Optimism in Q-learning: A Unified Overview and Empirical Comparison

Chenyu Liu

Department of Electrical & Computer Engineering

McGill University

Montréal, Québec, Canada

May 25, 2022

A report presented for the degree of Masters of Electrical Engineering

Abstract

In this report, we review the problem of reinforcement learning [1] in tabular and episodic environments. We focus on Q-learning, a model-free RL algorithm that directly updates value functions without modeling the environment. Previous work suggests that model-free RL algorithms may need more samples than model-based RL algorithms, hence, the design of sample-efficient Q-learning algorithms is an important area of research.

We review four existing Q-learning algorithms with provable regret guarantees presented in [2] [3] [4] using a unified framework. We provide a brief overview of optimism in Q-learning, and an empirical comparison of these algorithms in several environments.

Abrégé

Dans ce rapport, nous examinons le problème de l'apprentissage par renforcement [1] dans des environnements tabulaires et épisodiques. Nous nous concentrons sur le Q-learning, un algorithme RL sans modèle qui met directement à jour les fonctions de valeur sans modéliser l'environnement. Des travaux antérieurs suggèrent que les algorithmes RL sans modèle peuvent nécessiter plus d'échantillons que les algorithmes RL basés sur un modèle, par conséquent, la conception d'algorithmes d'apprentissage Q efficaces par échantillon est un domaine de recherche important.

Nous passons en revue quatre algorithmes [2] [3] [4] d'apprentissage Q existants avec des garanties de regret prouvables présentées à l'aide d'un cadre unifié. Nous donnons un bref aperçu de l'optimisme dans le Q-learning et une comparaison empirique de ces algorithmes dans plusieurs environnements.

Acknowledgements

First, I would like to thank my supervisor, Prof. Aditya Mahajan, for his encouragement and guidance during my study at McGill. His enthusiasm in research have made a huge impact on me, and his clear advice and suggestions helped me carry out the research. It is a great honor for me to complete the degree under his guidance.

I wish to thank all the students in the Systems and Control lab at McGill, their research works have broaden my view and horizon.

I also want to express my appreciation to McGill University, for providing excellent courses and warm atmosphere.

Finally, I would like to thank my family, who have given me the strongest support during my study.

Contents

List of Figures

List of Tables

List of Acronyms

MDP Markov Decision Process.

OFU Optimism in the Face of Uncertainty.

PAC Probably Approximately Correct.

RL Reinforcement Learning.

UCB Upper Confidence Bound.

Chapter 1

Introduction

In reinforcement learning (RL), an agent interacts with an unknown environment, and aims to maximize the sum of collected rewards. At each time step, the agent observes the current state and interacts with the environment by taking an action, then the environment yields a reward and the state evolves according to an unknown model, and the agent observes the next state and reward [1].

There are mainly two types of RL algorithms: model-based and model-free learning. In model-based RL algorithms, the agent learns a model based on the past experience, and chooses a policy based on this learned model. In model-free RL algorithms, the agent dispenses with the model, and only maintains a set of value functions and/or the policies. There has been a long debate on the relative pros and cons of the two approaches.

From the classical Q-learning algorithm [5], to modern DQN [6], A3C [7], and others,

most state-of-the-art RL algorithms use the model-free paradigm. Model-free RL algorithms have many advantages, such as being more efficient in time and space, and being more simple and flexible. These pros explain the great success of model-free algorithms in modern deep RL applications.

On the other hand, it is believed that model-free algorithms may suffer from worse learning performance, usually in terms of regret or sample complexity, than model-based algorithms, since model-based algorithms may be able to take advantage of the learned model. This has been empirically revealed in [8] [9]. Yet there is little theory to support such claim, which requires a more quantitative understanding of sample complexity. Hence, a natural and intriguing theoretical question about reinforcement learning algorithms is the following:

Can we design model-free algorithms with competitive learning efficiency?

Especially, **is Q-learning provably efficient?**

The answer remains elusive even in the basic tabular setting where the number of states and actions are finite. In this report, we review the literature on sample efficiency of Q-learning in episodic Markov Decision Process (MDP) formalism.

As widely seen in the research of multi-armed bandits problems, the key to obtaining good sample efficiency lies in managing the trade-off between exploration and exploitation. The agent needs a balanced strategy to explore the unknown environment while maximizing the sum of collected rewards.

One approach that allows these (both model-based and model-free) algorithms to achieve sample efficiency is the use of Optimism in the Face of Uncertainty (OFU). In the model-based setting, particularly the value iteration method, a recent line of research has taken the ideas of upper confidence bounds (UCB) in multi-armed bandits problem, and has obtained asymptotically optimal sample efficiency [10] [11].

In terms of efficiency of an algorithm, two related evaluation criteria are used. Some works provide Probably Approximately Correct (PAC) guarantee on the time required to achieve near-optimal performance [12]. Other works establish efficiency by showing that the regret of the algorithm, i.e., the total loss of sum of rewards incurred while learning, grows sub-linearly with respect to the number of interactions[1]. A unified view of PAC and regret is presented in [12].

In this report, we summarize several previous works under the category of Q-learning algorithm in [2] [3] [4]. To the best of our knowledge, this report is the first work to provide numerical evaluation of the above algorithms on several MDP environments.

[1] See Chapter 2 for the formal definitions.

