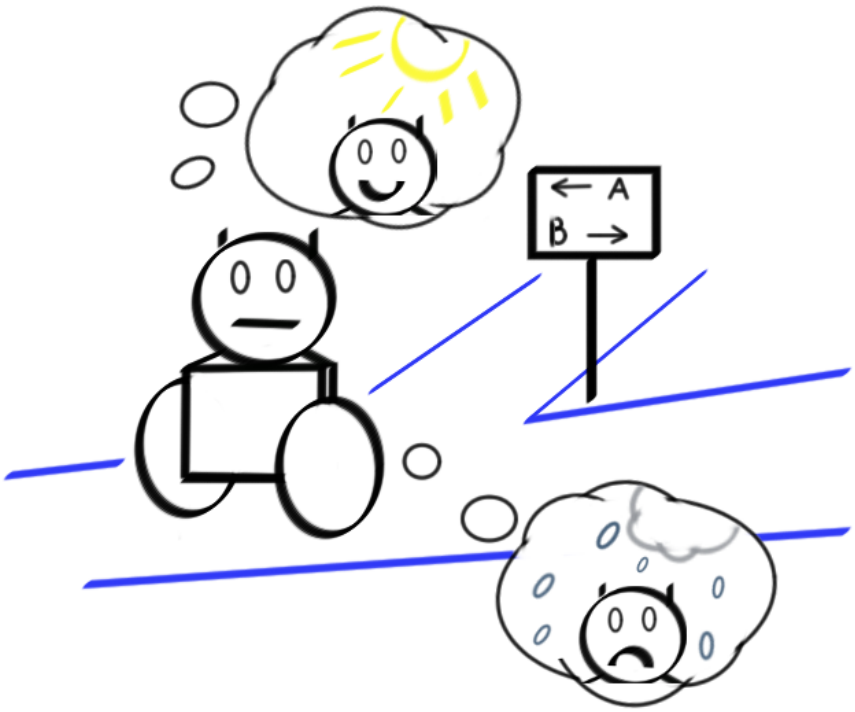
Multiple Q-learning for Model-free Online Reinforcement Learning

B.J.L. Haanstra

June 30, 2014



Should one go left or should one go right?
illustration by Maud Haanstra

As partial fulfillment for the degree: Master of Science
Master Artificial Intelligence, Vrije Universiteit
Amsterdam, The Netherlands, 2014

Comittee	Dr. Hado P. van Hasselt	University of Alberta
	Dr. Agoston E. Eiben	Vrije Universiteit

Contents

Chapter 1

Introduction

Over the past few decades, *artificial intelligence* (AI) has grown immensely as a field with numerous successes in building *intelligent systems*. Nowadays, many parts of our lives are being automated and assisted by such systems. For example, recognizing handwritten addresses on mail, detecting fraud, spam filtering, routing networks and Google search.

In the future, we expect, and may already witness, their increasing application in social, expedition, domestic, commercial and industrial domains. For example, self-driving cars have been developed in the past years, and robots are believed to take over various jobs from humans in the next few decades.

However, developing intelligent systems can be complicated, because one has to address how it should behave for every possible scenario that it may encounter. More often than not, it is difficult to foresee all scenarios and our understanding may differ from reality. In some cases, we may not even be sure yet how it should behave. For example, we do not yet know how to play games like Go and Chess perfectly, nor is it always trivial to program how humans do things such as telling apart dogs from cats. This may result in the system performing faulty or inefficiently, and potentially have harmful and critical consequences. It is therefore essential that the system is capable of adapting to the task at hand by itself. That is, it has the capability of *learning*.

Machine learning is a branch of AI that concerns itself with developing computational systems that learn to perform tasks from data. In this thesis, we focus on *reinforcement learning*: an area of machine learning that is concerned with *learning behaviour* for a *sequential decision problem* with unknown dynamics through *feedback signals*.

Reinforcement Learning

Reinforcement learning (RL) is about an *agent*, a decision-making entity, that interacts with an environment, a sequential decision problem, in which it endeavours to accomplish a task through a series of *actions*. Instance problems of this type are prevalent in both the physical and digital world. Figure 1.1 shows a few examples.

Whenever an agent interacts by executing an *action*, it observes a feedback signal and a new *state* of the environment. This feedback signal indicates how good the executed action was and is usually referred to as a *reward* in the literature. In RL, one is concerned with how an agent should choose actions in succession so as to maximise the *expected return*: the *expected sum of (discounted) rewards* (Sutton and Barto, 1998).[1]

[1] Alternatively, a feedback signal corresponds to the incurred *cost* of an action and one is concerned with minimizing the expected sum of costs (Bertsekas and Shreve, 2007).

Commonly, the environment is modelled as a *Markov decision process* (MDP) (Howard, 1960; Puterman, 1994). Such modelling choices involve assumptions about the environment, made by the *designer* of the agent, that allows learning and decision making to be manageable. The *policy* of the agent, which defines its decision making and thus its behaviour, is in general a function of the history of made observations and actions. For any MDP, there always exists an *optimal policy* that maximises the expected return (Bertsekas and Shreve, 2007; Puterman, 1994).

Naturally, one is interested in this optimal policy, which encompasses the *optimal actions*, and use it to *control* the environment. However, this optimal policy is typically not known in advance as the dynamics of the environment usually are not perfectly known. Namely, one is *uncertain* about the consequential rewards and states that follow one's actions. In some cases, these consequences can even be stochastic, resulting in different rewards and states every time the same action is tried.

Furthermore, the agent should not only consider the immediate reward of an action, but also its consequences in the long term due to the environment's sequential nature. For example, one can choose to withdraw one's money from the bank and benefit from it immediately, or one can choose to wait and benefit from a higher interest over the initial deposit at a later time. This idea also applies to sequential decision problems.

Learning to Control

The goal is thus to find the optimal policy. Assuming that the unknown environment can be interacted with, we can proceed by having the agent try out actions, according to its *behaviour policy*, to gather *samples* about the actions' consequences. From these samples, we can learn about the environment and, more importantly, *learn to control*.

However, unlike *supervised learning*, a different area of machine learning, the feedback signals in RL do not explicitly inform the agent about the optimal actions. Instead, it merely indicates numerically how good the tried actions were, and the agent has to discover by itself which course of actions will yield the most reward.

At the present, there exists a vast amount of RL algorithms that can learn about good and even optimal policies from interaction. These algorithms can be classified as follows. First, whether it learns a model of the environment (*model-based*) or not (*model-free*). Second, whether it memorizes all samples to compute a solution with every interaction (*batch*), or update an existing solution and discard samples immediately after (*online*) (Lange et al., 2012). Third, one can learn about the behaviour policy that the agent follows in the environment (*on-policy*) or learn about an *estimation policy* while following a different behaviour policy for interaction (*off-policy*). Finally, whether it explicitly learns a policy, a *value function* or both (Sutton and Barto, 1998).

Value functions are used to map states, or actions in states, for a policy to *values*. The

Figure 1.1: Examples of real-life sequential decision problems.

value of a state, or the *action-value* of an action in a state, corresponds to the expected return that the agent will accrue when it is in that state, or takes that action in that state, and follows the policy afterwards. Many RL algorithms learn value functions to determine which actions and policies are better by comparing values. Ultimately, an optimal policy can be easily derived from the *optimal action-values*.

Temporal-difference learning (TD) is a family of model-free value-based RL algorithms and is one of the most popular topics in the field of RL (Rummery and Niranjan, 1994; Sutton and Barto, 1998; van Hasselt, 2011; Watkins, 1989). They *estimate* values by *bootstrapping*: they use *value samples* that are based on other value estimates. Despite that this makes learning intrinsically biased and non-stationary in MDPs due to bootstrapping, TD tends to be computationally cheap and can conveniently learn optimal policies. *Q-learning* is arguably the most famous TD algorithm, which can be used to learn an optimal policy regardless of the behaviour policy by directly estimating the optimal action-values in a model-free online off-policy manner (Watkins, 1989).

1.1 Focus of this Thesis: Learning to Behave

In this thesis we are interested in *online reinforcement learning* (ORL). Here, we consider the case that an agent is deployed in an unknown environment and has to maximise its *online performance*: the expected return of the agent's behaviour policy (van Hasselt, 2011). Ergo, it matters how the agent *behaves* during its lifetime in the environment.

Given that the environment is initially unknown to the agent, it will unlikely know how to behave optimally from the start and therefore has to learn from interaction. However, just learning about good behaviour is not sufficient, as we want the agent to perform well while still learning.

Obviously, if we have the agent randomly roam through the environment, then it may gather samples that it can use to learn control, but it will also jeopardize its online performance. The goal of ORL is thus to develop an algorithm that a learning agent can use to choose actions that will consistently accrue a lot of reward during its lifetime.

The challenge that accompanies this is that the agent can only learn about actions by trying them out. In doing so, it risks squandering valuable interaction time when the chosen actions do not amount to good behaviour nor help it learn to improve its behaviour. Moreover, if the agent sticks to choosing the same actions, then it will always experience the same and risks missing out on learning better behaviour. This is known as the *exploration-vs-exploitation dilemma*: the agent has to balance the *exploration* of learning about good actions with the *exploitation* of known good actions (Sutton and Barto, 1998; Wiering and van Otterlo, 2012).

In the literature, there are many approaches that aim to explore efficiently. Commonly, they employ reinforcement learning algorithms to learn from samples and guide exploration through heuristics or statistical methods. Contemporary state-of-the-art algorithms are mainly model-based, and tend to be very computationally demanding as they account for every foreseeable future of actions and consider many plausible environments that the agent may be in (Duff, 2002; Li, 2012; Vlassis et al., 2012).

We believe that autonomous agents in the future will have to be completely self-reliant and rarely have access to an abundance of computational resources. For example, robots that work remotely in deep sea or space can not rely on a supercomputer doing all the heavy work. Therefore, we focus on online model-free value-based algorithms for ORL in MDPs that are computationally economical and estimate action-values.

Early online model-free attempts at exploring more efficiently than random have based action selection on combinations of action-value estimates and other statistics such as the number of times that an action was tried before (Thrun, 1992).

More recent and sophisticated alternatives pursue efficient exploration by accounting for one's *uncertainty* over the optimal action-values in some way. For example, using *Bayesian statistics* to quantify uncertainty explicitly (Dearden et al., 1998), using *interval estimation* or *probably approximately correct learning* to estimate the optimal action-values *optimistically* to direct exploration towards poorly understood actions that are potentially optimal (Kaelbling, 1993; Meuleau and Bourgine, 1999; Strehl et al., 2006).

Unfortunately, these online model-free approaches involve assumptions that usually do not hold in MDPs, have parameters that are not easily tuned for the online performance, or do not properly account for the biased and non-stationary nature of learning as they incorporate bootstrapping. As a result, these approaches may work well for certain MDPs, but are not suitable to function as an 'off-the-shelf' algorithm for ORL.

Our observation is that Q-learning, as well as other temporal-difference learning algorithms, does not involve such restrictive assumptions or critical parameter tuning. But on the other hand, it only provides a single estimate that does not easily lend itself for balancing exploration with exploitation.

We hypothesize that using *multiple estimates*, for the optimal action-values, can potentially improve the agent's action selection as the difference among the estimates can be used to signify our (un)certainty over the optimal action-values, and hence which actions are potentially optimal. I.e., when an optimal action-value is estimated at different values, then we are uncertain. This can be used as an incentive for exploration. We call our approach *Multiple Q-learning* (MQ) as the estimation is based on Q-learning.

However, if we update all the estimates identically, then action selection will be no different than using a single estimate. Thus, the addition of more estimates poses several challenges such as how the multiple estimates should be updated, how many to use, and how they are best used for ORL.

Research Questions

Our main research question and related questions can be formulated as follows:

1. "**Is it beneficial to use multiple action-value estimates for model-free online reinforcement learning?**"
   (a) *How should the multiple estimates be updated?*
   (b) *How many estimates should one use?*
   (c) *Which one of the considered exploration strategies, that translates the multiple estimates to an actual decision, works best?*

1.2 Outline Thesis

This thesis is constructed as follows. We formalize the concepts of reinforcement learning in Chapter 2. In Chapter 3, we discuss online reinforcement learning and related work. In Chapter 4, we develop our ideas for updating and using multiple estimates for ORL: *Multiple Q-learning* (MQ). In Chapter 5, we perform empirical experiments to investigate the design choices inherent to MQ, and compare its performance to other model-free alternatives. We discuss the results in Chapter 6 and conclude this thesis in Chapter 7.

Chapter 2

Background

In this chapter we formalize concepts of reinforcement learning (RL) that are necessary for understanding the remainder of this thesis: the agent, the environment and how one can learn from interaction. We strongly recommend the reader to read through this chapter, even when they are already familiar with these concepts.

For more background on sequential decision making and reinforcement learning we recommend Sutton and Barto (1998) and Buşoniu et al. (2010).

Notation

We hope that our mathematical notation is obvious from context, and otherwise refer the reader to the nomenclature at the end of this thesis.

Throughout, we use capital letters for functions and random variables[2], non-capital letters for non-random variables and random variates, and blackboard bold letters for sets and spaces.[3]

Outline Chapter

In the first section we introduce Markov decision processes for environments and how to model agents. In Section 2.2, we discuss learning algorithms and exploration strategies for learning control from interaction.

In the next chapter we discuss online reinforcement learning, where the agent's behaviour is evaluated, and discuss related work from the literature.

[2] A random variable is a variable whose value is subject to variations due to chance. A random variate is a particular outcome of a random variable.

[3] A *space* is a set with a structure added among its elements.

2.1 Markov Decision Processes

A *Markov decision process* (MDP) is an often-used discrete-time framework for modelling the dynamics of an environment, a sequential decision problem (SDP) (Howard, 1960; Puterman, 1994). An MDP is noted as a sextuple $M = (\mathbb{S}, \mathbb{A}, P, R, I, \gamma)$, where

- $\mathbb{S}$ is the state space; the set of states that the MDP can be in. A state contains information regarding the environment. E.g., when flying a helicopter, a state pertains to the position and orientation of the helicopter. Let $s \in \mathbb{S}$ denote a context-dependent state, and let s_t denote the observed state at timestep t.
- $\mathbb{A}$ is the action space; the set of actions available to the agent. E.g., the pilot can choose to yaw, roll or pitch the helicopter. Let a denote a context-dependent action, and let a_t denote the action taken at t.[4]
- $P : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$; is a transition function that outputs a probability for a *transition* to a *successor state* after taking an action in a state.[5] Let S_t denote the random variable for the state at t. E.g., after steering, the helicopter makes a turn.
- $R : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{R} \to [0, 1]$; is a reward function that outputs a probability for a numeric reward given a transition. Let R_t denote the random variable for the reward at t, and let r_t denote the observed reward at t. E.g., not crashing is good.
- $I : \mathbb{S} \to [0, 1]$; is an initial state function that outputs the probability that a state is the MDPs initial state s_0. E.g., the helicopter can take off at several locations.
- $\gamma \in [0, 1]$; is a discount rate parameter that weighs off imminent rewards versus long-term rewards.

