university of groningen / faculty of science and engineering / bernoulli institute

Exploring One-Step Fixed Horizon Q Learning in Tabular Stochastic Environments

Bachelor's Project Thesis

Stan Ferguson, s4674367, s.k.ferguson@student.rug.nl
Supervisors: Prof. R.F. Cunha

Abstract: In this study, we test Fixed Horizon Q-learning (FHQ) in tabular stochastic environments. FHQ was proposed as an alternative to regular (infinite horizon) Q learning by Asis et al. [2020] to break the deadly triad of reinforcement learning by countering bootstrapping to unreliable estimates. Instead of updating the value function with the entire episode return, a specific horizon length is set. We reproduce- and build on the previous research to better pinpoint what the (dis-)advantages of this method are, particularly on the performance of different horizon lengths and their computational cost over longer periods of time. We showed that, contrary to our initial assumption based on the previous research, the shorter horizons do not necessarily perform better in highly stochastic environments. We identify a trade-off between horizon length and computational cost, and find that α-decay is necessary for successful empirical convergence, which is not generally the case for most RL algorithms.

1 Introduction

Reinforcement learning (RL) is a framework in which an agent learns to make sequential decisions through interaction with an environment (Sutton and Barto [2018]). At each step, the agent selects an action, receives a reward, and transitions to a new state. Over time, it learns a policy that maximizes the expected cumulative reward, often by estimating value functions that predict long-term returns.

One of the key challenges in reinforcement learning is dealing with *stochasticity*: randomness in the environment's transitions or rewards. In highly stochastic settings, an action might not always result in the intended outcome. This unpredictability complicates learning, as the agent may struggle to distinguish between good and bad actions when outcomes are inconsistent.

Traditional Q-learning (Watkins and Dayan [1992]), a popular reinforcement learning algorithm (Hasselt [2010]), estimates the return from each state-action pair over the entire episode. Although this works well in deterministic environments, it can lead to instability and overestimation in stochastic environments (Thrun and Schwartz [1993], Hasselt [2010]). This is in part due to the use of *bootstrapping*—a technique where the value of a state-action pair is updated using estimates of future returns, rather than only observed rewards. While bootstrapping enables efficient learning by propagating value information backward through the state space, it can amplify estimation errors in stochastic environments. Thus, long-horizon methods (like regular Q-learning) that rely heavily on bootstrapped values are more likely to cause inaccurate value estimates.

To address this, Asis et al. [2020] introduced *Fixed-Horizon Q-learning* (FHQ), where the agent learns multiple action-value functions, each corresponding to a fixed number of steps (the horizon). Instead of estimating the return to the end of the episode, FHQ estimates returns up to a fixed horizon h. The intuition is that short-horizon targets are more stable in stochastic environments because they rely less on unpredictable future outcomes and reduce the reliance on bootstrapped estimates over many steps. In a sense, regular Q-learning does this with the discount factor γ, which can also scale the amount of weight that is attributed to future steps. However, FHQ provides a more direct control over the planning horizon and the extent of bootstrapping, making it a promising approach for learning in environments where long-term predictions are unreliable.

Initial experiments in Asis et al. [2020] with FHQ showed promising results: in highly stochastic environments, shorter horizons sometimes outperformed longer ones and they seemed to outperform regular discounted Q-learning. However, these findings were limited to early learning behavior and lacked broader analysis of convergence and computational cost.

1.1 Aim of This Project

In this thesis, we build on the work by Asis et al. [2020] and investigate FHQ-learning further. More specifically, we focus on the one-step FHQ-learning algorithm in a tabular stochastic environment; Tabular meaning that it is a simple environment with discrete, small, finite state and action spaces.

The primary goal is to extend knowledge on learning and (speed of) convergence of the different horizons in a stochastic environment. We hypothesize that short horizons are preferable in the highly stochastic setting (in line with Asis et al. [2020]), and that longer horizons would be preferable in a low-stochastic one. Therefore, we also want to investigate whether or not it can be beneficial to make this horizon adaptable to the amount of stochasticity in the state. So, if the local stochasticity between states varies, we can adapt the horizon length accordingly.

2 Background

2.1 Reinforcement Learning

As briefly discussed in the introduction, the goal in reinforcement learning (RL) is to develop successful behavior by interacting with an environment.

2.1.1 Markov Decision Process

This can mathematically be formulated as a so-called Markov Decision Process (MDP) (Sutton and Barto [2018]):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma) \tag{2.1}$$

Where $\mathcal{S}$ is the set of all states, $\mathcal{A}$ is the set of all actions, $\mathcal{P} = p(s', r|s, a)$ is the *transition function*. This is the probability of transitioning to next state s' with reward r when taking action a from state s. $\mathcal{R} = r(s, a, s')$ is the *reward function*. This is the reward for moving from state s to state s' with action a. Lastly, γ (where $\gamma \in [0, 1]$) is the discount factor, with which we tune how much future rewards are weighted.

So, from the agent's perspective, at each timestep t it finds itself in a state $s_t \in \mathcal{S}$ from which it can choose an action $a_t \in \mathcal{A}$. After taking said action, the agent will transition into a next state $s_{t+1} \sim (p(\cdot|s_t, a_t)$ which will give some reward $r_t = r(s_t, a_t, s_{t+1})$. After an episode ends, when the agent reaches the goal or terminates for some other reason, the agent can reflect on the taken actions. The goal is to learn a behavior pattern that leads to the optimal sequence of actions and obtain the maximum cumulative reward over an entire episode. Such a behavior pattern is called a *policy* π.

To achieve this, we try to maximize the expected total reward for each timestep, called the *return*. It is given as:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{2.2}$$

Where T is the final time step of the episode, t is the current timestep, R_t is the reward received at time t, γ is the discount factor and k is the index.

