

# REVISITING OVERESTIMATION BIAS OF Q-LEARNING: BREAKING BIAS PROPAGATION CHAINS DOES WELL

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

This paper revisits the overestimation bias of Q-learning from a new perspective, i.e., the breaking bias propagation chains. We make five-fold contributions. First, we analyze the estimation bias propagation chains of Q-learning, and find that the bias propagated from previous steps dominates the maximum Q-value estimation bias and slows the convergence speed, instead of the current bias. Second, we propose a novel positive-negative bias alternating algorithm called Alternating Q-learning (AQ). It breaks the unidirectional estimation bias propagation chains via alternately executing Q-learning and Double Q-learning. We show theoretically that there exist two suitable alternating parameters to eliminate the propagation bias. Third, we design an adaptive alternating strategy for AQ, obtaining Adaptive Alternating Q-learning (AdaAQ). It applies a softmax strategy with the absolute value of TD error to choose Q-learning or Double Q-learning for each state-action pair. Fourth, we extend AQ and AdaAQ to the large-scale settings with function approximation, i.e., including both discrete- and continuous-action Deep Reinforcement Learning (DRL). Fifth, both discrete- and continuous-action DRL experiments show that our method outperforms several baselines drastically; tabular MDP experiments reveal fundamental insights into why our method can achieve superior performance.

## 1 INTRODUCTION

As one of the most fundamental reinforcement learning algorithms, Q-learning Watkins et al. (1989) has been successfully applied to many real-world applications Zong et al. (2025); Arvanitidis & Alamaniotis (2024); Gao (2024) due to its simplicity and convergence guarantees under some mild assumptions Kearns & Singh (1998). However, Q-learning suffers from overestimation bias Thrun & Schwartz (1993), which can be exacerbated in DRL with nonlinear function approximation Mnih et al. (2015). This issue originates from the fact that the maximum Q-value is obtained by maximizing the stochastic estimations of Q-value. These stochastic estimations are caused by stochastic and unknown reward and state transition functions. Notably, the deadly triad Van Hasselt et al. (2018); Sutton et al. (2018) illustrates that the overestimation bias of Q-learning can be iteratively propagated via bootstrapping. Although this bias propagation phenomenon is well-known, most works Van Hasselt (2013); Peer et al. (2021); Schmitt-Förster & Sutter (2024); Tan et al. (2024c) focus on mitigating the overestimation bias instead of eliminating the bias propagation chains. For example, Double Q-learning Hasselt (2010) removes the overestimation bias via cross-validation, but may lead to underestimation bias. However, this underestimation bias can cause a slower learning speed and a larger performance penalty than the overestimation bias Ren et al. (2021); Li et al. (2023).

This paper provides a new perspective, i.e., the breaking bias propagation chains, to study the overestimation bias of Q-learning. In Section 3.2, we first set the asynchronous Q-value of Bellman optimality equation Silver (2015) as the ground truth. Then, we analyze the maximum Q-value estimation bias of Q-learning as Theorem 1, which consists of the current bias and the propagation bias. Notably, Corollary 1 demonstrates that the propagation bias rather than the current bias dominates the maximum Q-value estimation bias. That is to say, the propagation bias is the primary cause for the slow convergence speed. Example in Section 3.3 further illustrates that the ratio of propagation bias to current bias can be as high as 5 just after running Q-learning for four steps.

Based on the above analysis, this paper proposes a novel algorithm to break the unidirectional estimation bias propagation chains, called Alternating Q-learning. The main idea is to alternate between positive bias algorithms, such as Q-learning, and negative bias algorithms, such as Double Q-learning. Theorem 2 in Section 4.1 demonstrates that there exist two suitable alternating parameters to eliminate the propagation bias. To address the challenge in Alternating Q-learning, i.e., the optimal alternating parameters are unknown in advance and dynamically for different state-action pairs, we propose an adaptive alternating strategy, resulting in Adaptive Alternating Q-learning. It applies a softmax strategy with the absolute value of TD error to determine whether Alternating Q-learning should execute Q-learning or Double Q-learning for each state-action pair. Extensive experiment results show that our method outperforms several baselines drastically in tabular MDP, discrete-action DRL, and continuous-action DRL settings. In summary, this paper studies the overestimation bias of Q-learning from the breaking bias propagation chains perspective, and makes five key contributions as outlined in the abstract.

## 2 RELATED WORK

### 2.1 UNDERESTIMATION BIAS METHODS

Double Q-learning Hasselt (2010); Van Hasselt (2013) is one notable algorithm, which uses cross-validation to decouple the maximum Q-value estimation. This decoupling process is achieved by maintaining two independent Q-tables: one Q-table is used to select the optimal action that attains the maximum Q-value; the other Q-table is used to estimate the Q-value associated with the previously selected optimal action. As a result, Double Q-learning removes the overestimation bias of Q-learning, but may lead to underestimation bias. EBQL Peer et al. (2021) is a natural extension of Double Q-learning to ensembles, and the estimation bias of maximum Q-value is always negative. REDQ Chen et al. (2021) reduces the estimation bias via a minimum operation over multiple random Q-tables, and the default size of random subset is two. But it still maintains a negative bias throughout most rounds of learning.

### 2.2 CONTROL ESTIMATION BIAS METHODS

Weighted Double Q-learning Zhang et al. (2017) is a weighted combination of Q-learning and Double Q-learning, and controls the estimation bias through the weight parameter. Averaged Q-learning Anschel et al. (2017) averages multiple independent Q-tables to reduce the variance of Q-values, and finds that the estimation bias is inversely proportional to the number of Q-tables. With a finite number of Q-tables, the estimation bias of Averaged Q-learning is always positive. Softmax Q-learning Song et al. (2019) demonstrates that the estimation bias is proportional to the hyperparameter of softmax operation. Maxmin Q-learning Lan et al. (2020) uses a minimum operation over multiple independent Q-tables, and finds that the estimation bias is inversely proportional to the number of Q-tables. AdaEQ Wang et al. (2021) adjusts the ensemble size of Maxmin Q-learning with the approximation Q-value error to control the estimation bias, note that this adjustment method relies on the discounted MC return Li (2023). Self-Correcting Q-learning Zhu & Rigotti (2021) builds a self-correcting estimator with the current and last Q-values, and controls the estimation bias via dynamically adjusting the Pearson correlation coefficient between successive iterations. Balanced Q-learning Karimpanal et al. (2023) computes the optimistic and pessimistic biases with the maximum and minimum operations, respectively, and balances them to control the estimation bias via the balancing factor. AEQ Gong et al. (2023) uses the uncertainty of Q-values and the familiarity of sampling trajectories to control the estimation bias. AdaOrder Q-learning Tan et al. (2024c) uses the order statistic of multiple independent Q-tables to control the estimation bias, which can satisfy the fine-grained bias needs for different environments.

The maximum expected Q-value is impossible to compute without the underlying state transition probabilities and reward distributions Ishwaei D et al. (1985). Thus, there exist estimation bias and its propagation via bootstrapping. However, existing methods focus on the current estimation bias, instead of the propagation bias. Different from previous methods, this paper revisits the propagation process of estimation bias in Section 3.2, and finds that even though previous estimation biases can be diluted during propagation, the propagation bias still dominates the maximum Q-value estimation bias and slows the convergence speed.

### 3 PROBLEM ANALYSIS

#### 3.1 OVERESTIMATION BIAS OF Q-LEARNING

We consider an infinite-horizon MDP Chen et al. (2022), where each decision step is indexed by  $t \in \mathbb{N}$ . Let  $\mathcal{S}$  and  $\mathcal{A}$  denote the state space and the action space, respectively. Let  $P(s'|s, a)$  denote the state transition probability under  $(s, a) \in \mathcal{S} \times \mathcal{A}$ , where  $s' \in \mathcal{S}$  is the next state. Let  $R(s, a)$  denote the reward associated with  $(s, a)$ . Let  $s \mapsto \pi(s)$  denote the policy, where  $s \in \mathcal{S}$  and  $\pi(s) \in \mathcal{A}$ . Let  $Q_\pi(s, a)$  denote the expected cumulative discounted reward with discounting factor  $\gamma \in (0, 1)$  as:

$$Q_\pi(s, a) = \mathbb{E} [R(S_t, A_t) + \gamma R(S_{t+1}, A_{t+1}) + \gamma^2 R(S_{t+2}, A_{t+2}) + \dots | S_t = s, A_t = a],$$

where  $S_\kappa$  is generated from  $P(S_\kappa | S_{\kappa-1}, A_{\kappa-1})$  and  $A_\kappa = \pi(S_\kappa)$ ,  $\forall \kappa \geq t+1$ . Note that  $S_\kappa$  and  $A_\kappa$  are the state random variable and the action random variable, respectively. The learning objective is to find the optimal Q-value and the optimal policy. More specifically, the optimal Q-value is  $Q^*(s, a) = \max_{\pi \in \Pi} Q_\pi(s, a)$ , where  $\Pi$  denotes a set of all policy; the optimal policy is  $\pi^*(s) = \arg \max_{a \in \mathcal{A}} Q^*(s, a)$ ,  $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ .

The optimal Q-value is unknown in advance for model-free reinforcement learning. Q-learning maintains one Q-table  $Q_t(s, a)$  in each time step  $t$ , and uses this Q-table to estimate the optimal Q-value. More specifically, at each time step  $t$ , Q-learning uses  $\varepsilon$ -greedy policy Rodrigues Gomes & Kowalczyk (2009) with  $\arg \max_{a \in \mathcal{A}} Q_t(s, a)$ , i.e.,  $\varepsilon \in (0, 1)$ , to interact with the environment, obtains the sample data  $\{s, a, R(s, a), s'\}$  and updates the Q-table as:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \left[ R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t(s', a') - Q_t(s, a) \right], \quad (1)$$

where  $\alpha$  is the learning rate,  $Y_t(s, a) = R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t(s', a')$  is the target Q-value. Under some mild conditions Kearns & Singh (1998), the estimated Q-value of Q-learning is guaranteed to converge to the optimal Q-value, i.e.,  $\lim_{t \rightarrow +\infty} Q_t(s, a) = Q^*(s, a)$ ,  $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ .

We set the asynchronous Q-value, denoted by  $\hat{Q}_t(s, a)$ , of Bellman optimality equation as the ground truth Silver (2015), which updates as:

$$\hat{Q}_{t+1}(s, a) = \hat{Q}_t(s, a) + \alpha \left[ \mathbb{E} \left[ R(s, a) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_t(s', a') \right] - \hat{Q}_t(s, a) \right], \quad (2)$$

where  $\hat{Y}_t(s, a) = \mathbb{E} [R(s, a) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_t(s', a')]$  is the unbiased target Q-value. We focus on the maximum operation of Q-learning, which is the root cause of overestimation bias. Following Thrun & Schwartz (1993), although we assume that  $Q_t(s', a')$  is an unbiased estimator for  $\hat{Q}_t(s', a')$ ,  $\forall a' \in \mathcal{A}$ , according to Jensen's inequality Hansen & Pedersen (2003), we have:  $\mathbb{E} [\max_{a' \in \mathcal{A}} Q_t(s', a')] \geq \max_{a' \in \mathcal{A}} \mathbb{E} [Q_t(s', a')] = \max_{a' \in \mathcal{A}} \hat{Q}_t(s', a')$ . Figure 1(a) also verifies that this overestimation bias can slow the convergence speed of Q-learning.