Chapter 2

Preliminaries

2.1 Episodic Non-stationary Markov Decision Process

In this report, we consider tabular episodic Markov Decision Process (MDP) denoted by the tuple ($\mathcal{S}$,$\mathcal{A}$,H,$\mathbb{P}$,r), where $\mathcal{S}$ is the set of states with $|\mathcal{S}| = \mathsf{S}$, $\mathcal{A}$ is the set of actions with $|\mathcal{A}| = \mathsf{A}$. H is the horizon, i.e., the episode length, $\mathbb{P}$ is the horizon-dependent state transition matrix so that $\mathbb{P}_h(\cdot|s,a)$ gives the distribution over states if action a is taken at state s and step h[1]. $r_h : \mathcal{S} \times \mathcal{A} \times [H] \to [0,1]$ is the reward function[2]. Without loss of generality, we assume that all rewards are bounded by $[0,1]$, since any finite MDP can be rescaled to meet this setting.

At the beginning of each episode, an initial state s_1 is picked arbitrarily. Then, at each

[1]The non-stationarity implies the distribution $\mathbb{P}_h(\cdot|s,a)$ may be different at different step h.
[2]We use $[H]$ to denote the set $\{1,2,\ldots,H\}$.

step $h \in [H]$, the agent observes state $s_h \in \mathcal{S}$, selects an action $a_h \in \mathcal{A}$ based on the agent's policy, then collects reward $r_h(s_h, a_h)$, and transitions to a next state s_{h+1}, which is drawn from the distribution $\mathbb{P}_h(\cdot|s_h, a_h)$. Every episode ends at state s_{H+1}.

A policy π of an agent is a set of functions $\{\pi_h : \mathcal{S} \to \mathcal{A}\}_{h\in[H]}$. Then, the state value function at step h under policy π is denoted as $V_h^\pi : \mathcal{S} \to \mathbb{R}$, so that $V_h^\pi(s)$ indicates the expected sum of received rewards under policy π, starting from state s, and until the end of that episode. Note that we assume there is no discount in the summation. Mathematically:

$$V_h^\pi(s) := \mathbb{E}_\pi\left[\sum_{h'=h}^{H} r_{h'}(s_{h'}, \pi_{h'}(s_{h'}))\middle| s_h = s\right] \tag{2.1}$$

Similarly, we also define action value function at step h as $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, gives the expected sum of received rewards under policy π, starting from state s and action a, and until the end of that episode. Mathematically:

$$Q_h^\pi(s, a) := r_h(s, a) + \mathbb{E}_\pi\left[\sum_{h'=h+1}^{H} r_{h'}(s_{h'}, \pi_{h'}(s_{h'}))\middle| s_h = s, a_h = a\right] \tag{2.2}$$

Note that due to our assumption of bounded reward, $V_h^\pi(s)$ and $Q_h^\pi(s, a)$ are bounded between 0 and $H - h + 1$ for any triple (s, a, h).

Because the state, action and horizon are all finite, it is guaranteed that there always exists an optimal policy π^*, which gives the optimal value $V_h^*(s) = \max_\pi V_h^\pi(s)$ and $Q_h^*(s, a) = \max_\pi Q_h^\pi(s, a)$ for all triple (s, a, h).

Then, at step $h = H + 1$, $V_{H+1}^{\pi}(s) = 0$ and $V_{H+1}^{*}(s) = 0$.

And for steps $h \in [H]$, the Bellman equation and the Bellman optimality equation are:

$$\begin{cases} V_h^{\pi}(s) = Q_h^{\pi}(s, \pi_h(s)) \\ Q_h^{\pi}(s,a) = (r_h + \mathbb{P}_h V_{h+1}^{\pi})(s,a) \end{cases} \tag{2.3}$$

$$\begin{cases} V_h^{*}(s) = \max_{a \in A} Q_h^{*}(s,a) \\ Q_h^{*}(s,a) = (r_h + \mathbb{P}_h V_{h+1}^{*})(s,a) \end{cases} \tag{2.4}$$

2.2 Learning Regret

2.2.1 Definition

Notation. We use $O(\cdot)$ to denote the upper bound on an algorithm, i.e., the worst case performance. We use $\widetilde{O}(\cdot)$ to denote the simplified upper bound, in which lower order terms are hidden. We use $\Omega(\cdot)$ to denote the lower bound on an algorithm, i.e., the most efficient case possible. We also use $\text{poly}(\cdot)$ to denote the polynomial terms.

The agent interacts with the environment for K episodes $k = 1, 2, \ldots, K$. At the start of k-th episode, an initial state s_1^k is picked randomly, and the agent chooses a policy π_k. The

regret is then defined as:

$$\text{Regret}(K) = \sum_{k=1}^{K} \left[V_1^*(s_1^k) - V_1^{\pi_k}(s_1^k) \right] \tag{2.5}$$

Note that Regret(k) is the indicator of the convergence of the optimal policy. If we can observe that, for some episode index k^*, Regret(k) is non-increasing for all episodes $k > k^*$ (see the figures in Chapter 4), then from the definiton we could derive that $V_1^*(s_1^k) = V_1^{\pi_k}(s_1^k)$ for all $k > k^*$. Then, due to the uniqueness of optimal policy, we have $\pi_k = \pi^*$ for all $k > k^*$, i.e., the algorithm has converged to the optimal policy.