2.1.2 Value Functions

Within reinforcement learning, a key distinction exists between *value-based* and *policy-based* methods. Policy-based methods attempt to learn the optimal policy directly by optimizing parameters that define the policy. In contrast, value-based methods approach the problem by estimating so-called *value functions*, which represent the expected return from a given state or state-action pair under a particular policy. In this project, we are working with value functions, of which there are two commonly used types:

- **State-value function**:

  $$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \tag{2.3}$$

  Where $V_\pi(s)$ is the value function of state s under policy π, $\mathbb{E}_\pi[G_t \mid S_t = s]$ is the expected return G_t under policy π for state s at timestep t.

- **Action-value function**:

  $$Q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{2.4}$$

  Where $Q_\pi(s, a)$ is the state-action value function, $\mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ is the expected return G_t under policy π for taking action a from state s at timestep t.

  This state-action value is also called the *Q-value*.

We can approximate these value functions using the *Bellman equations*, which form the foundation for many reinforcement learning algorithms (Sutton and Barto [2018]), including *temporal difference* (TD) learning.

2.1.3 Temporal Difference (TD) learning

In one-step TD learning, a value estimate is updated using the observed reward and the estimated value of the next state known as the *TD target*, which is used by a subsequent update function for $V(s)$:

$$\hat{G}_t = R_{t+1} + \gamma V(S_{t+1}) \tag{2.5}$$

$$V(S_t) \leftarrow V(S_t) + \alpha \underbrace{\left[\hat{G}_t - V(S_t)\right]}_{\text{TD error}} \tag{2.6}$$

Where $\hat{G}_t$ is the TD target that consists of the reward R_{t+1} and the discounted (γ) state-value of the next state $V(S_{t+1})$. The update equation 2.6 updates the current state-value $V(S_t)$ with this target $\hat{G}_t$ and the existing value, which together form the *TD-error*. This is then factored by the learning rate α, where for higher values of α, more existing information is replaced by the TD-error estimate.

In action-value methods such as Q-learning (Watkins and Dayan [1992]), which is arguably one of the most popular TD learning methods (Hasselt [2010]), the target is an estimate for a state-action transition (so for example; moving up, down, left and right from state s have their own Q-values). The recursive update rule becomes:

$$\hat{G}_t = R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') \tag{2.7}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \underbrace{\left[\hat{G}_t - Q(S_t, A_t)\right]}_{\text{TD error}} \tag{2.8}$$

Where again, $\hat{G}_t$ is the target, R_{t+1} is the obtained reward and γ is the discount factor. However now, the next state-action value is chosen from all the actions that can be taken from the current state S_t by selecting the maximum Q-value of those actions: $\max_{a'} Q(S_{t+1}, a')$.

The current Q-value $Q(S_t, A_t)$ is then updated using this TD target and the existing Q-value (together the TD-error), and again scaled with the learning rate α.

In this project, we will work with Fixed-Horizon Q-learning, which is a version of regular Q-learning described above.

2.1.4 Tabular Environments

In environments with a small, finite number of states and actions, these value functions can be represented using a single table. This approach, known as the *tabular case*, stores a separate value for each state (in the case of $V(s)$) or each state-action pair (for $Q(s, a)$). Tabular methods are conceptually simple and easy to interpret. They allow for exact updates using temporal-difference learning rules, as each entry in the table can be updated independently based on observed transitions.

In this tabular case, convergence is generally easy to prove both theoretically and empirically under standard assumptions such as sufficient exploration and a decaying learning rate (Sutton and Barto [2018]). This makes tabular environments a useful testing ground for developing and evaluating reinforcement learning algorithms.

However, the tabular representation becomes infeasible in large or continuous state spaces, where the number of states and actions can be (infinitely) large (Sutton and Barto [2018]). In such cases, techniques such as *function approximation* can be used to estimate these value functions (Thrun and Schwartz [1993]).

2.1.5 Stochasticity

One of the main challenges in reinforcement learning is that of *stochasticity*, which refers to the inherent unpredictability in an environment's transition and reward functions (Sutton and Barto [2018], Hasselt [2010]). Returning to the MDP formulation in Section 2.1.1, this means that taking the same action a from a given state s_t does not always result in the same next state s_{t+1} and subsequent reward. The transition function is not deterministic, but rather a set of probabilities.

2.2 Fixed-Horizon Q-Learning

Traditional Q-learning aims to estimate the optimal action-value function $Q^*(s, a)$, which corresponds to the expected cumulative return from a given state-action pair over an entire episode. This assumes planning over potentially long or infinite horizons, which can lead to instability, especially in stochastic environments (2.1.5).

Fixed-Horizon Q-learning (FHQ), introduced by De Asis et al. (2020), proposes a modification: rather than estimating the expected return over the full episode, the algorithm estimates the return over a fixed number of future steps, referred to as the horizon h.

2.2.1 General Idea

In FHQ-learning, we keep track of a set of action-value functions $Q^h(s, a)$, one for each horizon $h \in \{1, 2, \ldots, H\}$, where H is the fixed furthest horizon. Each function Q^h stores a value for every possible state-action pair, forming a separate table of size $\mathcal{S} \times \mathcal{A}$. These tables are updated independently using the structure of the fixed-horizon targets. Recall (from section 2.1.4), that in regular value-based methods, we only have one such Q-table of size $\mathcal{S} \times \mathcal{A}$.

At each timestep t, the agent observes the current state s_t, selects an action a_t according to an exploration policy (e.g., ϵ-greedy with respect to Q^H), and receives a reward r_t and next state S_{t+1} from the environment. Then, for each horizon $h > 1$, the agent updates its estimate using the one-step target from horizon $h - 1$:

$$\hat{G}_t^h = R_{t+1} + \gamma \max_{a'} Q^{h-1}(S_{t+1}, a') \quad (2.9)$$

$$Q^h(S_t, A_t) \leftarrow Q^h(S_t, A_t) + \alpha \underbrace{\left[\hat{G}_t^h - Q^h(S_t, A_t)\right]}_{\text{TD error}} \quad (2.10)$$

Where $\hat{G}_t^h$ is the fixed-horizon TD target for horizon h — a one-step estimate of the return over h steps. It consists of the immediate reward R_{t+1} and the discounted (γ) maximum Q-value of the next state S_{t+1} under the shorter horizon $h-1$: $\max_{a'} Q^{h-1}(S_{t+1}, a')$. This ensures that longer-horizon estimates build upon more stable, shorter-horizon ones.

