# The Effect of Q-function Reuse on the Total Regret of Tabular, Model-Free, Reinforcement Learning

Volodymyr Tkachuk
University of Waterloo
Waterloo, Ontario
vtkachuk@uwaterloo.ca

Sriram Ganapathi Subramanian
University of Waterloo
Waterloo, Ontario
s2ganapa@uwaterloo.ca

Matthew E. Taylor
University of Alberta
Alberta Machine Intelligence Institute (Amii)
Edmonton, Alberta
matthew.e.taylor@ualberta.ca

## ABSTRACT

Some reinforcement learning methods suffer from high sample complexity causing them to not be practical in real-world situations. $Q$-function reuse, a transfer learning method, is one way to reduce the sample complexity of learning, potentially improving usefulness of existing algorithms. Prior work has shown the empirical effectiveness of $Q$-function reuse for various environments when applied to model-free algorithms. To the best of our knowledge, there has been no theoretical work showing the regret of $Q$-function reuse when applied to the tabular, model-free setting. We aim to bridge the gap between theoretical and empirical work in $Q$-function reuse by providing some theoretical insights on the effectiveness of $Q$-function reuse when applied to the $Q$-learning with UCB-Hoeffding algorithm. Our main contribution is showing that in a specific case if $Q$-function reuse is applied to the $Q$-learning with UCB-Hoeffding algorithm it has a regret that is independent of the state or action space. We also provide empirical results supporting our theoretical findings.

## KEYWORDS

Reinforcement Learning, Transfer Learning

## 1 INTRODUCTION

In reinforcement learning (RL), an agent interacts with an environment and tries to maximize its expected sum of rewards. Many algorithms, such as Q-learning with $\epsilon$-greedy exploration [5], can suffer from poor sample complexity [4]. This is a problem in real-world situations where an agent may receive a limited amount of samples to learn an optimal policy. Such real-world environments serve as motivation to reduce the sample complexity of RL algorithms.

Transfer learning (TL) is a method used in RL as one way to reduce an agent's training time [6]. The key idea is that an agent can learn a target task faster by transferring information from a previously learned source task, similar to how humans can learn algebra more quickly by transferring knowledge from previous tasks that require addition and multiplication. Although the concept of TL is intuitively appealing, its effectiveness has been mostly shown through empirical studies [7]. As such, we aim to provide new theoretical results for one TL method in RL, $Q$-function reuse.

$Q$-function reuse is the process of training an agent on a simple *source* Markov decision process (MDP) $\mathcal{M}_S$ and then transferring its learned $Q$-function to a more complex, yet related *target* MDP $\mathcal{M}_T$. The goal is to improve the sample complexity when compared to just training in $\mathcal{M}_T$ from the start. Sample complexity is loosely defined as how much data an agent must collect in order to learn a good policy [3]. If the agent was trained in $\mathcal{M}_S$ until convergence to the optimal policy, transferring the $Q$-function from $\mathcal{M}_S$ to $\mathcal{M}_T$ can sometimes be thought of as a near-optimal $Q$-function initialization in $\mathcal{M}_T$, since $\mathcal{M}_S$ is related to $\mathcal{M}_T$. Therefore, we propose that one method to study $Q$-function reuse is to study the effectiveness of near-optimal $Q$-function initialization.

Since the effectiveness of $Q$-function reuse has been mostly shown in the model-free setting [7], and it is easier to perform a theoretical analysis in the tabular domain, we choose to study the effects of $Q$-function reuse on a tabular, model-free algorithm that is provably efficient, $Q$-learning with UCB-Hoeffding [2]. In this work we study the setting where we are given the $Q$-function from some agent that has previously been trained on a simple MDP $\mathcal{M}_S$. We refer to this $Q$-function as the pre-trained $Q$-function from $\mathcal{M}_S$. We will answer the following question:

> Will the total regret of the $Q$-learning with UCB-Hoeffding algorithm be lower in a complex (target) MDP $\mathcal{M}_T$, if it is initialized with a pre-trained $Q$-function from a related, but simpler (source) MDP $\mathcal{M}_S$?

For our analysis, we assume the $Q$-function initialization is optimal for all but one value. Although this is a rather strong assumption, we believe it provides useful insights and a promising starting point for future work. In general, the $Q$-function initialization in a target MDP after $Q$-function reuse has some nearly optimal $Q$-values and some $Q$-values that are far from optimal. Therefore, to address the general case of $Q$-function reuse, future work might include increasing the number of not optimal $Q$-values to more than one and relaxing the optimal $Q$-function initialization for the remaining $Q$-values to some notion of sub-optimality. To the best of our knowledge, there has not been any theoretical work showing the total regret of $Q$-function reuse applied on a tabular, model-free algorithm. We perform a regret analysis, showing that the $Q$-learning with the UCB-Hoeffding algorithm [2], along with our initialization assumptions achieves a total regret of only $O(\sqrt{H^2T\iota'})$ (independent of the state and action space), while regular $Q$-learning with UCB-Hoeffding suffers a regret of $O(\sqrt{H^4SAT\iota})$ [2]. Empirical results are presented to support these theoretical claims.