2.2.2 Lower Bound

For the non-stationary episodic setting, the lower bound on regret is $\Omega(\sqrt{H^3SAK})$. Formally stated as the following theorem:

Theorem 2.1 *For any algorithm, there exists an H-episodic MDP with S states and A actions, such that for any episode number K, the lower bound on regret is* $\Omega(\sqrt{H^3SAK})$.

A rigorous proof the above theorem is presented in [13], in which a specific class of hard MDPs is constructed, and then learning is performed on it to obtain the lower bound on regret.

2.2.3 Algorithmic Upper Bound

In this report, we examine the four state-of-the-art optimistic Q-learning algorithms: UCB-H [2], UCB-B [2], UCB-ADVANTAGE [3], and UCBMQ [4]. All of the four algorithms adopt UCB-based exploration, and the main contribution of the above works is that they provide a high probability upper bound on regret with rigorous proof. The generalized theorem is stated as below:

Theorem 2.2 *For any $p \in (0,1)$, with probability $1-p$, algorithm's regret on any MDP is upper bounded by $\widetilde{O}(\sqrt{\text{poly}(H)SAK\ell})$.*

Note that some lower order terms and the term $\ell = \log(SAKH/p)$ are hidden in the $\widetilde{O}(\cdot)$ notation, and the accurate regret upper bound is presented in the table below.

Algorithm	**Regret Upper Bound**
UCB-H [2]	$O(\sqrt{H^5SAK\ell} + H^2SA)$
UCB-B [2]	$O(\sqrt{H^4SAK\ell} + \sqrt{H^9S^3A^3}\ell^2)$
UCB-ADVANTAGE [3]	$O(\sqrt{H^3SAK\ell} + \sqrt{H^3K\ell}\log(HK) + H^{33/4}S^2A^{3/2}K^{1/4}\ell)$
UCBMQ [4]	$O(\sqrt{H^3SAK\ell} + H^4SA\ell)$

Table 2.1: Table of regret upper bound.

As shown in the table, two state-of-the-art Q-learning algorithms, UCB-ADVANTAGE [3] and UCBMQ [4], can achieve asymptotic near-optimal performance, up to a poly-logarithmic factor of K and p.

Notably, UCB-based exploration is also proved to be efficient in model-based RL algorithms. Recently, a model-based algorithm with UCB exploration is proposed in [14],

and the algorithm regret approaches the lower bound up to logarithmic terms. The above results indicate that model-free RL algorithm is as competitive as model-based RL algorithm in terms of regret.

2.2.4 Relationship of Lower and Upper Bound

Theorem 2.2 points out that when performing certain algorithm on any MDP, the corresponding regret at any episode number K, i.e., Regret(K), is upper bounded. In contrast, **Theorem** 2.1 shows that, for any algorithm, any MDP parameters H, S, A and any episode number K, there exists a MDP with parameters H, S, A, such that when performing any algorithm on this MDP, Regret(K) is lower bounded. The key difference is that the episode number K is fixed here, instead of being a variable in **Theorem** 2.2. To verify the two theorems, one may recur the MDP proposed in [13], test different algorithms and oberserve Regret(K) at the given K.

Chapter 3

A Unified Algorithm Framework

In this report, we look at the four state-of-the-art optimistic Q-learning algorithms: UCB-H [2], UCB-B [2], UCB-ADVANTAGE [3], and UCBMQ [4]. We unify these algorithms into a generalized framework, which is shown as Algorithm 1.

We use (s_h^k, a_h^k) to denote state-action pair visited at episode k and step h. We use $Q_h(s, a)$ and $V_h(s)$ to denote the learned action value function and value function which the algorithm maintains. We use $N_h^k(s, a)$ to denote the visit counter of triple (s, a, h) up to the k-th episode, without misleading it is abbreviated as n.

Algorithm 1 contains five parameters: value initialization at line 1, action selection policy at line 5, update trigger set at line 9, learned action value function update rule at line 10, and learned value function update rule at line 11. The design of the above parameters is introduced in the following sections.

Algorithm 1 Unified Optimistic Q-learning

initialize: $Q_h(s,a) \leftarrow Q_h^0$, $V_h(s) \leftarrow V_h^0$, visit counter $N_h^0(s,a) \leftarrow 0$ and other variables.
for episode $k = 1, 2, \ldots, K$ **do**
  observe initial state s_1^k,
  for step $h = 1, 2, \ldots, H$ **do**
    take action $a_h^k \leftarrow \arg\max_a \overline{Q}_h^k(s_h^k, a)$,
    receive reward r_h^k and observe next state s_{h+1}^k,
    update visit counter $n := N_h^k(s_h^k, a_h^k) \leftarrow N_h^{k-1}(s_h^k, a_h^k) + 1$,
    update other accumulators,
    if $n \in$ update trigger set $\mathbb{T}$ **then**
      update $Q_h(s_h^k, a_h^k) \leftarrow Q_{new}$,
      update $V_h(s_h^k, a_h^k) \leftarrow V_{new}$,
    end if
  end for
end for

3.1 Value Initialization

At the beginning of Algorithm 1, for each triple (s, a, h), the learned action value function and value function which the algorithm maintains are initialized as Q_h^0 and V_h^0. The design of the four algorithms is summarized in the table below:

Algorithm	$\mathbf{Q_h^0}$	$\mathbf{V_h^0}$
UCB-H [2]	H	0
UCB-B [2]	H	0
UCB-ADVANTAGE [3]	$H-h+1$	$H-h+1$
UCBMQ [4]	0	H

Table 3.1: Table of value initialization.