The update equation modifies the current Q-value estimate $Q^h(S_t, A_t)$ using the TD target $\hat{G}_t^h$ and the existing estimate. Their difference (again) forms the *TD-error*, which is scaled by the learning rate α.

As we proceed through the Q-tables, longer horizons build on the (more stable) predictions of shorter ones. Over time, each table Q^h becomes a better estimate of the return over those h steps.

This avoids bootstrapping from the same estimate using the expected infinite return which, as mentioned earlier, can cause instability in regular Q-learning.

Now, we will describe the algorithmic procedure in detail and review the empirical findings of Asis et al. [2020], before diving into the focus of this project and how we have built on those findings.

2.2.2 Algorithm Description

The tabular, one-step FHQ-learning algorithm is shown in Algorithm 1. At each timestep, the agent observes a transition tuple $(s_t, a_t, r_{t+1}, s_{t+1})$, and performs updates for each horizon $h \in \{1, \ldots, H\}$.

The TD error δ is computed as the difference between the observed reward (plus the bootstrapped estimate from horizon $h-1$) and the current prediction from horizon h, like explained in the mathematical notation above. Action selection is done using an exploration strategy such as ϵ-greedy with respect to Q^H, the final horizon. The algorithm loops until the terminal state is reached.

2.2.3 Proof of Convergence

Asis et al. [2020] prove that FHQ-learning converges in the tabular setting under standard assumptions such as a decaying learning rate and sufficient exploration. For the formal proof and detailed assumptions, refer to the original paper.

Algorithm 2.1 Tabular One-step FHQ-Learning for estimating $Q^H \approx q_*^H$

```
1: Initialize $Q[h][s][a] \leftarrow 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $h = 0, 1, \ldots, H$
2: $s \sim p(s_0)$
3: $a \sim \mu(\cdot \mid s)$ (e.g., $\epsilon$-greedy w.r.t. $Q^H(s, \cdot)$)
4: $t \leftarrow 0$
5: while $t \neq t_{\max}$ do
6:     $(s', r) \sim p(s', r \mid s, a)$
7:     for $h = 1, 2, \ldots, H$ do
8:         $\delta \leftarrow r + \gamma \max_{a'} Q[h-1][s'][a'] - Q[h][s][a]$
9:         $Q[h][s][a] \leftarrow Q[h][s][a] + \alpha \cdot \delta$
10:    end for
11:    $s \leftarrow s'$
12:    $a \sim \mu(\cdot \mid s)$
13:    $t \leftarrow t + 1$
14: end while
```

2.2.4 Findings

In the research by Asis et al. [2020], fixed horizon methods were tested in multiple configurations. Both for state- (V_s) and state-action ($Q_{(s,a)}$) value functions, in a one-step and multi-step form and in tabular, linear and deep learning control problems. In this project, we are only focusing on the state-action ($Q_{(s,a)}$), one-step, tabular case.

For this specific case, tested in a highly stochastic environment, they concluded that:

"In a tabular control problem, we showed that greedifying with respect to estimates of a short, fixed horizon could outperform doing so with respect to longer horizons."

In the results section below, we will explain in more detail how they arrived at this conclusion and what experiment was performed. For now, it is important to know that this was the starting point for our research.

2.3 Our Approach

We hypothesized that, if short horizons could indeed be preferable in highly stochastic environments, then there might be motivation for this horizon to be adaptable to the amount of stochasticity in the environment.

This assumes the inverse to be true as well: that longer horizons would be beneficial in low-stochastic or deterministic environments. This seems like an intuitive, logical consequence as in those cases, previous experiences can be "trusted" by the agent, so there is no misleading information in the far-away horizons.

Thus, we want to dynamically adjust the planning horizon as a function of local, state-level stochasticity, rather than using a fixed horizon. Our hypothesis is that such an adaptive strategy can perform better than a fixed-horizon strategy in an environment with changing levels of stochasticity.

3 Methods

3.1 Environments

3.1.1 Slippery Maze

Our baseline environment is a grid-world maze, identical to- and taken from Asis et al. [2020]. The agent (a penguin) starts at a fixed location in a 9×9 grid and must reach a goal state (the fish) located in the bottom-right corner. The agent can move in four directions (up, down, left, right). However, the environment is highly stochastic: the intended action is only executed with probability 0.25, while with probability 0.75 the actual move is chosen uniformly at random across all four actions. This means that even the chosen action only succeeds with probability $0.25 + \frac{0.75}{4} = 0.4375$.

Each step leads to a reward of -1, and the episode ends once the goal is reached. The cumulative reward is therefore the negative of the episode length, and we use this quantity (mean episode length) as our main performance metric.

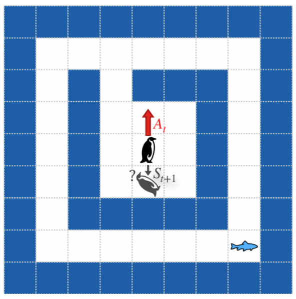


Figure 3.1: The slippery maze environment. The agent starts in the center; the goal is in the bottom-right. All actions are stochastic with slipperiness = 0.75. The agent is rewarded with -1 for each timestep.

The optimal policy takes about 70 steps on average to reach the goal state when we account for stochasticity. We calculated this by manually instructing an agent to take the shortest path, and averaging the return over 1000 identical runs.

Deterministic case We also performed experiments with this maze, but without any slipperiness. The transition function in that case is deterministic, meaning that each action has the agent end up in the intended follow-up state. In this case, the optimal path is exactly 14 steps, which can just be counted as the number of states between the start and goal states.

