q}^\tau_{\hat{\theta}}(x^\tau_{t+1}, \arg\max_{a_{t+1}} \hat{Q}_{\hat{\psi}}(x_{t+1}, a_{t+1})) \right] \right)$$

      end if
    end for
    Update target networks weights $\hat{Q}_{\hat{\psi}}$ and $\hat{Q}^\tau_{\hat{\theta}}$ via copying from Q_ψ and Q^τ_θ, respectively
  end for
end for
Output: Q-network Q_ψ

Algorithm 3 Auxiliary Delay Soft Actor-Critic (AD-SAC)

Input: actor π_ψ for delay Δ; actor π^τ_ϕ, critics $Q^\tau_{\theta_1}, Q^\tau_{\theta_2}$ and target critics $\hat{Q^\tau}_{\hat{\theta}_1}, \hat{Q^\tau}_{\hat{\theta}_2}$ for auxiliary delays Δ^τ; n-step n; replay buffer $\mathcal{D}$;
for each episode **do**
  Initial state buffer $(s_1, \cdots, s_{\Delta^\tau}, \cdots, s_\Delta)$ and action buffer $(a_1, \cdots, a_{\Delta^\tau}, \cdots, a_\Delta)$
  for each environment step t **do**
    Observe augment states $x_t = (s_1, a_1, \cdots, a_\Delta)$ and $x^\tau_t = (s_{\Delta^\tau}, a_{\Delta^\tau}, \cdots, a_\Delta)$
    Sample and take action $a_t = \underset{a \in \{\pi_\psi(x_t), \pi^\tau_\phi(x^\tau_t)\}}{\arg\min} Q^\tau_{\theta_i}(x^\tau_t, a)$
    Observe reward r_t and next state s_{t+1}
    Update state buffer $(s_1, \cdots, s_\Delta) \leftarrow (s_2, \cdots, s_\Delta, s_t)$ and action buffer $(a_1, \cdots, a_\Delta) \leftarrow (a_2, \cdots, a_\Delta, a_t)$
    Update replay buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t, x^\tau_t, a_t, r_t, x_{t+1}, x^\tau_{t+1}\}$
  end for
end for
for each batch $\{x_t, x^\tau_t, a_t, r_t : r_{t+n-1}, x_{t+n}, x^\tau_{t+n}\} \sim \mathcal{D}$ **do**
  Compute TD Target

$$\mathbb{Y} = \underset{\substack{\hat{a} \sim \pi^\tau_\phi(\cdot|x^\tau_{t+n}) \\ \hat{a} \sim \pi_\psi(\cdot|x_{t+n})}}{\mathbb{E}} \left[\sum_{i=0}^{n-1} \left[\gamma^i r_{t+i}\right] + \gamma^n \min\left(Q^\tau_{\theta_1}(x^\tau_{t+n}, \hat{a}) - \log \pi^\tau_\phi(\hat{a}|x^\tau_{t+n}), Q^\tau_{\theta_2}(x_{t+n}, \hat{a}) - \log \pi_\phi(\hat{a}|x_{t+1}),\right)\right]$$

  Update $Q^\tau_{\theta_i}(i = 1, 2)$ via applying gradient descent

$$\nabla_{\theta_i} \left[Q^\tau_{\theta_i}(x^\tau_t, a_t) - \mathbb{Y}\right]$$

  if Uniform$(0, 1) > 0.5$ **then**
    Update π^τ_ϕ via applying gradient descent

$$\nabla_\phi \underset{\hat{a} \sim \pi^\tau_\phi(\cdot|x^\tau_t)}{\mathbb{E}} \left[\log \pi^\tau_\phi(\hat{a}|x^\tau_t) - \min_{i=1,2} Q^\tau_{\theta_i}(x^\tau_t, \hat{a})\right]$$

  else
    Update π_ψ via applying gradient descent

$$\nabla_\psi \underset{\hat{a} \sim \pi_\psi(\cdot|x_t)}{\mathbb{E}} \left[\log \pi_\psi(\hat{a}|x_t) - \min_{i=1,2} Q^\tau_{\theta_i}(x^\tau_t, \hat{a})\right]$$

  end if
  Soft update target critics weights $Q^\tau_{\theta_1}, Q^\tau_{\theta_2}$ via copying from $\hat{Q^\tau}_{\hat{\theta}_1}, \hat{Q^\tau}_{\hat{\theta}_2}$, respectively
end for
Output: actor π_ψ

B. Theoretical Analysis

B.1. Performance Difference

Proposition B.1 (Lipschitz Continuous Q-value function Bound (Liotet et al., 2022)). *Consider a L_Q-LC Q-function Q^π of the L_π-LC policy π in the (L_P, L_R)-LC MDP, it satisfies that $\forall x \in \mathcal{X}$,*

$$\left| \mathop{\mathbb{E}}_{\substack{a_1\sim\mu \\ a_2\sim\upsilon}} \left[Q^\pi(x,a_1) - Q^\pi(x,a_2)\right] \right| \leq L_Q W_1\left(\mu || \upsilon\right)$$

where μ, υ are two arbitrary distributions over $\mathcal{X}$.

Lemma B.2 (General Delayed Performance Difference). *For policies π^τ and π, with delays $\Delta^\tau < \Delta$. Given any $x_t \in \mathcal{X}$, the performance difference is denoted as $I(x_t)$*

$$\begin{aligned} I(x_t) &= \mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)} \left[V^\tau(x_t^\tau)\right] - V(x_t) \\ &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{\hat{x}^\tau\sim b_\Delta(\cdot|\hat{x}) \\ a\sim\pi(\cdot|\hat{x}) \\ \hat{x}\sim d^\pi_{x_t}}} \left[V^\tau(\hat{x}^\tau) - Q^\tau(\hat{x}^\tau, a)\right] \end{aligned}$$

Proof.

$$\begin{aligned} &\underbrace{\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)} \left[V^\tau(x_t^\tau)\right] - V(x_t)}_{I(x_t)} \\ &= \underbrace{\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)} \left[V^\tau(x_t^\tau)\right] - \mathop{\mathbb{E}}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[\mathcal{R}_\tau(x_t^\tau, a_t) + \gamma \mathop{\mathbb{E}}_{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t)} \left[V^\tau(x_{t+1}^\tau)\right]\right]}_{A} \\ &+ \underbrace{\mathop{\mathbb{E}}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[\mathcal{R}_\tau(x_t^\tau, a_t) + \gamma \mathop{\mathbb{E}}_{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t)} \left[V^\tau(x_{t+1}^\tau)\right]\right] - V(x_t)}_{B} \end{aligned}$$

For A, we have

$$A = \mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)} \left[V^\tau(x_t^\tau)\right] - \mathop{\mathbb{E}}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[Q^\tau(x_t^\tau, a_t)\right]$$