## 2 PRELIMINARIES

We borrow standard notations from Jin et al. [2], that we will include in this section for quick reference. To provide a fair comparison between our algorithm and the $Q$-learning with UCB-Hoeffding algorithm, proposed by Jin et al. [2], we maintain a similar problem setting. Extending this work to other settings (e.g., stochastic reward, terminating states, discounting, etc.) is left for future work. We begin by describing the Markov decision process, MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$. The set of states is $\mathcal{S}$, with $|\mathcal{S}| = S$. The set of actions is $\mathcal{A}$, with $|\mathcal{A}| = A$. The number of steps per episode (horizon) is $H$. The transition dynamics are given by $\mathbb{P}$, where $\mathbb{P}_h(\cdot|x, a)$ gives the next state distribution if action $a$ was taken in state $x$ at step $h \in [H]$. The deterministic reward function is $r_h(x, a)$, which provides a reward in the range $[0, 1]$ for taking action $a$ in state $x$ at step $h$. The agent acts in this MDP for $K$ episodes. We let $T = KH$ denote the total number of steps the agent takes in the MDP.

For each episode $k \in [K]$ an initial state $x_1^k$ is chosen randomly. At each step $h$ and episode $k$ the agent observes a state $x_h^k$, takes action $a_h^k$, receives reward $r_h(x_h^k, a_h^k)$, and then transitions to its next state drawn from the distribution $\mathbb{P}_h(\cdot|x_h^k, a_h^k)$. The transition dynamics and reward function were chosen to depend on the step $h$ for generality and to remain consistent with prior work [2]. Note that if dependence on $h$ is not required then the transition dynamics and reward function can be set the same for all $h$ and all the results shown in this work will still hold. The episode ends when $x_{H+1}^k$ is reached.

There is a separate policy $\pi_h$ for each step $\{\pi_h : \mathcal{S} \to \mathcal{A}\}_{h\in[H]}$. We use $V_h^\pi : \mathcal{S} \to \mathbb{R}$ to denote the value function at step $h$ under policy $\pi$. The expected sum of rewards under policy $\pi$, from $x_h = x$ until the end of the episode is given by $V_h^\pi(x)$. This is represented as:

$$V_h^\pi(x) := \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}(x_{h'}, \pi_{h'}(x_{h'}))|x_h = x\right]$$

Similarly, we define the state-action value function as $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The expected sum of rewards under policy $\pi$, from state $x_h = x$ after taking action $a_h = a$ until the end of the episode is given by $Q_h^\pi(x, a)$. This is represented as:

$$Q_h^\pi(x, a) := r_h(x, a) + \mathbb{E}\left[\sum_{h'=h+1}^{H} r_{h'}(x_{h'}, \pi_{h'}(x_{h'}))|x_h = x, a_h = a\right]$$

Since the state space, action space, and horizon are all finite, there always exists an optimal policy $\pi^*$ which gives the optimal value $V_h^*(x) := \sup_\pi V_h^\pi(x)$, for all $x \in \mathcal{S}$ and $h \in [H]$ [1]. To simplify notation we will denote $[\mathbb{P}_h V_{h+1}](x, a) := \mathbb{E}_{x'\sim\mathbb{P}(\cdot|x,a)} V_{h+1}(x')$. Using this notation we have the Bellman equations and Bellman optimality equations as follows:

$$\begin{cases} V_h^\pi(x) = Q_h^\pi(x, \pi_h(x)) \\ Q_h^\pi(x, a) = (r_h + \mathbb{P}_h V_{h+1}^\pi)(x, a) \\ V_{H+1}^\pi(x) = 0 \quad \forall x \in \mathcal{S} \end{cases} \qquad \begin{cases} V_h^*(x) = \max_{a\in\mathcal{A}} Q_h^*(x, a) \\ Q_h^*(x, a) := (r_h + \mathbb{P}_h V_{h+1}^*)(x, a) \\ V_{H+1}^*(x) = 0 \quad \forall x \in \mathcal{S} \end{cases} \tag{1}$$

We will use $\pi_h^k$ to denote the agent's policy at episode $k$. Finally, the performance metric of interest is the total regret, defined as:

$$\text{Regret}(K) = \sum_{k=1}^{K} [V_1^*(x_1^k) - V_1^{\pi_h^k}(x_1^k)]$$

## 3 RESULTS

In this section we present our algorithm, $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization, a modified version of Algorithm 1 from Jin et al. [2]. We also introduce a theorem that shows the total regret of our algorithm is $O(\sqrt{H^2 T\iota'})$.

As a starting point to answering our question, presented in the introduction, we propose using an ideal $Q$-function initialization to model a possible pre-trained $Q$-function we might receive. We call this *Max-Optimal Initialization* because it is the maximum number of assumptions that can be made before the $Q$-function initialization becomes the optimal $Q$-function for all states, actions, and steps. In words, we initialize the $Q$-function to the optimal $Q$-function for all $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$ except for state $x1$, action $a1$ and step $h = 1$, which is initialized to $H$ (as per the $Q$-learning with UCB-Hoeffding algorithm). Since updating the $Q$-function at any of the optimal states, actions, and steps could potentially make it sub-optimal, we make the additional assumption that the $Q$-function is only updated for $(x1, a1, h = 1)$. We only keep track of how many times $(x1, a1, h = 1)$ was visited using a counter $N_1(x1, a1)$ initialized to 0. Since the $Q$-function is not updated for any other $(x, a, h)$, there is no need to keep track of how many times any other $(x, a, h)$ is visited. In mathematical notation the initialization can be stated as follows:

$$\begin{aligned} &Q_1(x1, a1) \leftarrow H \text{ and } N_1(x1, a1) \leftarrow 0 \\ &Q_h(x, a) \leftarrow Q_h^*(x, a) \text{ for all } (x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H] \backslash (x1, a1, h = 1) \end{aligned} \tag{2}$$