3.1.2 Non-Homogeneous Maze

In addition to the standard environment, we construct a custom variant with changing amounts of state-level stochasticity. Each tile in the maze has its own slipperiness probability, either 0 (deterministic) or 0.75 (stochastic). This non-homogeneous environment allows us to test our hypothesis: that the optimal planning horizon should vary depending on the local level of stochasticity.

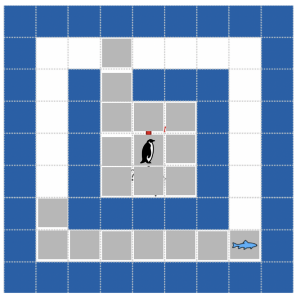


Figure 3.2: Custom non-homogeneous environment with varying slip probabilities across the grid. The white squares are slippery and the gray squares are not.

We designed it this way because, as seen in 3.2, the maze contains two slippery sections, one on the left and one on the right. Normally, the right path is quicker, but due to the slippery section on the right being longer, the optimal paths for the going around the maze left and right are: ≈ 31 and ≈ 35 respectively. Therefore, after accounting for the stochastic sections, the left path is actually quicker on average.

The motivation was that the short horizons would not be able to "look beyond" the initial obstruction and thus they would not be able to effectively converge (as quick or at all) to the optimal solution, following the left path.

3.2 Experimental Setup

All experiments are based on one-step, tabular FHQ-learning, as described in Section 2.2. We use ϵ-greedy action selection with $\epsilon = 0.1$, a common standard in RL that balances exploration and exploitation. At each timestep, the agent updates $Q^h(s, a)$ for every horizon $h \in \{8, 16, 32, 48\}$. We directly took these values for the horizon length from the original paper by Asis et al. [2020], and will compare performance between them.

Each configuration is evaluated across 100 independent runs to reduce variance. To assess both learning speed and overall convergence, we run experiments for both 100 and 1000 episodes. Additionally, we include analyses based on the computational cost, described in Section 3.4.

3.3 Hyperparameters

In this section, we will walk through all the hyperparameters involved.

3.3.1 learning rate α

First of all, the learning rate α is an important controlling parameter in all TD-learning algorithms (Sutton and Barto [2018]). In the research done by Asis et al. [2020], multiple values of α were compared, which seemed to suggest that the higher values of α (like 0.4-0.6) outperformed lower values (like 0.1) that are generally a good standard in reinforcement learning. That is because a learning rate of 0.5 means that during each update, we are replacing half of the information in our current estimate with the newly gathered reward. Generally, this leads to instable learning, as it is too reliant on (the accuracy of) the most recent experiences.

We will test multiple values of α, compare them in their speed of convergence and overall performance, and see if we can verify/reject the claim made by Asis et al. [2020], as cited in 2.2.4.

α-decay In the theoretical proof of convergence, α-decay is listed as a necessary requirement to ensure convergence. It is described as:

$$\alpha_t = \alpha_0 \cdot \lambda^t$$

where $\lambda < 1$ is the decay factor (we used $\lambda = 0.999$) and t is the timestep. Generally, α-decay is not considered necessary for empirical convergence results, so we have tested FHQ-learning both with and without α-decay.

3.3.2 Slipperiness

The slipperiness of the environment refers to the amount of stochasticity in the transition function. In the standard slippery maze, this is set to 0.75. We also test the deterministic case, where the environment is identical but the slipperiness is set to 0, and we test the non-homogeneous case which contains both of these for different states.

3.4 Evaluation Metrics

3.4.1 Episode Length

As each step yields a reward of -1, the total return per episode is equal to the negative of the episode length. Hence, minimizing episode length corresponds to maximizing performance. We track the mean episode length across runs as the main performance metric.

3.4.2 Computational cost

In the FHQ algorithm, the computational cost increases linearly with the horizon length. For each step in the environment, the agent performs h Q-value calculations. Thus, the agent with $h = 32$ performs four times as many Q-value calculations as the agent with $h = 8$ for each step in the environment. Therefore, the plotting of episodes on the x-axis can give a slightly distorted image of the time it has taken each horizon to converge effectively.

We define this computational cost as:

$$Computational\ Cost = h * \sum_{i=1}^{N} T_i \tag{3.1}$$

Where h is the horizon length, N are the number of episodes and T_i is the number of steps within episode i.

Besides plotting performance over episodes, we will also evaluate performance over this computational cost between the horizons.

4 Results

4.1 Preceding Results

The plotted results of this experiment by Asis et al. [2020] can be seen in figure 4.1.

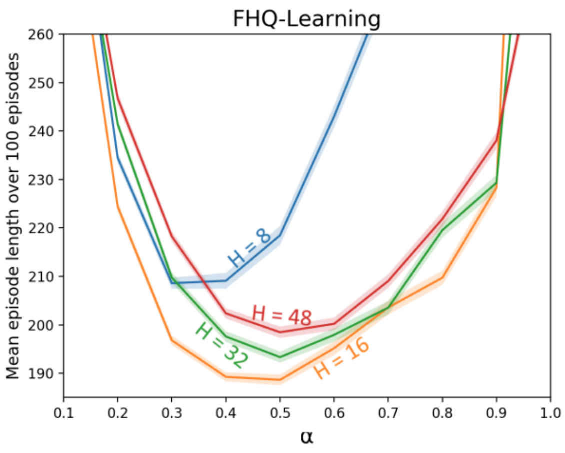


Figure 4.1: Mean episode lengths over 100 episodes of FHQ-learning with various step-sizes and horizons of interest. Results are averaged over 100 runs, and shaded regions represent one standard error.

Here, it is shown that for different step-sizes (= learning rate values), the horizons behave differently. As these results are the average return over the first 100 episodes, we cannot extract information about convergence, but we can say that an intermediate (shorter) horizon of 16 can outperform longer (and even shorter) horizons. Because, for some values of α, the mean episode length of horizon 16 over the first 100 episodes is lower than that of the other horizons.