And for B, note that $V(x_t) = \mathbb{E}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[\mathcal{R}_\tau(x_t^\tau, a_t)\right] + \gamma \mathbb{E}_{\substack{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[V(x_{t+1})\right]$, then we have

$$\begin{aligned} B &= \mathop{\mathbb{E}}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[\mathcal{R}_\tau(x_t^\tau, a_t) + \gamma \mathop{\mathbb{E}}_{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t)} \left[V^\tau(x_{t+1}^\tau)\right]\right] \\ &\quad - \mathop{\mathbb{E}}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[\mathcal{R}_\tau(x_t^\tau, a_t)\right] - \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[V(x_{t+1})\right] \\ &= \mathop{\mathbb{E}}_{\substack{x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[\gamma \mathop{\mathbb{E}}_{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t)} \left[V^\tau(x_{t+1}^\tau)\right]\right] - \mathop{\mathbb{E}}_{a_t\sim\pi(\cdot|x_t)} \left[\gamma \mathop{\mathbb{E}}_{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)} \left[V(x_{t+1})\right]\right] \\ &= \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t) \\ a_t\sim\pi(\cdot|x_t)}} \underbrace{\left[\mathop{\mathbb{E}}_{x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1})} \left[V^\tau(x_{t+1}^\tau)\right] - V(x_{t+1})\right]}_{I(x_{t+1})} \end{aligned}$$

The last step can be derived due to the fact that

$$\mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t) \\ x_t^\tau\sim b_\Delta(\cdot|x_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[x_{t+1}^\tau\right] = \mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t) \\ a_t\sim\pi(\cdot|x_t)}} \left[x_{t+1}^\tau\right]$$

Based on the above iterative equation, we have

$$
\begin{aligned}
I(x_t) &= \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} [V^\tau(x_t^\tau)] - \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi(\cdot|x_t)}} [Q^\tau(x_t^\tau, a_t)] + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t) \\ a_t \sim \pi(\cdot|x_t)}} [I(x_{t+1})] \\
&= \sum_{i=0}^{\infty} \gamma^i \mathop{\mathbb{E}}_{\substack{x_{t+i} \sim \mathcal{P}_\Delta(\cdot|x_{t+i-1}, a_{t+i-1}) \\ a_{t+i-1} \sim \pi(\cdot|x_{t+i-1})}} \left[\mathop{\mathbb{E}}_{x_{t+i}^\tau \sim b_\Delta(\cdot|x_{t+i})} \left[V^\tau(x_{t+i}^\tau) \right] - \mathop{\mathbb{E}}_{\substack{x_{t+i}^\tau \sim b_\Delta(\cdot|x_{t+i}) \\ a_{t+i} \sim \pi(\cdot|x_{t+i})}} \left[Q^\tau(x_{t+i}^\tau, a_{t+i}) \right] \right] \\
&= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{\hat{x}^\tau \sim b_\Delta(\cdot|\hat{x}) \\ a \sim \pi(\cdot|\hat{x}) \\ \hat{x} \sim d^\pi_{x_t}}} [V^\tau(\hat{x}^\tau) - Q^\tau(\hat{x}^\tau, a)]
\end{aligned}
$$

□

Theorem B.3 (Delayed Performance Difference Bound)**.** *For policies π^τ and π, with $\Delta^\tau < \Delta$. Given any $x_t \in \mathcal{X}$, if Q^τ is L_Q-LC, the performance difference between policies can be bounded as follow*

$$
\mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi(\cdot|x_t)}} [V^\tau(x_t^\tau) - Q^\tau(x_t^\tau, a_t)] \leq L_Q \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} [\mathcal{W}(\pi^\tau(\cdot|x_t^\tau) || \pi(\cdot|x_t))]
$$

Proof. We can rewrite the left hand side and apply the Proposition B.1 to get the result

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi(\cdot|x_t)}} [V^\tau(x_t^\tau) - Q^\tau(x_t^\tau, a_t)] \\
=& \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi^\tau(\cdot|x_t^\tau)}} [Q^\tau(x_t^\tau, a_t)] - \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi(\cdot|x_t)}} [Q^\tau(x_t^\tau, a_t)] \\
=& \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi^\tau(\cdot|x_t^\tau)}} [Q^\tau(x_t^\tau, a_t)] - \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi(\cdot|x_t)}} [Q^\tau(x_t^\tau, a_t)] \\
=& \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} \left[\mathop{\mathbb{E}}_{a_t \sim \pi^\tau(\cdot|x_t^\tau)} [Q^\tau(x_t^\tau, a_t)] - \mathop{\mathbb{E}}_{a_t \sim \pi(\cdot|x_t)} [Q^\tau(x_t^\tau, a_t)] \right] \\
\leq& L_Q \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} [\mathcal{W}(\pi^\tau(\cdot|x_t^\tau) || \pi(\cdot|x_t))]
\end{aligned}
$$

□

Theorem B.4 (Delayed Q-value Difference Bound)**.** *For policies π and π^τ, with $\Delta^\tau < \Delta$. Given any $x_t \in \mathcal{X}$, if Q^τ is L_Q-LC, the corresponding Q-value difference can be bounded as follow*

$$
\mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|x_t) \\ x_t^\tau \sim b_\Delta(\cdot|x_t)}} [Q^\tau(x_t^\tau, a_t) - Q(x_t, a_t)] \leq \frac{\gamma L_Q}{1-\gamma} \mathop{\mathbb{E}}_{\substack{\hat{x}^\tau \sim b_\Delta(\cdot|\hat{x}) \hat{x} \sim d^\pi_{x_{t+1}} \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t) \\ a_t \sim \pi(\cdot|x_t)}} [\mathcal{W}_1(\pi^\tau(\cdot|\hat{x}^\tau) || \pi(\cdot|\hat{x}))]
$$

Specially, for optimal policies $\pi_{(*)}$ and $\pi^\tau_{(*)}$, if $Q^\tau_{(*)}$ is L_Q-LC, the corresponding optimal Q-value difference can be bounded as follow

$$
\left\| \mathop{\mathbb{E}}_{x_t^\tau \sim b_\Delta(\cdot|x_t)} \left[Q^\tau_{(*)}(x_t^\tau, a_t) \right] - Q_{(*)}(x_t, a_t) \right\|_\infty \leq \frac{\gamma^2 L_Q}{(1-\gamma)^2} \mathop{\mathbb{E}}_{\substack{\hat{x}^\tau \sim b_\Delta(\cdot|\hat{x}) \\ \hat{x} \sim d^\pi_{x_{t+1}} \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t) \\ a_t \sim \pi_{(*)}(\cdot|x_t)}} \left[\mathcal{W}\left(\pi^\tau_{(*)}(\cdot|\hat{x}^\tau) \middle\| \pi_{(*)}(\cdot|\hat{x}) \right) \right]
$$