The Max-Optimal initialization can model the scenario where an agent is trained on some simple MDP $\mathcal{M}_0$ until convergence, but then a new action $a1$ is introduced at state $x1$ and step $h = 1$ (MDP $\mathcal{M}_F$). In $\mathcal{M}_F$ the agent is essentially initialized with an optimal $Q$-function for all $(x, a, h)$, except $(x1, a1, h = 1)$. We would like to highlight that in this scenario $\mathcal{M}_0$ is considered related to MDP $\mathcal{M}_F$ since it has the same transition dynamics and reward function for all but one state, action, and step. $\mathcal{M}_0$ is also considered simpler than $\mathcal{M}_F$ since it is exactly $\mathcal{M}_F$, except without $(x1, a1, h = 1)$. We present Algorithm 1, $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization, which combines the $Q$-learning with UCB-Hoeffding algorithm with the Max-Optimal Initialization assumptions. The modifications we made to the $Q$-learning with UCB-Hoeffding algorithm are shown in blue in Algorithm 1.

We now describe the steps performed in our algorithm. The $Q$-function and step counter $N_1(x1, a1)$ are initialized using Max-Optimal initialization. For each episode $k \in [K]$ the agent starts in a random state $x_1^k$. Then, for each step $h \in [H]$ and state $x_h^k \in \mathcal{S}$ the agent selects the action $a \in \mathcal{A}$ that maximizes its current estimate of $Q_h(x, a)$. The next state $x_{h+1}^k \in \mathcal{S}$ is sampled from $\mathbb{P}_h(\cdot|x, a)$. If the current state, action and step $(x_h^k, a_h^k, h)$ is $(x1, a1, h = 1)$, the

agent updates its $Q$-function using the following rule:

$$Q_h(x_h^k, a_h^k) \leftarrow (1-\alpha_t)Q_h(x_h^k, a_h^k) + \alpha_t[r_h(x_h^k, a_h^k) + V_{h+1}(x_{h+1}^k) + b_t] \tag{3}$$

where $t$ is the step counter for how many times the agent has visited $(x1, a1)$ at step $h = 1$, $b_t$ is the confidence bonus indicating the agents confidence in its $Q$-value at $(x1, a1, h = 1)$, and the learning rate $\alpha$ is $\alpha_t := \frac{H+1}{H+t}$. This choice of learning rate $\alpha_t$ is crucial to obtain a total regret that is not exponential in $H$ [2].

We removed the update $V_h(x_h) \leftarrow \min\{H, \max_{a' \in \mathcal{A}} Q_h(x_h, a')\}$ since the $Q$-function in our algorithm is only updated for $(x1, a1, h = 1)$, which requires knowledge of $V_{h+1}^k(x_{h+1}^k)$. But $V_{h+1}^k(x_{h+1}^k) = V_{h+1}^*(x_{h+1}^k)$ for $h \geq 1$ due to the Max-Optimal initialization, meaning we never need to update $V_{h+1}^k(x_{h+1}^k)$.

We present the following theorem for the $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization algorithm:

Theorem 1 (Hoeffding Max-Optimal). *There exists an absolute constant $c > 0$ such that, for any $p \in (0, 1)$, if we choose $b_t = c\sqrt{H^3\iota'/t}$, then with probability $1 - p$ the total regret of $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization (Algorithm 1) is at most $O(\sqrt{H^2T\iota'})$, where $\iota' := log(K/p)$.*

Note that we reserve $\iota$ for when we refer to the $Q$-learning with UCB-Hoeffding algorithm, where $\iota := log(SAT/p)$. In our algorithm this term is reduced to $\iota' := log(K/p)$ due to our added assumptions.

**Algorithm 1:** $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization

Initialize $Q_1(x1, a1) \leftarrow H$ and $N_1(x1, a1) \leftarrow 0$
Initialize $Q_h(x, a) \leftarrow Q_h^*(x, a)$ for all $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H] \backslash (x1, a1, h = 1)$
**for** *episode* $k = 1, ..., K$ **do**
  receive $x_1^k$
  **for** *step* $h = 1, ..., H$ **do**
    Take action $a_h^k \leftarrow \text{argmax}_{a'} Q_h(x_h^k, a')$, and observe $x_{h+1}^k$
    **if** $x_h^k = x1$ *and* $a_h^k = a1$ *and* $h = 1$ **then**
      $t = N_h(x_h^k, a_h^k) \leftarrow N_h(x_h^k, a_h^k) + 1$;
      $b_t \leftarrow c\sqrt{H^3\iota'/t}$
      $Q_h(x_h^k, a_h^k) \leftarrow (1-\alpha_t)Q_h(x_h^k, a_h^k) + \alpha_t[r_h(x_h^k, a_h^k) + V_{h+1}(x_{h+1}^k) + b_t]$
    **end**
  **end**
**end**

## 4 PROOF FOR Q-LEARNING WITH UCB-HOEFFING AND MAX-OPTIMAL INITIALIZATION

We provide a full proof of Theorem 1, following similar steps and notation to that mentioned in Jin et al. [2]. We first introduce some notation for convenience.