#### 3.2 BIAS PROPAGATION CHAINS

For compactness, we write  $Q_i, Y_i, \hat{Q}_i, \hat{Y}_i$  instead of  $Q_i(s, a), Y_i(s, a), \hat{Q}_i(s, a), \hat{Y}_i(s, a)$ ,  $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ . Then, we expand Equation (1) and Equation (2) as:

$$\begin{cases} Q_{t+1} = (1 - \alpha)^{t+1} Q_0 + \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} Y_i, \\ \hat{Q}_{t+1} = (1 - \alpha)^{t+1} \hat{Q}_0 + \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \hat{Y}_i. \end{cases} \quad (3)$$

We set  $Q_0 = \hat{Q}_0$  with the same initial Q-value;  $e_i = Q_i - \hat{Q}_i$  as the Q-value estimation error;  $Z_i = Y_i - \hat{Y}_i$  as the target Q-value estimation bias. Then, we have:

$$e_{t+1} = \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} Z_i. \quad (4)$$

To analyze the overestimation bias propagation chains of Q-learning, following Thrun & Schwartz (1993); Lan et al. (2020), we first make a common assumption as:

**Assumption 1** The  $Q$ -value estimation error obeys a uniform distribution as:

$$e_{t+1} \sim U(\mu_{t+1} - \xi_{t+1}, \mu_{t+1} + \xi_{t+1}),$$

where  $\mathbb{E}[e_{t+1}] = \sum_{i=0}^t \alpha(1-\alpha)^{t-i} \mathbb{E}[Z_i] = \mu_{t+1} \geq 0$  due to the positive bias of  $Q$ -learning;  $0 \leq \xi_{t+1} \leq \xi_t$  due to the increasing number of samples and convergence guarantees. Note that  $\forall a \in \mathcal{A}$ , the  $Q$ -value estimation error  $e_{t+1}$  are independent and identically distributed (i.i.d.).

Following the target  $Q$ -value estimation bias, we have:

$$Z_{t+1} = R(s, a) - \mathbb{E}[R(s, a)] + \gamma \left( \max_{a' \in \mathcal{A}} Q_{t+1} - \mathbb{E} \left[ \max_{a' \in \mathcal{A}} \hat{Q}_{t+1} \right] \right). \quad (5)$$

**Theorem 1** Based on Assumption 1, the expected maximum  $Q$ -value estimation bias is as:

$$\mathbb{E}[Z_{t+1}] = \underbrace{\gamma \xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1}}_{\text{cur-bias}} + \underbrace{\gamma \sum_{i=0}^t \alpha(1-\alpha)^{t-i} \mathbb{E}[Z_i]}_{\text{prop-bias}}.$$

Theorem 1 is a generalization of the first Lemma in Thrun & Schwartz (1993); we provide the proof in Appendix A. Theorem 1 demonstrates that in  $Q$ -learning, the maximum  $Q$ -value estimation bias consists of two components: the current bias (cur-bias) and the propagation bias (prop-bias).

**Corollary 1** When the discount factor  $\gamma \geq \frac{\xi_{t+1}}{\xi_t}$ , we have:

$$\lim_{t \rightarrow +\infty} \frac{\gamma \sum_{i=0}^t \alpha(1-\alpha)^{t-i} \mathbb{E}[Z_i]}{\gamma \xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1}} \geq 1.$$

We prove Corollary 1 in Appendix B. It illustrates that during  $Q$ -learning updates, the prop-bias progressively dominates the maximum  $Q$ -value estimation bias composition, instead of the cur-bias.

### 3.3 EXAMPLE

Consider a simple multi-armed bandit setting Zhang et al. (2017), which includes one state  $S$  with ten identical actions, and each action returns a reward following  $\mathcal{N}(0, 1)$ . Following previous works Zhang et al. (2017); Tan et al. (2024c), we set  $\gamma = 0.95$ ,  $\alpha = 0.5$ ,  $\varepsilon = \frac{1}{n(S)}$ ,  $Q_0(S, a) \sim \mathcal{N}(0, 1)$  for each action  $a$ , where  $n(S)$  is the visited number of state  $S$ .

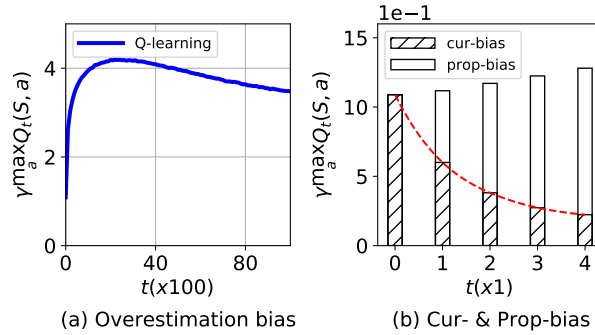


Figure 1: The discounted maximum  $Q$ -value of  $Q$ -learning.

Figure 1 shows the discounted maximum  $Q$ -value at state  $S$ , denoted by  $\gamma \max_a Q_t(S, a)$ , of  $Q$ -learning across steps  $t$ . All results are averaged over 10,000 runs. Note that the maximum expected  $Q$ -value is zero. According to Theorem 1, we compute prop-bias as  $\gamma \sum_{i=0}^{t-1} \alpha(1-\alpha)^{t-i-1} \max_a Q_i(S, a)$ , and compute cur-bias as  $\gamma \max_a Q_t(S, a) - \gamma \sum_{i=0}^{t-1} \alpha(1-\alpha)^{t-i-1} \max_a Q_i(S, a)$ . Figure 1(a) shows that  $Q$ -learning has an overestimation bias, which slows convergence speed. Figure 1(b) shows that prop-bias accumulates the previous positive bias, and progressively dominates the maximum  $Q$ -value estimation bias composition. More specifically, the ratio of prop-bias to cur-bias is  $\frac{1.06}{1.28-1.06} \approx 5$  at  $t = 4$ .



## 4 METHODS

### 4.1 ALTERNATING Q-LEARNING

Due to the overestimation bias of Q-learning, the prop-bias in Theorem 1 accumulates the unidirectional positive bias. Even though previous positive biases can be diluted during propagation, Corollary 1 shows that the prop-bias still dominates the maximum Q-value estimation bias. Example in Section 3.3 further illustrates that the prop-bias is the primary cause for the slow convergence speed in Q-learning. Similarly, Double Q-learning tends to accumulate the unidirectional negative bias, may lead to a slower convergence speed than Q-learning Ren et al. (2021); Li et al. (2023).

To break the above unidirectional estimation bias propagation chains and improve the convergence speed, we propose a novel positive-negative bias alternating execution framework. More specifically, this framework includes two hyperparameters  $(M, N)$ , where  $M \in \mathbb{N}^+$  is the number of alternating execution steps for the positive bias algorithms, such as Q-learning Watkins et al. (1989) and Averaged Q-learning Anschel et al. (2017);  $N \in \mathbb{N}^+$  is the corresponding steps for the negative bias algorithms, such as Double Q-learning Hasselt (2010) and EBQL Peer et al. (2021). Note that we set Q-learning and Double Q-learning as a pair of positive-negative bias algorithms, obtaining our Alternating Q-learning (AQ) as Algorithm 1.

---

#### Algorithm 1 Alternating Q-learning

---

```

1: Parameter:  $M, N$ 
2: Initialize:  $Q_0^1(s, a), Q_0^2(s, a), \forall (s, a) \in \mathcal{S} \times \mathcal{A}$ 
3: Get the starting state  $s$ 
4: for  $t = 0, 1, 2, \dots$  do
5:   Choose action  $a$  at state  $s$  by  $\varepsilon$ -greedy policy with  $\arg \max_{a \in \mathcal{A}} \frac{Q_t^1(s, a) + Q_t^2(s, a)}{2}$ 
6:   Take action  $a$ , get reward  $R(s, a)$  and next state  $s'$ 
7:   Randomly select one Q-table  $k$  from  $\{1, 2\}$  to update
8:   if  $t \bmod (M + N) < M$  then
9:      $Q_{t+1}^k(s, a) = Q_t^k(s, a) + \alpha [R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t^k(s', a') - Q_t^k(s, a)]$ 
10:  else
11:     $Q_{t+1}^k(s, a) = Q_t^k(s, a) + \alpha [R(s, a) + \gamma Q_t^{3-k}(s', \arg \max_{a' \in \mathcal{A}} Q_t^k(s', a')) - Q_t^k(s, a)]$ 
12:     $s \leftarrow s'$ 

```

---

AQ maintains two independent Q-tables  $Q_t^1(s, a), Q_t^2(s, a)$  in each time step  $t$ . When  $t \bmod (M + N) < M$ , AQ uses Q-learning to update, the target Q-value  $Y_t^k = R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t^k$ ; when  $t \bmod (M + N) \geq M$ , AQ uses Double Q-learning to update, the target Q-value  $Y_t^k = R(s, a) + \gamma Q_t^{3-k}(s', \arg \max_{a' \in \mathcal{A}} Q_t^k(s', a'))$ . To ensure fair comparison, we also consider two Q-tables  $\hat{Q}_t^1(s, a), \hat{Q}_t^2(s, a)$  for Bellman optimality equation as Equation (2), and the unbiased target Q-value  $\hat{Y}_t^k = \mathbb{E} [R(s, a) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}_t^k]$ . Following Section 3.2, we set the Q-value estimation error as:  $e_{t+1}^k = Q_{t+1}^k - \hat{Q}_{t+1}^k$ ; set the target Q-value estimation bias as:  $Z_i^k = Y_i^k - \hat{Y}_i^k$ ; and have:  $e_{t+1}^k = \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} Z_i^k$ , where  $e_{t+1}^k \sim U(\mu_{t+1}^k - \xi_{t+1}^k, \mu_{t+1}^k + \xi_{t+1}^k)$ . Note that the sign of  $\mathbb{E} [e_{t+1}^k] = \mu_{t+1}^k$  is unknown in advance due to the alternating mechanism.

**Theorem 2** *Under the above statement, the expected estimation bias of AQ is as follows.*

- When  $t \bmod (M + N) < M$ , we have:

$$\mathbb{E} [Z_{t+1}^k] = \underbrace{\gamma \xi_{t+1}^k \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1}}_{\text{cur-bias}} + \underbrace{\gamma \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \mathbb{E} [Z_i^k]}_{\text{prop-bias}}.$$

- When  $t \bmod (M + N) \geq M$ , we have:

$$\mathbb{E} [Z_{t+1}^k] = \underbrace{\gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} [\hat{Q}_{t+1}^k(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}')] p(a'_j = \hat{a}')}_{\text{cur-bias}} + \underbrace{\gamma \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \mathbb{E} [Z_i^k]}_{\text{prop-bias}}.$$

Note that  $\xi_{t+1}^k \geq 0$ ;  $a'_j = \arg \max_{a' \in \mathcal{A}} Q_{t+1}^k(s', a')$ ;  $\hat{a}' = \arg \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}^k(s', a')$ .

We prove Theorem 2 in Appendix C. It shows that when  $t \bmod (M + N) < M$ , AQ selects Q-learning to update, the cur-bias is positive, the prop-bias accumulates the positive bias; when  $t \bmod (M + N) \geq M$ , AQ selects Double Q-learning to update, the cur-bias is negative, the prop-bias accumulates the negative bias. Therefore, AQ breaks the unidirectional estimation bias propagation chains via alternately executing Q-learning and Double Q-learning. More specifically, the estimation bias is proportional to  $M$  and inversely proportional to  $N$ . When  $M \rightarrow +\infty$  and  $N = 1$ , AQ gets the upper bound, but is always smaller than Q-learning; when  $M = 1$  and  $N \rightarrow +\infty$ , AQ gets the lower bound, but is always larger than Double Q-learning. Due to that the cur-bias can vary between positive and negative, the prop-bias can theoretically be eliminated by two suitable parameters  $(M, N)$ .

#### 4.2 ADAPTIVE ALTERNATING Q-LEARNING

AQ alternately executes Q-learning and Double Q-learning via  $(M, N)$ . However, the optimal  $(M, N)$  are unknown in advance and dynamically for different state-action pairs. Therefore, we need to design an adaptive alternating strategy for AQ.