Proof. For policies π and π^τ, we can rewrite the left hand side as

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{a_t\sim\pi(\cdot|x_t)\\ x_t^\tau\sim b_\Delta(\cdot|x_t)}}\left[Q^\tau(x_t^\tau,a_t)-Q(x_t,a_t)\right]\\
=&\mathop{\mathbb{E}}_{\substack{a_t\sim\pi(\cdot|x_t)\\ x_t^\tau\sim b_\Delta(\cdot|x_t)}}\left[\mathcal{R}_\tau(x_t^\tau,a_t)+\gamma\mathop{\mathbb{E}}_{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t)}\left[V^\tau(x_{t+1}^\tau)\right]\right]-\mathop{\mathbb{E}}_{a_t\sim\pi(\cdot|x_t)}\left[\mathcal{R}_\Delta(x_t,a_t)+\gamma\mathop{\mathbb{E}}_{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)}\left[V(x_{t+1})\right]\right]\\
=&\gamma\mathop{\mathbb{E}}_{a_t\sim\pi(\cdot|x_t)}\left[\mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau\sim\mathcal{P}_{\Delta^\tau}(\cdot|x_t^\tau,a_t)\\ x_t^\tau\sim b_\Delta(\cdot|x_t)}}\left[V^\tau(x_{t+1}^\tau)\right]-\mathop{\mathbb{E}}_{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)}\left[V(x_{t+1})\right]\right]\\
=&\gamma\mathop{\mathbb{E}}_{\substack{x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)\\ a_t\sim\pi(\cdot|x_t)}}\underbrace{\left[\mathop{\mathbb{E}}_{x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1})}\left[V^\tau(x_{t+1}^\tau)\right]-V(x_{t+1})\right]}_{I(x_{t+1})}\\
\leq&\frac{\gamma L_Q}{1-\gamma}\mathop{\mathbb{E}}_{\substack{\hat{x}^\tau\sim b_\Delta(\cdot|\hat{x})\\ \hat{x}\sim d^\pi_{x_{t+1}}\\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)\\ a_t\sim\pi(\cdot|x_t)}}\left[\mathcal{W}(\pi^\tau(\cdot|\hat{x}^\tau)||\pi(\cdot|\hat{x}))\right]
\end{aligned}
$$

The last two steps are derived via applying Lemma B.2 and Theorem B.3 Then based on the above result, for optimal policies $\pi_{(*)}$ and $\pi^\tau_{(*)}$, we have

$$
\begin{aligned}
&\left|\left|\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{(*)}(x_t^\tau,a_t)\right]-Q_{(*)}(x_t,a_t)\right|\right|_\infty\\
=&\left|\left|\gamma\mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1})\\ x_{t+1}\sim\mathcal{P}_\Delta(x_t,a_t)}}\left[\max_{a_{t+1}}Q^\tau_{(*)}(x_{t+1}^\tau,a_{t+1})\right]-\gamma\mathop{\mathbb{E}}_{x_{t+1}\sim\mathcal{P}_\Delta(x_t,a_t)}\left[\max_{a_{t+1}}Q_{(*)}(x_{t+1},a_{t+1})\right]\right|\right|_\infty\\
=&||\gamma\mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1})\\ x_{t+1}\sim\mathcal{P}_\Delta(x_t,a_t)}}\left[\max_{a_{t+1}}Q^\tau_{(*)}(x_{t+1}^\tau,a_{t+1})\right]-\gamma\mathop{\mathbb{E}}_{\substack{a_{t+1}\sim\pi_{(*)}(\cdot|x_{t+1})\\ x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1})\\ x_{t+1}\sim\mathcal{P}_\Delta(x_t,a_t)}}\left[Q^\tau_{(*)}(x_{t+1}^\tau,a_{t+1})\right]\\
&+\gamma\mathop{\mathbb{E}}_{\substack{a_{t+1}\sim\pi_{(*)}(\cdot|x_{t+1})\\ x_{t+1}^\tau\sim b_\Delta(\cdot|x_{t+1})\\ x_{t+1}\sim\mathcal{P}_\Delta(x_t,a_t)}}\left[Q^\tau_{(*)}(x_{t+1}^\tau,a_{t+1})\right]-\gamma\mathop{\mathbb{E}}_{x_{t+1}\sim\mathcal{P}_\Delta(x_t,a_t)}\left[\max_{a_{t+1}}Q_{(*)}(x_{t+1},a_{t+1})\right]||_\infty\\
\leq&\gamma\left|\left|\mathop{\mathbb{E}}_{x_t^\tau\sim b_\Delta(\cdot|x_t)}\left[Q^\tau_{(*)}(x_t^\tau,a_t)\right]-Q_{(*)}(x_t,a_t)\right|\right|_\infty+\gamma\frac{\gamma L_Q}{1-\gamma}\mathop{\mathbb{E}}_{\substack{\hat{x}^\tau\sim b_\Delta(\cdot|\hat{x})\\ \hat{x}\sim d^\pi_{x_{t+1}}\\ x_{t+1}\sim\mathcal{P}_\Delta(\cdot|x_t,a_t)\\ a_t\sim\pi_{(*)}(\cdot|x_t)}}\left[\mathcal{W}\left(\pi^\tau_{(*)}(\cdot|\hat{x}^\tau)\middle|\middle|\pi_{(*)}(\cdot|\hat{x})\right)\right]
\end{aligned}
$$

□

B.2. Convergence Analysis

Here, we recall that we assume the action-space $\mathcal{A}$ is finite where $|\mathcal{A}|<\infty$. When the assumption is satisfied, for any $x\in\mathcal{X}$, the $L1$-Wasserstein distance between two delayed policies π and π^τ becomes the l_1 distance and it is bounded.

$$\mathop{\mathbb{E}}_{x^\tau\sim b_\Delta(\cdot|x)}\left[\mathcal{W}\left(\pi^\tau(\cdot|x^\tau)||\pi(\cdot|x)\right)\right]<\infty \tag{7}$$

Furthermore, the entropy of policy π is also bounded

$$\mathcal{H}\left(\pi(\cdot|x)\right)<\infty \tag{8}$$

Theorem B.5 (AD-VI Convergence Guarantee)**.** *Consider the bellman operator $\mathcal{T}$ in Eq. (3) and the initial Q-function $Q_{(0)}$: $\mathcal{X} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define a sequence $\{Q_{(k)}\}_{k=0}^{\infty}$ where $Q_{(k+1)} = \mathcal{T} Q_{(k)}$. As $k \to \infty$, $Q_{(k)}$ will converge to the fixed point $Q_{(\approx)}$ with Q^{τ} converges to $Q^{\tau}_{(*)}$. And for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$, we have*

$$Q_{(\approx)}(x_t, a_t) = \mathop{\mathbb{E}}_{x_t^{\tau} \sim b_{\Delta}(\cdot|x_t)} \left[Q^{\tau}_{(*)}(x_t^{\tau}, a_t) \right]$$