We denote by $\mathbb{I}[A]$ as the indicator function for an event $A$. Recall that $[\mathbb{P}_h V_{h+1}](x, a) := \mathbb{E}_{x' \sim \mathbb{P}_h(\cdot|x,a)} V_{h+1}(x')$. We now introduce its empirical counterpart, $[\hat{\mathbb{P}}_h^k V_{h+1}](x, a) := V_{h+1}(x_{h+1}^k)$, which is defined only for $(x, a) = (x_h^k, a_h^k)$. Recalling that $\alpha_t = \frac{H+1}{H+t}$, we introduce the following:

$$\alpha_t^0 = \prod_{j=1}^{t}(1-\alpha_j), \qquad \alpha_t^i = \alpha_i \prod_{j=i+1}^{t}(1-\alpha_j) \tag{4}$$

Recall that $\sum_j^{t<j}(\cdot) = 0$ and $\prod_j^{t<j}(\cdot) = 1$. Therefore, the following properties hold:

$$\sum_{i=1}^{t} \alpha_t^i = 1 \text{ and } \alpha_t^0 = 0 \text{ for } t \geq 1, \quad \sum_{i=1}^{t} \alpha_t^i = 0 \text{ and } \alpha_t^0 = 1 \text{ for } t = 0$$

The motivation for introducing this notation is to simplify the recursive $Q$-function update formula (as seen in equation (3)). From equation (3) and equation (4) we have:

$$Q_h^k(x, a) = \alpha_t^0 H + \sum_{i=1}^{t} \alpha_t^i [r_h(x, a) + V_{h+1}^*(x_{h+1}^{k_i}) + b_i] \tag{5}$$

which only applies for $(x1, a1, h = 1)$, $\forall k \in [K]$, as discussed in equation (3).

## Proof Details

We now introduce some Lemmas that will help us in the proof of Theorem 1. For completeness we repeat Lemme 4.1 exactly as stated in Jin et al. [2].

Lemma 4.1. *The following properties hold for $\alpha_t^i$:*

(a) $\frac{1}{\sqrt{t}} \leq \sum_{i=1}^t \frac{\alpha_t^i}{\sqrt{i}} \leq \frac{2}{\sqrt{t}}$ *for every* $t \geq 1$
(b) $\max_{i \in [t]} \alpha_t^i \leq \frac{2H}{t}$ *and* $\sum_{i=1}^t (\alpha_t^i)^2 \leq \frac{2H}{t}$ *for every* $t \geq 1$
(c) $\sum_{t=1}^{\infty} \alpha_t^i = 1 + \frac{1}{H}$ *for every* $i \geq 1$

Proof. See proof of Lemma 4.1 in Jin et al. [2] □

We now present modified versions of Lemma 4.2 and Lemma 4.3 from Jin et al. [2], which we will use in our proof of Theorem 1.

Lemma 4.2 (Difference in Q). *For $(x1, a1, h = 1) \in \mathcal{S} \times \mathcal{A} \times [H]$ and episode $k \in [K]$, let $t = N_1^k(x1, a1)$ and suppose $(x1, a1)$ was previously taken at step h=1 of episode $k_1, ..., k_t < k$. Then:*

$$(Q_1^k - Q_1^*)(x1, a1) = \alpha_t^0(H - Q_1^*(x1, a1)) + \sum_{i=1}^{t} \alpha_t^i \left[ [(\hat{\mathbb{P}}_1^{k_i} - \mathbb{P}_1)V_2^*](x1, a1) + b_i \right]$$

Proof of Lemma 4.2. From the Bellman optimality equation we have $Q_h^*(x, a) = (r_h + \mathbb{P}_h V_{h+1}^*)(x, a)$. Recalling that $[\hat{\mathbb{P}}_h^k V_{h+1}](x, a) := V_{h+1}(x_{h+1}^k)$, and the fact that $\sum_{i=0}^t \alpha_t^i = 1$, we have:

$$\begin{aligned} Q_h^*(x, a) &= r_h(x, a) + [\mathbb{P}_h V_{h+1}^*](x, a) \\ &= \alpha_t^0 Q_h^*(x, a) + \sum_{i=1}^{t} \alpha_t^i \left[ r_h(x, a) + [\mathbb{P}_h V_{h+1}^*](x, a) \right] \\ &= \alpha_t^0 Q_h^*(x, a) \\ &\quad + \sum_{i=1}^{t} \alpha_t^i \left[ r_h(x, a) + [(\mathbb{P}_h - \hat{\mathbb{P}}_h^{k_i})V_{h+1}^*](x, a) + V_{h+1}^*(x_{h+1}^{k_i}) \right] \end{aligned}$$

which is true for all $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$. Subtracting the above equation from formula (5) for $(x1, a1, h = 1)$, we obtain Lemma 4.2:

$$
\begin{aligned}
&(Q_1^k - Q_1^*)(x1, a1) \\
&= \alpha_t^0 (H - Q_1^*(x1, a1)) \\
&\quad + \sum_{i=1}^{t} \alpha_t^i \left[ (V_2^{k_i} - V_2^*)(x_2^{k_i}) + [(\hat{\mathbb{P}}_1^{k_i} - \mathbb{P}_1) V_2^*](x1, a1) + b_i \right] \\
&\overset{①}{=} \alpha_t^0 (H - Q_1^*(x1, a1)) \\
&\quad + \sum_{i=1}^{t} \alpha_t^i \left[ [(\hat{\mathbb{P}}_1^{k_i} - \mathbb{P}_1) V_2^*](x1, a1) + b_i \right]
\end{aligned}
$$

Where ① holds because $V_h^k(x_h^k) = V_h^*(x_h^k)$ for $h \geq 2, \forall k \in [K]$ □

Next, we present a modified version of Lemma 4.3 from Jin et al. [2], for $(x1, a1, h = 1)$, which shows that $Q_1^k$ is an upper bound on $Q_1^*$ with high probability.