Following Algorithm 1, we first define the TD error Zhang et al. (2021) of Q-learning in each time step  $t$  as:  $td_t^Q(s, a) = R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t^k(s', a') - Q_t^k(s, a)$ ; the corresponding TD error of Double Q-learning as:  $td_t^{DQ}(s, a) = R(s, a) + \gamma Q_t^{3-k}(s', \arg \max_{a' \in \mathcal{A}} Q_t^k(s', a')) - Q_t^k(s, a)$ . Although the TD error has inherent uncertainty during updates, the convergence guarantees of algorithms still support it as a reliable feedback signal for tracking the variation trend of maximum Q-value estimation bias. More specifically, a large  $td_t^Q(s, a)$  reflects significant positive bias of Q-learning, and AQ should switch to Double Q-learning to suppress this bias and prevent its propagation from enlarging prop-bias. Similarly, when  $td_t^{DQ}(s, a)$  is small, AQ should switch to Q-learning to counteract the significant negative bias of Double Q-learning. Thus, we apply a softmax strategy based on the absolute value of TD error to compute the alternating execution probabilities as:

$$\mathbb{P}_t^{(s,a)}[Q] = \frac{e^{\tau|td_t^Q(s,a)|}}{e^{\tau|td_t^Q(s,a)|} + e^{\tau|td_t^{DQ}(s,a)|}}, \mathbb{P}_t^{(s,a)}[DQ] = \frac{e^{\tau|td_t^{DQ}(s,a)|}}{e^{\tau|td_t^Q(s,a)|} + e^{\tau|td_t^{DQ}(s,a)|}}, \quad (6)$$

where  $\tau \geq 0$  denotes the temperature parameter;  $\mathbb{P}_t^{(s,a)}[Q]$  and  $\mathbb{P}_t^{(s,a)}[DQ]$  represent the alternating probabilities to Q-learning and Double Q-learning, respectively.

Incorporating the above softmax strategy into AQ, we obtain Adaptive Alternating Q-learning (AdaAQ) as Algorithm 2. Note that at line 8, AQ selects Q-learning or Double Q-learning via categorical sampling.

---

#### Algorithm 2 Adaptive Alternating Q-learning

---

```

1: Initialize:  $Q_0^1(s, a), Q_0^2(s, a), \forall (s, a) \in \mathcal{S} \times \mathcal{A}$ 
2: Get the starting state  $s$ 
3: for  $t = 0, 1, 2, \dots$  do
4:   Choose action  $a$  at state  $s$  by  $\varepsilon$ -greedy policy with  $\arg \max_{a \in \mathcal{A}} \frac{Q_t^1(s, a) + Q_t^2(s, a)}{2}$ 
5:   Take action  $a$ , get reward  $R(s, a)$  and next state  $s'$ 
6:   Randomly select one Q-table  $k$  from  $\{1, 2\}$  to update
7:   Compute  $\mathbb{P}_t^{(s,a)}[Q], \mathbb{P}_t^{(s,a)}[DQ]$  as Equation (6)
8:   Select algorithm via categorical sampling:  $AQ \sim \text{Cat}([Q, DQ], [\mathbb{P}_t^{(s,a)}[Q], \mathbb{P}_t^{(s,a)}[DQ]])$ 
9:   if  $AQ == Q$  then
10:     $Q_{t+1}^k(s, a) = Q_t^k(s, a) + \alpha [R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q_t^k(s', a') - Q_t^k(s, a)]$ 
11:   else
12:     $Q_{t+1}^k(s, a) = Q_t^k(s, a) + \alpha [R(s, a) + \gamma Q_t^{3-k}(s', \arg \max_{a' \in \mathcal{A}} Q_t^k(s', a')) - Q_t^k(s, a)]$ 
13:    $s \leftarrow s'$ 

```

---

### 4.3 EXTENSION TO DRL: DISCRETE- AND CONTINUOUS-ACTION SPACES

Following the previous DRL algorithms Silver et al. (2014); Mnih et al. (2015), we represent the Q-function by a neural network for high-dimensional environments, and try to extend our methods to DRL. More specifically, for the discrete-action DRL settings, such as Atari Bellemare et al. (2013), we set DQN Mnih et al. (2015) and DDQN Van Hasselt et al. (2016) as a pair of positive-negative bias algorithms, and extend AQ and AdaAQ to Alternating DQN (ADQN) as Appendix D.1 and Adaptive Alternating DQN (AdaADQN) as Appendix D.2, respectively; for the continuous-action DRL settings, such as Mujoco Brockman et al. (2016), we set DDPG Silver et al. (2014) and TD3 Fujimoto et al. (2018) as a pair of positive-negative bias algorithms, and extend AQ and AdaAQ to Alternating DDPG (ADDPG) as Appendix E.1 and Adaptive Alternating DDPG (AdaADDPG) as Appendix E.2, respectively.

## 5 EXPERIMENTS

### 5.1 TABULAR MDP EXPERIMENTS

**MDP environments:** (1) Multi-armed bandit is shown in Section 3.3. (2) Roulette is adapted from Lee & Powell (2019). Like Tan et al. (2024b), we simplify Roulette to 13 actions: six 2 : 1 bets on 12 numbers (win 0.3158); six 1 : 1 bets on 18 numbers (win 0.4737); one 1 : 1 bet on nothing (win 0.5). (3) Gridworld  $3 \times 3$  and  $4 \times 4$  Zhu & Rigotti (2021) have four cardinal actions for each state, with start (southwest) and goal (northeast) positions. The agent resets to the start state upon reaching the goal, while boundary-violating actions maintain the current state. Non-goal states yield equiprobable stochastic rewards ( $-12$  or  $10$ ), while goal-state yields equiprobable stochastic rewards ( $-30$  or  $40$ ).

**Parameter settings:** Following Hasselt (2010); Penttiliotis & Wiering (2021); Tan et al. (2024a), we set  $\gamma = 0.95$ ,  $\varepsilon = \frac{1}{n(s)^{0.5}}$  by default;  $\alpha = \frac{1}{n(s,a)^{0.8}}$  for Multi-armed bandit and Roulette;  $\alpha = \frac{1}{n(s,a)^{1.0}}$  for Gridworld, where  $n(s)$  and  $n(s, a)$  are the visited number of state  $s$  and state-action pair  $(s, a)$ , respectively. Note that all experiment results are averaged over 1,000 runs.

**Comparison baselines:** We vary  $M = 1, 2, 4, 8, 16$ ;  $N = 1, 2, 4, 8, 16$  for our AQ;  $\tau = 1, 2, 5, 10, 100$  for our AdaAQ, and set  $\tau = 1$  by default. We consider eight comparison baselines as: Averaged Q-learning (AvgQ) Anschel et al. (2017), Maxmin Q-learning (MQ) Lan et al. (2020), Self-Correcting Q-learning (SCQ) Zhu & Rigotti (2021), Softmax Q-learning (SoftQ) Song et al. (2019), Weighted Double Q-learning (WDQ) Zhang et al. (2017), REDQ Chen et al. (2021), EBQL Peer et al. (2021), AdaEQ Wang et al. (2021). For a fair comparison, we set the number of Q-tables of AvgQ and MQ as 2; the temperature parameter of SoftQ as 1. For other baselines, we set the self-correcting parameter of SCQ as 2; the adaptive adjustment parameter of WDQ as 1; the number of Q-tables of REDQ, EBQL, AdaEQ as 10; which are recommended and fine-tuned.

Figure 2 shows the maximum Q-value, the probability of betting nothing denoted by  $\Pr[\text{leave}]$ , and the average reward per step of our methods in tabular MDP environments. **AQ:** Figure 2(a-b) show that the maximum Q-value of AQ is proportional to  $M$  and inversely proportional to  $N$ , and it always lies between that of Q-learning and Double Q-learning. This implies that AQ can effectively break the unidirectional estimation bias propagation chains. **AdaAQ:** Figure 2(c-d) show that the maximum Q-value curves of AdaAQ with different  $\tau$  overlap, and converge to zero faster than that of AQ. This implies that AdaAQ is not sensitive to  $\tau$ , and can provide an adaptive alternating strategy for AQ. **Comparison:** Figure 2(e-h) show that AdaAQ can estimate the maximum expected Q-value more accurately than the eight baselines, resulting in better policy and higher reward. Appendix F.1 provides the table of comparison results.

Table 1 shows the results of our AQ and AdaAQ with different parameters, where all values are evaluated in the final round. From the maximum Q-value perspective, AdaAQ(1), i.e.,  $4.14 \times 10^{-3}$ , is one level improvement over AQ(4, 8), i.e.,  $-7.04 \times 10^{-2}$ , in Multi-armed bandit. From the policy perspective, compared to AQ(4, 4), AdaAQ(100) improves by  $93.59\% - 87.43\% = 6.16\%$  in Roulette. From the reward perspective, AdaAQ(10) outperforms AQ(4, 4) by  $\frac{-0.12 - (-0.492)}{0.492} = 75.60\%$  in Gridworld  $3 \times 3$ ; AdaAQ(5) outperforms AQ(4, 4) by  $\frac{-0.476 - (-0.73)}{0.73} = 34.79\%$  in Gridworld  $4 \times 4$ . Appendix F.2 provides the figure of our AQ and AdaAQ with different parameters.

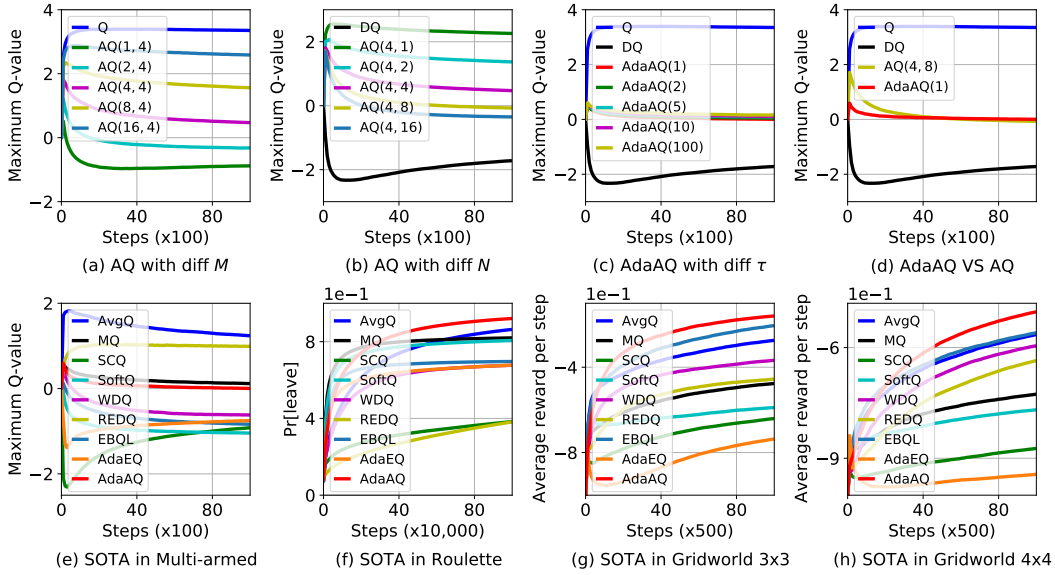


Figure 2: The performance of our AQ and AdaAQ in tabular MDP settings.

Table 1: The results of our AQ and AdaAQ with different parameters.