At the beginning of interaction with an MDP, an agent observes an initial state s_0 and chooses an action a_0. From that point on, at each discrete *timestep* $t = 0, 1, 2, \ldots$, it observes a successor state $s_{t+1} \sim P(s_t, a_t, \cdot)$ and a reward $r_{t+1} \sim R(s_t, a_t, s_{t+1}, \cdot)$, and chooses a new action a_{t+1} accordingly. Figure 2.1 illustrates this *cycle* of observations and interactions between agent and environment.

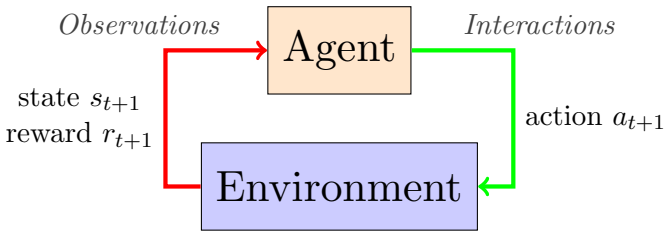


Figure 2.1: Schematic diagram of the interaction cycle between agent and environment.

A *stochastic MDP* is an MDP where the transition function, reward function or initial state function is stochastic. Conversely, an MDP is deterministic when all three are deterministic. Notice that for a deterministic MDP, S_t and R_t can still be stochastic when actions are selected stochastically, which we discuss in the next subsection.

Over time, the made observations and actions accumulate to a *history*. If the MDP or decision making is stochastic, then the resulting history is stochastic. We let $H_{0:t}$

[4] Alternatively, the actions available in a state can result from a function on the state.

[5] For each state s and action a, the probability of transitioning to a successor state s' respects constraints $Pr(s' \mid s, a) \geq 0$ and $\sum_{s' \in \mathbb{S}} Pr(s' \mid s, a) = 1$.

denote the random variable over the history from timestep 0 up to t and define it as

$$H_{0:t} = \langle S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, R_t, S_t \rangle \ , \tag{2.1}$$

where A_t is the random variable over the action taken at timestep t.

The duration of the interaction is the *horizon*. If the interaction stops after a finite number of timesteps, we say the problem has a *finite horizon* and otherwise an *infinite horizon*. Some MDPs have *terminal states*, which divide interaction into *episodes*. Whenever a terminal state is reached, it ends the current episode and a new one starts by resetting the MDP to an initial state. E.g., a maze has an exit as terminal state.

An MDP makes two important assumptions about the environment. First, it assumes that an agent can always perfectly observe the state of the environment. And second, it assumes that the *Markov property* holds (Howard, 1960; Puterman, 1994):

The future is conditionally independent of the past given the present.

That is, we assume that

$$Pr(R_{t+1}, S_{t+1} \mid S_t, A_t) = Pr(R_{t+1}, S_{t+1} \mid H_{0:t}, A_t) \ .$$

We can exploit this assumption in both learning and decision making as we do not have to account for the past when we consider the present.

Types of Markov decision processes and generalizations

A *finite MDP* has a finite state space and finite action space, and a *continuous MDP* has a continuous state space or action space. For an *ergodic MDP*, all states can be reached from all other states in a finite number of transitions. A *stationary MDP* is an MDP for which the parameters are fixed and independent of the timestep.

A *partially observable Markov decision process* (POMDP) does not assume that the state is perfectly observable and hence generalizes the MDP framework. This introduces an uncertainty over the state of the environment and complicates decision making.

Multi-armed bandit problems can be considered as a special type of MDP: it always starts in the same state and each action leads to a terminal state. The name is inspired by one-armed bandits, which are machines found in casino's where one pulls a lever in the hope of getting a winning combination (Sutton and Barto, 1998).

In this thesis, we will only consider finite ergodic stationary MDPs with infinite horizon that have more than one state and are fully observable.

2.1.1 Policies and Objective Functions

In decision making, one seeks to optimise an objective that is chosen prior to any interaction with the problem. For example: reaching a specific state or maximise some reward signal over the horizon. In this subsection we discuss a common objective for RL and model the decision making of an agent.

We model the decision making of an agent as a *policy function*, or *policy*, and denote it by π. In general, a policy maps histories and actions to probabilities:

$$\pi : \mathbb{H} \times \mathbb{A} \to [0, 1] \ ,$$

where $\mathbb{H}$ is the space of all possible histories. For an MDP, it suffices that a policy is a mapping from states and actions to probabilities. A Markovian policy is defined as:

$$\pi : \mathbb{S} \times \mathbb{A} \to [0, 1] \ .$$

At each timestep, an agent will choose a single action available in the current state according to its probability,

$$A_t \sim \pi(H_{0:t}, \cdot) \quad \text{or} \quad A_t \sim \pi(S_t, \cdot) \ ,$$

which can be interpreted as the agent's preference for that action.

A policy is stochastic when in some state more than one action has a non-zero probability of being selected. Conversely, a deterministic policy has in each state a single action with probability one.

For a learning agent, its preference over actions can change over time and is therefore time-dependent and typically not Markovian as it is the result of the history. We use Π to denote the set of all policies, and π_t to refer to how actions are chosen at timestep t.

2.1.1.1 Objective Functions

We measure the performance of a policy π on an MDP M by using an *objective function*. In reinforcement learning, we commonly want to maximise the *expected sum of discounted rewards*, which we will from now on refer to as the *expected return* (Szepesvári, 2010). The expected return of a π on an M for an infinite horizon is defined by[6]

$$J(\pi, M) = \mathop{\mathbb{E}}_{\pi, M}\left\{\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right\} \ , \tag{2.2}$$

where $\sum_{t=0}^{\infty} \gamma^t R_{t+1}$ functions as a single random variable called the *return*, which depends on the stochasticity of rewards, transitions, initial states and the policy. We use $\mathbb{E}_{\pi,M}\{X\}$ to denote the expectation of a random variable X when following π on M.

The discount rate parameter, $\gamma \in [0, 1]$, weighs off immediate rewards versus long-term rewards (Sutton and Barto, 1998). I.e., a reward received n steps in the future only contributes γ^n of that reward. As γ goes to one, future rewards contribute more.

Conceptually, $J(\pi, M)$ represents the *value* of a policy. An objective function thus enables one to compare and prefer policies based on their values.

A policy that optimises the objective function is called an *optimal policy*, and is denoted by π^*_M. We write π^* when M is clear from context. For any MDP, a deterministic Markovian π^* always exists (Bertsekas and Shreve, 2007; Puterman, 1994). For the expected return over an infinite horizon, π^* has the maximal possible value:

$$\pi^* = \arg\max_{\pi \in \Pi} J(\pi, M) \ . \tag{2.3}$$

2.1.1.2 Control and Prediction

In reinforcement learning, we often want to address the *problem of control*: finding an optimal policy which optimises the chosen objective function. The *problem of prediction* is about estimating the value of a policy $J(\pi, M)$.[7] Notice that one can use the latter to solve the former: we can compare and choose the policy with the highest value.

In this thesis, we are concerned with how an agent behaves in an environment. This necessitates that one addresses the problem of control.

[6] For a finite horizon, the optimal policy becomes time-dependent.

[7] This is also known as *policy evaluation* in the literature.

2.1.2 Value Functions and Q-Functions

In this subsection we first introduce *value functions* and then *action-value functions*. The latter, more conveniently called *Q-functions*, are an extension to the former. These functions can be used for prediction and control (Sutton and Barto, 1998).

A value function maps each state to a value that corresponds with the output of the objective function. We use V_M^π to denote the value function when following π on M, and let $V_M^\pi(s)$ denote the *state-value* of a state s. We omit M when it is clear from context. A state-value for the expected return of a policy on M is defined by

$$V_M^\pi(s) = \mathop{\mathbb{E}}_{\pi,M}\left\{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\,\middle|\, S_t = s\right\} \ ,$$

where $\mathbb{E}_{\pi,M}\{X \mid Y\}$ is the conditional expectation of X given Y when following π on M.

Thus, a state-value indicates how good it is to be in its respective state in terms of the expected return. If we want to perform well, we prefer states with high state-values.

The value function of π^* is called the *optimal value function* and is denoted by $V_M^{\pi^*}$ and more conveniently V^*. The *optimal state-value* of a state is denoted by $V^*(s)$. It holds that for each state, $V^*(s)$ is equal or higher than for other policies:

$$\forall_{\pi\in\Pi}, \forall_{s\in\mathbb{S}} : V^*(s) \geq V^\pi(s) \ .$$

2.1.2.1 Q-functions

An *action-value function*, which we will call a *Q-function* from now on, is a simple extension to a value function to account for actions as well. It is used to map combinations of states and actions to values (Sutton and Barto, 1998). We will refer to a single combination as a *state-action pair*, and its value as a (policy) *action-value*.

We use Q_M^π to denote the Q-function when following π on M, and let $Q_M^\pi(s,a)$ denote the action-value of the state-action pair (s,a). We omit M when it is clear from context. We define an action-value for the expected return by

$$Q_M^\pi(s,a) = \mathop{\mathbb{E}}_{\pi,M}\left\{\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\,\middle|\, S_t = s, A_t = a\right\} \ . \tag{2.4}$$

Similarly, an action-value indicates the value of an action in a state in terms of the resulting expected return for a policy. Hence, we prefer actions with higher action-values.

The Q-function of π^* is called the *optimal Q-function*. It is denoted by Q^{π^*} and more conveniently Q^*. The *optimal action-value* of (s,a) is denoted by $Q^*(s,a)$. It holds that for each state-action pair, $Q^*(s,a)$ is equal or higher than for other policies:

$$\forall_{\pi\in\Pi}, \forall_{s\in\mathbb{S}}, \forall_{a\in\mathbb{A}} : Q^*(s,a) \geq Q^\pi(s,a) \ . \tag{2.5}$$

Notice that state-values and action-values are related to each other as follows:

$$V^*(s) \equiv \max_{a\in\mathbb{A}} Q^*(s,a) \quad \text{and} \quad V^\pi(s) \equiv \sum_{a\in\mathbb{A}} \pi(s,a) Q^\pi(s,a) \ .$$

It is straightforward to derive policies from Q-functions. For instance, a *greedy policy* of any Q-function Q is a policy that selects in each state a *greedy action*, which is an

action that has the highest action-value. The action-probability for an action is then:

$$\pi^{greedy(Q)}(s,a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases}$$
$$\mathbb{M} = \{a \in \mathbb{A} \mid \forall_{a' \in \mathbb{A}} : Q(s,a) \geq Q(s,a')\}$$
$$= \arg\max_{a \in \mathbb{A}} Q(s,a) \ , \tag{2.6}$$

where $\mathbb{M}$ is the set of actions that share the same highest action-value, and $|\mathbb{M}|$ is its cardinality. Other ways to derive a policy are described in Section 2.2.2.

We can derive π^* by using a greedy policy with respect to Q^*. Interestingly, Q^* is sufficient but not necessary for π^*, because any of the action-values can be changed as long as the same actions keep the highest action-values (Bradtke, 1994).

2.1.2.2 Bellman Equations

For MDPs, action-values (and state-values) share an interesting recursive relation between them, clarified by the so-called *Bellman equation* (Bellman, 1957). The Bellman equation for an action-value is derived as follows:

$$\begin{aligned} Q^\pi(s,a) &= \mathop{\mathbb{E}}_{\pi,M}\Big\{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\Big\} \\ &= \mathop{\mathbb{E}}_{\pi,M}\Big\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s, A_t = a\Big\} \\ &= \mathop{\mathbb{E}}_{\pi,M}\Big\{R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a\Big\} \\ &= \mathop{\mathbb{E}}_{\pi,M}\Big\{R_{t+1} + \gamma \sum_{a' \in \mathbb{A}} \pi(S_{t+1}, a') Q^\pi(S_{t+1}, a') \mid S_t = s, A_t = a\Big\} \ . \end{aligned} \tag{2.7}$$

The Bellman equation version for π^* is called the *Bellman optimality equation*, and is formulated as:

$$\begin{aligned} Q^*(s,a) &= \mathop{\mathbb{E}}_{\pi^*,M}\Big\{R_{t+1} + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a\Big\} \\ &= \mathop{\mathbb{E}}_{\pi^*,M}\Big\{R_{t+1} + \gamma \max_{a' \in \mathbb{A}} Q^*(S_{t+1}, a') \mid S_t = s, A_t = a\Big\} \ . \end{aligned} \tag{2.8}$$

Clearly, an action-value can be alternatively defined as the expectation over immediate rewards and the action-values of successor state-action pairs. Similar equations can be derived for state-values, which we conveniently omit.

For the problem of prediction, we are interested in Q^π, whereas for the problem of control we want to know Q^*. Reinforcement learning algorithms commonly exploit these recursive relations for learning state-values and action-values.

2.2 Learning to Control

For the control problem, we want to know an optimal policy that maximises the expected return on an MDP of interest, denoted by M, see Equation 2.3. But usually, this optimal policy is initially unknown, and has to be found among Π. Figure 2.2 illustrates a fictive policy value landscape over policy space.

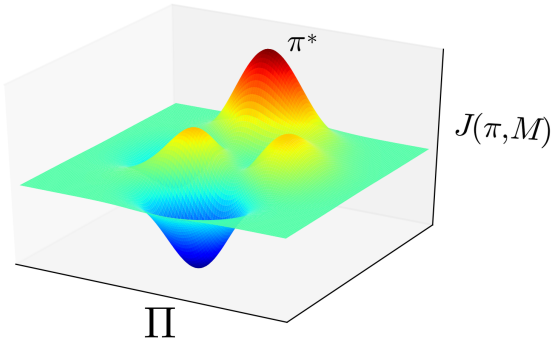


Figure 2.2: A fictive policy value landscape over policy space. We wish to find π^*.