3.2 Action Selection Policy

We define $\overline{Q}_h^k(s,a)$ as the upper bound on $Q_h^*(s,a)$, i.e., the upper bound on optimal action value function of state-action pair (s,a) at step h. Then, at each step, as shown in our framework, the agent simply selects the action with the highest value of upper bound on corresponding optimal action value function.

In UCB-H [2], UCB-B [2] and UCB-ADVANTAGE [3], the current learned action value function $Q_h^k(s,a)$ is always a proper upper bound for all state-action pair (s,a) and at any episode k step h[1], simply as: $\overline{Q}_h^k(s,a) = Q_h^k(s,a)$.

In UCBMQ [4], the upper bound is designed to be the sum of the current learned action value function and a confidence bonus[2]: $\overline{Q}_h^k(s,a) = Q_h^k(s,a) + b_h^k(s,a)$.

Then, the confidence bonus is derived from the Bernstein concentration inequality and defined as:

$$b_h^k(s,a) := 2\sqrt{W_h^k(s,a)\frac{\zeta}{n_h^k}} + 53H^3\frac{\zeta}{n_h^k}\log(K) + \sum_{t=1}^{k}\frac{n_h^t(H+n_h^t)}{(n_h^t-1)\log(K)}(M_{s,a,h}^{t-1}(s_{h+1}^t) - V_{h+1}^t(s_{h+1}^t)), \tag{3.1}$$

where $\zeta = \log(32eHSA(2K+1)/p)$ with $p \in (0,1)$. When a state-action pair (s,a) is visited at step h for n times at episode $k_1, k_2, \ldots, k_n$, respectively, then, an empirical variance

[1] See Lemma 4.3, Lemma C.4 in [2], and Proposition 4 in [3] for formal statement and proof.
[2] See Lemma 1 in [4] for formal statement and proof.

of learned value function is constructed as:

$$W_h^k(s,a) := \frac{1}{n}\sum_{i=1}^{n}[V_{h+1}^{k_i}(s_{h+1}^{k_i}) - \frac{1}{n}\sum_{j=1}^{n} V_{h+1}^{k_j}(s_{h+1}^{k_j})]^2. \tag{3.2}$$

3.3 Update Trigger Set

The update trigger set is the trigger set of indices to update the state value function and action value function. When the visit counter of triple (s,a,h) is in the set, the update will be triggered.

In the setting of UCB-H [2], UCB-B [2] and UCBMQ [4]. the update trigger set $\mathbb{T}$ is simply the positive integer set. Intuitively, each time when a triple (s,a,h) is visited and the new sample is collected, its corresponding learned value function $V_h(s)$ and action value function $Q_h(s,a)$ are updated according to the update rule.

However, UCB-ADVANTAGE [3] adopts an novel update design, called stage-based update framework. For each triple (s,a,h), the samples observed for the triple is divided into consecutive stages, and the learned value function $V_h(s)$ and action value function $Q_h(s,a)$ are only updated when the stage comes to an end. In other words, they remain the same during each stage. The update trigger set is constructed as follows: Define $e_1 = H$, and $e_{i+1} = \lfloor(1+1/H)e_i\rfloor$. Then, $\mathbb{T} := \{\sum_{i=1}^{j} e_j | j = 1,2,3\dots\}$.

The property of such design is that the length of each stage roughly increases exponentially with growth rate of $(1+1/H)$. One benefit of the stage-based update

framework is that it helps to reduce the number of the updates to the action value function. It is studied by [15], where the authors propose a lazy update scheme of algorithm UCB-H [2]. This technique is also used in [14], where a similar stage-based update framework is successfully applied to a model-based algorithm to help achieve near-optimal performance.

3.4 Learned Action Value Function Update Rule

Since our framework is a Q-learning type algorithm, the update rule of learned action value function plays an essential role.

3.4.1 UCB-H and UCB-B

In terms of learned Q-value update, UCB-H [2] and UCB-B [2] adopt the same paradigm, that a confidence bonus b_n is directly added to the learned Q-value to encourage exploration. At episode k and step h, the learned action value function is updated as follows:

$$Q_h(s_h^k, a_h^k) \leftarrow (1-\alpha_n) \cdot Q_h(s_h^k, a_h^k) + \alpha_n \cdot [r_h^k + V_{h+1}(s_{h+1}^k) + b_n], \tag{3.3}$$

where the learning rate $\alpha_n = (H+1)/(H+n)$, and n is the shorthand of $N_h^k(s,a)$, which is the visit counter of triple (s,a,h) up to the k-th episode.

Then, the difference between UCB-H [2] and UCB-B [2] is solely the design of confidence bonus b_n. In UCB-H [2], the confidence bonus is derived from Hoeffding-type martingale

concentration inequality, and defined as:

$$b_n = c \cdot \sqrt{H^3 \ell / n}, \tag{3.4}$$

where c is an absolute constant with $c > 0$, and $\ell = \log(SAKH/p)$ with $p \in (0, 1)$.