Algorithm	Maximum Q-value	Pr[leave]	Average reward per step	
	Multi-armed bandit	Roulette	Gridworld $3 \times 3$	Gridworld $4 \times 4$
Q-learning	3.35e0	40.83%	-8.55e-1	-9.41e-1
Double Q-learning	-1.72e0	26.15%	-6.65e-1	-8.45e-1
AQ(1,4)	-8.82e-1	49.88%	-5.35e-1	-7.76e-1
AQ(2,4)	-3.21e-1	83.73%	-5.16e-1	-7.44e-1
AQ(4,4)	4.72e-1	87.43%	-4.92e-1	-7.30e-1
AQ(8,4)	1.56e0	77.83%	-5.98e-1	-7.74e-1
AQ(16,4)	2.59e0	56.04%	-7.33e-1	-8.63e-1
AQ(4,1)	2.26e0	63.32%	-5.81e-1	-7.97e-1
AQ(4,2)	1.37e0	82.70%	-5.27e-1	-7.47e-1
AQ(4,8)	-7.04e-2	85.69%	-5.80e-1	-7.89e-1
AQ(4,16)	-3.50e-1	78.26%	-5.98e-1	-8.28e-1
AdaAQ(1)	<b>4.14e-3</b>	92.00%	-1.60e-1	-5.04e-1
AdaAQ(2)	5.01e-3	90.53%	-1.57e-1	-4.97e-1
AdaAQ(5)	9.32e-3	90.78%	-1.25e-1	<b>-4.76e-1</b>
AdaAQ(10)	1.08e-2	91.24%	<b>-1.20e-1</b>	-4.93e-1
AdaAQ(100)	1.61e-2	<b>93.59%</b>	-1.50e-1	-5.09e-1

## 5.2 DISCRETE-ACTION DRL EXPERIMENTS

We choose three discrete-action DRL experiment environments from PLE Urtans & Nikitenko (2018) and MinAtar Young & Tian (2019): Pixelcopter, Breakout, Asterix. Appendix G.1 provides the experiment settings. Figure 3 shows the average score per episode of our AdaADQN in discrete-action DRL settings, where the score is averaged over the last 100 episodes and the shaded area represents one standard error. One can observe that the average score curves of AdaADQN lie at the top, and are not sensitive to  $\tau$ . Appendix G.2 provides the table of comparison results between our AdaADQN and ten baselines, where all values are evaluated in the final round. More specifically, our AdaADQN improves the average score per episode over baselines by at least  $\frac{37.89-30.13}{30.13} = 25.76\%$ ,

$\frac{13.89-10.64}{10.64} = 30.55\%$ , and  $\frac{16.13-9.77}{9.77} = 65.10\%$  in Pixelcopter, Breakout, and Asterix, respectively. In addition, Appendix G.3 provides the results of our ADNQ and AdaADQN with different parameters.

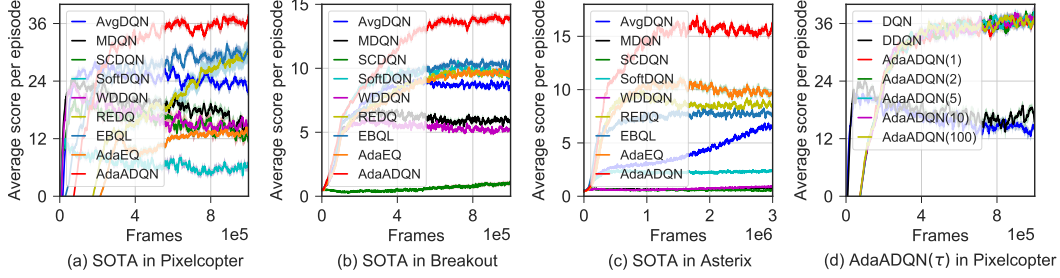


Figure 3: The average score per episode of our AdaADQN in discrete-action DRL settings.

### 5.3 CONTINUOUS-ACTION DRL EXPERIMENTS

We choose three continuous-action DRL experiment environments from Mujoco Brockman et al. (2016): Hopper, Ant, and Walker2d. Appendix H.1 provides the experiment settings. Figure 4 shows the average return of our AdaADDPG in continuous-action DRL settings, where the average return is averaged over the last 10 episodes. One can observe that the average return curves of AdaADDPG lie at the top, and are not sensitive to  $\tau$ . Appendix H.2 provides the table of comparison results between our AdaADDPG and five baselines, where all values are evaluated in the final round. More specifically, our AdaADDPG improves the average return over baselines by at least  $\frac{3296.87-3033.64}{3033.64} = 8.68\%$ ,  $\frac{3321.97-2818.44}{2818.44} = 17.87\%$ ,  $\frac{4786.20-4248.04}{4248.04} = 12.67\%$  in Hopper, Ant, Walker2d, respectively. In addition, Appendix H.3 provides the results of our ADDPG and AdaADDPG with different parameters.

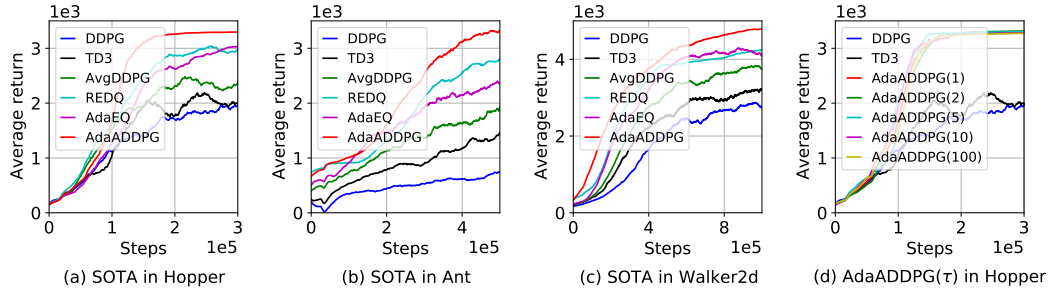


Figure 4: The average return of our AdaADDPG in continuous-action DRL settings.

## 6 CONCLUSION

In this paper, we analyze the overestimation bias propagation process of Q-learning, and find that the prop-bias rather than the cur-bias dominates the maximum Q-value estimation bias. Then, we propose AQ, which breaks the unidirectional estimation bias propagation chains via alternately executing positive-negative bias algorithms. Based on AQ, we design an adaptive alternating strategy, leading to AdaAQ. More specifically, it applies a softmax strategy with the absolute value of TD error to determine whether AQ should execute Q-learning or Double Q-learning. We also extend AQ and AdaAQ to both discrete- and continuous-action DRL settings. Extensive experiment results show that our method outperforms several baselines drastically in tabular MDP, discrete-action DRL, and continuous-action DRL settings. More specifically, our method improves the average score or return over baselines by at least 65.10% in Asterix and 17.87% in Ant, respectively.

## REFERENCES

- Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pp. 176–185. PMLR, 2017.
- Athanasios Ioannis Arvanitidis and Miltiadis Alamaniotis. Optimal economic dispatch scheduling in competitive energy market utilizing a greedy q-learning algorithm. In *2024 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, pp. 1–5. IEEE, 2024.
- Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.
- G Brockman, V Cheung, L Pettersson, J Schneider, J Schulman, J Tang, and W Zaremba. Openai gym: A toolkit for developing and comparing reinforcement learning algorithms, 2016.
- Liyu Chen, Rahul Jain, and Haipeng Luo. Learning infinite-horizon average-reward markov decision process with constraints. In *International Conference on Machine Learning*, pp. 3246–3270. PMLR, 2022.
- Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.
- Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.
- Tianci Gao. Optimizing robotic arm control using deep q-learning and artificial neural networks through demonstration-based methodologies: A case study of dynamic and static conditions. *Robotics and Autonomous systems*, 181:104771, 2024.
- Xiaoyu Gong, Shuai Lü, Jiayu Yu, Sheng Zhu, and Zongze Li. Adaptive estimation q-learning with uncertainty and familiarity. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 3750–3758, 2023.
- Frank Hansen and Gert K Pedersen. Jensen’s operator inequality. *Bulletin of the London Mathematical Society*, 35(4):553–564, 2003.
- Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.
- Bhaeiyal Ishwaei D, Divakar Shabma, and K Krishnamoorthy. Non-existence of unbiased estimators of ordered parameters. *Statistics: A Journal of Theoretical and Applied Statistics*, 16(1):89–95, 1985.
- Thommen George Karimpanal, Hung Le, Majid Abdolshah, Santu Rana, Sunil Gupta, Truyen Tran, and Svetha Venkatesh. Balanced q-learning: Combining the influence of optimistic and pessimistic targets. *Artificial Intelligence*, 325:104021, 2023.
- Michael Kearns and Satinder Singh. Finite-sample convergence rates for q-learning and indirect algorithms. *Advances in neural information processing systems*, 11, 1998.
- Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=Bkg0u3Etwr>.
- Donghun Lee and Warren B Powell. Bias-corrected q-learning with multistate extension. *IEEE Transactions on Automatic Control*, 64(10):4011–4023, 2019.
- Shengbo Eben Li. Model-free indirect rl: Monte carlo. In *Reinforcement Learning for Sequential Decision and Optimal Control*, pp. 41–65. Springer, 2023.
- Sicen Li, Qinyun Tang, Yiming Pang, Xinmeng Ma, and Gang Wang. Realistic actor-critic: A framework for balance between value overestimation and underestimation. *Frontiers in Neurorobotics*, 16:1081242, 2023.

- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. Ensemble bootstrapping for q-learning. In *International conference on machine learning*, pp. 8454–8463. PMLR, 2021.
- Andreas Pentaliotis and Marco A Wiering. Variation-resistant q-learning: Controlling and utilizing estimation bias in reinforcement learning for better performance. In *ICAART (2)*, pp. 17–28, 2021.
- Zhizhou Ren, Guangxiang Zhu, Hao Hu, Beining Han, Jianglun Chen, and Chongjie Zhang. On the estimation bias in double q-learning. *Advances in Neural Information Processing Systems*, 34: 10246–10259, 2021.
- Eduardo Rodrigues Gomes and Ryszard Kowalczyk. Dynamic analysis of multiagent q-learning with  $\epsilon$ -greedy exploration. In *Proceedings of the 26th annual international conference on machine learning*, pp. 369–376, 2009.
- Peter Schmitt-Förster and Tobias Sutter. Regularized q-learning through robust averaging. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 43742–43764, Vienna, Austria, 2024. ICML.
- David Silver. Lecture 3: Planning by dynamic programming. *UCL Course on RL*, 2015.
- David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.
- Zhao Song, Ron Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. In *International conference on machine learning*, pp. 5916–5925. PMLR, 2019.
- Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 2018.
- Tao Tan, Hong Xie, and Liang Feng. Q-learning with heterogeneous update strategy. *Information Sciences*, 656:119902, 2024a.
- Tao Tan, Hong Xie, and Liang Feng. Q-learning with heterogeneous update strategy. *Information Sciences*, 656:119902, 2024b.
- Tao Tan, Hong Xie, and Defu Lian. Adaptive order q-learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 4946–4954, 2024c.
- Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pp. 255–263. Psychology Press, 1993.
- Evalds Urtans and Agris Nikitenko. Survey of deep q-network variants in pygame learning environment. In *Proceedings of the 2018 2nd international conference on deep learning technologies*, pp. 27–36, 2018.
- Hado Van Hasselt. Estimating the maximum expected value: an analysis of (nested) cross validation and the maximum sample average. *arXiv preprint arXiv:1302.7175*, 2013.
- Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.
- Hang Wang, Sen Lin, and Junshan Zhang. Adaptive ensemble q-learning: Minimizing estimation bias via error feedback. *Advances in neural information processing systems*, 34:24778–24790, 2021.

- Christopher John Cornish Hellaby Watkins et al. Learning from delayed rewards. 1989.
- Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 59:60, 2019.
- Sheng Zhang, Zhe Zhang, and Siva Theja Maguluri. Finite sample analysis of average-reward td learning and  $q$ -learning. *Advances in Neural Information Processing Systems*, 34:1230–1242, 2021.
- Zongzhang Zhang, Zhiyuan Pan, and Mykel J Kochenderfer. Weighted double  $q$ -learning. In *IJCAI*, pp. 3455–3461, 2017.
- Rong Zhu and Mattia Rigotti. Self-correcting  $q$ -learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11185–11192, 2021.
- Zefang Zong, Tao Feng, Jingwei Wang, Tong Xia, and Yong Li. Deep reinforcement learning for demand-driven services in logistics and transportation systems: A survey. *ACM Transactions on Knowledge Discovery from Data*, 19(4):1–42, 2025.