One option is to *learn* the values of policies and actions, see Equations 2.2 and 2.4. These values are dependent on the consequences of actions: they are defined as the expectation over possible returns that depend on M and a policy. In turn, these values enable us to compare and infer which policy controls M better, see Equation 2.5. For this purpose, many different algorithms have been developed.

Planning with a Model

If an accurate model of M is known, we can simulate the consequences of actions. It is then straightforward to compute the values of actions and policies, and to *plan* an optimal policy.[8] For this scenario, many methods such as dynamic programming and meta-heuristics can be used (Sutton and Barto, 1998; Wiering and van Otterlo, 2012).

Unfortunately, a model is usually unavailable as the transition and reward functions are impractical or expensive to acquire (Sutton and Barto, 1998). Opting for a different model is not proper, because its solutions may have adverse results. For example, knowledge on how to drive a car will arguably not help to win a game of Chess.

Reinforcement Learning

The alternative is to experience the consequences through interaction with M. The agent observes *sample rewards* and *sample transitions* as according to its *behaviour policy*. *Reinforcement learning algorithms* use these samples to learn about the dynamics of M, the value of policies and actions, and how to control. As an agent can only learn from samples that it has gathered, it has to discover the optimal actions through experimentation with every action. For this, it can use an *exploration strategy* to guide its behaviour in M. Figure 2.3 illustrates a diagram of a reinforcement learning agent.

Types of Reinforcement Learning Algorithms

Here we briefly describe a few classifications of reinforcement learning algorithms.

Model-based reinforcement learning (MBRL) algorithms use the samples to estimate a model of M, which can then be used to plan an estimate optimal policy. *Model-free reinforcement learning* (MFRL) algorithms learn without estimating any models.[9]

[8] In the literature, using a known model to arrive at a solution is called *planning* (Wiering and van Otterlo, 2012).

[9] Hence, MFRL and MBRL algorithms are respectively called *direct* and *indirect* algorithms (Sutton and Barto, 1998).

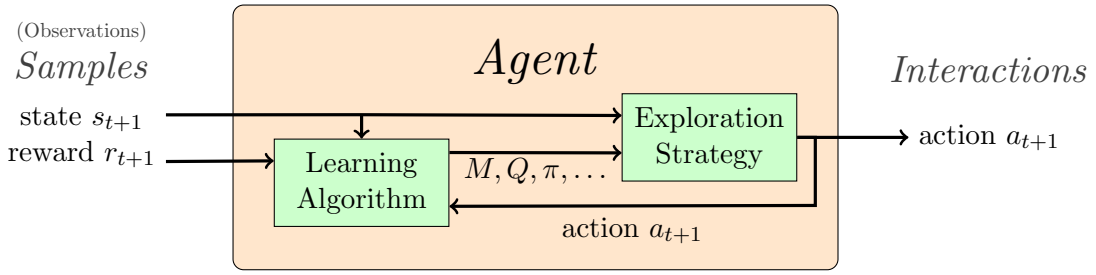


Figure 2.3: Diagram of a reinforcement learning agent.

Among MFRL algorithms, one has the option to estimate a Q-function, policy or both. *Actor-critic* algorithms estimate both a Q-function (critic) and a policy (actor). The critic *evaluates* the actions chosen by the actor, and the actor changes its action selection based on the critic's evaluation, which is the estimated value of the action.

Commonly, one only estimates a Q-function and keeps the policy implicit. Such *value-based reinforcement learning* algorithms learn control by improving the implicit policy towards another policy that is estimated to be better.

MBRL algorithms have the disadvantage that they require more computational resources: a model has to be memorized and estimated in addition to a Q-function or policy (Strehl et al., 2006). Also, for the optimal policy, it is not necessary to memorize the model parameters of states and actions that are suboptimal (van Hasselt, 2011). Ergo, keeping track of a model is sometimes superfluous for learning control.

As last, algorithms can be categorized by how they memorize samples and when they learn a model, Q-function or policy (Lange et al., 2012). *Batch* algorithms typically memorize and use all samples to make a new estimate with every new sample. *Online* algorithms update an *existing* estimate with every new sample, and immediately discard the sample after the update. A reason to use the latter is that the number of samples may accumulate indefinitely, which may strain the agent's computational resources.

In this thesis, we will focus on *temporal-difference learning*, which is a family of model-free value-based reinforcement learning algorithms. Arguably, they are one of the most computationally efficient for learning control when a model is not known.

We will not consider *function approximation* as we assume finite MDPs. This allows value functions, Q-functions and policies to be represented by a table, with one entry per state or state-action pair. We note that most of the work discussed in this thesis can be adopted for situations where approximation is imperative such as continuous MDPs.

Outline Section

In the remainder of this section we describe temporal-difference learning and two commonly-used exploration strategies.

In the next chapter, we discuss the challenges that accompany learning when the performance matters during interaction.

2.2.1 Temporal-Difference Learning

Temporal-difference (TD) learning algorithms *bootstrap* value estimates by using samples that are based on other value estimates as inspired by the Bellman equations (Sutton and Barto, 1998). We will only consider estimates for action-values by using Q-functions.

Let Q_t denote a *Q-function estimate* at timestep t, where Q_0 is initialized arbitrarily. In general, a TD algorithm updates the *action-value estimate* of the state-action pair that was visited at timestep t, denoted by $Q_t(S_t, A_t)$, as follows:[10]

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t(S_t, A_t)\big(X_t - Q_t(S_t, A_t)\big) \ , \tag{2.9}$$

where $Q_{t+1}(S_t, A_t)$ is the updated action-value estimate, and X_t is a *value sample* observed at t and is discussed below. The *learning rate* $\alpha_t(S_t, A_t) \in [0, 1]$ weighs off the new sample with previous samples for the same state-action pair.

TD algorithms are typically used in an online manner: whenever the agent interacts and observes a sample reward R_{t+1} and transition S_{t+1}, a value sample X_t is constructed and used to update the relevant estimate. Once updated, these samples are discarded.

2.2.1.1 Value Samples: Q-learning and Expected Sarsa

There are many different choices for X_t. We will only discuss *Q-learning* and *Expected Sarsa*, which are TD algorithms that can be used to learn control in MDPs.[11]

Q-learning is a popular algorithm that estimates Q^* directly (Watkins, 1989; Watkins and Dayan, 1992). Expected Sarsa is an algorithm that estimates Q^π of the behaviour policy (van Seijen et al., 2009). They define X_t as follows:[12]

$$\textbf{\textit{Q-learning}} \qquad X_t = R_{t+1} + \gamma \max_{a' \in \mathbb{A}} Q_t(S_{t+1}, a') \tag{2.10}$$

$$\textbf{\textit{Expected Sarsa}} \qquad X_t = R_{t+1} + \gamma \sum_{a' \in \mathbb{A}} \pi(S_{t+1}, a') Q_t(S_{t+1}, a') \tag{2.11}$$

In the literature of reinforcement learning, one contrasts between the policy followed by the agent for obtaining sample rewards and transitions, called the *behaviour policy*, and the policy that one learns about, called the *estimation policy* (van Hasselt, 2011).

Q-learning is an *off-policy* algorithm: it learns about a *greedy policy* regardless of the agent's behaviour policy. Expected Sarsa is an *on-policy* algorithm: the estimation policy is the same as the behaviour policy.

Q-learning learns control by implicitly improving the policy that it learns about: it is updated towards the action in the successor state that has the highest estimate. For Expected Sarsa, the behaviour policy itself has to become more greedy in order to improve the policy. In both cases, this results in a non-stationary learning task.

Throughout the years, various TD algorithms have been researched. Most of them learn Q^* or Q^π with fewer samples, but are more computationally demanding. The discussion of these algorithms lies out of the scope of this thesis.

2.2.1.2 Learning Rates

The learning rate α_t is used to weigh off new value samples with previous value samples. There are many schemes available for the learning rate. For example, if the MDP is deterministic we can set the learning rate at one at all times.

[10] TD algorithms can be viewed as *incremental estimators* as they incrementally update a previous estimate with a new sample.

[11] Expected Sarsa is an algorithmic improvement of the original Sarsa algorithm which we will not discuss in this thesis (Rummery and Niranjan, 1994; van Seijen et al., 2009).

[12] Notice that X_t can be regarded as a random variable defined in terms of the other random variables R_{t+1}, S_{t+1} and another estimate. Consequently, Q_t is also a random variable.

Usually, the MDP is stochastic and a lower value should be used. For instance, one can use a *hyperharmonic learning rate scheme*:

$$\alpha_t(s_t, a_t) = \frac{1}{n_t(s_t, a_t)^w} \ , \tag{2.12}$$

where $n_t(s_t, a_t)$ is the number of times the estimate for state-action pair (s_t, a_t) has been updated by timestep t, and $w \in (0.5, 1]$ is a tunable parameter of the scheme.

Even-Dar and Mansour (2003) showed that $0.5 < w < 1$ (hyperharmonic) works better than $w = 1$ (harmonic). This is because the value samples are based on other estimates that may change over time, and hence the distribution of the value samples are non-stationary as well as not independent and identically distributed (not i.i.d.).

2.2.1.3 Initialization

Commonly, action-value estimates are initialized at zero as there is no prior knowledge. An alternative is to initialize the estimates optimistically at a high value. For example, one assumes that the maximal observable reward, denoted by R_{max}, for the MDP is known. An optimistic initial estimate is then to suppose that R_{max} is observed at every timestep: $Q_0(s, a) = Q_{max} = \frac{R_{max}}{1-\gamma} = \sum_{t=0}^{\infty} \gamma^t R_{max}$. Notice that initialization may affect convergence speed due to how it biases learning (Sutton and Barto, 1998).

2.2.1.4 Bias and Variance

Despite that sample rewards and transitions are unbiased, the value samples used by TD algorithms are usually biased as they are drawn from a *bootstrap distribution*. Namely, estimates are used instead of the true action-values (see Equations 2.7 & 2.8):[13]

$$Q_t(S_{t+1}, \cdot) \quad \text{instead of} \quad Q^*(S_{t+1}, \cdot) \text{ or } Q^\pi(S_{t+1}, \cdot) \ .$$

However, *bootstrapping* has several advantages. First, we do not need to wait until we have a sample return, consisting of a(n infinite) number of sample rewards, before we can update estimates. This can improve learning speed for prediction and control. Second, we need to know the true action-values (or the optimal policy) to obtain unbiased samples, which are generally unknown initially. Third, the bootstrap distribution's variance is smaller than the variance of the distribution over returns (van Hasselt, 2011).

A second bias occurs when we change the implicit estimation policy towards an action that is perceived as better than it actually is. I.e., for Q-learning, we estimate the optimal action-value based on the maximum action-value estimate (van Hasselt, 2010):

$$\max_{a' \in \mathbb{A}} Q_t(S_{t+1}, a') \quad \text{instead of} \quad Q_t(S_{t+1}, a^*) \ ,$$

where a^* denotes the optimal action in the successor state.

This *overestimation bias* becomes apparent when we severely overestimate action-values of suboptimal actions, due to an extremely high value sample or improper initialization. As a result, other estimates can also become overestimated as value samples will be inflated due to bootstrapping, and may remain inaccurate as well as mislead action-selection for an extended period of time. *Double Q-learning* addresses this overestimation bias, but may suffer from an underestimation bias instead (van Hasselt, 2010).

[13] For multi-armed bandit problems, one can simply use the sample rewards to obtain unbiased value samples for learning values.

2.2.1.5 Convergence

It is proven that, for Q-learning and Expected Sarsa, the initial Q_0 converges to Q^* in the limit with probability one (Tsitsiklis, 1994; van Seijen et al., 2009) (see App. C),

$$\lim_{t\to\infty} Pr(Q_t = Q^*) = 1 \ , \tag{2.13}$$

given that the behaviour policy followed guarantees *infinite exploration*; each action in each state is tried infinitely often in the limit (van Hasselt, 2011):

$$\forall_{(s,a)\in\mathbb{S}\times\mathbb{A}} : \sum_{t=0}^{\infty} \pi_t(s,a) = \infty \ , \tag{2.14}$$

and the *Robbin-Monro's conditions* for stochastic approximation are satisfied:

$$\forall_{(s,a)\in\mathbb{S}\times\mathbb{A}} : \ \sum_{t=0}^{\infty} \alpha_t(s,a) = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} [\alpha_t(s,a)]^2 < \infty \ , \tag{2.15}$$

which is the case for a hyperharmonic learning rate scheme when $w \in (0.5, 1]$. These conditions assure that an estimate, regardless of initialization, can reach any possible number and converges in the limit (Robbins and Monro, 1951). For Expected Sarsa, the behaviour policy also has to become *greedy in the limit* (van Hasselt, 2011):

$$\forall_{s\in\mathbb{S}} : \lim_{t\to\infty} \sum_{a\in\mathbb{A}} \pi_t(s,a) Q_t(s,a) = \lim_{t\to\infty} \max_{a\in\mathbb{A}} Q_t(s,a) \ . \tag{2.16}$$

This makes Expected Sarsa and Q-learning consistent but biased estimators for Q^*, as their bootstrap bias and overestimation bias eventually disappear.[14]

2.2.2 Exploration Strategies

In this section we describe strategies that can be used for guiding exploration through the MDP. Conceptually, they can be viewed as a way to map Q-functions to behaviour policies. From these policies, an action is then selected according to its action-probability.

We mention two common strategies: *epsilon-greedy* and *Boltzmann action selection* (Watkins, 1989; Sutton and Barto, 1998). They are computationally efficient, easy-to-understand and can guarantee infinite exploration.

2.2.2.1 Epsilon Greedy

Epsilon-greedy, henceforth noted as ϵ-*greedy*, is an often-used heuristic method that uses a parameter ϵ to regulate exploration (Sutton and Barto, 1998; Watkins, 1989).[15]

At each timestep, ϵ-greedy chooses a greedy action with probability $1-\epsilon$, and chooses a random action uniformly with probability ϵ. The behaviour policy at t is then:

$$\pi_t(s_t,a) = \begin{cases} \frac{(1-\epsilon_t)}{|\mathbb{M}|} + \frac{\epsilon_t}{|\mathbb{A}|} & \iff a \in \mathbb{M} \\ \frac{\epsilon_t}{|\mathbb{A}|} & \iff a \notin \mathbb{M} \end{cases}$$
$$\mathbb{M} = \arg\max_{a\in\mathbb{A}} Q_t(s_t,a) \qquad , \tag{2.17}$$

where ϵ_t is the parameter's value at t, and $|\mathbb{A}|$ is the cardinality of the action space.