Meanwhile, the confidence bonus of UCB-B [2] is derived from Bernstein concentration inequality. When state-action pair (s, a) is visited at step h for n times at episode index $k_1, k_2, \ldots, k_n$, respectively. We also consider an empirical variance of learned value function that can be computed by the algorithm:

$$W_h(s, a) := \frac{1}{n} \sum_{i=1}^{n} [V_{h+1}^{k_i}(s_{h+1}^{k_i}) - \frac{1}{n} \sum_{j=1}^{n} V_{h+1}^{k_j}(s_{h+1}^{k_j})]^2. \tag{3.5}$$

Then, the corresponding bonus $b_n(s, a, h)$ is defined as follows:

$$\begin{cases} \beta_n(s, a, h) := \min \left\{ c_1 (\sqrt{\frac{H\ell}{n}(W_h(s, a) + H)} + \frac{\ell}{n}\sqrt{H^7 SA}), c_2 \sqrt{\frac{H^3 \ell}{n}} \right\}, \\ b_1(s, a, h) := \frac{1}{2}\beta_1(s, a, h), \\ b_n(s, a, h) := \frac{1}{2\alpha_n}\beta_n(s, a, h) - \frac{1 - \alpha_n}{2\alpha_n}\beta_{n-1}(s, a, h), \end{cases} \tag{3.6}$$

where c_1, c_2 are absolute constants with $c_1, c_2 > 0$.

3.4.2 UCB-ADVANTAGE

UCB-ADVANTAGE [3] introduces the reference-advantage decomposition technique, which aims to learn an accurate estimation of the optimal value function, and denote it by the reference value function V_h^{ref}.

As introduced in the previous subsection, the algorithm adopts a stage-based update framework. It keeps two types of accumulators to facilitate the update: global accumulators that use the samples in all stages, and intra-stage accumulators that only use the samples in the latest stage. In the following equations, we use $\overset{+}{\leftarrow}$ to denote the accumulating process.

There are three global accumulators: $n_h(s,a)$ is used to keep the total visit number of triple (s,a,h), μ_h^{ref} and σ_h^{ref} are the sum and quadratic sum of the reference value function, defined as:

$$\begin{cases} \mu_h^{ref} := \mu_h^{ref}(s_h,a_h) \overset{+}{\leftarrow} V_{h+1}^{ref}(s_{h+1}), \\ \sigma_h^{ref} := \sigma_h^{ref}(s_h,a_h) \overset{+}{\leftarrow} \left(V_{h+1}^{ref}(s_{h+1})\right)^2. \end{cases} \tag{3.7}$$

There are four intra-stage accumulators only counting the current stage and reset as 0 at the beginning of each stage, denoted by a hat. $\hat{n}_h(s,a)$ is the visit counter of triple (s,a,h) in current stage, and:

$$\begin{cases} \hat{\mu}_h := \hat{\mu}_h(s_h,a_h) \overset{+}{\leftarrow} V_{h+1}(s_{h+1}) - V_{h+1}^{ref}(s_{h+1}). \\ \hat{v}_h := \hat{v}_h(s_h,a_h) \overset{+}{\leftarrow} V_{h+1}(s_{h+1}). \\ \hat{\sigma}_h := \hat{\sigma}_h(s_h,a_h) \overset{+}{\leftarrow} \left(V_{h+1}(s_{h+1}) - V_{h+1}^{ref}(s_{h+1})\right)^2, \end{cases} \tag{3.8}$$

Then, at episode k step h, the Q-value update rule is defined as:

$$Q_h(s_h^k, a_h^k) \leftarrow \min\left\{ r_h^k + \frac{\hat{v}_h}{\hat{n}_h} + \bar{b},\, r_h^k + \frac{\hat{\mu}_h}{\hat{n}_h} + \frac{\mu_h^{ref}}{n} + b,\, Q_h(s_h^k, a_h^k) \right\}. \tag{3.9}$$

The two exploration bonuses are defined as:

$$\begin{cases} b := c_1\sqrt{\dfrac{\sigma_h^{ref}/n - (\mu_h^{ref}/n)^2}{n}\ell} + c_2\sqrt{\dfrac{\hat{\sigma}_h^{ref}/\hat{n} - (\hat{\mu}_h^{ref}/\hat{n})^2}{\hat{n}}\ell} + c_3\left(\dfrac{H\ell}{n} + \dfrac{H\ell}{\hat{n}} + \dfrac{H\ell^{3/4}}{n^{3/4}} + \dfrac{H\ell^{3/4}}{\hat{n}^{3/4}}\right). \\ \bar{b} := 2\sqrt{\dfrac{H^2\ell}{\hat{n}}}, \end{cases} \tag{3.10}$$

where c_1, c_2, c_3 are absolute constant and $\ell = \log(2/p)$ with $p \in (0, 1)$.

Finally, when $\sum_a n_h(s_h^k, a) = c_4 H^6 SA\ell$ for a large enough universal constant $c_4 > 0$, the reference value function is updated as:

$$V_h^{ref}(s_h^k) \leftarrow V_h(s_h^k). \tag{3.11}$$

3.4.3 UCBMQ

In UCBMQ [4], a momentum term is added into the learned action value function update, the paper name it as the bias-value function and denote it by $M_{s,a,h}^k(s')$. The bias-value function are initialized as H, and when state-action pair (s, a) is visited at step h for

n times at episode index $k_1, k_2, \ldots, k_n$, respectively, its update rule is as follows:

$$M^k_{s,a,h}(s') = \eta_n V^{k-1}_{h+1}(s') + (1-\eta_n) M^{k-1}_{s,a,h}(s'), \tag{3.12}$$

where we define $\eta_n = (H+1)/(H+n)$, and n is the visit counter of the triple (s,a,h) up to k-th episode. Then, the update rule of learned action value function is given by:

$$Q_h(s^k_h, a^k_h) \leftarrow (1-\alpha_n) Q_h(s^k_h, a^k_h) + \alpha_n \left(r^k_h + V_{h+1}(s^k_{h+1})\right) + \gamma_n \left(V_{h+1}(s^k_{h+1}) - M^k_{s,a,h}(s^k_{h+1})\right), \tag{3.13}$$

where $\alpha_n = 1/n$, and $\gamma_n = \frac{n-1}{n} \cdot \frac{H}{H+n}$.