## A PROOF TO THEOREM 1

In Assumption 1,  $e_{t+1} \sim U(\mu_{t+1} - \xi_{t+1}, \mu_{t+1} + \xi_{t+1})$  is a uniform random variable. Its probability density function (PDF) and cumulative distribution function (CDF) are given by:

$$\begin{cases} f(x) = \frac{1}{2\xi_{t+1}}, \\ F(x) = \frac{1}{2} + \frac{x - \mu_{t+1}}{2\xi_{t+1}}, x \in [\mu_{t+1} - \xi_{t+1}, \mu_{t+1} + \xi_{t+1}]. \end{cases} \quad (7)$$

Following the first Lemma in Thrun & Schwartz (1993), according to Equation (5), the expected maximum Q-value estimation bias for Q-learning is as:

$$\begin{aligned} \mathbb{E}[Z_{t+1}] &= \mathbb{E}[R(s, a)] - \mathbb{E}[R(s, a)] + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q_{t+1} - \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}\right] \\ &= \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} e_{t+1}\right] \\ &= \gamma \int_{\mu_{t+1} - \xi_{t+1}}^{\mu_{t+1} + \xi_{t+1}} x |\mathcal{A}| f(x) [F(x)]^{|\mathcal{A}|-1} dx \\ &= \gamma \int_{\mu_{t+1} - \xi_{t+1}}^{\mu_{t+1} + \xi_{t+1}} x |\mathcal{A}| \frac{1}{2\xi_{t+1}} \left[\frac{1}{2} + \frac{x - \mu_{t+1}}{2\xi_{t+1}}\right]^{|\mathcal{A}|-1} dx \\ &= \gamma 2\xi_{t+1} |\mathcal{A}| \int_0^1 \left[y - \frac{1}{2} + \frac{\mu_{t+1}}{2\xi_{t+1}}\right] [y]^{|\mathcal{A}|-1} dy \\ &= \gamma 2\xi_{t+1} |\mathcal{A}| \int_0^1 \left([y]^{|\mathcal{A}|} - \frac{1}{2} [y]^{|\mathcal{A}|-1} + \frac{\mu_{t+1}}{2\xi_{t+1}} [y]^{|\mathcal{A}|-1}\right) dy \\ &= \gamma 2\xi_{t+1} |\mathcal{A}| \left(\frac{1}{|\mathcal{A}|+1} - \frac{1}{2|\mathcal{A}|} + \frac{\mu_{t+1}}{2\xi_{t+1}|\mathcal{A}|}\right) \\ &= \gamma \xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1} + \gamma \mu_{t+1}. \end{aligned} \quad (8)$$

According to Equation (4) and Assumption 1, we have:

$$\mathbb{E}[e_{t+1}] = \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \mathbb{E}[Z_i] = \mu_{t+1}. \quad (9)$$

Substituting Equation (9) into Equation (8), we have:

$$\begin{aligned} \mathbb{E}[Z_{t+1}] &= \gamma \xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1} + \gamma \mu_{t+1} \\ &= \underbrace{\gamma \xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1}}_{\text{cur-bias}} + \underbrace{\gamma \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \mathbb{E}[Z_i]}_{\text{prop-bias}}. \end{aligned} \quad (10)$$

According to Equation (10), we can get the expected maximum Q-value estimation bias of Q-learning as Theorem 1.

## B PROOF TO COROLLARY 1

Following Theorem 1, according to Equation (8) and Equation (10), the ratio of propagation bias to current bias is as:

$$\lim_{t \rightarrow +\infty} \frac{\gamma \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \mathbb{E}[Z_i]}{\gamma \xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1}} = \lim_{t \rightarrow +\infty} \frac{\sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \left[ \gamma \xi_i \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1} + \gamma \mu_i \right]}{\xi_{t+1} \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + 1}} \quad (11)$$

According to Assumption 1, the  $\mu_{t+1} \geq 0$  due to the positive bias of Q-learning; the  $0 \leq \xi_{t+1} \leq \xi_t$  due to the increasing number of samples and convergence guarantees; where  $\forall t \in \mathbb{N}$ . Then, when the discount factor  $\gamma \geq \frac{\xi_{t+1}}{\xi_t}$ , we have:

$$\begin{aligned}
\lim_{t \rightarrow +\infty} \frac{\gamma \sum_{i=0}^t \alpha (1-\alpha)^{t-i} \mathbb{E}[Z_i]}{\gamma \xi_{t+1} \frac{|\mathcal{A}|-1}{|\mathcal{A}|+1}} &= \lim_{t \rightarrow +\infty} \frac{\sum_{i=0}^t \alpha (1-\alpha)^{t-i} \left[ \gamma \xi_i \frac{|\mathcal{A}|-1}{|\mathcal{A}|+1} + \gamma \mu_i \right]}{\xi_{t+1} \frac{|\mathcal{A}|-1}{|\mathcal{A}|+1}} \\
&\geq \lim_{t \rightarrow +\infty} \frac{\sum_{i=0}^t \alpha (1-\alpha)^{t-i} \gamma \xi_i \frac{|\mathcal{A}|-1}{|\mathcal{A}|+1}}{\xi_{t+1} \frac{|\mathcal{A}|-1}{|\mathcal{A}|+1}} \\
&\geq \lim_{t \rightarrow +\infty} \sum_{i=0}^t \alpha (1-\alpha)^{t-i} \frac{\xi_i}{\xi_t} \\
&\geq \lim_{t \rightarrow +\infty} \sum_{i=0}^t \alpha (1-\alpha)^{t-i} \\
&= 1.
\end{aligned} \tag{12}$$

According to Equation (12), we can get the ratio of propagation bias to current bias as Corollary 1.

## C PROOF TO THEOREM 2

When  $t \bmod (M+N) < M$ , AQ uses Q-learning to update, following Theorem 1 in Section 3.2, the expected estimation bias of AQ is as:

$$\begin{aligned}
\mathbb{E}[Z_{t+1}^k] &= \mathbb{E}[R(s, a)] - \mathbb{E}[R(s, a)] + \gamma \mathbb{E} \left[ \max_{a' \in \mathcal{A}} Q_{t+1}^k - \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}^k \right] \\
&= \gamma \mathbb{E} \left[ \max_{a' \in \mathcal{A}} e_{t+1}^k \right] \\
&= \underbrace{\gamma \xi_{t+1}^k \frac{|\mathcal{A}|-1}{|\mathcal{A}|+1}}_{\text{cur-bias}} + \underbrace{\gamma \sum_{i=0}^t \alpha (1-\alpha)^{t-i} \mathbb{E}[Z_i^k]}_{\text{prop-bias}},
\end{aligned} \tag{13}$$

where  $\xi_{t+1}^k \geq 0$ . For the cur-bias, which is positive; for the prop-bias, which may include the previous steps positive or negative bias due to the alternating execution strategy.

When  $t \bmod (M+N) \geq M$ , AQ uses Double Q-learning to update, following the Lemma 1 in Hasselt (2010), the expected estimation bias of AQ is as:

$$\begin{aligned}
\mathbb{E}[Z_{t+1}^k] &= \mathbb{E}[R(s, a)] - \mathbb{E}[R(s, a)] + \\
&\quad \gamma \left( \mathbb{E} \left[ Q_{t+1}^{3-k} \left( s', \arg \max_{a' \in \mathcal{A}} Q_{t+1}^k(s', a') \right) \right] - \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}^k(s', a') \right) \\
&= \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ Q_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') \right] p(a'_j = \hat{a}'),
\end{aligned} \tag{14}$$

where  $a'_j = \arg \max_{a' \in \mathcal{A}} Q_{t+1}^k(s', a')$  and  $\hat{a}' = \arg \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}^k(s', a')$ . Recall that both AQ and Bellman optimality equation randomly select one Q-table to update, that is to say, both the two Q-tables are independent and identically distributed (i.i.d.), then, we have:

$$\begin{cases} \mathbb{E}[Q_{t+1}^{3-k}] = \mathbb{E}[Q_{t+1}^k], \\ \mathbb{E}[\hat{Q}_{t+1}^{3-k}] = \mathbb{E}[\hat{Q}_{t+1}^k], \\ \mathbb{E}[Q_{t+1}^{3-k} - \hat{Q}_{t+1}^{3-k}] = \mathbb{E}[Q_{t+1}^k - \hat{Q}_{t+1}^k] = \mathbb{E}[e_{t+1}^k]. \end{cases} \tag{15}$$

Substituting Equation (15) into Equation (14), we have:

$$\begin{aligned}
\mathbb{E}[Z_{t+1}^k] &= \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ Q_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') \right] p(a'_j = \hat{a}') \\
&= \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ \hat{Q}_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') + Q_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^{3-k}(s', a'_j) \right] p(a'_j = \hat{a}') \\
&= \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ \hat{Q}_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') \right] p(a'_j = \hat{a}') + \\
&\quad \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ Q_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^{3-k}(s', a'_j) \right] p(a'_j = \hat{a}') \\
&= \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ \hat{Q}_{t+1}^{3-k}(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') \right] p(a'_j = \hat{a}') + \gamma \mathbb{E} \left[ Q_{t+1}^{3-k} - \hat{Q}_{t+1}^{3-k} \right] \\
&= \gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ \hat{Q}_{t+1}^k(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') \right] p(a'_j = \hat{a}') + \gamma \mathbb{E} [e_{t+1}^k] \\
&= \underbrace{\gamma \sum_{j=1}^{|\mathcal{A}|} \mathbb{E} \left[ \hat{Q}_{t+1}^k(s', a'_j) - \hat{Q}_{t+1}^k(s', \hat{a}') \right] p(a'_j = \hat{a}')}_{\text{cur-bias}} + \underbrace{\gamma \sum_{i=0}^t \alpha (1 - \alpha)^{t-i} \mathbb{E} [Z_i^k]}_{\text{prop-bias}}.
\end{aligned} \tag{16}$$

For the cur-bias, which is negative; for the prop-bias, which may include the previous steps positive or negative bias due to the alternating execution strategy.

According to Equation (13) and Equation (16) we can get the expected maximum Q-value estimation bias of AQ as Theorem 2.

## D EXTENSION TO DRL: DISCRETE-ACTION SPACES

### D.1 ALTERNATING DQN

We set DQN Mnih et al. (2015) and DDQN Van Hasselt et al. (2016) as a pair of positive-negative bias algorithms, and extend our AQ in Section 4.1 to the discrete-action DRL settings, leading to Alternating DQN (ADQN) as Algorithm 3, where  $M \in \mathbb{N}^+$  is the number of alternating execution steps for DQN;  $N \in \mathbb{N}^+$  is the corresponding steps for DDQN;  $\theta$  is the Q-function network;  $\bar{\theta}$  is the target Q-function network;  $|\mathcal{D}|$  is the buffer size;  $|\mathcal{B}|$  is the batch size; and  $V \in \mathbb{N}^+$  is the updated steps for  $\bar{\theta}$ .

### D.2 ADAPTIVE ALTERNATING DQN

Following Section 4.2, according to Algorithm 3, we first denote the TD error of DQN Mnih et al. (2015) and DDQN Van Hasselt et al. (2016) as:

$$\begin{cases} td_t^{DQN}(s, a) = R(s, a) + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \bar{\theta}_t) - Q(s, a; \theta_t), \\ td_t^{DDQN}(s, a) = R(s, a) + \gamma Q\left(s', \arg \max_{a' \in \mathcal{A}} Q(s', a'; \theta_t); \bar{\theta}_t\right) - Q(s, a; \theta_t). \end{cases} \tag{17}$$

Then, we apply a softmax strategy based on the absolute value of TD error to compute the alternating execution probabilities as:

**Algorithm 3 Alternating DQN**


---

```

1: Parameter:  $M, N$ 
2: Initialize:  $\theta_0, \bar{\theta}_0 = \theta_0$ 
3: Get the starting state  $s$ 
4: for  $t = 0, 1, 2, \dots$  do
5:   Choose action  $a$  at state  $s$  by  $\varepsilon$ -greedy policy with  $\arg \max_{a \in \mathcal{A}} Q(s, a; \theta_t)$ 
6:   Take action  $a$ , get reward  $R(s, a)$  and next state  $s'$ 
7:   Store  $\{s, a, R(s, a), s'\}$  in  $\mathcal{D}$ 
8:   Randomly sample a set of  $\{s_B, a_B, R(s_B, a_B), s'_B\}$  from  $\mathcal{D}$ 
9:   for  $\{s_j, a_j, R(s_j, a_j), s'_j\} \in \{s_B, a_B, R(s_B, a_B), s'_B\}$  do
10:    if  $t \bmod (M + N) < M$  then
11:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma \max_{a'_j \in \mathcal{A}} Q(s'_j, a'_j; \bar{\theta}_t)$ 
12:    else
13:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma Q(s'_j, \arg \max_{a'_j \in \mathcal{A}} Q(s'_j, a'_j; \theta_t); \bar{\theta}_t)$ 
14:     $\theta_{t+1} \approx \arg \min_{\theta_t} \mathbb{E}_{j \in [1, 2, \dots, |B|]} [Y(s_j, a_j) - Q(s_j, a_j; \theta_t)]^2$ 
15:    if  $(t + 1) \bmod V == 0$  then
16:       $\bar{\theta}_{t+1} \leftarrow \theta_{t+1}$ 
17:     $s \leftarrow s'$ 

```

---

$$\begin{cases} \mathbb{P}_t^{(s,a)}[DQN] = \frac{e^{\tau |td_t^{DQN}(s,a)|}}{e^{\tau |td_t^{DQN}(s,a)|} + e^{\tau |td_t^{DDQN}(s,a)|}}, \\ \mathbb{P}_t^{(s,a)}[DDQN] = \frac{e^{\tau |td_t^{DDQN}(s,a)|}}{e^{\tau |td_t^{DQN}(s,a)|} + e^{\tau |td_t^{DDQN}(s,a)|}}, \end{cases} \quad (18)$$

where  $\tau \geq 0$  denotes the temperature parameter;  $\mathbb{P}_t^{(s,a)}[DQN]$  and  $\mathbb{P}_t^{(s,a)}[DDQN]$  represent the alternating probabilities to DQN and DDQN, respectively.