The chance of choosing a greedy action increases as ϵ decreases. When $\epsilon = 0$, it will be identical to a greedy policy, and is unlikely to satisfy infinite exploration.

[14] A consistent estimator is a rule for an estimate that converges to the true parameter in the limit.

[15] Also known as *semi-uniform exploration* (Dearden et al., 1998; Thrun, 1992).

2.2.2.2 Boltzmann Action Selection

Boltzmann action selection is another often-used heuristic for exploration (Sutton and Barto, 1998; Watkins, 1989).[16] It uses a tunable parameter β for mapping action-values to probabilities. Different from ϵ-greedy, the resulting action-probabilities are proportionate to that of the greedy action. The behaviour policy at t is then:

$$\pi_t(s_t, a) = \frac{\exp(Q_t(s_t, a)/\beta_t)}{\sum_{b \in \mathbb{A}} \exp(Q_t(s_t, b)/\beta_t)} \ , \tag{2.18}$$

where β_t is the *temperature parameter* at t, and $\exp(\cdot)$ is the exponential function.

The parameter β is used to intensify the difference between actions' probabilities as according to their (estimated) action-values. As β goes to zero, Boltzmann action selection becomes more greedy. Conversely, as β goes to infinity, all actions will have an equal chance of being selected.

[16] Also known as *softmax action selection* (Sutton and Barto, 1998).

Chapter 3

Online Reinforcement Learning

In this thesis, we consider *online reinforcement learning* (ORL) (Li, 2012). Here, an agent is deployed in an unknown environment and has to autonomously *learn to behave* well without an explicit teacher. The goal is to develop an agent that autonomously maximises its *online performance*: the *expected return* of the *behaviour policy* followed by the agent in its environment (van Hasselt, 2011).[17] Formally, we want it to maximise:

$$J(\pi^b, M) = \mathop{\mathbb{E}}_{\pi^b, M}\left\{\sum_{t=0}^{\infty} \gamma^t R_{t+1}\right\} \ ,$$

where π^b henceforth denotes the behaviour policy of the agent and M is an MDP that is initially unknown to the agent. Notice that π^b is typically not Markovian for a learning agent as it adapts its behaviour based on what it observed in the past.

In this chapter, we discuss how to address ORL and related work from the literature. In the next chapter, we introduce our own approach to ORL: *Multiple Q-learning*.

Exploration versus Exploitation

Ideally, the environment is known and one can find an optimal policy prior to interaction. However in ORL, we are uncertain about the environment and thus the consequences of actions on the agent's online performance. In order to behave well, the agent has to learn about its environment by trying out actions and discover which ones are best.

Evidently, if the agent always selects the same actions, then it will always experience the same and learn nothing new (Wiering, 1999). Unless it behaves optimally, we want it to improve its behaviour by considering other actions: *exploration*.

Unfortunately, exploration is not without cost or risk. An action may turn out to be worse than a known good action, causing the agent to miss out on reward. On the other hand, if the agent ceases exploration prematurely, then it may get stuck *exploiting* suboptimal actions and miss out on reward as well (Kaelbling, 1993).[18]

The challenge to optimising the online performance is thus to appropriately balance the exploration of learning about actions and the exploitation of known good actions. In the literature, this is known as the *exploration-vs-exploitation dilemma*, and is one of the core challenges of (online) reinforcement learning (Sutton and Barto, 1998).[19]

[17] This is closely related to *regret minimization*, where the regret is defined as the optimal performance minus the online performance (Li, 2012). A reason to consider just the online performance is that the optimal performance does not have to be known, which can be a daunting computational task in itself.

[18] This is known as the *sticking problem* (Kaelbling, 1993).

[19] In the field of control engineering, this is also known as *dual control* (Sutton and Barto, 1998).

Sufficient Conditions for Learning to Behave Optimally

The ulterior goal is that the agent will eventually act optimally in its respective environment autonomously. In order to guarantee that the agent behaves optimally when starting from scratch, there are four sufficient conditions that have to be satisfied:

- Opportunity: during the agent's lifetime in the environment, the knowledge that the agent can learn from earlier interactions should have some use in future states. Otherwise the agent can only rely on sheer luck for good performance.
- Capability: the agent should obviously be capable of learning the optimal policy. I.e., the learning algorithm used by the agent should converge to the optimal policy.
- Infinite exploration: if the agent ceases exploration after a finite number of timesteps, then there is a probability that the samples it has observed do not allow the agent to identify the optimal policy without failure. E.g., if the agent observed only negative rewards for an optimal action that has an expected positive value, then it may disregard that action. This probability reduces to zero given infinite exploration.
- Greedy in the limit: the agent will eventually have to act greedy with respect to the optimal policy that it has learned.

In this thesis, we assume that the environment is a finite ergodic stationary MDP with infinite horizon. This means that the agent can visit all state-action pairs infinitely often and the value of actions and policies do not change over time, which implies that the agent can learn and exploit the optimal policy during interaction. For this scenario, most of the algorithms in the literature, including Q-learning and Expected Sarsa, are capable of learning the optimal policy. The last two conditions can be satisfied by properly arranging the exploration strategy of the agent.

Efficient Exploration in the Literature

For our scenario, it is trivial to satisfy infinite exploration and greedy in the limit by using ϵ-greedy when its parameter follows some harmonic series, $\epsilon_t = \frac{c}{n^w}$, where $c \in (0, 1]$, n is monotonously increasing, and $w \in (0, 1]$ (see Section 4.1.1.3) (van Hasselt, 2011).

However, such an approach can jeopardize the online performance, because ϵ-greedy randomly chooses actions that may not amount to good behaviour. Obviously, we want the agent to quickly learn to behave as well as possible. This requires not only an adequate learning algorithm that can learn from samples, but also involves choosing actions that lead the agent to quickly improve its behaviour and thus high rewards.

Interestingly, an action is not necessarily bad for the online performance when its expected return under an optimal policy is low, because when still learning it can provide information that leads to the agent to improve its behaviour. Hence, actions have a certain *value of information* for a learning agent (Duff, 2002). For example in Figure 3.1, the agent is certain that it is in one of several known environments. Once the agent makes an observation that is only consistent with one of them (e.g. by moving up or completely to the left if it is in the red maze) it can instantly recognize the environment it is in and subsequently deduce the optimal policy (that leads to the chest with gold). In other words, those actions have a high value of information.

Unfortunately, computing an action's exact value of information is impractical as it requires one to know, or make assumptions about, the environment and how the agent will change its behaviour in the future. It is for this reason that approaches in the literature balance exploration with exploitation by other means: choosing actions

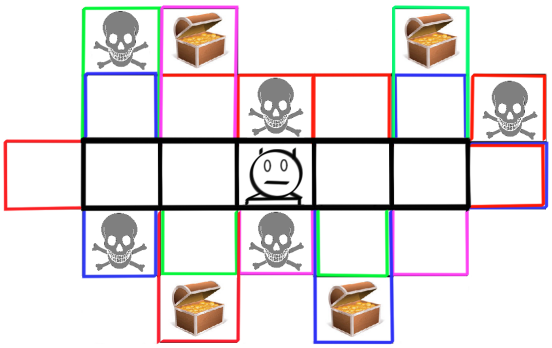


Figure 3.1: Example where the agent is uncertain about the environment and thus the optimal policy for reaching the chest with gold. The black squares are already experienced by the agent. The other coloured squares belong to different hypothetical environments, which have the black squares in common, that the agent might be in.

randomly, account for the uncertainty over the optimal action-values, using statistics related to the learning process, and so on.

Contemporary state-of-the-art algorithms compute the values of actions for the online performance by considering, or even simulating, the consequences for the agent with respect to a set of probable environments or a probability distribution over environments, and choose the action with the highest value (Duff, 2002; Li, 2012; Vlassis et al., 2012).

Scope of this Thesis

Regrettably, these state-of-the-art algorithms are mainly model-based and computationally demanding as they have to solve one or more hypothetical MDPs with every new observation. We argue that autonomous agents will not always have access to heaps of computational resources. For example, a robot on the planet Mars will likely have to be computationally economical as it has to work energy-efficiently in seclusion.

In this thesis, we will therefore consider online model-free value-based reinforcement learning algorithms for ORL, which are commonly computationally cheap as they utilize bootstrapping that provides several advantages for learning, see Section 2.2.1.4.

Thus, our research focuses on how a learning agent should autonomously choose actions to perform robustly online despite imposed limitations on its learning capacities.

Outline Chapter

In the following two sections we divide related work into two families of exploration strategies: *undirected* and *directed* (Thrun, 1992). Section 3.1 describes *undirected exploration* that proceeds by choosing actions randomly, such as ϵ-greedy and Boltzmann action selection. Section 3.2 describes *directed exploration* that methodically guide exploration such as preferring actions that have not been tried recently. In the next chapter, we discuss and motivate our own approach: *Multiple Q-learning*.

3.1 Undirected Exploration

In this section we discuss exploration strategies that are undirected: ϵ-greedy and Boltzmann action selection. They explore by choosing actions randomly.

In the literature, both good and bad results have been reported when using ϵ-greedy. Kuleshov and Precup (2010) have shown that it can outperform more sophisticated algorithms on multi-armed bandit problems.

However for MDPs, Koenig and Simmons (1993) show that it is not ideal for guiding exploration, because it is undirected: it may incidentally ignore what has been learned and choose an action randomly without regard for the consequences. That is, it chooses actions *absentmindedly*, which can jeopardize the online performance as actions that are bad and do not improve behaviour can be selected. In some cases, it may even require exponentially many timesteps to learn a good policy (Whitehead and Ballard, 1991).

For example, an agent has to choose N correct actions in succession to reach the exit in a maze. The chance of reaching the exit when it uses an optimal Q-function with ϵ-greedy is then $(1-\epsilon)^N * 100\%$. E.g., $N = 10$ and $\epsilon = 0.1$ results in 34.87% chance.

The tuning of ϵ-greedy is not trivial (Sutton and Barto, 1998). If we set ϵ too high, then it may try suboptimal actions too often. But if ϵ is too low, then it will take much longer before actions are explored sufficiently to get accurate estimates, and may be subject to the sticking problem. As Boltzmann action selection uses a parameter to regulate exploration in a similar manner as ϵ-greedy, it can face the same difficulties.

According to Sutton and Barto (1998) and van Seijen et al. (2009), Expected Sarsa can have a higher online performance than Q-learning for some MDPs when using ϵ-greedy. This is because Expected Sarsa learns on-policy and will implicitly account for the costs of exploration, which can result in a safer behaviour policy with more reward accrued during deployment.

3.2 Directed Exploration

In this section, we discuss directed exploration strategies that choose actions methodically rather than absentmindedly. Typically, this is done by choosing the greedy action in terms of the estimated action-value plus some *exploration-value*.

An exploration-value functions as a substitution for the value of information, and is used for encouraging or discouraging exploration of actions (Thrun, 1992; Wiering, 1999). They are often based on assumptions or statistics related to learning. The behaviour policy at t for a directed exploration strategy is generally a greedy policy:

$$\pi_t^b(s_t,a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases}, \quad \mathbb{M} = \arg\max_{a \in \mathbb{A}} \big[Q_t(s_t,a) + p \cdot E_t(s_t,a) \big] \ ,$$

where π_t^b denotes how actions are selected for the behaviour at timestep t, $E_t(s_t,a)$ is the exploration-value for state-action pair (s_t,a) at t and p is a parameter that scales the exploration-value in proportion to the action-value estimate.

Another option is to incorporate exploration-values within the value samples that are used for estimating action-values (Meuleau and Bourgine, 1999; Sutton, 1988). The idea is that the exploration-values will propagate to other action-value estimates due to bootstrapping, which encourages exploration globally rather than locally. *Global directed exploration* then proceeds by following a greedy policy with respect to the action-value estimates that implicitly contain the exploration-values.

Difficulties for Directed Exploration

The tuning of p and the choice of exploration-values are not trivial as an inappropriate balance between exploration-values and action-value estimates can cause the agent to optimise something else than the online performance (Thrun, 1992).

If the value of p or choice of exploration-value has little impact on action selection, then the agent may not explore sufficiently. Conversely, if p or the exploration-value obscure the action-value estimate, then the agent will choose actions that accrue the most exploration-values instead of actions that lead to a lot of reward (Wiering, 1999).

A possible solution is to start with a high exploration-value and p, and have them decline to zero over time. This has the nice feature that the agent will automatically exploit actions with respect to the action-value estimate and gradually explore less. However, this leads to the question to how they should decline, and how the sticking problem is properly averted with a good online performance.

Directed Exploration in the Literature

In the literature, various directed exploration strategies have been researched, where the exploration-value is based on recency, frequency, error estimation, and so on (Thrun, 1992; Wiering, 1999). For example, a frequency-based exploration-value is $E_t(s,a) = -n_t(s,a)$, where $n_t(s,a)$ is the number of visits to (s,a) by t. This discourages the exploration of actions that have been tried frequently.

For multi-armed bandit problems, the family of upper confidence bound algorithms is arguably the most famous to guide exploration (Auer et al., 2002). The simplest algorithm, UCB1 (Auer et al., 2002), uses:[20]

$$E_t(s,a) = \sqrt{\frac{2log[n_t(s)]}{n_t(s,a)}} \ ,$$

where $n_t(s) = \sum_{a'} n_t(s,a')$. An action's exploration-value increases indefinitely with the timesteps, but drops severely and increases slower whenever the action is tried.

To the best of our knowledge, these algorithms have only been extensively researched on multi-armed bandit problems, and are in some cases outperformed by ϵ-greedy (Auer et al., 2002; Kuleshov and Precup, 2010). For MDPs, they have only been adopted for online planning which lies out of the scope of this thesis (Kocsis and Szepesvári, 2006).

Other sophisticated approaches in the literature guide exploration by accounting for one's uncertainty over the optimal action-values. This is done using statistical frameworks that involve assumptions or tunable parameters to arrive at a decision. We briefly describe several algorithms for ORL that adopt one of the following frameworks: *Bayesian inference*, *interval estimation* and *probably approximately correct learning*.