Lemma 4.3 (bound on $(Q_1^k - Q_1^*)(x1, a1)$). *There exists an absolute constant $c > 0$ such that, for any $p \in (0, 1)$, letting $b_t = c\sqrt{H^3\iota'/t}$, we have $\beta_t = 2\sum_{i=1}^{t} \alpha_t^i b_i \leq 4c\sqrt{H^3\iota'/t}$, where $\iota' = log(K/p)$, and with probability at least $1 - p$, the following holds for $(x1, a1, h = 1)$, $\forall k \in [K]$:*

$$0 \leq (Q_1^k - Q_1^*)(x1, a1) \leq \alpha_t^0 H + \beta_t$$

*where $t = N_1^k(x1, a1)$ and $k_1, ..., k_t < k$ are the episodes where $(x1, a1)$ was taken at step $h = 1$.*

Proof of Lemma 4.3. For $(x1, a1, h = 1)$, let us denote $k_0 = 0$ and denote

$$k_i = \min \left( \{k \in [K] \mid k \geq k_{i-1} \wedge (x_1^k, a_1^k) = (x1, a1)\} \cup \{K + 1\} \right)$$

In words, this means that $k_i$ is the episode when $(x1, a1)$ was taken at step $h = 1$ for the $i$th time, and $k_i$ equals $K+1$ if $(x1, a1)$ was taken for fewer than $i$ times. The random variable $k_i$ can be thought of as a stopping time. If we let $\mathcal{F}_i$ (filtration) be the $\sigma$-algebra generated by all the random variables until episode $k_i$, and step $h = 1$. Then, $\left(\mathbb{I}[k_i \leq K] \cdot [(\hat{\mathbb{P}}_1^{k_i} - \mathbb{P}_1) V_2^*](x1, a1)\right)_{i=1}^{\tau}$ is a Martingale difference sequence w.r.t. the filtration $\{\mathcal{F}_i\}_{i \geq 0}$. In words, the filtration $\{\mathcal{F}_i\}_{i \geq 0}$ can be thought of as a sequence of increasing information about state $x1$, action $a1$ at step $h = 1$, where the filtration satisfies $\mathcal{F}_1 \subseteq \mathcal{F}_2 ... \subseteq \mathcal{F}_i$ and the three properties that define a $\sigma$-algebra. By Azuma-Hoeffding and a union bound, we have with probability at least $1 - p$:

$$
\begin{aligned}
\forall \tau \in [K] : \left| \sum_{i=1}^{\tau} \alpha_\tau^i \cdot \mathbb{I}[k_i \leq K] \cdot [(\hat{\mathbb{P}}_1^{k_i} - \mathbb{P}_1) V_2^*](x1, a1) \right| \\
\leq \frac{cH}{2} \sqrt{\sum_{i=1}^{\tau} (\alpha_\tau^i)^2 \cdot \iota'} \leq c\sqrt{\frac{H^3\iota'}{\tau}}
\end{aligned} \tag{6}
$$

for some absolute constant c. Because inequality (6) holds for all fixed $\tau \in [K]$ uniformly, it also holds for $\tau = t = N_1^k(x1, a1) \leq K$, which is a random variable, where $k \in [K]$. Also note $\mathbb{I}[k_i \leq K] = 1$ for all $i \leq N_1^k(x1, a1)$. We now have:

$$
\left| \sum_{i=1}^{t} \alpha_\tau^i \cdot [(\hat{\mathbb{P}}_1^{k_i} - \mathbb{P}_1) V_2^*](x1, a1) \right| \leq c\sqrt{\frac{H^3\iota'}{\tau}} \text{ where } t = N_1^k(x1, a1) \tag{7}
$$

If we choose $b_t = c\sqrt{H^3\iota'/t}$ for the same constant $c$ as in inequality (7), we have $\beta_t/2 = \sum_{i=1}^{t} \alpha_t^i b_i \in [c\sqrt{H^3\iota'/t}, 2c\sqrt{H^3\iota'/t}]$ according to Lemma 4.1.a. Then the right-hand side of Lemma 4.3 follows immediately from Lemma 4.2 and inequality (7). The left-hand side also follows from Lemma 4.2. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. We follow a similar procedure to Jin et al. [2], except we do not have to decompose the regret into a recursive form in $h$ since we are interested in a fixed $h = 1$. We denote $\delta_1^k := (V_1^k - V_1^{\pi_1^k})(x1)$.

By Lemma 4.3, we have that with at least $1 - p$ probability, $Q_1^k(x1, a1) \geq Q_1^*(x1, a1)$ and thus $V_1^k(x1) \geq V_1^*(x1)$. We also know that $Q_1^k(x, a) = Q_1^*(x, a)$, $\forall (x, a) \in \mathcal{S} \times \mathcal{A} \backslash (x1, a1)$ and thus $V_1^k(x) = V_1^*(x)$, $\forall x \in \mathcal{S} \backslash x1$. The regret can be upper bounded:

$$
\begin{aligned}
\text{Regret}(K) = \sum_{k=1}^{K} (V_1^* - V_1^{\pi_1^k})(x_1^k) &\leq \sum_{k=1}^{K} (V_1^k - V_1^{\pi_1^k})(x_1^k) \\
&\overset{①}{\leq} \sum_{k=1}^{K} (V_1^k - V_1^{\pi_1^k})(x1) = \sum_{k=1}^{K} \delta_1^k
\end{aligned} \tag{8}
$$

where inequality ① holds because $\sum_{k=1}^{K} (V_1^k - V_1^{\pi_1^k})(x_1^k) = 0$, $\forall x \in \mathcal{S} \backslash x1$.