Based on above analysis, we extend our AdaAQ in Section 4.2 to the discrete-action DRL settings, leading to Adaptive Alternating DQN (AdaADQN) as Algorithm 4. Note that at line 10, ADQN selects DQN or DDQN via categorical sampling.

**Algorithm 4 Adaptive Alternating DQN**


---

```

1: Initialize:  $\theta_0, \bar{\theta}_0 = \theta_0$ 
2: Get the starting state  $s$ 
3: for  $t = 0, 1, 2, \dots$  do
4:   Choose action  $a$  at state  $s$  by  $\varepsilon$ -greedy policy with  $\arg \max_{a \in \mathcal{A}} Q(s, a; \theta_t)$ 
5:   Take action  $a$ , get reward  $R(s, a)$  and next state  $s'$ 
6:   Store  $\{s, a, R(s, a), s'\}$  in  $\mathcal{D}$ 
7:   Randomly sample a set of  $\{s_B, a_B, R(s_B, a_B), s'_B\}$  from  $\mathcal{D}$ 
8:   for  $\{s_j, a_j, R(s_j, a_j), s'_j\} \in \{s_B, a_B, R(s_B, a_B), s'_B\}$  do
9:     Compute  $\mathbb{P}_t^{(s_j, a_j)}[DQN], \mathbb{P}_t^{(s_j, a_j)}[DDQN]$  as Equation (18)
10:     $ADQN \sim \text{Cat}\left([DQN, DDQN], \left[\mathbb{P}_t^{(s_j, a_j)}[DQN], \mathbb{P}_t^{(s_j, a_j)}[DDQN]\right]\right)$ 
11:    if  $ADQN == DQN$  then
12:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma \max_{a'_j \in \mathcal{A}} Q(s'_j, a'_j; \bar{\theta}_t)$ 
13:    else
14:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma Q(s'_j, \arg \max_{a'_j \in \mathcal{A}} Q(s'_j, a'_j; \theta_t); \bar{\theta}_t)$ 
15:     $\theta_{t+1} \approx \arg \min_{\theta_t} \mathbb{E}_{j \in [1, 2, \dots, |B|]} [Y(s_j, a_j) - Q(s_j, a_j; \theta_t)]^2$ 
16:    if  $(t + 1) \bmod V == 0$  then
17:       $\bar{\theta}_{t+1} \leftarrow \theta_{t+1}$ 
18:     $s \leftarrow s'$ 

```

---

## E EXTENSION TO DRL: CONTINUOUS-ACTION SPACES

### E.1 ALTERNATING DDPG

We set DDPG Silver et al. (2014) and TD3 Fujimoto et al. (2018) as a pair of positive-negative bias algorithms, and extend our AQ in Section 4.1 to the continuous-action DRL settings, leading to Alternating DDPG (ADDPG) as Algorithm 5, where  $M \in \mathbb{N}^+$  is the number of alternating execution steps for DDPG;  $N \in \mathbb{N}^+$  is the corresponding steps for TD3;  $\theta$  is the Q-function network;  $\bar{\theta}$  is the target Q-function network;  $\phi$  is the policy-function network;  $\bar{\phi}$  is the target policy-function network;  $|\mathcal{D}|$  is the buffer size;  $|\mathcal{B}|$  is the batch size; and  $\rho \in (0, 1)$  is the smoothing coefficient.

---

**Algorithm 5 Alternating DDPG**


---

```

1: Parameter:  $M, N$ 
2: Initialize:  $\theta_0 = \{\theta_0^1, \theta_0^2\}, \bar{\theta}_0 = \theta_0$ 
3: Initialize:  $\phi_0, \bar{\phi}_0 = \phi_0$ 
4: Get the starting state  $s$ 
5: for  $t = 0, 1, 2, \dots$  do
6:   Choose action  $a$  at state  $s$  by  $\pi(s; \phi_t)$  with the exploration noise  $\epsilon \in \mathcal{N}(0, 0.1)$ 
7:   Take action  $a$ , get reward  $R(s, a)$  and next state  $s'$ 
8:   Store  $\{s, a, R(s, a), s'\}$  in  $\mathcal{D}$ 
9:   Randomly sample a set of  $\{s_{\mathcal{B}}, a_{\mathcal{B}}, R(s_{\mathcal{B}}, a_{\mathcal{B}}), s'_{\mathcal{B}}\}$  from  $\mathcal{D}$ 
10:  Randomly select one Q-function network  $k$  from  $\{1, 2\}$  to update
11:  for  $\{s_j, a_j, R(s_j, a_j), s'_j\} \in \{s_{\mathcal{B}}, a_{\mathcal{B}}, R(s_{\mathcal{B}}, a_{\mathcal{B}}), s'_{\mathcal{B}}\}$  do
12:    if  $t \bmod (M + N) < M$  then
13:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma Q(s'_j, \pi(s'_j; \bar{\phi}_t); \bar{\theta}_t^k)$ 
14:    else
15:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma \min_{i \in \{1, 2\}} Q(s'_j, \pi(s'_j; \bar{\phi}_t); \bar{\theta}_t^i)$ 
16:     $\theta_{t+1}^k \approx \arg \min_{\theta_t^k} \mathbb{E}_{j \in [1, 2, \dots, |\mathcal{B}|]} [Y(s_j, a_j) - Q(s_j, a_j; \theta_t^k)]^2$ 
17:    Update policy-function network by gradient ascent
18:     $\phi_{t+1} \approx \arg \max_{\phi_t} \mathbb{E}_{j \in [1, 2, \dots, |\mathcal{B}|]} [Q(s_j, \pi(s_j; \phi_t); \theta_{t+1}^k)]$ 
19:     $\bar{\theta}_{t+1}^k \leftarrow \rho \theta_{t+1}^k + (1 - \rho) \bar{\theta}_t^k$ 
20:     $\bar{\phi}_{t+1} \leftarrow \rho \phi_{t+1} + (1 - \rho) \bar{\phi}_t$ 
21:     $s \leftarrow s'$ 

```

---

### E.2 ADAPTIVE ALTERNATING DDPG

Following Section 4.2, according to Algorithm 5, we first denote the TD error of DDPG Silver et al. (2014) and TD3 Fujimoto et al. (2018) as:

$$\begin{cases} td_t^{DDPG}(s, a) = R(s, a) + \gamma Q(s', \pi(s'; \bar{\phi}_t); \bar{\theta}_t^k) - Q(s, a; \theta_t^k), \\ td_t^{TD3}(s, a) = R(s, a) + \gamma \min_{i \in \{1, 2\}} Q(s', \pi(s'; \bar{\phi}_t); \bar{\theta}_t^i) - Q(s, a; \theta_t^k). \end{cases} \quad (19)$$

Then, we apply a softmax strategy based on the absolute value of TD error to compute the alternating execution probabilities as:

$$\begin{cases} \mathbb{P}_t^{(s, a)}[DDPG] = \frac{e^{\tau |td_t^{TD3}(s, a)|}}{e^{\tau |td_t^{DDPG}(s, a)|} + e^{\tau |td_t^{TD3}(s, a)|}}, \\ \mathbb{P}_t^{(s, a)}[TD3] = \frac{e^{\tau |td_t^{DDPG}(s, a)|}}{e^{\tau |td_t^{DDPG}(s, a)|} + e^{\tau |td_t^{TD3}(s, a)|}}, \end{cases} \quad (20)$$

where  $\tau \geq 0$  denotes the temperature parameter;  $\mathbb{P}_t^{(s, a)}[DDPG]$  and  $\mathbb{P}_t^{(s, a)}[TD3]$  represent the alternating probabilities to DDPG and TD3, respectively.

Based on above analysis, we extend our AdaAQ in Section 4.2 to the continuous-action DRL settings, leading to Adaptive Alternating DDPG (AdaADDPG) as Algorithm 6. Note that at line 12, ADPPG selects DDPG or TD3 via categorical sampling.

---

**Algorithm 6 Adaptive Alternating DDPG**


---

```

1: Initialize:  $\theta_0 = \{\theta_0^1, \theta_0^2\}, \bar{\theta}_0 = \theta_0$ 
2: Initialize:  $\phi_0, \bar{\phi}_0 = \phi_0$ 
3: Get the starting state  $s$ 
4: for  $t = 0, 1, 2, \dots$  do
5:   Choose action  $a$  at state  $s$  by  $\pi(s; \phi_t)$  with the exploration noise  $\epsilon \in \mathcal{N}(0, 0.1)$ 
6:   Take action  $a$ , get reward  $R(s, a)$  and next state  $s'$ 
7:   Store  $\{s, a, R(s, a), s'\}$  in  $\mathcal{D}$ 
8:   Randomly sample a set of  $\{s_B, a_B, R(s_B, a_B), s'_B\}$  from  $\mathcal{D}$ 
9:   Randomly select one Q-function network  $k$  from  $\{1, 2\}$  to update
10:  for  $\{s_j, a_j, R(s_j, a_j), s'_j\} \in \{s_B, a_B, R(s_B, a_B), s'_B\}$  do
11:    Compute  $\mathbb{P}_t^{(s_j, a_j)}[DDPG], \mathbb{P}_t^{(s_j, a_j)}[TD3]$  as Equation (20)
12:     $ADDPG \sim \text{Cat}\left([DDPG, TD3], \left[\mathbb{P}_t^{(s_j, a_j)}[DDPG], \mathbb{P}_t^{(s_j, a_j)}[TD3]\right]\right)$ 
13:    if  $ADDPG == DDPG$  then
14:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma Q(s'_j, \pi(s'_j; \bar{\phi}_t); \bar{\theta}_t^k)$ 
15:    else
16:       $Y(s_j, a_j) = R(s_j, a_j) + \gamma \min_{i \in \{1, 2\}} Q(s'_j, \pi(s'_j; \bar{\phi}_t); \bar{\theta}_t^i)$ 
17:     $\theta_{t+1}^k \approx \arg \min_{\theta_t^k} \mathbb{E}_{j \in [1, 2, \dots, |\mathcal{B}|]} [Y(s_j, a_j) - Q(s_j, a_j; \theta_t^k)]^2$ 
18:    Update policy-function network by gradient ascent
19:     $\phi_{t+1} \approx \arg \max_{\phi_t} \mathbb{E}_{j \in [1, 2, \dots, |\mathcal{B}|]} [Q(s_j, \pi(s_j; \phi_t); \theta_{t+1}^k)]$ 
20:     $\bar{\theta}_{t+1}^k \leftarrow \rho \theta_{t+1}^k + (1 - \rho) \bar{\theta}_t^k$ 
21:     $\bar{\phi}_{t+1} \leftarrow \rho \phi_{t+1} + (1 - \rho) \bar{\phi}_t$ 
22:   $s \leftarrow s'$ 

```

---

## F ADDITIONAL TABULAR MDP EXPERIMENTS

### F.1 COMPARISON WITH TEN BASELINES

Table 2 shows the comparison results in tabular MDP settings, where all values are evaluated in the final round. We consider ten comparison baselines are as follows: Q-learning (Q) Watkins et al. (1989), Double Q-learning (DQ) Hasselt (2010), Averaged Q-learning (AvgQ) Anschel et al. (2017), Maxmin Q-learning (MQ) Lan et al. (2020), Self-Correcting Q-learning (SCQ) Zhu & Rigotti (2021), Softmax Q-learning (SoftQ) Song et al. (2019), Weighted Double Q-learning (WDQ) Zhang et al. (2017), REDQ Chen et al. (2021), EBQL Peer et al. (2021), AdaEQ Wang et al. (2021). From the maximum Q-value perspective, AdaAQ is closer to zero than baselines, and only a slight overestimation of  $4.14 \times 10^{-3}$  in Multi-armed bandit. From the policy perspective, AdaAQ achieves at least  $92.00\% - 86.28\% = 5.72\%$  over baselines in Roulette. From the reward perspective, AdaAQ outperforms baselines by at least  $\frac{0.16 - (-0.206)}{0.206} = 22.33\%$  and  $\frac{-0.504 - (-0.561)}{0.561} = 10.16\%$  in Gridworld  $3 \times 3$  and  $4 \times 4$ , respectively.