3.2.1 Bayesian Reinforcement Learning

In this subsection we discuss the adoption of Bayesian inference for reinforcement learning: *Bayesian reinforcement learning* (BRL). In Bayesian inference, one maintains an explicit probability distribution over quantities that one is uncertain about and updates this distribution in the light of new experiences using Bayes' rule:

$$Pr(\theta \mid X) = \frac{Pr(X \mid \theta) Pr(\theta)}{Pr(X)} \ ,$$

[20] Notice that in multi-armed bandit problems, one does not need to keep track of the state.

where $Pr(\theta \mid X)$ is the posterior probability of an uncertain parameter θ given X are observations, $Pr(X \mid \theta)$ is the likelihood probability, $Pr(\theta)$ is the prior probability and $Pr(X)$ is the probability of the observations.

The idea of using Bayesian inference to solve MDPs was already researched in the 1950's and 1960's (Martin, 1967). Since then, various algorithms have been proposed for both prediction and control, for a variety of uncertain quantities such as policies, MDPs and action-values (Vlassis et al., 2012). Hence, there also exists a distinction in BRL between model-based and model-free algorithms. Figure 3.2 shows an example where one quantifies one's uncertainty for the optimal action-values of two actions by probability distributions over possible values.

The advantage of Bayesian inference is that it meshes well with decision making: one can make excellent trade-offs between exploitation and exploration, and allow for a natural way to encode prior knowledge (Vlassis et al., 2012). Unfortunately, BRL algorithms can generally not guarantee optimal or even good performance for any specific MDP, because the prior involves assumptions about the MDP. If the prior is inappropriate, then convergence to good solutions is slow or unlikely (Vlassis et al., 2012).

In the literature, there are only a few model-free Bayesian reinforcement learning algorithms: *Bayesian Q-learning* by Dearden et al. (1998) that learns in a Q-learning fashion, and on-policy algorithms that use Gaussian processes (Engel et al., 2005). We will not discuss the latter as they are more computationally demanding and focus on the prediction problem rather than guiding exploration.

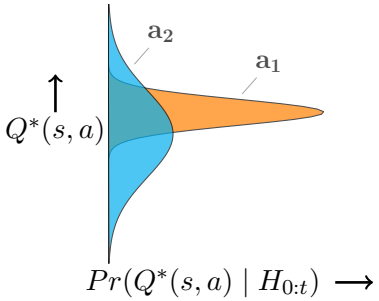


Figure 3.2: An example where one's uncertainty over the actions' optimal action-values is manifested as a probability distribution: $Pr(Q^*(s,a) \mid H_{0:t})$ denotes the probability over possible values for the optimal action-value given the history so far. Here, one is quite certain about $Q^*(s,a_1)$, but is still uncertain about $Q^*(s,a_2)$. This can then be an incentive for choosing the second action as it is potentially better.

3.2.1.1 Bayesian-adaptive Markov decision process

But first, we discuss the *Bayes-adaptive Markov decision process* (BAMDP), which solution optimally balances exploration with exploitation on an unknown MDP (Duff, 2002).[21] A BAMDP is a *meta-problem* formulation, where one assumes a prior over possible MDPs and computes a posterior probability distribution with each new experience.

Although the MDP is unknown, the BAMDP is fully known in advance. This allows one to compute action-values by simulating the consequences of the agent's behaviour in many possible environments, without requiring any interaction with the actual MDP. It

[21] In the literature, a BAMDP has also been described as a POMDP, where states are augmented with explicit values of the MDP's parameters that are not observable (Wang et al., 2005; Vlassis et al., 2012).

is then straightforward to plan an optimal policy with respect to the prior that implicitly trades-off exploration with exploitation perfectly.

Unfortunately, solving the BAMDP exactly is intractable for all but the smallest problems (Duff, 2002). Therefore, many have tried to approximate its solution and several of them can be regarded as state-of-the-art for ORL (Duff, 2002; Wang et al., 2005; Vlassis et al., 2012). The discussion of these algorithms lies out of the scope as they are model-based.

3.2.1.2 Bayesian Q-learning

In this subsection we discuss Bayesian Q-learning (BQL) by Dearden et al. (1998). This is a model-free BRL algorithm that retains and updates for each state-action pair an *estimate probability distribution* over possible values for its optimal action-value.

It is assumed that, for each state-action pair, the distribution of the return under an optimal policy is a Gaussian distribution with an unknown mean and unknown precision, with the optimal action-value being equal to the mean.[22] The uncertainty over a state-action pair's optimal action-value is then quantified by assuming a normal-gamma distribution (NG) as prior over the Gaussian's parameters.

As sample rewards and transitions are observed, the NG's parameters of a state-action pair are updated and one's belief over the optimal action-value changes. Interestingly, BQL resembles Q-learning as it updates the estimates in an online manner, and essentially bootstraps as it uses the NG of the action in the successor state which optimal action-value is believed to be highest. The implication is that it can also suffer from a bootstrap bias: the probability for some values of the parameters can be lower or higher than supposed as the NG of the successor is non-stationary.

For exploration, Dearden et al. (1998) consider a *myopic value of perfect information* and *Q-value sampling*. The former chooses the action that maximises the estimated expected action-value plus a value that corresponds to how much the policy would change if it were to be selected, and is directly computed from its NG posterior. It remains myopic as it does not account for how the posteriors will actually change.

The latter is, in fact, the old heuristic *Thompson sampling* (TS) (Thompson, 1933). Recently, it has been reintroduced in the field of BRL (Strens, 2000; Dearden et al., 1998) and is regarded as state-of-the-art for multi-armed bandit problems (Chapelle and Li, 2011). TS works by choosing the greedy action with respect to a single randomly sampled value from one's posterior. For BQL, a single possible action-value for each action is sampled proportionally to its estimated probability, and then the action with the highest sampled value is chosen. It is only proven for TS that BQL causes the behaviour policy to converge to the optimal policy asymptotically (Dearden, 2000).

3.2.2 Interval Estimation

Another approach to guiding action-selection based on uncertainty is *interval estimation* (IE) (Kaelbling, 1993; Meuleau and Bourgine, 1999; Wiering, 1999). Here, an *interval of probable values* is estimated instead of a *point estimate* like Q-learning does.

For example, we estimate an interval for the optimal action-value $Q^*(s,a)$ bounded by a lower-bound estimate $\underline{Q_t}(s,a)$ and an upper-bound estimate $\overline{Q_t}(s,a)$. All values that lie between these bounds are regarded as *probable* for the optimal action-value.

[22] The precision of a Gaussian distribution is the inverse of its variance (Dearden et al., 1998).

For RL, IE has been adopted by both model-based and model-free algorithms. However, to the best of our knowledge, IE has been mostly used for estimating upper-bounds of optimal action-values as they can be high for two reasons (Kaelbling, 1993):

1. Given the samples, we are still uncertain, and the interval is large and the upper-bound is high.
2. Given the samples, the action must be good, and the whole interval is high.

The idea is thus that one can balance exploration with exploitation by choosing the action with the highest action-value upper-bound estimate. As more samples are obtained, the bounds can become tighter and the uncertainty about the optimal action-value diminishes. Notice that the same risk of ceasing exploration and getting stuck with a suboptimal greedy action also applies in this case (Kaelbling, 1993).

Model-based approaches usually estimate an interval or set of probable MDPs and then compute an optimistic upper-bound accordingly (Li, 2012). Model-free approaches estimate an upper-bound directly from value samples. We will discuss two model-free approaches that adopt ideas from IE to balance exploration with exploitation.

3.2.2.1 Interval Estimation Q-learning

IE for RL was first considered by Kaelbling (1993), where both parametric and non-parametric methods for model-free interval estimation for the optimal action-values were mentioned, but only a parametric method on multi-armed bandit problems was thoroughly investigated.

Parametric IE makes assumptions about the stochastic process from which samples are drawn. For example, the distribution of value samples is assumed to follow a Gaussian distribution, which allows elegant algebraic expressions for estimating intervals.

Non-parametric IE does not make such assumptions, but uses statistical resampling techniques. Although this can be viewed as an advantage over parametric IE, it may require all samples to be memorized in order to properly estimate intervals (Kaelbling, 1993). This may stress the computational resources of the learning agent.

Both parametric and non-parametric IE approaches commonly assume that the samples are independent and identically distributed. This assumption holds for multi-armed bandit problems, but usually not for value samples that are drawn from a bootstrapping distribution. For this reason, Kaelbling (1993) suggests to decay computed intervals over time to accommodate for non-stationary value samples and encourage the exploration of actions that have not been tried recently, which unfortunately introduces new parameters that have to be tuned.

3.2.2.2 Meuleau's Global Interval Estimation Q-learning

In Meuleau and Bourgine (1999), a number of variants that employ ideas from IE for the optimal action-values were researched. All variants assume that the rewards, for each state-action pair, follow a Gaussian distribution with an unknown mean and a variance that is either unknown or assumed to be known.

We will only discuss the best-performing variant in their experiments: *global worst-case interval estimation Q-learning* (IEQL), which assumes the Gaussian distributions' have a known variance equal to or higher than $\sigma_w = \frac{R_{max}-R_{min}}{2(1-\gamma)}$, where R_{min} and R_{max} are respectively the minimal and maximal observable rewards of the MDP and are assumed to be known. Incidentally, this variant also uses the least computation and

memory of all their considered variants. Other variants that they considered, estimates σ_w for each state-action pair individually, but performed worse in the experiments.

IEQL updates upper-bound estimates in a Q-learning fashion. It uses $Q_t(s_t, a_t)$ as an estimate for the upper-bound for the optimal action-value of (s_t, a_t) and is updated using value samples that bootstrap, but with an additional bonus value:

$$X_t = R_t + \gamma \max_a Q_t(S_{t+1}, a) + \delta_t(1 - \gamma) \ , \tag{3.1}$$

where $\delta_t = \sigma_w \frac{z_\theta}{\sqrt{n_t(s_t,a_t)}}$, and z_θ is the z-score for a prior chosen confidence level θ for the upper-bound, and $n_t(s_t, a_t)$ is the number of visits made to (s_t, a_t) up to timestep t. This δ_t functions as a bonus value that declines to zero as $n_t(s_t, a_t)$ goes to infinity.

Clearly, IEQL is a global directed exploration approach, because it incorporates δ_t in the update as an exploration-value and hence shares the same difficulties. However, more troubling is that it assumes a worst-case scenario uniformly for all state-action pairs as δ_t changes only with the number of visits. For instance, if an MDP has an action that rarely leads to an exceptional R_{max} (e.g. winning a lottery) and at other times a low R_{min} (e.g. not winning), then σ_w is extremely high and all actions have to be tried many times before the bonus dissipates.

3.2.3 Probably Approximately Correct Learning

In this subsection we discuss the adoption of *probably approximately correct* (PAC) learning for reinforcement learning. The PAC learning framework was proposed by Valiant (1984) for mathematical analysis on the theory of learning: given an arbitrary set of samples, the goal is to choose a generalizing function that exhibits low error on unseen samples with high probability.

The idea of PAC learning is interesting for reinforcement learning as it can provide theoretical bounds regarding the *sample complexity*: the number of timesteps for which the agent does not act near-optimally[23] (Strehl et al., 2006, 2009; Li, 2012). That is, we can derive how often an agent has to interact in order to guarantee a *certain level of performance with a certain probability.*

Commonly, this is achieved by using theoretical results such as the *Hoeffding's inequality* that provides an upper-bound on the probability that the sum of random samples deviates from the true expected value regardless of the underlying distribution of the samples (Li, 2012). In RL, this inequality is used to derive an upper-bound on the probability that the action-value estimate deviates from the optimal action-value.

We discuss two model-free online value-based algorithms from the literature that are based on PAC learning: *Delayed Q-learning* by Strehl et al. (2006) and *Model-free Action Elimination* (MFAE) by Even-Dar et al. (2003).

3.2.3.1 Delayed Q-learning

Delayed Q-learning closely resembles Q-learning, except that it delays the update of an action-value estimate of a state-action pair until it has obtained m value samples for it (Strehl et al., 2006). This has an averaging effect that partly mitigates the variance of the estimate update, although it still biases estimation as it bootstraps.

[23] Near-optimally is formally defined as a condition that holds when the difference between the estimated Q-function and the optimal Q-function on an infinity norm is small: $||Q_t - Q^*||_\infty < \epsilon$.

Delayed Q-learning follows the philosophy from *optimism-in-the-face-of-uncertainty*: assume that the action-value is maximal until sufficient evidence suggests otherwise (Li, 2012). Given that the action-values are optimistically initialized, based on an assumed R_{max}, Delayed Q-learning assures that $Q_t(s_t, a_t) \geq Q^*(s_t, a_t)$ holds at each timestep with high probability (Strehl et al., 2006). Similar to IE, one can then balance exploration with exploitation by simply acting greedy with respect to the action-value estimates.

Delayed Q-learning requires one to specify the number m of value samples for the desired level of accuracy of the estimation; with a higher number for m resulting in more accurate action-value estimates (Strehl et al., 2006). However, this can jeopardize the online performance as it is usually not necessary to know the action-values very accurately in order to behave properly. I.e., spending a lot of time to learn that a suboptimal action is indeed suboptimal is arguably not efficient. Another particularity is that it is only guaranteed to perform near-optimally with a probability, which implies that there is a chance that it may suffer from the sticking problem.

3.2.3.2 Model-Free Action Elimination

Unlike Delayed Q-learning, MFAE does not delay updates. It updates a lower-bound estimate $\underline{Q_t}(s, a)$ and an upper-bound estimate $\overline{Q_t}(s, a)$ for the optimal action-value simultaneously in a TD manner, which are respectively initialized at a pessimistic value and an optimistic value based on an assumed R_{max} (Even-Dar et al., 2003).

To guarantee PAC-bounds, the value samples include an exploration-value that is defined in terms of the MDP's size, desired level of accuracy, number of visits to the state-action pair and another tunable parameter. Hence, MFAE can also be regarded as a global directed exploration approach:

$$\begin{aligned} \underline{X_t} &= R_{t+1} + \gamma \max_{a' \in \mathbb{A}} \underline{Q_t}(S_{t+1}, a') - B(S_t, A_t) \\ \overline{X_t} &= R_{t+1} + \gamma \max_{a' \in \mathbb{A}} \overline{Q_t}(S_{t+1}, a') + B(S_t, A_t) \ , \end{aligned} \tag{3.2}$$

where $B(S_t, A_t)$ is the exploration-value, and $\underline{X_t}$ and $\overline{X_t}$ are the value samples for respectively $\underline{Q_t}(S_t, A_t)$ and $\overline{Q_t}(S_t, A_t)$. The updates gradually become identical to Q-learning as the exploration-value declines to zero in the limit.