Proof. The update rule of the bellman operator $\mathcal{T}$ is as follows

$$Q(x_t, a_t) \leftarrow \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^{\tau} \sim b_{\Delta}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[Q^{\tau}(x_{t+1}^{\tau}, \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1})) \right]$$

and the right hand side can be written as

$$\underbrace{\mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^{\tau} \sim b_{\Delta}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[Q^{\tau}(x_{t+1}^{\tau}, \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1})) - \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) \right]}_{r^{\tau}(x_t, a_t)} + \gamma \mathop{\mathbb{E}}_{x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)} \max_{a_{t+1}} Q(x_{t+1}, a_{t+1})$$

Next, we prove that $r^{\tau}(x_t, a_t)$ is bounded

$$\begin{aligned}
r^{\tau}(x_t, a_t) &= \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^{\tau} \sim b_{\Delta}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[Q^{\tau}(x_{t+1}^{\tau}, \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1})) - \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) \right] \\
&= \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^{\tau} \sim b_{\Delta}(\cdot|x_{t+1}) \\ a_{t+1} = \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[Q^{\tau}(x_{t+1}^{\tau}, a_{t+1}) - \max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) \right] \\
&\leq \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \frac{\gamma L_Q}{1 - \gamma} \mathop{\mathbb{E}}_{\substack{\hat{x}^{\tau} \sim b_{\Delta}(\cdot|\hat{x}) \\ \hat{x} \sim d^{\pi}_{x_{t+2}} \\ x_{t+2} \sim \mathcal{P}_{\Delta}(\cdot|x_{t+1}, a_{t+1}) \\ a_{t+1} = \arg\max_{a_{t+1}} Q(x_{t+1}, a_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[\mathcal{W}\left(\pi^{\tau}(\cdot|\hat{x}^{\tau}) || \pi(\cdot|\hat{x}) \right) \right] \\
&< \infty
\end{aligned}$$

The last two steps are derived via applying Lemma B.4 and Eq. (7), respectively. Then, we can get the property of convergence by applying original VI (Sutton & Barto, 2018), and the convergence is related to the Q^{τ}.

For the Q^{τ}, we know that it converges to $Q^{\tau}_{(*)}$ as it is updated by the original VI rule, and it satisfies that

$$\mathop{\mathbb{E}}_{x_t^{\tau} \sim b_{\Delta}(\cdot|x_t)} \left[Q^{\tau}_{(*)}(x_t^{\tau}, a_t) \right] = \mathcal{R}_{\tau}(x_t^{\tau}, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^{\tau} \sim b_{\Delta}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[\max_{a_{t+1}} Q^{\tau}_{(*)}(x_{t+1}^{\tau}, a_{t+1}) \right]$$

Without loss of generality, we can assume that the update of $Q_{(k)}$ is based on the $Q^{\tau}_{(*)}$, then the converged fixed point of $Q_{(k)}(k \to \infty)$, denoted as $Q_{(\approx)}$, satisfies the equation as followed:

$$Q_{(\approx)}(x_t, a_t) = \mathcal{R}_{\Delta}(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^{\tau} \sim b_{\Delta}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_{\Delta}(\cdot|x_t, a_t)}} \left[Q^{\tau}_{(*)}(x_{t+1}^{\tau}, \arg\max_{a_{t+1}} Q_{(\approx)}(x_{t+1}, a_{t+1})) \right]$$

then we can get the fixed point

$$
\begin{aligned}
Q_{(\approx)}(x_t, a_t) &= \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau_{(*)}(x^\tau_{t+1}, \arg\max_{a_{t+1}} Q_{(\approx)}(x_{t+1}, a_{t+1})) \right] \\
&= \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau_{(*)}(x^\tau_{t+1}, \arg\max_{a_{t+1}} \mathop{\mathbb{E}}_{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1})} \left[Q^\tau_{(*)}(x^\tau_{t+1}, a_{t+1}) \right]) \right] \\
&= \mathop{\mathbb{E}}_{x^\tau_t \sim b_\Delta(\cdot|x_t)} \left[\mathcal{R}_\tau(x^\tau_t, a_t) \right] + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[\max_{a_{t+1}} Q^\tau_{(*)}(x^\tau_{t+1}, a_{t+1}) \right] \\
&= \mathop{\mathbb{E}}_{x^\tau_t \sim b_\Delta(\cdot|x_t)} \left[Q^\tau_{(*)}(x^\tau_t, a_t) \right]
\end{aligned}
$$

□

Lemma B.6 (Policy Evaluation Convergence Guarantee)**.** *Consider the soft bellman operator $\mathcal{T}^\pi$ in Eq. (4) and the initial delayed Q-value function $Q_{(0)}$: $\mathcal{X} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define a sequence $\{Q_{(k)}\}_{k=0}^{\infty}$ where $Q_{(k+1)} = \mathcal{T}^\pi Q_{(k)}$. Then for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$, as $k \to \infty$, $Q_{(k)}(x_t, a_t)$ will converge to the fixed point*

$$
\mathop{\mathbb{E}}_{x^\tau_t \sim b_\Delta(\cdot|x_t)} \left[Q^\tau_{soft}(x^\tau_t, a_t) \right] - \log \pi(a_t|x_t)
$$

Proof. The update rule of the soft bellman operator $\mathcal{T}^\pi$ is as follows

$$
Q(x_t, a_t) \leftarrow \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x^\tau_{t+1}, a_{t+1}) - \log \pi(a_{t+1}|x_{t+1}) \right]
$$

Similar to AD-VI, we rewrite the right-hand side as

$$
\underbrace{\mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x^\tau_{t+1}, a_{t+1}) - Q(x_{t+1}, a_{t+1}) - \log \pi(a_{t+1}|x_{t+1}) \right]}_{r^\pi(x_t, a_t)} + \gamma \mathop{\mathbb{E}}_{\substack{a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q(x_{t+1}, a_{t+1}) \right]
$$

Similarly, we prove that $r^\pi(x_t, a_t)$ is bounded

$$
\begin{aligned}
r^\pi(x_t, a_t) &= \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x^\tau_{t+1}, a_{t+1}) - Q(x_{t+1}, a_{t+1}) - \log \pi(a_{t+1}|x_{t+1}) \right] \\
&= \mathcal{R}_\Delta(x_t, a_t) + \gamma \mathop{\mathbb{E}}_{\substack{x^\tau_{t+1} \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x^\tau_{t+1}, a_{t+1}) - Q(x_{t+1}, a_{t+1}) \right] - \gamma \log \mathcal{H}(\pi(\cdot|x_{t+1})) \\
&\leq \mathcal{R}_\Delta(x_t, a_t) + \frac{\gamma L_Q}{1-\gamma} \mathop{\mathbb{E}}_{\substack{\hat{x}^\tau \sim b_\Delta(\cdot|\hat{x}) \\ \hat{x} \sim d^\pi_{x_{t+2}} \\ x_{t+2} \sim \mathcal{P}_\Delta(\cdot|x_{t+1}, a_{t+1}) \\ a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[\mathcal{W}(\pi^\tau(\cdot|\hat{x}^\tau) || \pi(\cdot|\hat{x})) \right] - \gamma \log \mathcal{H}(\pi(\cdot|x_{t+1})) \\
&< \infty
\end{aligned}
$$