### F.2 AQ AND ADAAQ WITH DIFFERENT PARAMETERS

Figure 5 shows the probability of betting nothing, i.e.,  $\Pr[\text{leave}]$ , and the average reward per step of our AQ and AdaAQ with different parameters. **AQ:** Figure 5(a,e,i) show that AQ(4, 4) lies at the top, and AQ( $M$ , 4) are always higher than Q-learning. This implies that AQ achieves better policy and higher reward than Q-learning via breaking the unidirectional positive bias propagation chains. Figure 5(b,f,j) show that AQ(4, 4) lies at the top, and AQ(4,  $N$ ) are always higher than Double Q-learning. This implies that AQ achieves better policy and higher reward than Double Q-learning via breaking the unidirectional negative bias propagation chains. **AdaAQ:** Figure 5(c,g,k) show

Table 2: The comparison results of eleven algorithms in tabular MDP settings.

Algorithm	Maximum Q-value	Pr[leave]	Average reward per step	
	Multi-armed bandit	Roulette	Gridworld $3 \times 3$	Gridworld $4 \times 4$
Q	3.35e0	40.83%	-8.55e-1	-9.41e-1
DQ	-1.72e0	26.15%	-6.65e-1	-8.45e-1
AvgQ	1.24e0	86.28%	-2.75e-1	-5.66e-1
MQ	1.18e-1	81.94%	-4.77e-1	-7.27e-1
SCQ	-9.22e-1	38.26%	-6.41e-1	-8.74e-1
SoftQ	-1.04e0	80.59%	-5.90e-1	-7.69e-1
WDQ	-6.19e-1	67.66%	-3.68e-1	-5.96e-1
REDQ	9.91e-1	38.04%	-4.56e-1	-6.36e-1
EBQL	-8.37e-1	69.65%	-2.06e-1	-5.61e-1
AdaEQ	-7.53e-1	67.72%	-7.38e-1	-9.44e-1
<b>AdaAQ (Ours)</b>	<b>4.14e-3</b>	<b>92.00%</b>	<b>-1.60e-1</b>	<b>-5.04e-1</b>

that AdaAQ with different  $\tau$  overlap, and always lie higher than Q-learning and Double Q-learning. This implies that AdaAQ is not sensitive to  $\tau$ , and achieves better policy and higher reward than Q-learning and Double Q-learning. **Comparison:** Figure 5(d,h,l) show that AdaAQ(1) lies higher than AQ(4, 4). This implies that AdaAQ solves the challenge in Alternating Q-learning, i.e., the optimal alternating parameters ( $M, N$ ) are unknown in advance and dynamically for different state-action pairs, via the softmax strategy with the absolute value of TD error.

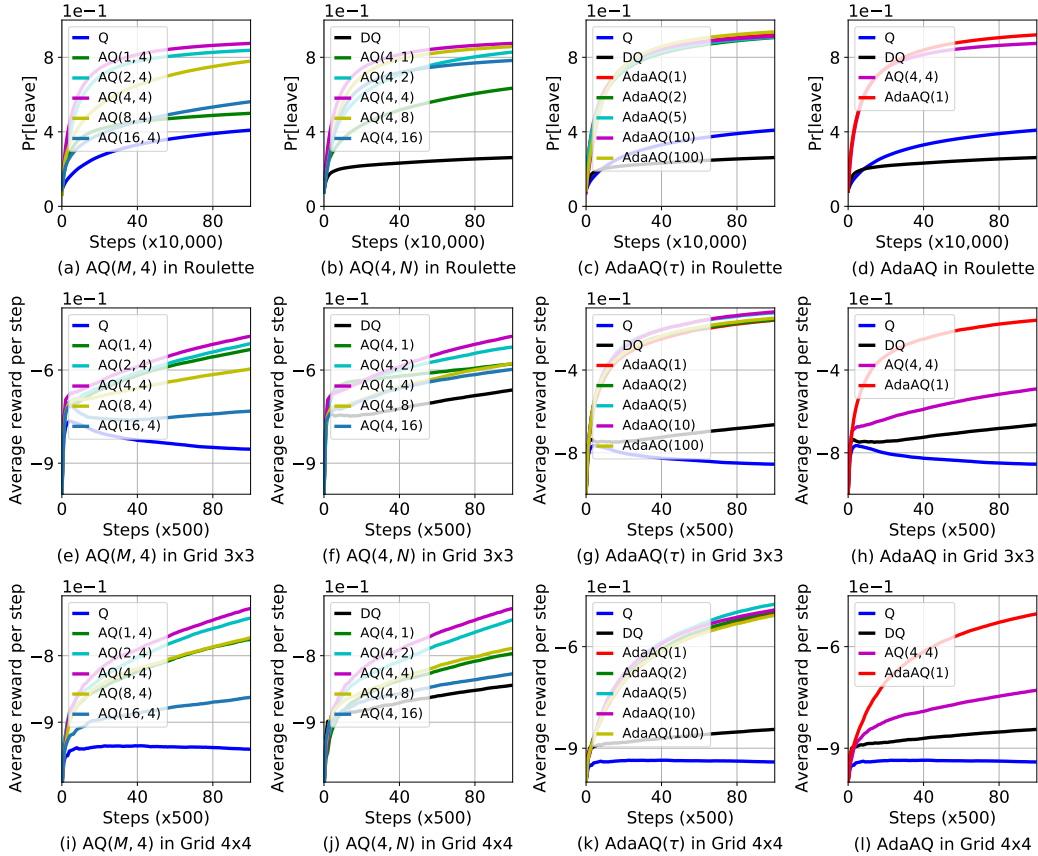


Figure 5: The performance of our AQ and AdaAQ with different parameters in Roulette, Gridworld (Grid)  $3 \times 3$  and  $4 \times 4$ .

## G ADDITIONAL DISCRETE-ACTION DRL EXPERIMENTS

### G.1 DISCRETE-ACTION DRL EXPERIMENT SETTINGS

**Discrete-action DRL environments:** We choose three discrete-action DRL experiment environments from PLE Urtans & Nikitenko (2018) and MinAtar Young & Tian (2019): Pixelcopter, Breakout, Asterix.

**Parameter settings:** We reuse the neural networks in Lan et al. (2020), and set  $\gamma = 0.99$ ,  $|\mathcal{B}| = 32$  by default. Pixelcopter uses  $|\mathcal{D}| = 10,000$ ,  $V = 200$ ,  $\alpha = 0.001$ ;  $\varepsilon$  decreases linearly from 1.0 to 0.01 in 1,000 steps, then fixes to 0.01. Breakout and Asterix use  $|\mathcal{D}| = 100,000$ ,  $V = 1,000$ ,  $\alpha = 0.01$ ;  $\varepsilon$  decreases linearly from 1.0 to 0.1 in 100,000 steps, then fixes to 0.1. Note that all experiment results are averaged over 20 runs.

**Comparison baselines:** We vary  $M = 1, 2, 4, 8, 16$ ;  $N = 1, 2, 4, 8, 16$  for our ADQN; vary  $\tau = 1, 2, 5, 10, 100$  for our AdaADQN, and set  $\tau = 1$  by default. We consider eight comparison baselines as: Averaged DQN (AvgDQN) Anschel et al. (2017), Maxmin DQN (MDQN) Lan et al. (2020), Self-Correcting DQN (SCDQN) Zhu & Rigotti (2021), Softmax DQN (SoftDQN) Song et al. (2019), Weighted Double DQN (WDDQN) Zhang et al. (2017), REDQ Chen et al. (2021), EBQL Peer et al. (2021), AdaEQ Wang et al. (2021). For a fair comparison, we set the number of Q-function network of AvgDQN and MDQN as 2; the temperature parameter of SoftDQN as 1. For other baselines, we set the self-correcting parameter of SCDQN as 2; the adaptive adjustment parameter of WDDQN as 1; the number of Q-function network of REDQ, EBQL, AdaEQ as 10; which are recommended and fine-tuned.

### G.2 COMPARISON WITH TEN BASELINES

Table 3: The comparison results of eleven algorithms in discrete-action DRL settings.

Algorithm	Average score per episode		
	Pixelcopter	Breakout	Asterix
DQN Mnih et al. (2015)	13.99	4.97	0.71
DDQN Van Hasselt et al. (2016)	15.44	5.51	0.86
AvgDQN Anschel et al. (2017)	22.11	8.68	6.46
MDQN Lan et al. (2020)	13.89	5.94	0.80
SCDQN Zhu & Rigotti (2021)	11.26	0.90	0.57
SoftDQN Song et al. (2019)	6.85	9.74	2.32
WDDQN Zhang et al. (2017)	13.85	5.23	0.91
REDQ Chen et al. (2021)	30.03	9.53	8.37
EBQL Peer et al. (2021)	30.13	10.64	7.60
AdaEQ Wang et al. (2021)	12.69	9.87	9.77
<b>AdaADQN(Ours)</b>	<b>37.89</b>	<b>13.89</b>	<b>16.13</b>

### G.3 ADQN AND ADAADQN WITH DIFFERENT PARAMETERS

Figure 6 shows the average score per episode of our ADQN and AdaADQN with different parameters, where the score is averaged over the last 100 episodes and the shaded area represents one standard error. **ADQN:** Figure 6(a,e,i) show that ADQN(4, 4) lies at the top, and ADQN( $M$ , 4) are always higher than DQN. This implies that ADQN achieves higher average score than DQN via breaking the unidirectional positive bias propagation chains. Figure 6(b,f,j) show that ADQN(4, 4) lies at the top, and ADQN(4,  $N$ ) are always higher than DDQN. This implies that ADQN achieves higher average score than DDQN via breaking the unidirectional negative bias propagation chains. **AdaADQN:** Figure 6(c,g,k) show that AdaADQN with different  $\tau$  overlap, and always lie higher than DQN and DDQN. This implies that AdaADQN is not sensitive to  $\tau$ , and achieves higher average score than DQN and DDQN. **Comparison:** Figure 6(d,h,l) show that AdaADQN(1) lies higher than ADQN(4, 4). This implies that AdaADQN solves the challenge of ADQN via the softmax strategy with the absolute value of TD error in discrete-action DRL settings.