In the paper, two (similar) claims are made based on these results:

1. In the results section: *"For FHQ-learning, it can be seen that if the final horizon is unreasonably short (H = 8), the agent performs poorly. However, H = 16 does considerably better than if it were to predict further into the future."*

2. In the discussion: *"In a tabular control problem, we showed that greedifying with respect to estimates of a short, fixed horizon could outperform doing so with respect to longer horizons."*

While the results indeed show a difference in performance that could suggest better results with shorter horizons, more tests need to be done to confidently conclude that short horizons are preferable in this stochastic tabular control problem. Mainly, information is missing on the (speed of) convergence of the different horizons, and the stability of their behavior. The current results are averaged over the first 100 episodes, but that includes the initial learning phase in which the Q values are not yet tuned. This learning phase is normally not included in evaluating performance. Rather, it is good practice to first train all the different models (per horizon), and then evaluate their performance on the fixed Q value estimates.

4.1.1 Reproduction

In Appendix A, we can see our reproduction of these results. The result for horizons 16, 32 and 48 are similar, but horizon 8 performs different to the results by Asis et al. [2020]. We will elaborate on this further in the discussion below.

4.2 Convergence and stability

From the results by Asis et al. [2020] in figure 4.1, one could assume that α values of 0.4-0.6 are optimal for performance of most horizons. In figure 4.2, we can see a more detailed representation of these first 100 episodes with $\alpha = 0.5$. All horizons show improvement in their performance over time, but the average episode length is still substantially above the optimal path of ≈ 70 steps.

In appendix 4.4a, an extended run of this experiment can is added with 1000 episodes for each horizon. Here, we see that horizon 8 diverges over time, while the other horizons keep improving the policy. Still, all horizons seem to approximate a policy that averages around ≈ 100 steps and not the optimal path of ≈ 70 steps.

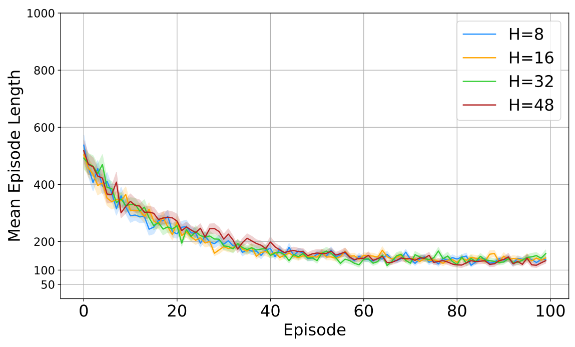


Figure 4.2: Mean episode length +- standard error over 100 identical runs for 100 episodes. α is set to 0.5 and decayed with 0.999 per step in the environment.

4.2.1 Lower Learning Rates Improve Stability and Convergence

To explore whether we could improve (the stability of) performance, we ran the same experiment with $\alpha = 0.1$. For the first 100 episodes, the results can be seen in 4.3. Learning is indeed lower than with $\alpha = 0.5$ (as seen in 4.2), which explain the difference in initial performance over these 100 episodes in 4.1.

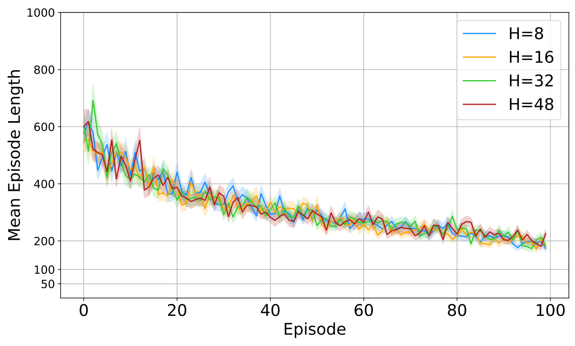
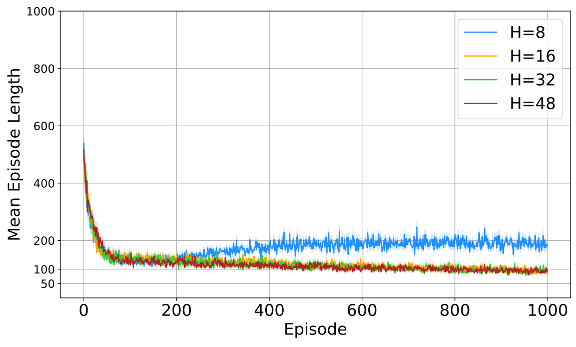


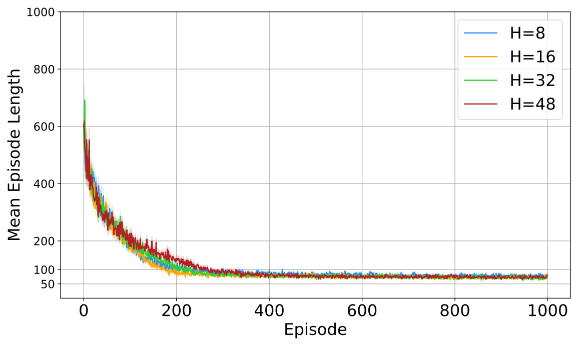
Figure 4.3: Mean episode length +- standard error over 100 identical runs for 100 episodes. α is set to 0.1 and decayed with 0.999 per step in the environment.

However, looking at the figures 4.4a and 4.4b, we can see that with $\alpha = 0.1$, FHQ-learning converges to a more successful policy over a longer period of time, making it preferable to achieve the best and most stable performance. If we look at the average episode length over the last 100 of these 1000 episodes, as shown in table 4.1, we can see a clear difference in performance between the two values of α.

These results confirm that while higher learning rates (such as $\alpha = 0.5$) can enable faster early learning in the initial phase, they are unstable and do not converge effectively over time. A lower learning rate (like $\alpha = 0.1$) provides more consistent convergence with less overall variance.