The last step can be derived due to Eq. (7) and Eq. (8). Then, we can get the result from the original policy evaluation (Sutton & Barto, 2018). Similarly, without loss of generality, we can assume that Q^τ has converged to $Q^\tau_{(soft)}$, and can get the fixed point easily:

$$
\mathop{\mathbb{E}}_{x^\tau_t \sim b_\Delta(\cdot|x_t)} \left[Q^\tau_{(soft)}(x^\tau_t, a_t) \right] - \log \pi(a_t|x_t)
$$

□

Lemma B.7 (Soft Policy Improvement Guarantee)**.** *Consider the policy update rule in Eq. (5), and let π_{old}, π_{new} be the old policy and new policy improved from the old one respectively. Then for any $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$, we have $Q_{old}(x_t, a_t) \leq Q_{new}(x_t, a_t)$.*

Proof. As

$$\pi_{new} = \arg\min_{\pi' \in \Pi} \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi'(\cdot|x_t)}} [\log \pi'(a_t|x_t) - Q^\tau(x_t^\tau, a_t)]$$

So

$$\mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi_{new}(\cdot|x_t)}} [\log \pi_{new}(a_t|x_t) - Q^\tau(x_t^\tau, a_t)] \leq \mathop{\mathbb{E}}_{\substack{x_t^\tau \sim b_\Delta(\cdot|x_t) \\ a_t \sim \pi_{old}(\cdot|x_t)}} [\log \pi_{old}(a_t|x_t) - Q^\tau(x_t^\tau, a_t)]$$

Then, we have

$$\begin{aligned}
Q_{old}(x_t, a_t) &= r_t + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi_{old}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x_{t+1}^\tau, a_{t+1}) - \log \pi_{old}(a_{t+1}|x_{t+1})\right] \\
&\leq r_t + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi_{new}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau(x_{t+1}^\tau, a_{t+1}) - \log \pi_{new}(a_{t+1}|x_{t+1})\right] \\
&= Q_{new}(x_t, a_t)
\end{aligned}$$

□

Theorem B.8 (Soft Policy Iteration Convergence Guarantee)**.** *Applying policy evaluation in Eq. (4) and policy improvement in Eq. (5) repeatedly to any given policy $\pi \in \Pi$, it converges to $\pi_{(\approx)}$ such that $Q^\pi(x_t, a_t) \leq Q^\pi_{(\approx)}(x_t, a_t)$ for any $\pi \in \Pi$, $(x_t, a_t) \in \mathcal{X} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.*

Proof. Based on Lemma B.6 and Lemma B.7, we have

$$\begin{aligned}
Q^\pi(x_t, a_t) &= r_t + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau_{(soft)}(x_{t+1}^\tau, a_{t+1}) - \log \pi(a_{t+1}|x_{t+1})\right] \\
&\leq r_t + \gamma \mathop{\mathbb{E}}_{\substack{x_{t+1}^\tau \sim b_\Delta(\cdot|x_{t+1}) \\ a_{t+1} \sim \pi_{(\approx)}(\cdot|x_{t+1}) \\ x_{t+1} \sim \mathcal{P}_\Delta(\cdot|x_t, a_t)}} \left[Q^\tau_{(soft)}(x_{t+1}^\tau, a_{t+1}) - \log \pi_{(\approx)}(a_{t+1}|x_{t+1})\right] \\
&= Q^\pi_{(\approx)}(x_t, a_t)
\end{aligned}$$

□

C. Stochastic MDP Case: Selection of Auxiliary Delays Δ^τ

We give the following Theorem C.2 to exemplify that the naive selection (e.g. $\Delta^\tau = 0$) might cause a larger performance gap and potential approximation error.

First of all, we introduce a stochastic MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho \rangle$ (Liotet et al., 2022) where

- $\mathcal{S} = \mathbb{R}$
- $\mathcal{A} = \mathbb{R}$
- $\mathcal{P}(s'|s,a) = \mathcal{N}(s + \frac{a}{L_\pi}, \sigma^2)$ which means that $s' = s + \frac{a}{L_\pi} + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- $\mathcal{R}(s,a) = -L_Q L_\pi |s + \frac{a}{L_\pi}|$

Lemma C.1 (value function upper bound (Liotet et al., 2022))**.** *Let $L_\pi, L_Q > 0$, in the MDP defined above with a L_π-LC optimal policy and L_Q-LC optimal value function, given $\Delta > 0$, for any Δ-delayed policy π and any augmented state $x \in \mathcal{X}$, it's value function V^π has following upper bound*

$$\begin{aligned} V^\pi(x) &\leq -\frac{L_Q L_\pi}{1-\gamma} \frac{\sqrt{2\Delta}}{\sqrt{\pi}} \sigma \\ &= V^\pi_{(*)}(x) \end{aligned} \tag{9}$$

where $V^\pi_{(*)}(x)$ is the value function of the optimal Δ-delayed policy $\pi_{(*)}$.

Then, we can extend Lemma C.1 in the context of our AD-RL. Given any delay Δ and auxiliary delays Δ^τ ($<\Delta$), the corresponding optimal value functions are $V_{(*)}$ and $V^\tau_{(*)}$, respectively. For $V_{(*)}$ and $V^\tau_{(*)}$, we can derive the performance difference in the following Theorem.

Theorem C.2. *For optimal policies $\pi_{(*)}$ and $\pi^\tau_{(*)}$, for any $x_t \in \mathcal{X}$, their corresponding performance difference $I(x_t)$ is*

$$I(x_t) = \frac{L_Q L_\pi}{1-\gamma} \sigma \frac{\sqrt{2}}{\sqrt{\pi}} \left[\sqrt{\Delta} - \sqrt{\Delta^\tau} \right]$$