3.5 Learned Value Function Update Rule

The update rule of the learned value function is shown as below. In UCB-H [2] and UCB-B [2], the learned value function is designed to be upper bounded by H, note that this is due to the assumption that reward is bounded between $[0,1]$. In UCBMQ [4], the learned value function is designed to be non-negative and non-increasing.

Algorithm	$\boldsymbol{V_{new}}$
UCB-H [2]	$\min\{H, \max_{a\in\mathcal{A}} Q_h(s^k_h, a)\}$
UCB-B [2]	$\min\{H, \max_{a\in\mathcal{A}} Q_h(s^k_h, a)\}$
UCB-ADVANTAGE [3]	$\max_{a\in\mathcal{A}} Q_h(s^k_h, a)$
UCBMQ [4]	$\min\left\{V_h(s^k_h), \max\{0, \max_{a\in\mathcal{A}} Q_h(s^k_h, a) + b^k_h(s^k_h, a)\}\right\}$

Table 3.2: Table of learned value function update rule.

3.6 Optimism in Q-learning

In this section, we present a brief overview of optimism in Q-learning based on our algorithm framework. The principle of optimism in the face of uncertainty (OFU) is usually implemented as the use of UCB-type exploration (i.e., action selection), and UCB-type exploration depends on two aspects: value initialization and design of UCB exploration bonus. We look at the above algorithms and combine our framework to explain these two aspects.

For value initialization, as mentioned in Section 3.1, in UCB-H, UCB-B and UCB-ADVANTAGE, the learned action value function is initialized as H or $H - h + 1$. Recall that the reward is bounded by 1 and the episode length is H, therefore, the optimal action value function is upper bounded by $H - h + 1$ (or H). Hence, such overestimated initialization is optimistic and is directly used for greedy action selection. In UCBMQ, the learned action value function is initialized as 0, this is due to UCBMQ adopts different action selection policy, which an additional bonus is added to the learned action value function, then the sum is used for greedy action selection. Readers may refer to Section 3.2 for details, and similar implementation could also be found in [16].

For the design of UCB exploration bonus, usually there are two inequalities involved: Hoeffding concentration inequality and Bernstein concentration inequality. UCB-H adopts Hoeffding concentration inequality, and the result exploration bonus is rather primitive, while UCB-B, UCB-ADVANTAGE and UCBMQ all adopt Bernstein concentration

inequality, which outputs more sophisticated exploration bonus. There is also difference in the deployment of UCB exploration bonus. In UCBMQ, the bonus is only used in greedy action selection, while in the rest algorithms, the bonus is directly added into the new estimate of learned action value function, to maintain the optimism of learned action value function with respect to the optimal value. Readers may refer to Section 3.4 for details.

Chapter 4

Numerical Comparisons

In this chapter, we provide empirical results of the four algorithms presented in the report to the RiverSwim environment [17], and the GARNET environment [18].

4.1 RiverSwim Environment

In this section, we compare the algorithms' performance on RiverSwim environment [17].

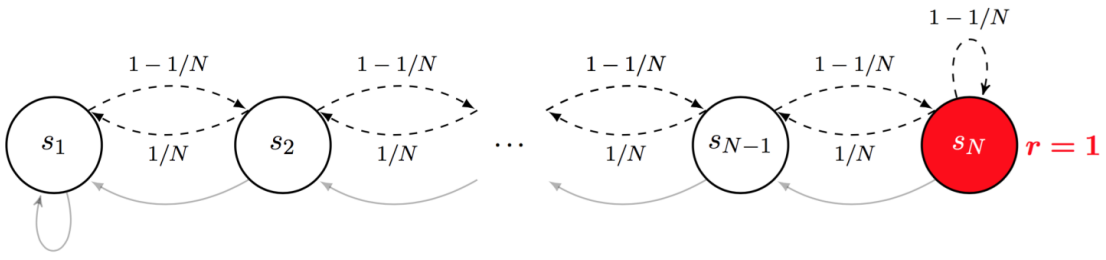

Figure 4.1: Illustration of RiverSwim MDP

The environment contains a long chain of N states, and each episode is of length $N-1$. The agent's initial state is the leftmost state s_1. At each step, the agent's actions are to swim left or right. Actions left always succeed in moving the agent to the next state at left, but actions right only succeed with probability $1-1/N$. All states have zero reward except for the rightmost state s_N which gives a reward of 1. Hence, the unique optimal policy is to go right at every step, and all other policies give zero reward.