(a) $\alpha = 0.5$



(b) $\alpha = 0.1$

Figure 4.4: Mean episode length +- standard error over 100 identical runs for 1000 episodes. for both $\alpha = 0.5$ and $\alpha = 0.1$. α is decayed with 0.999 per step in the environment.

Horizon H	$\alpha = 0.1$	$\alpha = 0.5$
8	77.26 ± 0.59	188.48 ± 1.82
16	72.39 ± 0.47	97.63 ± 0.89
32	72.29 ± 0.51	94.68 ± 0.88
48	73.26 ± 0.56	94.66 ± 1.17

Table 4.1: Mean and standard error episode length over 100 identical runs, of the final 100 episodes of an experiment with 1000 episodes for all horizons with $\alpha = 0.1$ and $\alpha = 0.5$.

4.2.2 Statistical Analysis of Respective Performance

To find out whether or not the difference in performance between the four horizons with $\alpha = 0.1$ is statistically significant, we perform a one-way ANOVA test, which gives an F-statistic of 17.76 with $p = 1.49e^{-9}$. This tells us there is a significant difference in performance. To analyze what that difference is exactly, we perform a Tukey-HSD test, as depicted in appendix B.

We can conclude that horizon 8 performs significantly worse than the other horizons. Horizons 16, 32 and 48 show no significant difference in performance among them. Thus we can conclude that, contrary to what was suggested in the paper, the short horizons do not outperform the longer horizons.

4.2.3 Rejection of Proposed Research Direction

Based on the results above, we have to reject the assumption that short horizons consistently outperform longer horizons in highly stochastic environments. As this assumption was the foundation for our initial idea to adapt the planning horizon based on state-level stochasticity, we chose not to pursue this direction further.

4.2.4 Deterministic environment

In the deterministic case, where the environment is identical but there is no slipperiness, all horizons perform similarly. They all converge to a policy with an average episode length of ≈ 30 steps, while the optimal path is only 14 steps. From these results, we also have to reject our second assumption: that longer horizons would be preferable in low-stochastic settings. The results can be seen in appendix C.

This further strengthens the rejection of our proposed method, in which we wanted to adapt the length of the horizon to the amount of stochasticity.

4.2.5 Non-homogeneous environment

In the non-homogeneous environment introduced in section 3.1.2, we ran several exploratory experiments using fixed-horizon lengths. However, as discussed in sections 4.2.3 and 4.2.4, the foundational assumptions for adaptive horizon selection could not be verified. The results for this environment, as seen in appendix D did not show any meaningful patterns or differences between the horizons.

4.3 α-decay

An interesting finding we came across was that, without α-decay, the horizons do not manage to converge as effectively, and especially the performance of the shortest horizon ($h = 8$) worsens. While α-decay is given as an assumption in the proof by Asis et al. [2020], it is generally not considered necessary when empirically testing convergence of reinforcement learning algorithms (Chen et al. [2021]). In 4.5, the results can be seen for an experiment identical to that in 4.4b, but without decaying α. We can see a slight divergence of $h = 8$ over time in the plot, which is supported by the performance table over the last 100 episodes in 4.2.

4.4 Computational cost

In 4.6, we have plotted the mean episode length against the computational cost of the agent in the

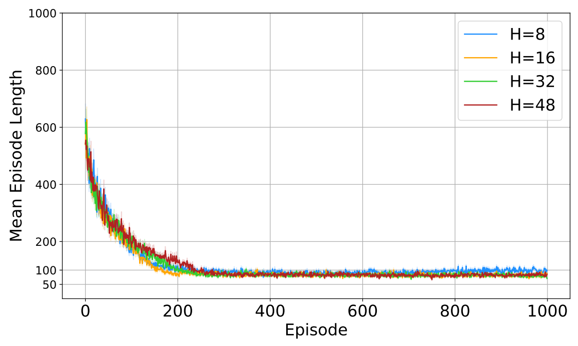


Figure 4.5: Mean episode length +- standard error over 100 identical runs for 1000 episodes with $\alpha = 0.1$ and not decayed.

Horizon H	$\alpha = 0.1$ (decayed)	$\alpha = 0.1$
8	77.26 ± 0.59	98.18 ± 8.02
16	72.39 ± 0.47	82.07 ± 8.23
32	72.29 ± 0.51	81.04 ± 7.91
48	73.26 ± 0.56	82.96 ± 9.67

Table 4.2: Mean and standard error episode length over 100 identical runs of the final 100 episodes of an experiment with 1000 episodes for all horizons with decayed and non-decayed α.

same environment. As explained above in section 3.4.2, the short horizons perform less calculations per step in the environment, leading to a lower overall computational cost per episode. As all horizons converge after a similar amount of episodes (seen in figure 4.4b), that means the shorter horizons are preferable from a computational viewpoint.

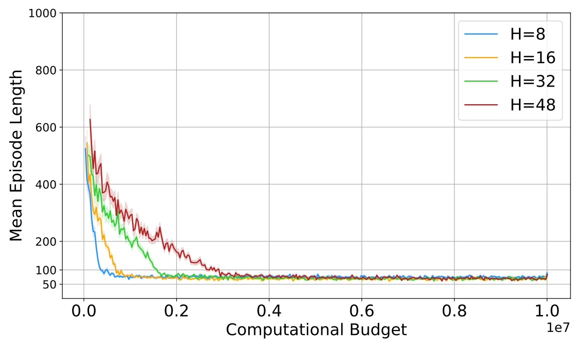


Figure 4.6: Mean episode length +- standard error over 100 identical runs, plotted against the computational cost. $\alpha = 0.1$ and decayed with 0.999 per step in the environment.