Proof. By applying Lemma C.1, we can get that

$$\begin{aligned} V_{(*)}(x_t) &= \mathop{\mathbb{E}}_{a_t \sim \pi_{(*)}(\cdot|x_t)} [Q_{(*)}(x_t, a_t)] \\ &= -\frac{L_Q L_\pi}{1-\gamma} \frac{\sqrt{2\Delta}}{\sqrt{\pi}} \sigma \end{aligned}$$

and

$$\begin{aligned} \mathop{\mathbb{E}}_{x^\tau_t \sim b_\Delta(\cdot|x_t)} \left[V^\tau_{(*)}(x^\tau_t) \right] &= \mathop{\mathbb{E}}_{\substack{a_t \sim \pi^\tau_{(*)}(\cdot|x^\tau_t) \\ x^\tau_t \sim b_\Delta(\cdot|x_t)}} \left[Q^\tau_{(*)}(x^\tau_t, a_t) \right] \\ &= -\frac{L_Q L_\pi}{1-\gamma} \frac{\sqrt{2\Delta^\tau}}{\sqrt{\pi}} \sigma \end{aligned}$$

then based on Lemma 5.2, we can derive that

$$\begin{aligned} I(x_t) &= \mathop{\mathbb{E}}_{x^\tau_t \sim b_\Delta(\cdot|x_t)} [V^\tau(x^\tau_t)] - V(x_t) \\ &= \frac{L_Q L_\pi}{1-\gamma} \sigma \frac{\sqrt{2}}{\sqrt{\pi}} \left[\sqrt{\Delta} - \sqrt{\Delta^\tau} \right] \end{aligned}$$

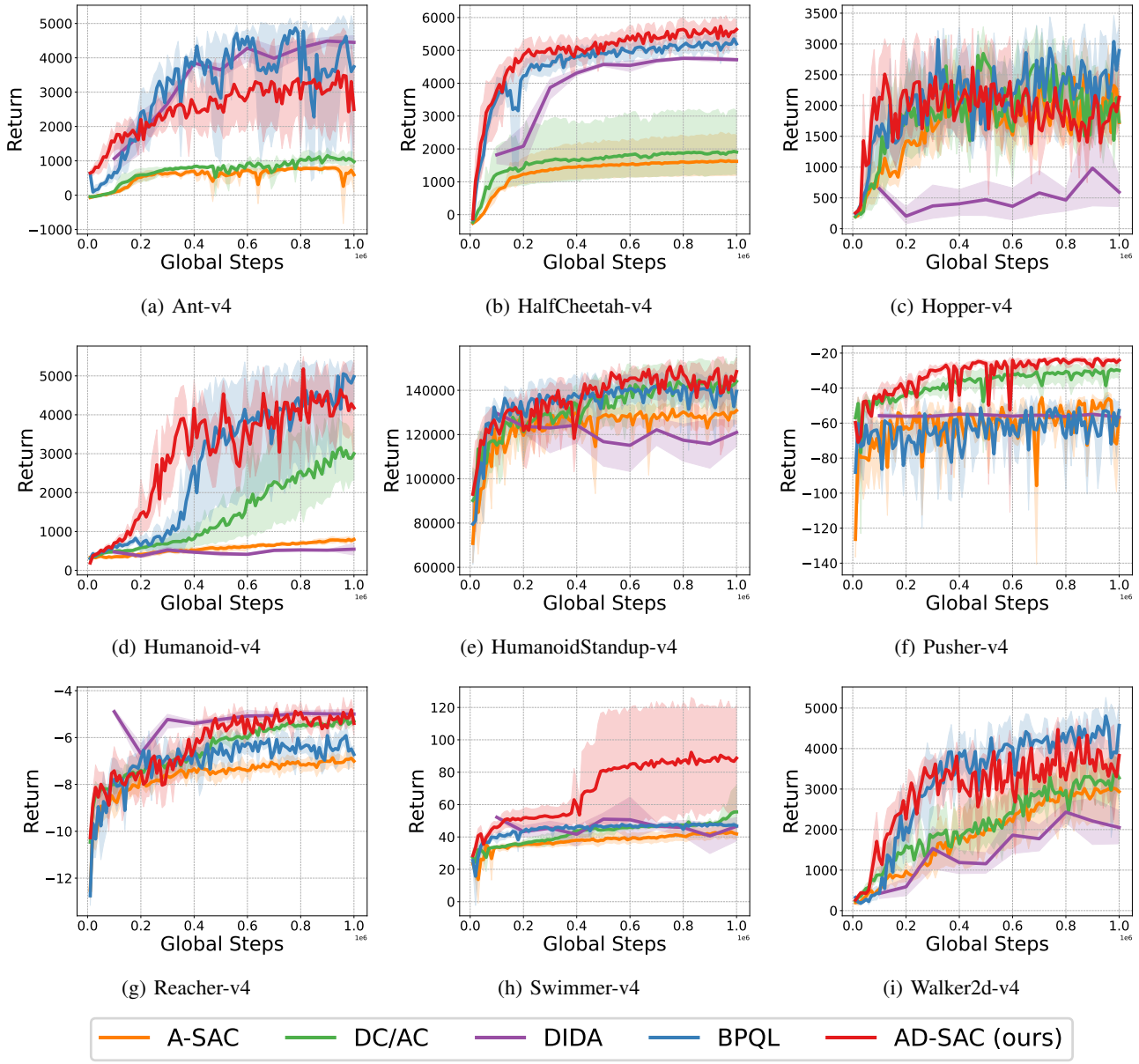
□

From Theorem C.2, we can observe that when the difference between delays (Δ^τ and Δ) increases, the performance difference becomes larger. In other words, if selecting a extremely short Δ^τ (e.g., $\Delta^\tau = 0$) for a long Δ, the value difference between $Q^\tau_{(*)}$ and $Q_{(*)}$ might be enlarged, and introducing potential estimation bias. Based on the analysis presented above, it actually exists a trade-off between sample efficiency (shorter Δ^τ is better) and approximation error (Δ^τ is closer to Δ).

D. MuJoCo: Additional Experimental Results

Table 5. Results of MuJoCo tasks with 5, 25 and 50 delays for 1M global time-steps. Each method was evaluated with 10 trials and is shown with the standard deviation denoted by $\pm$. The best performance is in blue.