For any fixed $k$, $\in [K]$, let $t = N_1^k(x1, a1)$ and suppose $(x1, a1)$ was previously taken at step $h = 1$ of episode $k_1, ..., k_t < k$, Then we have:

$$
\begin{aligned}
\delta_1^k = (V_1^k - V_1^{\pi_1^k})(x1) &\overset{①}{=} (Q_1^k - Q_1^{\pi_1^k})(x1, a_1^k) \\
&= (Q_1^k - Q_1^*)(x1, a_1^k) + (Q_1^* - Q_1^{\pi_1^k})(x1, a_1^k) \\
&\overset{②}{=} (Q_1^k - Q_1^*)(x1, a1) + (Q_1^* - Q_1^{\pi_1^k})(x1, a1) \\
&\overset{③}{\leq} \alpha_t^0 H + \beta_t + [\mathbb{P}_1 (V_2^* - V_2^{\pi_1^k})](x1, a1) \\
&\overset{④}{=} \alpha_t^0 H + \beta_t
\end{aligned} \tag{9}
$$

where $\beta = 2\sum \alpha_t^i b_i \leq O(1)\sqrt{H^3\iota'/t}$. Equality ① holds because $V_1^k(x1) = \max_{a' \in \mathbb{A}} Q_1^k(x1, a') = Q_1^k(x1, a_1^k)$. Equality ② holds because $Q_1^k(x1, a_1^k) = Q_1^*(x1, a_1^k)$, $\forall a_1^k \in \mathcal{A} \backslash a1$ and $Q_1^*(x1, a_1^k) = Q_1^{\pi_h^k}(x1, a_1^k)$, $\forall a_1^k \in \mathcal{A} \backslash a1$. Inequality ③ holds with at least $1 - p$ probability by Lemma 4.3 and the Bellman equations (1). Finally, equality ④ holds since $[\mathbb{P}_h (V_{h+1}^{\pi_h^k} - V_{h+1}^*)](x1, a1) = 0$ for $h \geq 1, \forall k \in [K]$. We now compute the summation $\sum_{k=1}^{K} \delta_1^k$. Denoting $n_1^k = N_1^k(x1, a1)$, we have:

$$\sum_{k=1}^{K} \alpha_{n_1^k}^0 H = \sum_{k=1}^{K} H \cdot \mathbb{I}[n_1^k = 0] \leq H \tag{10}$$

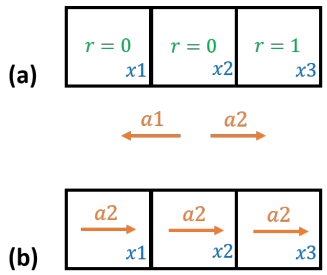


**Figure 1: The environment the agent was trained in can be seen in (a). For the purpose of simplifying the image we use the notation $r(x1) = 0$, $r(x2) = 0$, and $r(x3) = 1$ to represent the reward the agent receives for transitioning into states $x1, x2, x3$ respectively. The optimal policy is shown in (b).**

We also have:

$$\sum_{k=1}^{K} \beta_{n_1^k} \leq O(1) \cdot \sum_{k=1}^{K} \sqrt{\frac{H^3\iota'}{n_1^k}} = O(1) \sum_{n=1}^{N_1^K(x1,a1)} \sqrt{\frac{H^3\iota'}{n}} \overset{①}{\leq} O(\sqrt{H^2T\iota'}) \tag{11}$$

where inequality ① is true because the left-hand side of ① is maximized when $N_1^k(x1, a1) = K$, and $\sum_{n=1}^{K} \sqrt{\frac{1}{n}}$ can be bounded by $O(\sqrt{K})$. Taking the sum from $k = 1$ to $K$ for (9) and plugging in (10) and (11) we have:

$$\sum_{k=1}^{K} \delta_1^k \leq O(H + \sqrt{H^2T\iota'}) \overset{①}{=} O(\sqrt{H^2T\iota'})$$

where ① is true since $\sqrt{H^2T\iota'} \geq H$.

Recall that Lemma 4.3 was applied twice in this proof (once in equation (8) and once in ③ in equation (9)). Note that $(1-p)^2 = 1-2p+p^2 \geq 1-2p$, where the last inequality is applied because the $p^2$ term is small when compared to $2p$. In summary, we have that $\sum_{k=1}^{K} \delta_1^k \leq O(\sqrt{H^2T\iota'})$ holds with probability at least $1-2p$. Note the term $p$ cannot be greater than $1/2$ to ensure the probability $1-2p$ is non-negative. This can be achieved by re-scaling $p \to p/2$ to reduce it's range from $[0, 1]$ to $[0, 1/2]$. As such, re-scaling $p$ to $p/2$ finishes the proof. □

## 5 EXPERIMENTAL SETUP

In the previous section we proved the theoretical total regret of the $Q$-learning with Max-Optimal Initialization algorithm is bounded by $O(\sqrt{H^2T\iota'})$, while the total regret of the $Q$-learning with UCB-Hoeffding algorithm is bounded by $O(\sqrt{H^4SAT\iota})$ [2]. This result implies that our statement of interest holds true theoretically. In this section our goal is to empirically show that results are consistent with our theoretical findings. We do this by choosing a simple tabular environment to compare both algorithms. The simple environment will allow for easier interpretation of results. We expect the empirical results to hold for larger and more complex environments.