5 Discussion

5.1 Trade-off Between Performance and Computational Cost

Our results indicate a clear trade-off between the computational efficiency and the performance of different horizon lengths in Fixed-Horizon Q-learning. While short horizons like $H = 8$ learn quickly and are computationally cheap per episode, they fail to converge effectively and result in significantly worse long-term performance compared to longer horizons. On the other hand, horizons $H = 16$, $H = 32$, and $H = 48$ all show similar performance, but at increasing computational costs.

Thus, we conclude that a moderate horizon like $h = 16$ shows the best balance in performance and computational cost.

5.2 Adaptable Horizon Length

A central goal of this project was to explore dynamic horizon adjustment based on state-level stochasticity. However, our experiments showed that the underlying assumptions for this idea do not hold: short horizons did not consistently outperform longer ones in stochastic environments, nor did longer horizons outperform shorter ones in deterministic settings. Thus, reducing the planning horizon dynamically when state-level stochasticity is low does not appear to be a feasible research direction, at least in the tabular, one-step FHQ-learning case.

5.3 α-Decay

Although α-decay is typically not necessary for convergence in many RL algorithms, our results suggest it plays a critical role in stabilizing FHQ-learning, especially for short horizons. Without decay, learning became less stable and performance deteriorates significantly.

5.4 Limitations and Future Work

5.4.1 Investigating the inverse

From these results, one could conclude that investigating the inverse could be an interesting research direction. We showed that longer horizons perform better in highly stochastic environments, but in deterministic ones, all horizons perform similarly. So, one could argue that it could be beneficial to dynamically adjust the horizon based on state-level stochasticity, but instead reduce the horizon when stochasticity is low to save computational cost.

5.4.2 Failure to reproduce exactly

As discussed in the results above, we were not able to reproduce the experiment by Asis et al. [2020] exactly. Some of the parameters were not exactly defined in the initial research, like whether or not α was decayed, or the value of ϵ in ϵ-decay. Testing more configurations of these hyperparameters could get us results that are closer to those by Asis et al. [2020]

5.4.3 Values of h

While we tested a reasonable range of horizons ($H = 8$ to $H = 48$), further experiments with more extreme- and intermediate values could be done to get an even better understanding of this tradeoff.

5.4.4 Complex environments

Additionally, while we focused exclusively on tabular environments, extending this analysis to linear or deep function approximation settings could test whether the findings generalize.

Finally, while adaptive horizon selection may not be fruitful in this setting, it could still be worth exploring in more complex domains where the dynamics vary more dramatically across the state space.

6 Conclusion

In this thesis, we investigated the performance and stability of Fixed-Horizon Q-learning (FHQ) in tabular stochastic environments. Contrary to earlier findings by Asis et al. [2020], we found that short planning horizons do not necessarily lead to better performance in highly stochastic settings. Instead, a moderate horizon length such as $h = 16$ provided the best trade-off between computational cost and learning performance.

We further showed that α-decay plays a critical role in stabilizing learning and achieving convergence—especially for shorter horizons. Our empirical results rejected the initial hypothesis that the planning horizon should be dynamically adjusted based on state-level stochasticity, as we found no consistent relationship between local environment stochasticity and optimal horizon length.

Overall, our findings suggest that while FHQ-learning offers an elegant approach to mitigating instability in value estimation, adaptive horizon selection may not be beneficial in simple tabular environments. Future work should investigate whether these results generalize to more complex settings involving function approximation or deep learning.

References

Kristopher De Asis, Alan Chan, Silviu Pitis, Richard S. Sutton, and Daniel Graves. Fixed-horizon temporal difference methods for stable reinforcement learning, 2020. URL `https://arxiv.org/abs/1909.03906`.

Yifei Chen, Lambert Schomaker, and Marco Wiering. An investigation into the effect of the learning rate on overestimation bias of connectionist q-learning. In Ana Paula Rocha, Luc Steels, and Jaap van den Herik , editors, *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, volume 2, pages 107–118. SciTePress, February 2021. ISBN 978-989-758-484-8. doi:10.5220/0010227301070118. 13th International Conference on Agents and Artificial Intelligence ; Conference date: 04-02-2021 Through 06-02-2021.

Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL `https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf`.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. 10 1993.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992. doi:10.1007/BF00992698. URL `https://doi.org/10.1007/BF00992698`.