Task	Delay-free SAC	Random	Delays	A-SAC	DC/AC	DIDA	BPQL	AD-SAC (ours)
Ant-v4	5053.30	-123.88	5	$0.18_{\pm 0.01}$	$0.25_{\pm 0.05}$	$0.89_{\pm 0.03}$	$0.96_{\pm 0.03}$	$0.72_{\pm 0.25}$
			25	$0.07_{\pm 0.07}$	$0.19_{\pm 0.02}$	$0.29_{\pm 0.07}$	$0.57_{\pm 0.11}$	$0.66_{\pm 0.04}$
			50	$0.02_{\pm 0.04}$	$0.19_{\pm 0.02}$	$0.19_{\pm 0.05}$	$0.38_{\pm 0.07}$	$0.48_{\pm 0.06}$
HalfCheetah-v4	5322.74	-304.29	5	$0.35_{\pm 0.15}$	$0.40_{\pm 0.23}$	$0.90_{\pm 0.01}$	$1.00_{\pm 0.06}$	$1.07_{\pm 0.06}$
			25	$0.04_{\pm 0.01}$	$0.16_{\pm 0.07}$	$0.12_{\pm 0.03}$	$0.87_{\pm 0.04}$	$0.71_{\pm 0.12}$
			50	$0.12_{\pm 0.17}$	$0.12_{\pm 0.13}$	$0.15_{\pm 0.03}$	$0.73_{\pm 0.17}$	$0.74_{\pm 0.10}$
Hopper-v4	2455.18	12.93	5	$1.02_{\pm 0.28}$	$1.16_{\pm 0.25}$	$0.40_{\pm 0.40}$	$1.25_{\pm 0.09}$	$1.07_{\pm 0.30}$
			25	$0.13_{\pm 0.04}$	$0.19_{\pm 0.04}$	$0.27_{\pm 0.08}$	$1.21_{\pm 0.18}$	$0.86_{\pm 0.25}$
			50	$0.04_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.09_{\pm 0.01}$	$0.71_{\pm 0.13}$	$0.72_{\pm 0.03}$
Humanoid-v4	5269.34	135.47	5	$0.13_{\pm 0.02}$	$0.59_{\pm 0.17}$	$0.08_{\pm 0.04}$	$0.96_{\pm 0.05}$	$0.98_{\pm 0.07}$
			25	$0.05_{\pm 0.01}$	$0.04_{\pm 0.01}$	$0.07_{\pm 0.00}$	$0.12_{\pm 0.01}$	$0.25_{\pm 0.16}$
			50	$0.04_{\pm 0.01}$	$0.03_{\pm 0.01}$	$0.07_{\pm 0.00}$	$0.08_{\pm 0.01}$	$0.10_{\pm 0.01}$
HumanoidStandup-v4	129827.70	35743.26	5	$1.02_{\pm 0.08}$	$1.16_{\pm 0.12}$	$1.00_{\pm 0.00}$	$1.13_{\pm 0.07}$	$1.22_{\pm 0.03}$
			25	$0.97_{\pm 0.09}$	$1.03_{\pm 0.03}$	$0.97_{\pm 0.02}$	$1.09_{\pm 0.05}$	$1.15_{\pm 0.08}$
			50	$0.90_{\pm 0.02}$	$1.02_{\pm 0.07}$	$0.89_{\pm 0.06}$	$1.06_{\pm 0.04}$	$1.12_{\pm 0.02}$
Pusher-v4	-56.03	-148.51	5	$1.11_{\pm 0.02}$	$1.29_{\pm 0.05}$	$1.01_{\pm 0.01}$	$1.06_{\pm 0.08}$	$1.36_{\pm 0.01}$
			25	$0.49_{\pm 0.32}$	$1.12_{\pm 0.02}$	$1.04_{\pm 0.01}$	$1.07_{\pm 0.06}$	$1.29_{\pm 0.03}$
			50	$0.00_{\pm 0.05}$	$1.13_{\pm 0.01}$	$1.04_{\pm 0.02}$	$1.09_{\pm 0.05}$	$1.23_{\pm 0.02}$
Reacher-v4	-5.89	-44.11	5	$0.97_{\pm 0.01}$	$1.02_{\pm 0.00}$	$1.03_{\pm 0.00}$	$1.00_{\pm 0.01}$	$1.03_{\pm 0.01}$
			25	$0.96_{\pm 0.02}$	$1.00_{\pm 0.00}$	$0.98_{\pm 0.01}$	$0.87_{\pm 0.05}$	$0.98_{\pm 0.02}$
			50	$0.86_{\pm 0.02}$	$0.89_{\pm 0.01}$	$0.93_{\pm 0.02}$	$0.90_{\pm 0.02}$	$0.91_{\pm 0.03}$
Swimmer-v4	49.41	-3.18	5	$0.88_{\pm 0.09}$	$1.11_{\pm 0.30}$	$1.05_{\pm 0.01}$	$0.97_{\pm 0.02}$	$1.82_{\pm 0.78}$
			25	$0.72_{\pm 0.02}$	$0.78_{\pm 0.12}$	$0.93_{\pm 0.09}$	$1.36_{\pm 0.56}$	$2.52_{\pm 0.40}$
			50	$0.69_{\pm 0.04}$	$0.68_{\pm 0.06}$	$0.87_{\pm 0.03}$	$2.23_{\pm 0.55}$	$2.71_{\pm 0.14}$
Walker2d-v4	3999.54	1.48	5	$0.76_{\pm 0.21}$	$0.85_{\pm 0.12}$	$0.61_{\pm 0.07}$	$1.20_{\pm 0.11}$	$1.12_{\pm 0.09}$
			25	$0.12_{\pm 0.02}$	$0.26_{\pm 0.08}$	$0.10_{\pm 0.02}$	$0.59_{\pm 0.30}$	$0.72_{\pm 0.11}$
			50	$0.11_{\pm 0.02}$	$0.11_{\pm 0.02}$	$0.08_{\pm 0.01}$	$0.23_{\pm 0.10}$	$0.00_{\pm 0.11}$