The environment used is a 1-dimensional gridworld, with 3 states, and 2 actions (see Figure 1 (a)). The actions are left and right, and when the agent takes an action that causes it to hit a wall it remains in the same state. All transitions and rewards are deterministic. The agent receives a reward of 1 if its next state is the right most state and a reward of 0 otherwise. The agent interacts with the environment for exactly 3 steps each episode. The optimal policy is to go right ($a2$) in all states for all steps (See Figure 1 (b)). Formally this setting can be represented by a MDP$(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, where there are 3 states $\mathcal{S} = \{x1, x2, x3\}$, 2 actions $\mathcal{A} = \{a1, a2\}$, the horizon is $H = 3$, and the transition dynamics and reward function are as follows for all $h \in [H] = \{1, 2, 3\}$:

$$\begin{cases} \mathbb{P}_h(x1|x1, a1) = 1 \\ \mathbb{P}_h(x2|x1, a2) = 1 \\ \mathbb{P}_h(x1|x2, a1) = 1 \\ \mathbb{P}_h(x3|x2, a2) = 1 \\ \mathbb{P}_h(x2|x3, a1) = 1 \\ \mathbb{P}_h(x3|x3, a2) = 1 \end{cases} \quad \text{and} \quad \begin{cases} r_h(x1, a1) = 0 \\ r_h(x1, a2) = 0 \\ r_h(x2, a1) = 0 \\ r_h(x2, a2) = 1 \\ r_h(x3, a1) = 0 \\ r_h(x3, a2) = 1 \end{cases}$$

Recall that our algorithm makes two important changes to the $Q$-learning with UCB-Hoeffding algorithm from Jin et al. [2]. Namely, the initialization is changed according to equation (2) and the $Q$-function is only updated if the current state, action, and step is $(x1, a1, h = 1)$. We refer to these two changes as assumption 1 (A1) and assumption 2 (A2) respectively. We train an agent using three different algorithms:

(1) $Q$-learning with UCB-Hoeffding [2]
(2) $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization without A2
(3) $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization (Algorithm 1)

From the above enumeration, points 1 and 3 have already been discussed in detail and are shown explicitly as Algorithm 1 in this work and Algorithm 1 in Jin et al. [2] respectively. The goal of point 2 is to show that A2 is crucial for our theoretical regret bound (as seen in Theorem 1). Since A2 is removed in point 2, we expect the total regret to be greater than that of our algorithm. An important detail is that the $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization without A2 algorithm is the same as our algorithm except with A2 removed, but also with $N_h(x, a)$ initialized to 0 for all $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H]$. Since now the $Q$-function is updated for all states, actions, and steps, a visit count $N_h(x, a)$ must be kept for all $(x, a, h)$.

An agent was trained using each algorithm for 500 episodes, $K = 500$. We set the probability term $p = 0.05$, corresponding to a probability of at least $1 - p = 0.95$ of obtaining a total regret of $O(\sqrt{H^2T\iota'})$ as mentioned in Theorem 1. The constant used for the bonus $b_t$ was set to $c = 0.1$. For the $Q$-learning with UCB-Hoeffding and Max-Optimal initialization algorithm we set $(x1, a1, h = 1)$ as the non-optimal $Q$-value.

Recall that going left ($a1$) in the left-most state $x1$ causes the agent to hit a wall and remain in state $x1$. Action $a1$ is not the optimal action in state $x1$, since the agent will only receive a reward of 1 for transitioning into the right-most state, which going left, $a1$ does not help achieve. The optimal action is for the agent to go right ($a2$) from state $x1$. Intuitively, the above initialization causes the agent to have low confidence in $(x1, a1, h = 1)$ and therefore the agent will explore $(x1, a1, h = 1)$ until it is confident enough

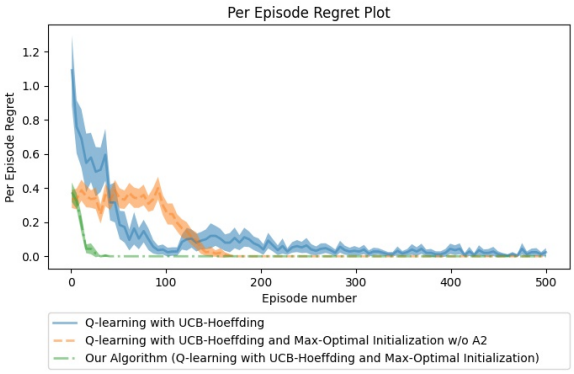


**Figure 2: A Per Episode Regret plot of algorithm UCB-H, UCB-H MO, and UCB-H MO A2 averaged over 50 independent runs. The shaded regions represent a 95% confidence interval.**

that $(x1, a1)$ actually provides it with less total reward than $(x1, a2)$ at step $h = 1$.