A Reproduction of results

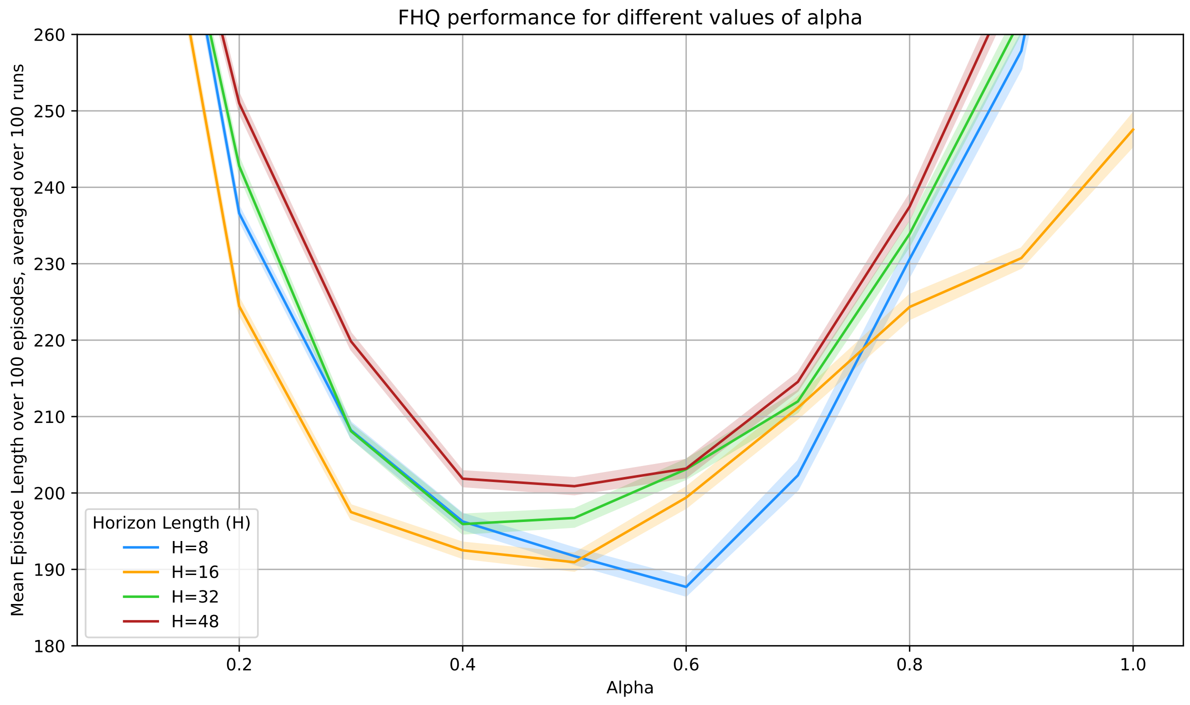


Figure A.1: Our reproduction of the results by Asis et al. [2020]; Mean episode lengths over 100 episodes of FHQ-learning with various step-sizes and horizons of interest. Results are averaged over 100 runs, and shaded regions represent one standard error.

B Tukey HSD table

Table B.1: Tukey HSD test following the one-way ANOVA test for episode length differences between horizons [8, 16, 32, 48].

Gr 1	Gr 2	Mean Diff	p-adj	Low	Upp	Rej
8	16	4.87	0.000	2.33	7.41	**Yes**
8	32	4.97	0.000	2.43	7.51	**Yes**
8	48	4.00	0.000	1.46	6.54	**Yes**
16	32	0.10	0.998	-2.44	2.64	No
16	48	-0.87	0.684	-3.41	1.67	No
32	48	-0.97	0.615	-3.51	1.57	No

C Deterministic environment

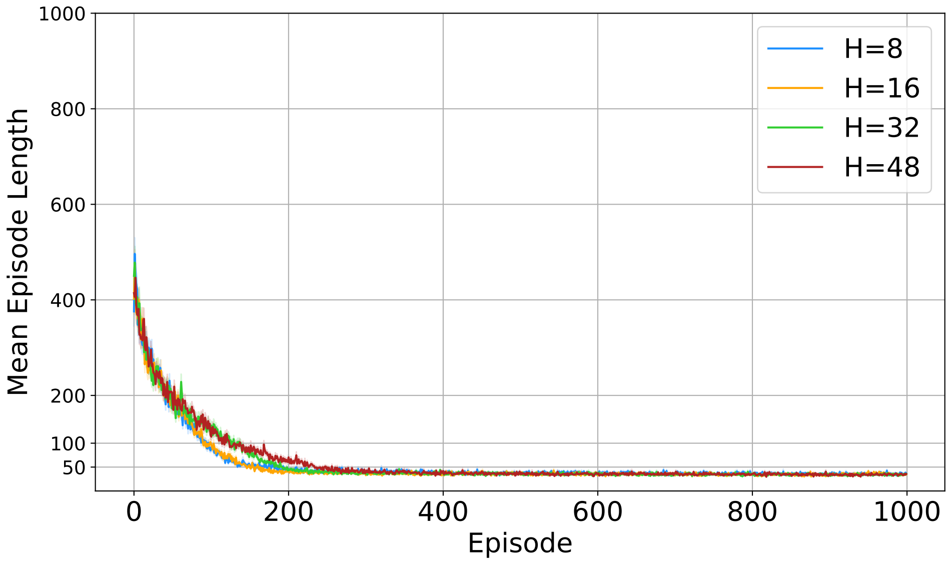


Figure C.1: Mean episode length +- standard error over 100 identical runs for 1000 episodes in the deterministic environment (slipperiness = 0). α is set to 0.1 and decayed with 0.999 at each step.

D Non-homogeneous environment

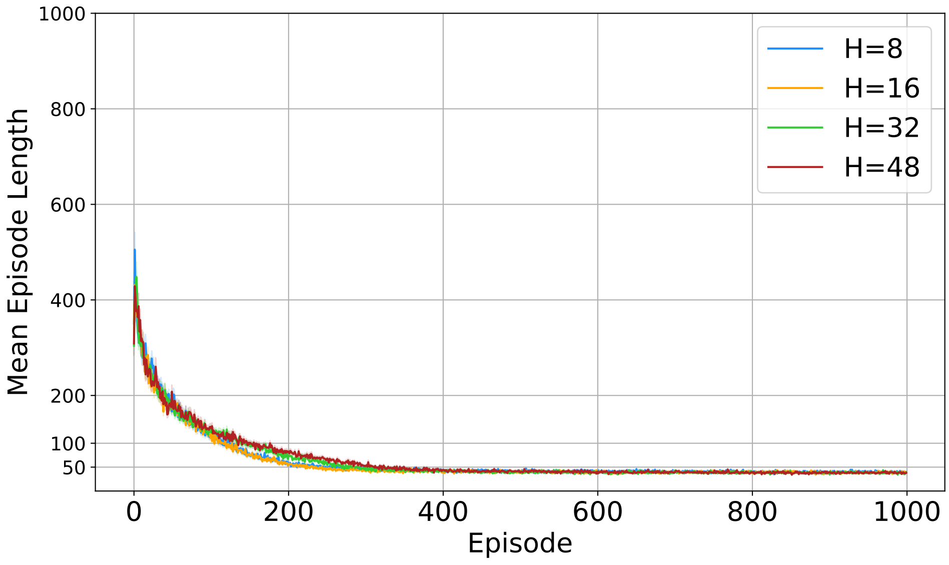


Figure D.1: Mean episode length +- standard error over 100 identical runs for 1000 episodes in the non-homogeneous environment (as described in 3.1.2). α is set to 0.1 and decayed with 0.999 at each step.