MFAE excludes actions from consideration which upper-bound estimate is lower than all other actions' lower-bound estimates, but is agnostic regarding how actions are further selected. Regardless, one can prove that MFAE learns the optimal action-values probably approximately correct (Even-Dar et al., 2003). Similar to delayed Q-learning, parameter selection may jeopardize the online performance, and optimal behaviour is not guaranteed as the optimal actions have a chance of being excluded.

Chapter 4

Multiple Q-learning

In this thesis we are interested in ORL where the agent is computationally constrained. This does not only call for easy-to-compute and easy-to-memorize autonomous learning algorithms, but also requires the agent to robustly adapt its behaviour to the unknown environment, even when it receives misleading observations. That is, the agent should not immediately disregard an action indefinitely when it led to an uncommon poor outcome as it might be the optimal action. Also, the agent should not behave too conservatively and overexplore suboptimal actions.

In the previous chapter, we discussed several algorithms that can be used to manage the agent's behaviour. Despite that they satisfy the imposed computational demands, being online and model-free, they face various difficulties when addressing ORL.

For instance, ϵ-greedy and Boltzmann action selection can work well in practice, but their undirected approach to exploration is commonly regarded as a motive for researching and employing directed exploration with the aspiration of better online performance. However, existing online model-free directed exploration approaches involve assumptions that usually do not hold in MDPs, have parameters that are not easily tuned for ORL, or do not properly account for the biased and non-stationary nature of learning when they incorporate bootstrapping. As a result, they can prematurely disregard optimal actions or behave too conservatively.

Namely, Bayesian Q-learning does not explicitly account for the effect that bootstrapping has on the posterior probabilities and requires an appropriately tuned prior. Likewise, interval estimation employ ingenious methods to estimate upper-bounds, but are not easily adopted for non-stationary and biased value samples.

As for Delayed Q-learning, MFAE and IEQL; they direct exploration by effectively overestimating the optimal action-values. This is achieved by either delaying updates or biasing the value samples as according to assumptions regarding the MDP and parameters that are not altered during deployment. Essentially, they assume a worst case scenario uniformly for all action-values, where exploration conservatively diminishes with the number of updates, but still may suffer from the sticking problem.

Nevertheless, these directed exploration strategies are still interesting as they automatically decrease exploration as the uncertainty over the optimal action-value dissipates. Still, we believe that it is favourable to have an algorithm for guiding exploration that is not critically dependent on tuning parameters and does not make restrictive assumptions. That is, we want the agent to behave robustly with respect to what it actually observes rather than introducing biases to encourage exploration.

An interesting starting point to develop such an algorithm is to consider Q-learning

as it does not explicitly include any exploration bonus to the value samples nor makes any assumptions about the underlying dynamics of the environment. In addition, there is a lot of literature available and it is computationally light. The only issue with using Q-learning for ORL is that it merely provides a single estimate for the optimal Q-function, which can not indicate when it is accurate and thus when acting greedy with respect to it will result in optimal behaviour. Evidently, a single estimate can also not indicate which actions will lead to improved behaviour.

We are interested in whether it is useful to have *multiple estimates* for guiding an agent's action selection. The idea is that each estimate acts as a hypothesis for the optimal Q-function and use the difference among them as a guideline to balance exploration with exploitation: when the estimated action-values for an action are different, then we are uncertain about whether the action is optimal and use it to encourage exploration. Conceptually, this corresponds to Bayesian Q-learning and Interval Estimation Q-learning, except that we approach this non-parametrically: we employ a number of estimates that are updated in a Q-learning manner.

Outline Chapter

In this chapter we describe our own directed exploration approach for ORL through the estimation of optimal action-values: *Multiple Q-learning* (MQ).

The addition of more estimates adds a whole new dimension of possible algorithms as there are numerous ways to update and combine them. Obviously, we can not research all of them due to time and space constraints. In the first section, we will formulate ideas for updating multiple estimates and using them for guiding exploration. In the second section, we integrate these ideas into an algorithm and discuss its computational requirements and convergence properties.

In the next chapter, we perform a number of experiments to investigate the design choices inherent to MQ and compare it to other discussed model-free algorithms.

4.1 The Concept of Multiple Q-learning

In this section we discuss options for updating multiple estimates and using them to guide action selection.

Essentially, our approach is to base action selection on the variety of multiple estimates. This means that the largest factor on the action selection originates from the learning algorithm. In other words, the way we update the multiple estimates should be such that it amounts to a good online performance. Naturally, the estimates should converge to the optimal action-value consistently and as quickly as possible, with the hope that the agent automatically decreases exploration as the estimates become more similar in order to behave optimally. Unfortunately, at the first sight it is not apparent how multiple estimates should be updated to achieve exactly this desired behaviour.

In order to properly discuss the possibilities, we introduce some notation for multiple estimates. We use an index i to refer to the i-th Q-function Q_t^i, value sample X_t^i and learning rate α_t^i. Let $\{Q_t^i\}_{i=1}^N$ denote a finite set consisting of N Q-functions. The behaviour policy of the agent at a timestep, denoted by π_t^b, is then derived from these estimates using an exploration strategy. The update of an estimate and the behaviour

policy for MQ are then characterized by the following equations:

Section 4.1.3 — α_t^i; X_t^i — Section 4.1.2

$$Q_{t+1}^i(S_t, A_t) = Q_t^i(S_t, A_t) + \alpha_t^i(S_t, A_t)\big(X_t^i - Q_t^i(S_t, A_t)\big) \tag{4.1}$$

$$\pi_t^b = \mathit{ExplorationStrategy}(\{Q_t^i\}_{i=1}^N) \ , \tag{4.2}$$

Section 4.1.1 — $\mathit{ExplorationStrategy}$

where π_t^b is the result of an exploration strategy that is in some sense greedy with respect to the set of Q-functions, and the value sample X_t^i is based on a sample reward R_{t+1} and a sample transition S_{t+1}.

We can conjecture about the outcome of updating estimates by rewriting the temporal-difference learning update as a direct equation. Let $Q_T^i(s,a)$ be the i-th final estimate for the action-value of (s,a) given a previous (e.g. initial) estimate $Q_t^i(s,a)$, a number of value samples $X_t^i, X_{t+1}^i, \ldots, X_T^i$ and learning rates $\alpha_t^i, \alpha_{t+1}^i, \ldots, \alpha_T^i$:

$$Q_T^i(s,a) = Q_t^i(s,a)\prod_{k=t}^{T}(1-\alpha_k^i) + \sum_{k=t}^{T}\big(X_k^i \cdot \alpha_k^i \prod_{l=k+1}^{T}(1-\alpha_l^i)\big) \ , \tag{4.3}$$

We can clearly see that $Q_T^i(s,a)$ is the sum of the value samples and the previous estimate scaled by different products of learning rates. I.e., the products of learning rates regulate how much the value sample and previous estimate contribute to the final estimate.

Prior to considering the possibilities, we can identify a number of principles that we believe are paramount for estimation and subsequently online reinforcement learning. First, in order to assure convergence to the optimal Q-function in the limit it is critical that the value samples eventually are unbiased and the learning rates satisfy the Robbin-Monro's conditions, see Equation 2.15. Second, it is evident that we should not update all Q-functions identically as the multiple estimates will behave equivalently to a single estimate. Third, in order to assure rapid convergence we prefer the estimation that is the least biased and has the least variance. Fourth, fewer parameters is better as this means being less dependent on tuning parameters. Fifth, the optimal action-value of the optimal action is unlikely to be underestimated by all estimates. This last one is the most important, but also the most difficult one to assure with respect to the other principles. For instance, we can easily overestimate the optimal action-values of all actions at infinity, but this will unlikely help in guiding exploration.

A first possible option is to have different initial estimates. If we keep the learning rates and value samples identical for all estimates, then the final estimates will only differ as much as the difference between the initial estimates scaled by the product of learning rates,

$$Q_T^i(s,a) - Q_T^j(s,a) = (Q_t^i(s,a) - Q_t^j(s,a))\prod_{k=t}^{T}(1-\alpha_k^i) \ , \tag{4.4}$$

which is unlikely appropriate for ORL as the initial estimates are generally based on our gut-feeling or assumptions rather than the actual environment. This also means that setting the learning rates at a lower value is not adequate for our task.

Another option is to make the value samples different among the multiple estimates,

$$\forall_{i \neq j}, \exists_{t \leq T} : X_t^i \neq X_t^j \ .$$

For example, we can introduce different bonus (or penalty) factors in the value samples similar to directed exploration approaches such as IEQL+ and MFAE. However, this will introduce a parameter that has to be tuned and may introduce a larger bias. An alternative is to introduce a noise factor to the value samples, which increases variance among the estimates, but this will also introduce another parameter left to be tuned.

A third option is to randomly update a number of estimates at each timestep, which corresponds to varying the learning rates among the estimates.

$$\forall_{i \neq j}, \exists_{t \leq T} : \alpha_t^i \neq \alpha_t^j \ .$$

Whenever an estimate is not to be updated at a timestep, the learning rate for that estimate at that timestep is set to zero. This causes the multiple estimates to be based on different subsets of the sample rewards and transitions.

The advantage of this is that the variety of the estimates is not solely dependent on the different learning rates, but also on the stochasticity of the sample rewards and transitions. E.g., if the distributions of the sample reward and transition for a state-action pair is deterministic, then the variety quickly decreases.

We believe that the third option is the best way to introduce variety among the estimates. However, the question is then how many of the estimates should be updated. If we only update one per timestep, then the effective learning rate is very low for all estimates, and it would require many more samples. On the other hand, we also should not update all of the estimates as this leaves them susceptible to an extreme value sample that causes them to severely overestimate or underestimate the optimal action-value.

Our idea is to update the estimates using *online bootstrapping*, which is an online approximation to non-parametric bootstrapping[24] that in turn is used to approximate the sampling distribution of statistics (Efron, 1979; Oza and Russell, 2001).[25] We further discuss the adoption of this idea in Subsection 4.1.3. Here, we will only note that online bootstrapping can be used to obtain confidence intervals, but only does so reliably for independent and identically distributed samples. In our case, where we base estimation on Q-learning, the value samples are commonly dependent and non-stationary as the estimation policy changes over time. The implication is that the estimates can be far off and may underestimate the optimal action-values. However, the other discussed model-free value-based algorithms are confronted with the same complications. We are interested in how well our approach will perform and further develop how we update and use the estimates in the next subsections.

4.1.1 Exploration Strategies

In this subsection we discuss exploration strategies that can be used with multiple estimates. Naturally, one can choose the mean of the estimates and use ϵ-greedy or Boltzmann action selection. However, we will then not use all available information, such as the spread of the estimates, and the agent will essentially behave similar to just ϵ-greedy with a single estimate.

We will consider three exploration strategies that intent to utilize the set of estimates to their full potential, and describe them in the following subsections. The common

[24] Notice that bootstrapping here differs from bootstrapping in TD-learning as one uses samples from the *empirical distribution* rather than samples that are based on *other estimates*. However, both apply the same idea: *bootstrap oneself up* using samples from another known distribution instead of the unknown true distribution.

[25] See Appendix B.

disadvantage of those exploration strategies is that they do not guarantee infinite exploration, and hence convergence to optimality. One can address this by combining them with ϵ-greedy to always have a tiny chance to explore a non-greedy action. We describe this combination in the subsection 4.1.1.3.

4.1.1.1 Empirical Upper-Confidence Bound

In this subsection we describe the idea of guiding action selection by computing an *empirical upper-confidence bound* (EUCB) for each available action using the set of Q-functions learned by MQ. This idea is inspired by interval estimation, see Section 3.2.2.

Following this exploration strategy, the agent chooses at each timestep the action with the highest EUCB: the mean over all its action-value estimates plus a tunable parameter times the standard-deviation over all its action-value estimates. Thus, the behaviour policy derived by EUCB at a timestep t is

$$\pi_t^b(s_t, a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases} \qquad \mu_a = \frac{1}{N}\sum_{i=1}^{N} Q_t^i(s_t, a)$$

$$\mathbb{M} = \arg\max_{a \in \mathbb{A}} (\mu_a + p \cdot \sigma_a) \qquad \sigma_a = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} \big(Q_t^i(s_t, a) - \mu_a\big)^2}\ , \tag{4.5}$$

where N is the number of Q-functions, $\mathbb{M}$ is the set of all actions with the highest computed EUCB, μ_a and σ_a are the mean and standard-deviation of the estimates for a state-action pair, and p is a tunable parameter that scales the standard-deviation.[26]

Ideally, as the estimates for a state-action pair tend to converge to a value, the standard deviation decreases. As a result, exploration automatically decreases and decision making gradually prefers to choose the action with the highest mean.

However, if p is zero, then action-selection is only based on the mean of the action-value estimates. One can expect that this may not lead to sufficient exploration and a value for p higher than zero is desired. It is possible to set p at an extremely high value and have the agent continue to explore actions until the different estimates are alike.

4.1.1.2 Thompson Sampling

In Section 3.2.1.2, we discussed Thompson sampling (TS) as an exploration strategy that proceeds by randomly sampling a value according to a posterior distribution over possible optimal action-values. For Bayesian Q-learning, one has a distribution per state-action pair and randomly samples a value from each available action's distribution, and then chooses the action with the highest sampled value. The same idea can be adopted for MQ, by randomly sampling values from $\{Q_t^i\}_{i=1}^N$. We identify two ways to proceed:

1. *Q-function Thompson sampling* (QTS): we uniform randomly sample a single Q-function from the set and act greedy with respect to it.
2. *Action-value Thompson sampling* (ATS): for each available action, we uniform randomly choose a Q-function from the set and select the action's corresponding action-value estimate. Once we have an estimate for each action, we act greedy with respect to these estimates.

[26] Notice that this corresponds to a corrected sample standard deviation, which can still be biased.