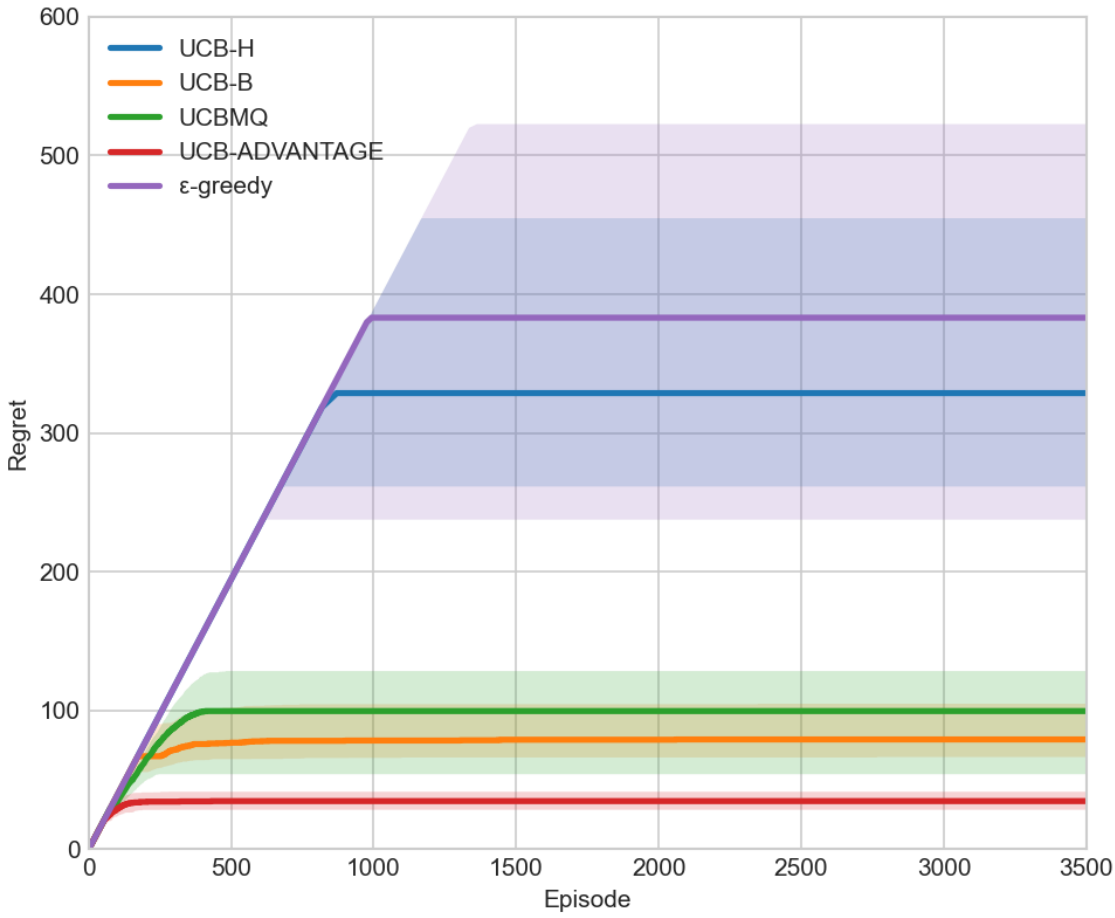

Figure 4.2: RiverSwim with $S = 9$

We show the performance of all algorithms on RiverSwim with 9 states, and we also add the $\epsilon-$greedy Q-learning for comparison. The experiment contains 100 runs, we plot the median value of the algorithms at each episode $k \in [1, K]$, and the Interquartile Range (IQR) is plotted as shadow.

4.2 Garnet Environment

Garnet [18] is an acronym for Generic Average Reward Non-stationary Environment Testbed, it is a class of randomly constructed stationary MDPs characterized by tuple (n, m, b, h). The parameters n and m are the number of states and actions respectively, the parameter b is the number of possible next states, and the parameter h is the episode length. At the beginning of each episode, the initial state is randomly selected from the state set. For each state-action pair, the possible next states are selected randomly from all states, then the state transition probability is drawn uniformly from the set. A sparse reward is constructed such that only 20% of the state-action pairs have reward 1, elsewhere the reward is 0. We set the discount factor $\lambda = 1$.

We test the algorithms on the Garnet problem with parameters $(10, 3, 1, 5)$. The experiment contains 100 runs, we plot the median value of the algorithms at each episode $k \in [1, K]$, and the Interquartile Range (IQR) is plotted as shadow. The figure is shown as below.