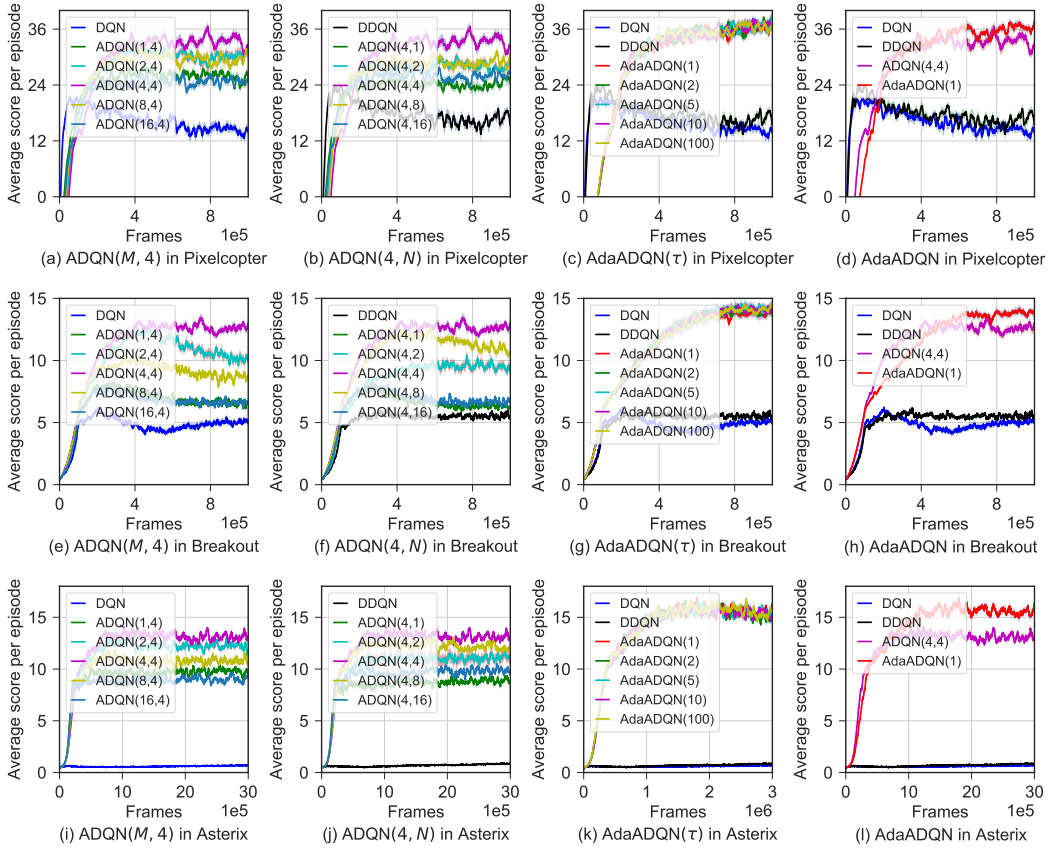


Figure 6: The average score per episode of our ADQN and AdaADQN with different parameters.

Table 4 shows the results of our ADQN and AdaADQN with different parameters, where all values are evaluated in the final round. One can observe that AdaADQN improves the average score per episode over ADQN by at least  $\frac{38.25-32.33}{32.33} = 18.31\%$ ,  $\frac{14.37-12.81}{12.81} = 12.18\%$ , and  $\frac{16.13-13.16}{13.16} = 22.57\%$  in Pixelcopter, Breakout, and Asterix, respectively.

## H ADDITIONAL CONTINUOUS-ACTION DRL EXPERIMENTS

### H.1 CONTINUOUS-ACTION DRL EXPERIMENT SETTINGS

**Continuous-action DRL environments:** We choose three continuous-action DRL experiment environments from Mujoco Brockman et al. (2016): Hopper, Ant, and Walker2d.

**Parameter settings:** We reuse the neural networks in Chen et al. (2021), and set  $\gamma = 0.99$ ;  $\alpha = 0.0003$ ;  $|\mathcal{D}| = 1,000,000$ ;  $|\mathcal{B}| = 256$ ;  $\rho = 0.005$  for all environments. Note that all experiment results are averaged over 20 runs.

**Comparison baselines:** We vary  $M = 1, 2, 4, 8, 16$ ;  $N = 1, 2, 4, 8, 16$  for our ADDPG; vary  $\tau = 1, 2, 5, 10, 100$  for our AdaADDPG, and set  $\tau = 1$  by default. We consider five comparison baselines as: DDPG Silver et al. (2014), TD3 Fujimoto et al. (2018) Averaged DDPG (AvgDDPG) Ansel et al. (2017), REDQ Chen et al. (2021), AdaEQ Wang et al. (2021). For a fair comparison, we set the number of Q-function network of DDPG, TD3, and AvgDDPG as 2. For other baselines, we set the number of Q-function network of REDQ and AdaEQ as 10. Note that these hyperparameters of comparison baselines are recommended and fine-tuned.

Table 4: The results of our ADQN and AdaADQN with different parameters.

Algorithm	Average score per episode		
	Pixelcopter	Breakout	Asterix
DQN Mnih et al. (2015)	13.99	4.97	0.71
DDQN Van Hasselt et al. (2016)	15.44	5.51	0.86
ADQN(1,4)	25.66	6.44	9.58
ADQN(2,4)	31.19	10.07	11.89
ADQN(4,4)	32.33	12.81	13.16
ADQN(8,4)	30.59	8.48	10.64
ADQN(16,4)	25.93	6.66	9.44
ADQN(4,1)	24.42	6.14	8.74
ADQN(4,2)	28.80	9.38	10.76
ADQN(4,8)	30.55	10.48	12.19
ADQN(4,16)	26.51	6.91	9.64
AdaADQN(1)	37.89	13.89	<b>16.13</b>
AdaADQN(2)	37.25	14.14	14.98
AdaADQN(5)	<b>38.25</b>	14.30	14.92
AdaADQN(10)	36.72	14.14	15.11
AdaADQN(100)	37.72	<b>14.37</b>	15.66

## H.2 COMPARISON WITH FIVE BASELINES

Table 5: The comparison results of six algorithms in continuous-action DRL settings.

Algorithm	Average return		
	Hopper	Ant	Walker2d
DDPG Silver et al. (2014)	1969.16	746.07	2733.22
TD3 Fujimoto et al. (2018)	1989.43	1466.34	3197.60
AvgDDPG Anschel et al. (2017)	2382.33	1922.99	3756.41
REDQ Chen et al. (2021)	2977.59	2818.44	4248.04
AdaEQ Wang et al. (2021).	3033.64	2354.71	4103.50
<b>AdaADDPG(Ours)</b>	<b>3296.87</b>	<b>3321.97</b>	<b>4786.20</b>

## H.3 ADDPG AND ADAADDPG WITH DIFFERENT PARAMETERS

Figure 7 shows the average return of our ADDPG and AdaADDPG with different parameters, where the average return is averaged over the last 10 episodes. **ADDPG:** Figure 7(a,e,i) show that ADDPG(4, 4) lies at the top, and ADDPG( $M$ , 4) are always higher than DDPG. This implies that ADDPG achieves higher average return than DDPG via breaking the unidirectional positive bias propagation chains. Figure 7(b,f,j) show that ADDPG(4, 4) lies at the top, and ADDPG(4,  $N$ ) are always higher than TD3. This implies that ADDPG achieves higher average return than TD3 via breaking the unidirectional negative bias propagation chains. **AdaADDPG:** Figure 7(c,g,k) show that AdaADDPG with different  $\tau$  overlap, and always lie higher than DDPG and TD3. This implies that AdaADDPG is not sensitive to  $\tau$ , and achieves higher average return than DDPG and TD3. **Comparison:** Figure 7(d,h,l) show that AdaADDPG(1) lies higher than ADDPG(4, 4). This implies that AdaADDPG solves the challenge of ADDPG via the softmax strategy with the absolute value of TD error in continuous-action DRL settings.

Table 6 shows the results of our ADDPG and AdaADDPG with different parameters, where all values are evaluated in the final round. One can observe that AdaADDPG improves the average return over ADDPG by at least  $\frac{3315.33-3098.15}{3098.15} = 7.01\%$ ,  $\frac{3450.58-2868.30}{2868.30} = 20.30\%$ , and  $\frac{4952.42-4604.37}{4604.37} = 7.56\%$  in Hopper, Ant, and Walker2d, respectively.

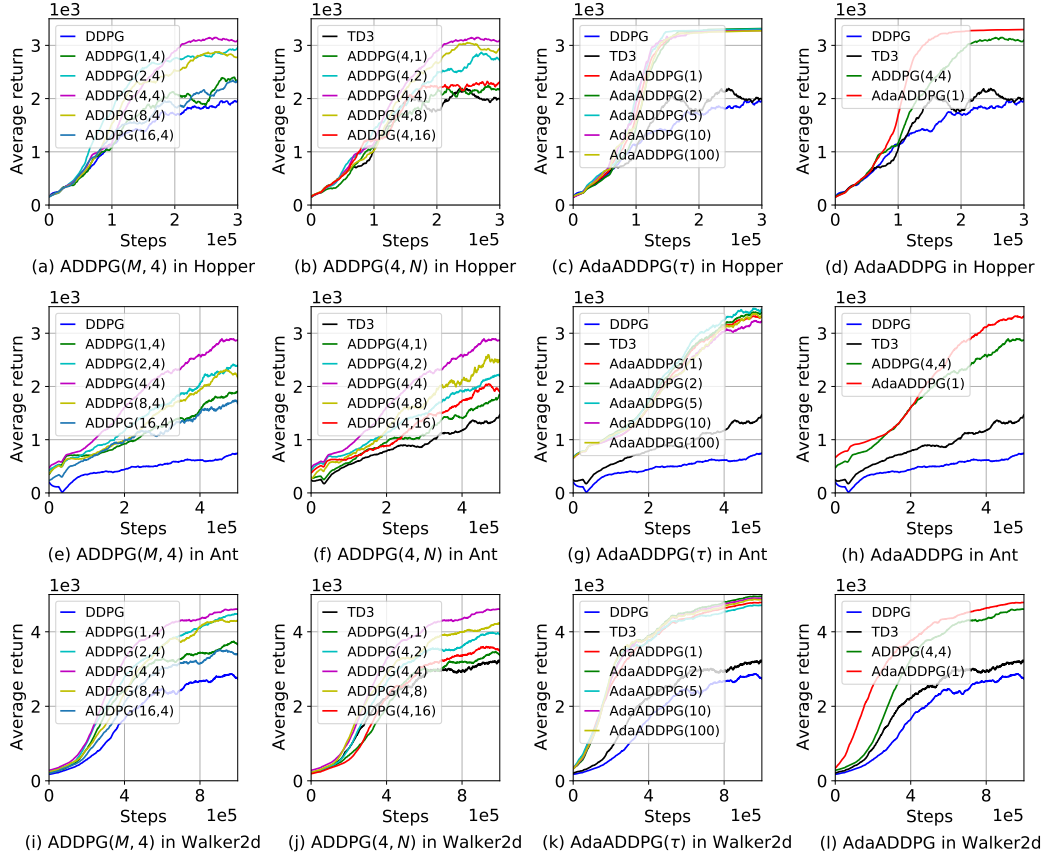


Figure 7: The average return of our ADDPG and AdaADDPG with different parameters.

Table 6: The results of our ADDPG and AdaADDPG with different parameters.

Algorithm	Average return		
	Hopper	Ant	Walker2d
DDPG Silver et al. (2014)	1969.16	746.07	2733.22
TD3 Fujimoto et al. (2018)	1989.43	1466.34	3197.60
ADDPG(1,4)	2319.00	1903.71	3656.51
ADDPG(2,4)	2961.98	2397.20	4469.58
ADDPG(4,4)	3098.15	2868.30	4604.37
ADDPG(8,4)	2774.16	2164.15	4298.21
ADDPG(16,4)	2332.61	1678.27	3404.21
ADDPG(4,1)	2178.42	1844.16	3371.05
ADDPG(4,2)	2713.00	2193.61	3927.11
ADDPG(4,8)	2941.71	2499.53	4225.42
ADDPG(4,16)	2317.31	1899.04	3474.53
AdaADDPG(1)	3296.87	3321.97	4786.20
AdaADDPG(2)	<b>3315.33</b>	3390.50	<b>4952.42</b>
AdaADDPG(5)	3306.26	<b>3450.58</b>	4711.62
AdaADDPG(10)	3278.40	3228.09	4901.93
AdaADDPG(100)	3273.39	3335.35	4864.64