One way to understand the difference between ATS and QTS is that the former constructs an extended set with additional Q-functions from $\{Q_t^i\}_{i=1}^N$ by recombination of the action-value estimates, whereas QTS randomly selects a Q-function from $\{Q_t^i\}_{i=1}^N$. The behaviour policy at t for both exploration strategies are then as follows:

$$\pi_t^b(s,a) = \frac{1}{|\{Q\}|} \sum_{Q \in \{Q\}} \pi^{greedy(Q)}(s,a)$$

$$\pi^{greedy(Q)}(s,a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases} , \quad \mathbb{M} = \arg\max_a Q(s,a)$$

$$\text{where QTS: } \{Q\} = \{Q_t^i\}_{i=1}^N \tag{4.6}$$

$$\text{ATS: } \{Q\} = \{Q \in \mathbb{Q} \mid \forall_{a \in \mathbb{A}}, \exists_{Q_t^i \in \{Q_t^i\}_{i=1}^N} : Q_t^i(s,a) = Q(s,a)\} , \tag{4.7}$$

where $\pi^{greedy(Q)}$ is the greedy policy with respect to Q and $\mathbb{Q}$ is the set of all Q-functions.

Although ATS corresponds to how TS is used for Bayesian Q-learning, QTS seems to be a viable alternative because it can exploit the correlations between the action-value estimates of the same Q-function.

4.1.1.3 Combination with Epsilon-Greedy

The exploration strategies in the previous subsections do not guarantee infinite exploration and therefore may not assure convergence of the behaviour policy to optimality. This is because the estimates will eventually converge at some value and the discussed exploration strategies will stick to choosing the same greedy actions. In order to guarantee convergence to optimality, we require them to explore infinitely often and have them become greedy in the limit (see Equations 2.14 & 2.16). For this purpose, we can combine them with ϵ-greedy as follows:

$$\pi_t^b(s_t,a) = \begin{cases} (1-\epsilon_t) + \frac{\epsilon_t}{|\mathbb{A}|} & \text{if } a = \widehat{a}_t \\ \frac{\epsilon_t}{|\mathbb{A}|} & otherwise \end{cases} , \tag{4.8}$$

where $\widehat{a}_t$ is the action selected by the exploration strategies that one uses for guiding exploration, such as EUCB, ATS or QTS.

Assuming that the used exploration strategy will choose greedy actions, we can informally prove that infinite exploration and greedy in the limit are ensured when ϵ_t follows a harmonic series. I.e., $\epsilon_t = \frac{c}{n^w}$, where $c \in (0,1]$, n is the output of a monotonously increasing positive function of t, e.g. n is the number of times state s has been visited by t, and $w \in (0,1]$. We first show that it satisfies infinite exploration:

$$\sum_{t=0}^{\infty} \pi_t^b(s,a) \geq \sum_{t=0}^{\infty} \frac{\epsilon_t}{|\mathbb{A}|} = \sum_{n=0}^{\infty} \frac{\frac{c}{n^w}}{|\mathbb{A}|} = \frac{c}{|\mathbb{A}|} \sum_{n=0}^{\infty} \frac{1}{n^w} = \frac{c}{|\mathbb{A}|} \cdot H_\infty^w = \infty , \tag{4.9}$$

where H_∞^w notes a generalized harmonic number of order ∞ of w. Using an integral test, one can testify that $H_\infty^w = \infty$ (diverges) given $w \leq 1$ (Cohen and W.J., 1979). The first inequality holds because the action-probability of an action is always as least as low as a non-greedy action (the otherwise case). The other equalities are rewritings.

For greedy in the limit, we only have to show that ϵ_t goes to zero in the limit, which implies that the probability of non-greedy actions goes to zero in the limit:

$$\lim_{t \to \infty} \epsilon_t = \lim_{n \to \infty} \frac{c}{n^w} = 0 , \tag{4.10}$$

because $c > 0$ and $w > 0$. However, we require $c \in (0, 1]$ to assure that the probabilities of actions will properly sum up to one at each timestep and infinite exploration.

The purpose of this combination is that we can set c at a very low value, e.g. 0.00001, to trivially guarantee infinite exploration and greedy in the limit, while the agent largely relies on the suggested exploration strategy for guiding exploration. Notice that convergence of the behaviour policy to the optimal policy still depends on how the estimates are updated. Also, notice that this combination can be used with other exploration strategies, such as the algorithms discussed in Chapter 3, and one can find other ways to set ϵ_t to assure infinite exploration and greedy in the limit.

4.1.2 Value Samples

In this subsection we discuss how we define the value samples for MQ from sample rewards and transitions. Using a temporal-difference learning basis, this involves the estimates of values for actions in successor states and an estimation policy.

Initially, the value samples will be unavoidably biased as the optimal action-values and the optimal policy are not known, see Section 2.2.1. This bias should eventually disappear in order to accurately estimate the optimal action-values. For TDL, this is the case when several technical conditions hold such as the action-value estimates should converge to the optimal action-value, and the estimation policy becomes congruent to the optimal policy. Therefore, we define the i-th value sample for MQ similarly to TDL:

$$X_t^i = R_{t+1} + \gamma \sum_{a'} \overset{\text{Section 4.1.1.2}}{\overset{\downarrow}{\pi_t^i}}(S_{t+1}, a') \overset{\text{Section 4.1.1.1}}{\overset{\downarrow}{Q_t^i}}(S_{t+1}, a') \ , \tag{4.11}$$

where π_t^i denotes the estimation policy for the i-th Q-function. In this way, diversity among the estimates can emerge when their estimation policy or successor action-value estimates differ.

Other options for defining the value samples are also possible. For instance, we can include bonus factors in the value sample like directed exploration methods such as IEQL+ and MFAE. We refrain from including such bonuses as it introduces another factor that has to be tuned, and can bias the estimates as well as exploration.

In the remainder of this subsection we discuss the initialization of the estimates and estimation policy in order to learn about the optimal action-values.

4.1.2.1 Initialization

Here, we discuss the initialization of the action-value estimates for MQ. This is important as it can affect estimation in two ways. First, the initial estimate for an action-value affects its subsequent estimates when all subsequent learning rates are less than one, see Equation 4.4. Second, they are used for bootstrapping in value samples as the true optimal action-values are not known. Fortunately, the initial estimate does not necessarily prevent convergence to the optimal action values as long as we assure that several technical conditions hold for TDL, see Section 2.2.1.5.

In our case, we can introduce an incentive for an exploration strategy to try actions by spreading the initial estimates over an interval. For this, we assume that the minimal and maximal observable rewards R_{min} and R_{max} of the MDP are known and spread the initial estimates for action-values over the interval $[Q_{min}, Q_{max}]$, where $Q_{min} = R_{min}/(1 - \gamma)$

and $Q_{max} = R_{max}/(1-\gamma)$. One way to initialize the action-value estimates for the i-th Q-function is then:

$$\forall_{(s,a)\in\mathbb{S}\times\mathbb{A}} : Q_0^i(s,a) = Q_{min} + (i-1)\cdot\frac{Q_{max}-Q_{min}}{N-1} \quad . \tag{4.12}$$

For example, we assume $[R_{min}, R_{max}] = [1, 10]$, $\gamma = 0.9$ and have $N = 10$ Q-functions. All state-action pairs' estimates of the first ($i = 1$) Q-function are then initialized at 10. The second ($i = 2$) initialized at 20, third ($i = 3$) at 30, and so on until the last ($i = 10$) Q-function's estimates initialized at 100.

There are many other possibilities for initialization, especially when combined with other measures to encourage diversity among the estimates such as different learning rates. For example, initialize all estimates at the same Q_{max}, or each initial action-value estimate is set to a randomly drawn value from the interval $[Q_{min}, Q_{max}]$. We will refrain from researching these options and consider them as possible future work.

4.1.2.2 Estimation Policy

In this subsection we describe options for the estimation policy used by MQ. Its purpose is similar to that for Q-learning: improving the implicit policy towards the optimal policy. In turn, action-value estimates for the optimal action-values are updated in accordance with an implicit policy that is estimated to be better.

A natural choice for the estimation policy is to use a greedy policy for each Q-function individually, which corresponds to updating them exactly like Q-learning. We refer to this estimation policy as *Individual Max*:

$$\textit{Individual Max} \quad \pi_t^i(s,a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases} \qquad \mathbb{M} = \arg\max_{a\in\mathbb{A}} Q_t^i(s,a) \quad . \tag{4.13}$$

Compatibility Estimation Policy with Behaviour Policy

However, an important aspect in choosing the estimation policy is that it is compatible with the behaviour policy. Namely, in TDL one only updates estimates of state-action pairs that are visited. Thus, if the behaviour policy does not visit the state-action pairs of action-value estimates that the estimation policy includes in its value samples, then these estimates can remain inaccurate.

For example, we update an estimate using a value sample that is based on an action-value estimate in a successor state with value 10, but we never try this action to verify that its value is indeed 10, and therefore the resulting estimates can be incorrect.

Table 4.1 demonstrates an example where this happens for MQ when we use EUCB for the behaviour policy and Individual Max as estimation policy. The estimates do not change as the same action is repeatedly selected. This can be addressed in two ways. The first option is to consider a different exploration strategy. For example, combining EUCB with ϵ-greedy or Boltzmann to assure infinite exploration, although this only averts the issue as one has to wait until it randomly opts for a different action. A second option is to consider a different estimation policy instead.

Mean Max

One possible alternative is to use all estimates to determine a common estimation policy for policy improvement. For instance in Bayesian Q-learning, the action with the highest

Actions	Q-function Estimates $Q_t^1(s,a)$	$Q_t^2(s,a)$	$Q_t^3(s,a)$	$Q_t^4(s,a)$	$\mu_{\mathbf{a}}$	$\sigma_{\mathbf{a}}$
a_1	10	10	0	0	5	5.7735
a_2	11.11	5	0	0	4.0275	5.2773
a_3	5	11.11	0	0	4.0275	5.2773
$\max_a Q_t^i(s,a)$	11.11	11.11	0	0		
$X^i = r + \gamma \max_a Q_t^i(s,a)$	10	10	0	0		
Resulting $Q_{t+1}^i(s,a_1)$	10	10	0	0		

Table 4.1: Example where learning gets stuck when using EUCB with Individual Max. Action a_1 has $Q^*(s,a_1) = 0$ as it always leads to a reward of $r = 0$ and a transition to the same state $P(s,a_1,s) = 1$, and the value sample X^i for each Q-function is 10 or 0 with $\gamma = 0.9$. EUCB always selects a_1 as it has the highest mean and variance regardless of how $p > 0$ is tuned, which means that its estimates do not change as the estimates of actions a_2 and a_3 do not change. If any other action has a higher optimal action-value, then the agent suffers from the sticking problem due to an incompatibility between estimation and behaviour policies. Notice that this scenario can occur with various MDP dynamics: a_1 does not always have to result in a reflexive transition, and the estimates may have different values.

mean over all estimates is chosen (Dearden et al., 1998).[27] We think this can also viable for MQ, because we are interested in choosing the action that is expected to have the highest value. We refer to this estimation policy as *Max Mean*:

$$\textit{Max Mean} \quad \pi_t^i(s,a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases} \qquad \mathbb{M} = \arg\max_{a \in \mathbb{A}} \sum_{j=1}^{N} Q_t^j(s,a) \ . \tag{4.14}$$

Individual Max updates the implicit policy individually for each estimate, whereas *Mean Max* effectively updates a single implicit policy that is shared by all estimates. We will include both estimation policies in our experiments as it is immediately apparent which one will work better for ORL. Nevertheless, both cause the estimates to be updated in an off-policy manner, hence the name *Multiple Q-learning*.

Other options for the estimation policy are also possible. For example, we can choose the action with the highest estimate of all actions, or choose the action with the highest mean and variance over the different estimates. Depending on the exploration strategy, it is also possible to learn on-policy. We consider these options as future work.

4.1.3 Learning Rates

As mentioned at the start of this section, the learning rate scheme will largely decide how the estimates change and thereby the behaviour of the agent. In this subsection we discuss the choice of the learning rate scheme for MQ.

An essential principle for the learning rate scheme is that it satisfies the Robbin-Monro's conditions for stochastic approximation, see Equation 2.15. This assures that every number is reachable for the estimate and that it converges in the limit.

[27] Oddly, they do not consider the possibility that multiple actions can have the same mean but different variances, and how tie-breakers are dealt with. We account for this by assigning a positive probability to all actions that share the highest mean.

One may notice that the value samples defined for MQ are similarly as those defined for Q-learning and Expected Sarsa, see Section 2.2.1. It is therefore reasonable to use similar learning rate schemes for MQ, such as the hyperharmonic learning rate scheme. We keep track of the number of updates for all estimates of MQ. The learning rate scheme for an action-value estimate is then defined by

$$\alpha_t^i = \frac{1}{n_t^i(s_t, a_t)^w} \ ,$$

where $n_t^i(s_t, a_t)$ is the number of times the action-value estimate of state-action pair (s_t, a_t) for the i-th Q-function has been updated at timestep t, and $w \in (0.5, 1]$ is the same tunable parameter of the scheme.

However for MQ, the learning rate scheme is also used to randomly distribute samples among the estimates. For this, we intend to use *online bootstrapping* (Oza and Russell, 2001).

Online Bootstrapping

Online bootstrapping is an approximation to the famous bootstrap resampling technique by Efron (1979). The intention of both is to perform statistical inference on the true underlying distribution by performing inference on *value samples*, that are obtained by sampling from the sample data with replacement and may contain duplicates.[28] For example, the value sample $\{1, 1, 3\}$ can be drawn from the sample data $\{1, 2, 3, 4\}$. Mathematically, the number of duplicates of one instance in the value sample follows a binomial distribution (Oza and Russell, 2001):

$$m \sim Binom(N, \frac{1}{N}) \ ,$$

where m is the number of duplicates and N is the size of the sample data.

Normally, one performs bootstrapping when the sample data is final (no new samples will be added) and contains a vast number of samples. Online bootstrapping assumes that samples come in a indefinite lasting continuous stream and have to be processed online. To account for this, the samples can be processed one at a time and generate m duplicates by drawing randomly from a binomial distribution. For an infinite number of samples, the distribution of m tends to a $Poisson(1)$ distribution (Oza and Russell, 2001) (see Appendix B):

$$m \sim Poisson(1), \quad Pr(M = m) = \frac{exp(-1)}{m!} \ . \tag{4.15}$$

We employ online bootstrapping as follows. Whenever we construct a value sample as according to Equation 4.1, we duplicate the value sample according to a $Poisson(1)$ distribution and update the respective action-value estimate using a hyperharmonic learning rate scheme. Abstractly, this corresponds to updating an estimate several times in a single timestep.