As a way to measure performance per episode we introduce:

$$\text{Per Episode Regret} = \text{PER}(k) = V_1^*(x_1^k) - V_1^{\pi_h^k}(x_1^k)$$

When the PER is summed over all episodes it gives the total regret. In our experiments, the PER was averaged over 50 independent runs for each algorithm.

## 6 EXPERIMENTAL RESULTS

Figure 2 shows the results of training an agent in the setting mentioned in the previous section. For convenience, we will refer to the $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization algorithm as UCB-H MO, and the $Q$-learning with UCB-Hoeffding and Max-Optimal Initialization without A2 algorithm as UCB-H MO A2, and to the $Q$-learning with UCB-Hoeffding as UCB-H. In Figure 2 we observe that the per episode regret converges to zero fastest for UCB-H MO. A faster convergence to zero implies that the total regret of UCB-H MO is lower than that of UCB-H MO A2 and UCB-H, since the total regret is just the sum of the PER over all episodes. Recall from the theoretical results that the total regret of UCB-H MO is $O(\sqrt{H^2T\iota'})$, while UCB-H has a total regret of $O(\sqrt{H^4SAT\iota})$. Therefore, this is the expected behaviour of UCB-H MO when compared to UCB-H based on our theoretical results. Since UCB-H MO also converges faster than UCB-H MO A2 it supports our earlier claim that A2 is crucial to the theoretical regret bound we obtain (see Theorem 1).

## 7 DISCUSSION AND FUTURE WORK

In summary, we showed that the total regret of our algorithm is upper bounded by $O(\sqrt{H^2T\iota'})$, which is a tighter regret bound than the $Q$-learning with UCB-Hoeffding algorithm, $O(\sqrt{H^4SAT\iota})$. The total regret bound of our algorithm is independent of the state and action space and reduces the dependence on the horizon from $H^4$ to $H^2$. The state and action space Independence arises since the transferred $Q$-function is only sub-optimal at one state and action. The reduction in the horizon factor by a quadratic factor arises due to two reasons. The first is because the transferred $Q$-function is only sub-optimal at one time step $h = 1$. The second reason is because the sub-optimal time step is the first time step (i.e., $h$ is not 2 or greater) which allows us to omit an otherwise necessary induction step over time steps in the theorem proof. This result provides theoretical justification for applying $Q$-function reuse on the $Q$-function with UCB-Hoeffding algorithm. Although we make the strong assumption that all but one of the Q-values are optimal, we believe that this provides a solid starting point for future work to build upon.

We believe some interesting future directions are:

(1) Increasing the number of non-optimal $Q$-values. In this work it was assumed that the $Q$-function was optimal for all but one state, action, and step $(x1, a1, h = 1)$. One possible next step would be to assume that the $Q$-values of two or more states, actions, and steps are non-optimal. This might make the analysis more complex because the proof of Lemma 4.2 assumed that $V_h^k(x_h^k) = V_h^*(x_h^k)$ for $h \geq 2, \forall k \in [K]$ which would no longer be true if one of the non-optimal $Q$-values occurred for $h \geq 2$.

(2) This work assumes that the $Q$-functions was optimal for all but one state, action, and step $(x1, a1, h = 1)$. This assumption is generally unrealistic for a transferred $Q$-function because an agent rarely learns the optimal $Q$-function in an environment (instead, it often only reaches a near-optimal one). Such cases can potentially be modelled by assuming some sub-optimality of the $Q$-function initialization. For instance, for some subset of states, actions, and steps it could be assumed that $Q_h(x, a) \geq Q_h^*(x, a) - \epsilon,\ \forall (x, a, h), \epsilon \geq 0$.

(3) Allowing the $Q$-function to be updated for some $(x, a, h) \in \mathcal{S} \times \mathcal{A} \times [H] \backslash (x1, a1, h = 1)$. In this work it was assumed that we had prior knowledge of which states, actions, and steps the $Q$-function was optimally initialized (i.e. $Q_h(x, a) = Q_h^*(x, a),\ \forall \mathcal{S} \times \mathcal{A} \times [H] \backslash (x1, a1, h = 1)$) and therefore we were able to explicitly choose to not perform $Q$-function updates at those states, actions and steps. It may not always be the case that this information is known. As such, the algorithm would not update the $Q$-function for all states, actions and steps. Modifications to the analysis done in this work would have to be made to provide an upper bound on the regret of such an algorithm because a $Q$-function update performed at an optimal state, action, and step might depend on the $Q$-function at a non-optimal state, action, and step. Such an update can potentially change the value of the $Q$-function at the optimal state, action, and step, causing for an increase in the number of non-optimal $Q$-values.

## ACKNOWLEDGMENTS

We would like to thank Laura Petrich, Su Zhang, Srijita Das, Saiful Islam, and Jiamin He, as well as the anonymous reviewers, for their feedback. Part of this work has taken place in the Intelligent Robot Learning Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii), CIFAR, and NSERC.

# REFERENCES


[1] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. 2017. Minimax regret bounds for reinforcement learning. *arXiv preprint arXiv:1703.05449* (2017).

[2] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. 2018. Is Q-learning provably efficient. In *Advances in Neural Information Processing Systems*. 4863–4873.

[3] Sham Machandranath Kakade et al. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation. University of London London, England.

[4] Michael Kearns and Satinder Singh. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning* 49, 2-3 (2002), 209–232.

[5] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[6] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, 7 (2009).

[7] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. 2020. Transfer Learning in Deep Reinforcement Learning: A Survey. *arXiv preprint arXiv:2009.07888* (2020).