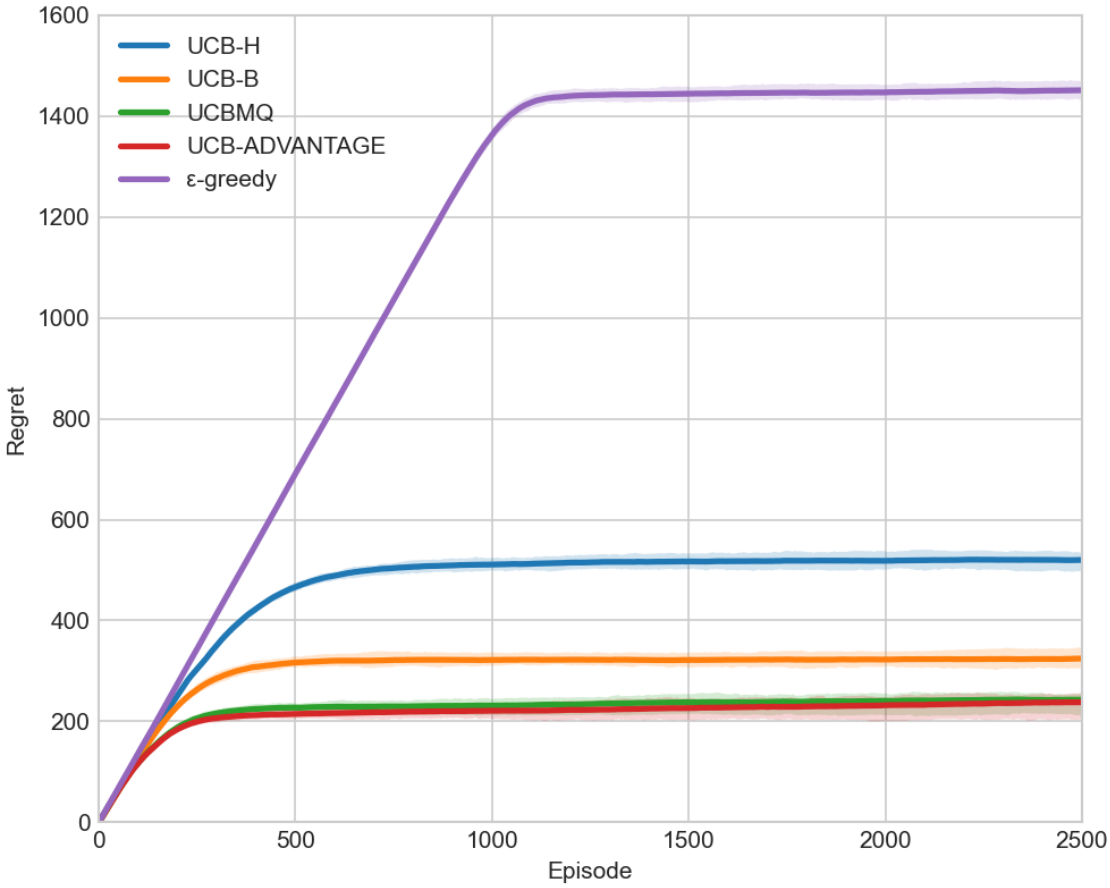

Figure 4.3: Garnet with parameter $(10, 3, 1, 5)$

4.3 Conclusion

In the above figures, we evaluate the algorithm performance by two criteria: the number of episodes to find the optimal policy, and the final regret value.

We could observe that the UCB-H algorithm has the worst performance, it is not

surprising since UCB-H uses primitive design of exploration bonus and thus has the largest regret upper bound in the above algorithms. Meanwhile, UCB-ADVANTAGE outperforms all other algorithms in every scenario, this result corresponds to the theoretical analysis that UCB-ADVANTAGE has a near-optimal regret upper bound.

We must point out that the above experiments require careful parameters tuning, and inappropriate parameter values would lead to learning failure.

Chapter 5

Conclusion and Future Works

To summarize, in this report we provide comprehensive analysis of several works of optimistic Q-learning algorithms with provably regret. We propose a generalized framework which recaps these algorithms, and we show how these general results can be used to facilitate the design of new optimistic model-free algorithms. To the extent of our knowledge, this is also the first study to evaluate their performance by presenting numerical experiments.

Future work includes general analysis of the proof of upper bound on regret, numerical comparison of optimistic Q-learning algorithms with state-of-the-art model-based algorithms, design of new optimistic Q-learning algorithms based on the proposed framework.

Another potential future work is to apply UCB exploration to continuous domains such as continuous state and action spaces, and provide performance guarantee similar to this report, readers may refer to [19] for existing work.

Bibliography

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

[2] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is q-learning provably efficient?," *Advances in neural information processing systems*, vol. 31, 2018.

[3] Z. Zhang, Y. Zhou, and X. Ji, "Almost optimal model-free reinforcement learningvia reference-advantage decomposition," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15198–15207, 2020.

[4] P. Ménard, O. D. Domingues, X. Shang, and M. Valko, "Ucb momentum q-learning: Correcting the bias without forgetting," in *International Conference on Machine Learning*, pp. 7609–7618, PMLR, 2021.

[5] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

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

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[8] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, Citeseer, 2011.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[10] M. G. Azar, I. Osband, and R. Munos, "Minimax regret bounds for reinforcement learning," in *International Conference on Machine Learning*, pp. 263–272, PMLR, 2017.

[11] P. Auer, T. Jaksch, and R. Ortner, "Near-optimal regret bounds for reinforcement learning," *Advances in neural information processing systems*, vol. 21, 2008.

[12] C. Dann, T. Lattimore, and E. Brunskill, "Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[13] O. D. Domingues, P. Ménard, E. Kaufmann, and M. Valko, "Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited," in *Algorithmic Learning Theory*, pp. 578–598, PMLR, 2021.

[14] Z. Zhang, X. Ji, and S. Du, "Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon," in *Conference on Learning Theory*, pp. 4528–4531, PMLR, 2021.

[15] Y. Bai, T. Xie, N. Jiang, and Y.-X. Wang, "Provably efficient q-learning with low switching cost," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] T. Rashid, B. Peng, W. Boehmer, and S. Whiteson, "Optimistic exploration even with a pessimistic initialisation," *arXiv preprint arXiv:2002.12174*, 2020.

[17] A. L. Strehl and M. L. Littman, "An analysis of model-based interval estimation for markov decision processes," *Journal of Computer and System Sciences*, vol. 74, no. 8, pp. 1309–1331, 2008.

[18] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[19] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *International conference on learning and intelligent optimization*, pp. 433–445, Springer, 2011.