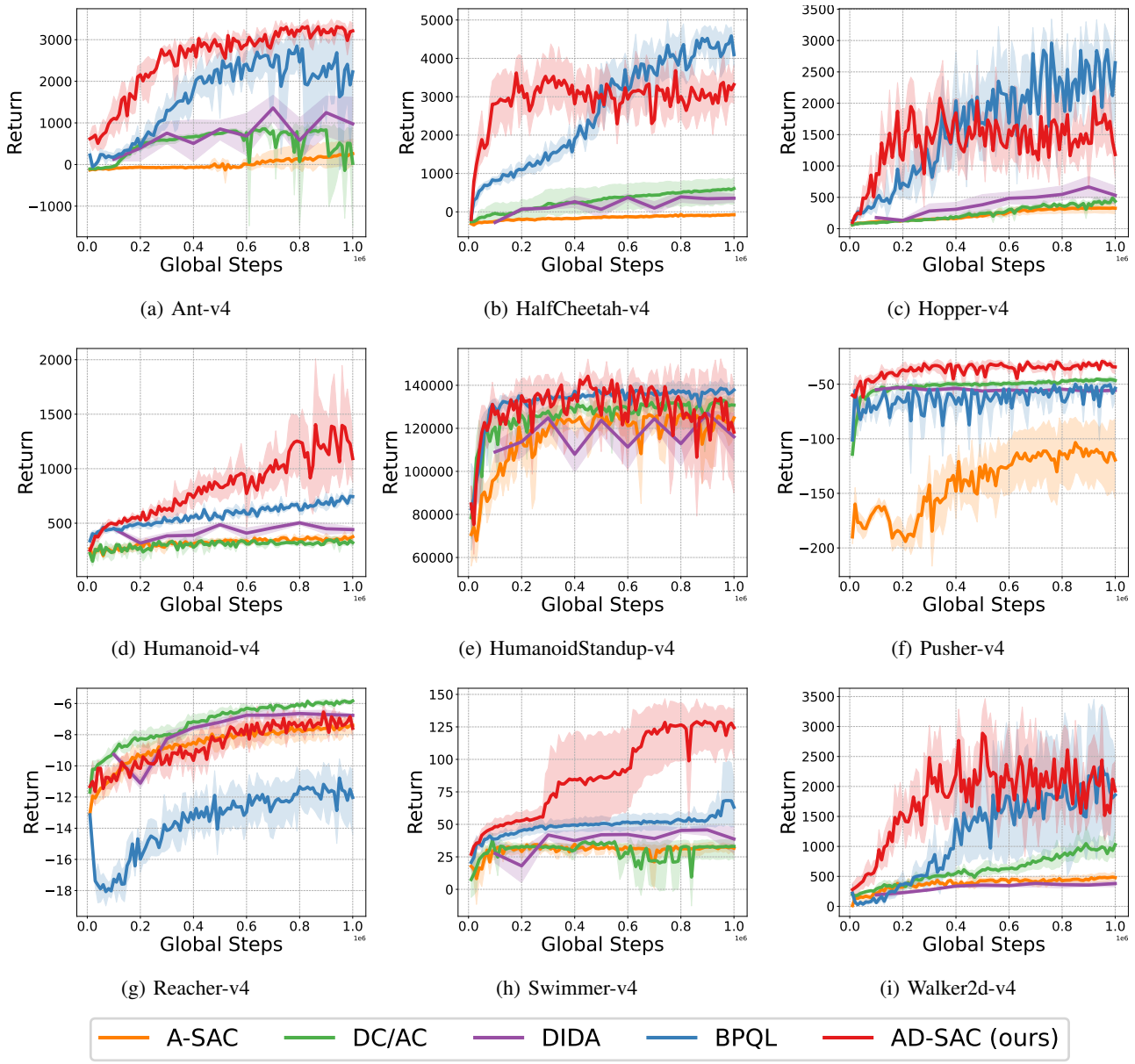


(a) Ant-v4 (b) HalfCheetah-v4 (c) Hopper-v4

(d) Humanoid-v4 (e) HumanoidStandup-v4 (f) Pusher-v4

(g) Reacher-v4 (h) Swimmer-v4 (i) Walker2d-v4

Figure 4. Results of MuJoCo tasks with 5 delays for learning curves. The shaded areas represented the standard deviation (10 seeds).

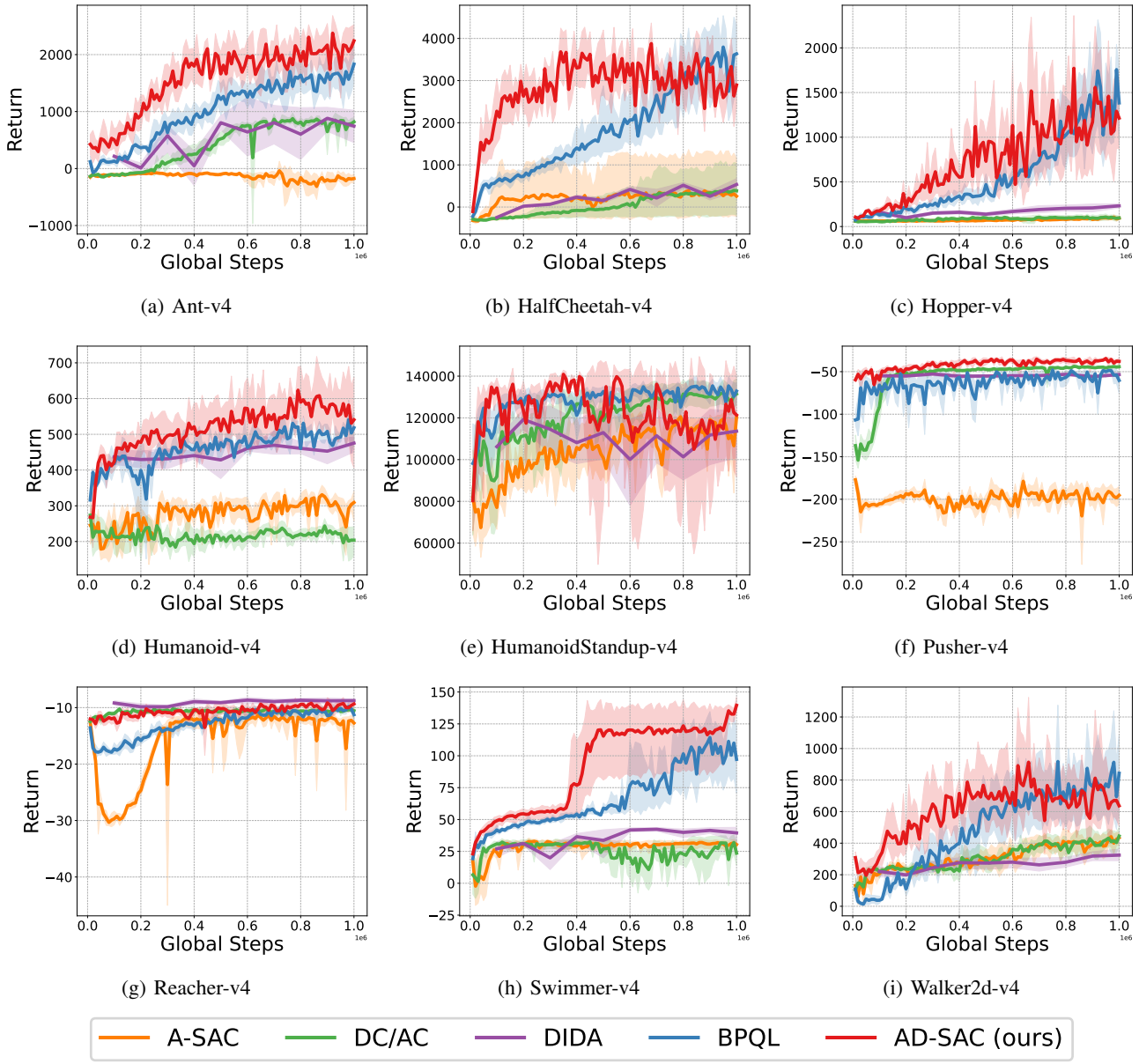


(a) Ant-v4 (b) HalfCheetah-v4 (c) Hopper-v4

(d) Humanoid-v4 (e) HumanoidStandup-v4 (f) Pusher-v4

(g) Reacher-v4 (h) Swimmer-v4 (i) Walker2d-v4

Figure 5. Results of MuJoCo tasks with 25 delays for learning curves. The shaded areas represented the standard deviation (10 seeds).



(a) Ant-v4 (b) HalfCheetah-v4 (c) Hopper-v4

(d) Humanoid-v4 (e) HumanoidStandup-v4 (f) Pusher-v4

(g) Reacher-v4 (h) Swimmer-v4 (i) Walker2d-v4

Figure 6. Results of MuJoCo tasks with 50 delays for learning curves. The shaded areas represented the standard deviation (10 seeds).