Because the duplicated value samples are identical in value, we can equivalently compute a learning rate that aggregates them and update an estimate only once per timestep. This learning rate then weighs off all the duplicates with the previous value

[28] We use value samples here as well as it corresponds to the terminology used in statistics.

samples. First, let us note that the contributions of the value samples are regulated by the learning rates and sum up to one:

$$1 = \prod_{t=0}^{T}(1-\alpha_t^i) + \sum_{t=0}^{T}\Big(\alpha_t^i \prod_{k=t+1}^{T}\big(1-\alpha_k^i\big)\Big) \ .$$

From this equation, we can derive an aggregated learning rate scheme for online bootstrapping using the hyperharmonic learning rate scheme as foundation:

$$\alpha_t^i = 1 - \prod_{k=1}^{m}\Big(1 - \frac{1}{\big(n_t^i(s_t,a_t)+k\big)^w}\Big) \ , \tag{4.16}$$

which we call the *stochastic learning rate scheme*.

As for the parameter w of the stochastic learning rate scheme, we argue that its value should be lower than one for the same reasons as for other temporal-difference learning algorithms, see Section 2.2.1. We further note that MQ is not limited to this specific scheme, and other schemes that do not include a tunable parameter can be used.

4.2 The Multiple Q-learning Algorithm

In this section we discuss the *Multiple Q-learning* algorithm, as a result of the ideas from the previous section. Algorithm 1 shows the pseudo-code for Multiple Q-learning when used by a reinforcement learning agent.

Algorithm 1: Multiple Q-learning

input: MDP $(\mathbb{S}, \mathbb{A}, P, R, I, \gamma)$,
Number of Q-functions N

Initialization
$\{Q_0^i\}_{i=1}^N \leftarrow$ Eq. 4.12 — See Sec. 4.1.2.1
$\forall_{s\in\mathbb{S}}, \forall_{a\in\mathbb{A}}, \forall_{i\in\{1,\dots,N\}} : n_0^i(s,a) \leftarrow 0$ — (number of updates)

Interaction with the MDP
for $t \leftarrow 0, 1, 2, \dots$ **do**
  Observe s_t
  $\pi_t^b \leftarrow$ Exploration Strategy on $\{Q_t^i\}_{i=1}^N$ — See Sec. 4.1.1
  Execute $a_t \leftarrow \pi_t^b(s_t, \cdot)$
  Observe r_{t+1} and s_{t+1}
  Update Estimates
  for $i \leftarrow \{1, \dots, N\}$ **do**
    Policy Improvement — See Sec. 4.1.2.2
    $\mathbb{M} \leftarrow \arg\max_a Q_t^i(s_{t+1}, a)$ or $\arg\max_a \sum_{j=1}^N Q_t^j(s_{t+1}, a)$
    $\forall_{a\in\mathbb{A}} : \pi_t^i(s_{t+1}, a) \leftarrow \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases}$
    Value Sample — See Sec. 4.1.2
    $X_t^i \leftarrow r_{t+1} + \gamma \sum_a \pi_t^i(s_{t+1}, a) Q_t^i(s_{t+1}, a)$
    Learning Rate — See Sec. 4.1.3
    $m \leftarrow Poisson(1)$
    $\alpha_t^i(s_t, a_t) \leftarrow 1 - \prod_{k=1}^m \left(1 - \frac{1}{\left(n_t^i(s_t,a_t)+k\right)^w}\right)$
    $n_{t+1}^i(s_t, a_t) \leftarrow n_t^i(s_t, a_t) + m$
    Update Estimate
    $Q_{t+1}^i(s_t, a_t) \leftarrow Q_t^i(s_t, a_t) + \alpha_t^i(s_t, a_t)\big(X_t^i - Q_t^i(s_t, a_t)\big)$
  end
end

Conceptually, the algorithm merely updates a number of action-value estimates at each timestep. The algorithm still involves four design choices: the number of Q-functions, the learning rate, the estimation policy, and the exploration strategy. In the next chapter, we perform some experiments to investigate their effect on the online performance. In the remainder of this section we discuss the algorithm's computational requirements and its convergence.

4.2.1 Computational and Space Complexity

In this subsection we briefly discuss *computational complexity* and *space complexity* of MQ. In Strehl et al. (2006) they note that the latter is about how much space is required to implement the algorithm, and the former is about the amount of operations needed to execute the learning algorithm at each timestep. In addition, they define a model-free algorithm as an algorithm with space complexity $o(|\mathbb{S} \times \mathbb{A} \times \mathbb{S}|)$.

The computational complexity of MQ is $O(N \cdot |\mathbb{A}|)$. This is because at most N estimates are updated, and for each estimate one has to iterate over all actions' estimates in the successor state for computing the estimation policy that is required for the value sample. This is comparable to other discussed model-free online algorithms in Chapter 3, where one also requires to iterate over actions to find the maximum action-value and updates one or more variables in the light of new experience.

The space complexity of MQ is $O(N \cdot |\mathbb{S} \times \mathbb{A}|)$. This follows from the fact that a Q-function is memorized as a table which has space complexity $O(|\mathbb{S} \times \mathbb{A}|)$, using one action-value per state-action pair, and MQ has N Q-functions. All other discussed model-free online algorithms have space complexity $O(|\mathbb{S} \times \mathbb{A}|)$ as they memorize a constant number of variables per state-action pair. In practice, the amount of space that MQ uses will depend on the number of Q-functions. For a small number of Q-functions, the used amount of memory is comparable to other algorithms. E.g., MQ with three Q-functions keeps track of six variables per state-action pair (the action-value estimate and the number of updates for the learning rate), whereas Delayed Q-learning keeps track of five variables per state-action pair (Strehl et al., 2006).

4.2.2 Convergence to Q^* and π_*

In this subsection we briefly discuss the convergence properties of MQ. This pertains to the convergence of each estimate to the optimal Q-function and the behaviour policy of the agent to the optimal policy. We first address the former and finish with the latter. We will only note a theorem and a conjecture for convergence, whose respective proof and proof sketch are conveniently located in Appendix C.3.

4.2.2.1 Convergence of Q_t^i to Q^*

In order to prove that each Q-function estimate under MQ converges to the optimal Q-function, we can adopt established proof for the convergence of temporal-difference learning methods, such as Q-learning and Expected Sarsa.

Here, we state a theorem for the convergence of MQ when using Individual Max as estimation policy (see Eq.4.13) and a conjecture that MQ converges when using Mean Max as estimation policy (see Eq. 4.14). Although both the theorem and conjecture are comparable in the assumptions they make, their working have subtle different implications and hence their proofs. The structure of the theorem and the conjecture, as well as their proofs, generally follow the same structure as the proof of convergence of (General) Q-learning, see Appendix C.

Theorem 1. (Convergence of Multiple Q-learning with Individual Max, Eq. 4.13)
All Q-function estimates converge to the optimal Q-function under Multiple Q-learning with Individual Max as estimation policy, as defined by (shortened)

$$Q_{t+1}^i(s_t, a_t) = Q_t^i(s_t, a_t) + \alpha_t^i(s_t, a_t)\big(r_{t+1} + \gamma \max_a Q_t^i(s_{t+1}, a) - Q_t^i(s_t, a_t)\big) \ ,$$

whenever the following assumptions hold:

1. $\mathbb{S}$ *and* $\mathbb{A}$ *are finite,*
2. *For each Q-function:* $\forall_{i\in[1,N]} : \alpha_t^i(s_t, a_t) \in [0, 1]$, $\sum_{t=1}^{\infty} \alpha_t^i(s_t, a_t) = \infty$, $\sum_{t=1}^{\infty}[\alpha_t^i(s_t, a_t)]^2 < \infty$ *w.p.1, and* $\forall(s, a) \neq (s_t, a_t) : \alpha_t^i(s, a) = 0$,
3. *The behaviour policy ensures infinite exploration,*
4. *The reward function of the MDP is bounded;* $Var\{r_{t+1} \mid P_t\} < \infty$,

where $P_t = \{Q_0, s_0, a_0, \alpha_0, r_1, s_1, a_1, \ldots, s_t, a_t\}$.

The proof for this theorem, see Appendix C.3, generally follows from the fact that Multiple Q-learning with Individual Max corresponds to a number of independent Q-learning algorithms, for which it is known that it converges to the optimal Q-function under the same assumptions. Notice that the order of action-value updates is not critical for convergence, provided that the behaviour policy ensures infinite exploration, and the learning rates for each Q-function for all state-action pairs satisfy the Robbin-Monro's conditions. This is the case when we use exploration strategies such as ϵ-greedy or a combination thereof, see Subsection 4.1.1.3, and the stochastic learning rate used by MQ in Algorithm 1 is based on the hyperharmonic learning rate for which we know it satisfies those conditions.

Conjecture 1. (Convergence of Multiple Q-learning with Mean Max, Eq. 4.14)
All Q-function estimates converge to the optimal Q-function under Multiple Q-learning with Mean Max as estimation policy, as defined by

$$Q_{t+1}^i(s_t, a_t) = Q_t^i(s_t, a_t) + \alpha_t^i(s_t, a_t)\big(r_{t+1} + \gamma \sum_a \pi_t^i(s_{t+1}, a) Q_t^i(s_{t+1}, a) - Q_t^i(s_t, a_t)\big)$$

$$\pi_t^i(s, a) = \begin{cases} \frac{1}{|\mathbb{M}|} & \iff a \in \mathbb{M} \\ 0 & \iff a \notin \mathbb{M} \end{cases} \qquad \mathbb{M} = \arg\max_{a\in\mathbb{A}} \sum_{j=1}^{N} Q_t^j(s, a) \ ,$$

whenever the following assumptions hold:

1. $\mathbb{S}$ *and* $\mathbb{A}$ *are finite,*
2. *For each Q-function:* $\forall_{i\in[1,N]} : \alpha_t^i(s_t, a_t) \in [0, 1]$, $\sum_{t=1}^{\infty} \alpha_t^i(s_t, a_t) = \infty$, $\sum_{t=1}^{\infty}[\alpha_t^i(s_t, a_t)]^2 < \infty$ *w.p.1, and* $\forall(s, a) \neq (s_t, a_t) : \alpha_t^i(s, a) = 0$,
3. *The behaviour policy ensures infinite exploration,*
4. *The reward function of the MDP is bounded;* $Var\{r_{t+1} \mid P_t\} < \infty$.

A proof sketch for this conjecture can be found in Appendix C.3. The idea is to exploit a similar structure as established proof of convergence for temporal-difference learning. The subtle difference that it has compared to Theorem 1 is that the estimation policy depends on multiple estimates. Provided that the estimation policy converges, or the Q-function estimates converge, the estimation policy will be equivalent to a greedy policy as Theorem 1 and Q-learning. We think that established proof for convergence of single estimates with temporal-difference learning can be extended to multiple estimates.

4.2.2.2 Convergence of π_t^b to π^*

In order to show that the behaviour policy π_t^b of the agent converges to π^* in the limit, we generally have to ensure the two following conditions (Singh et al., 2000):

1. The learning algorithm of the agent ensures that it learns the optimal Q-function or optimal policy,
2. The behaviour policy will act greedy in the limit with respect to the learned optimal Q-function or policy, see Eq. 2.16.

The former condition implies that we learn the optimal policy for the environment, whereas the latter ensures that the agent will actually follow the learned policy. However, those conditions generally rely on various other necessary conditions to hold such as infinite exploration, the estimation policy being greedy (in the limit), and other technical conditions regarding the agent's learning algorithm and the environment such as ergodicity of the MDP.

For finite ergodic stationary MDPs with an infinite horizon, we know that temporal-difference learning algorithms such as Q-learning and Expected Sarsa can converge to the optimal Q-function. In conjunction with a behaviour policy that ensures infinite exploration and greedy in the limit, the behaviour policy will also converge to the optimal policy (Singh et al., 2000).

Assuming that the Q-function estimates under Multiple Q-learning also converge to the optimal Q-function, as described in the previous subsection, we only have to ensure that the exploration strategy acts greedy in the limit with respect to the Q-function estimates and satisfies infinite exploration. For the exploration strategies proposed in Section 4.1.1, EUCB, ATS and QTS do not satisfy infinite exploration on their own, but are greedy in the limit. Fortunately, we have shown in Subsection 4.1.1.3 that it is trivial to ensure infinite exploration and greedy in the limit by combining an exploration strategy with ϵ-greedy, where ϵ follows some hyperharmonic series.

Therefore, we informally conclude that the behaviour policy under Multiple Q-learning is likely to converge to the optimal policy in the limit given that the same necessary conditions as for Q-learning and Expected Sarsa are satisfied.

Chapter 5

Experiments

In this chapter we describe our experiments for Multiple Q-learning's design choices and its comparison to other model-free online alternatives. The purpose of these experiments is to answer our research questions, see Section 1.1. In general, we wish to investigate the effectiveness of Multiple Q-learning for online reinforcement learning.

Outline Chapter

In the first section, we describe our general methodology for conducting experiments, which research questions they intend to answer and how they are generally structured.

In the remaining sections we separately describe three experiments. For each experiment, we mention their intention and elaborate their experimental setup. For reasons of readability, we also report and briefly discuss the results of each experiment in their respective sections.

In the next chapter, we discuss the Multiple Q-learning algorithm as a whole, draw conclusions from the results and give suggestions for future research.

5.1 Methodology

In this section, we describe our methodology to answer our main research question:

> "**Is it beneficial to use multiple action-value estimates for model-free online reinforcement learning?**"

In the previous chapter we introduced and motivated the Multiple Q-learning algorithm, which uses multiple action-value estimates to guide exploration, that we intent to use to answer our main research question. Here, we define 'beneficial' as having a better online performance in general than a baseline algorithm for model-free online reinforcement learning in MDPs. The underlying idea is that one can readily employ our proposed algorithm in an unknown MDP and perform better than the baseline algorithm.

Our approach is to conduct empirical experiments where the online performance is measured by computer simulation. This will spawn results that are used to test hypotheses, which are formulated in Section 5.4, and to answer our main research question.

Unfortunately, measuring the online performance as the expected return over an infinite horizon, as in Equation 3, is not easily done. Namely, the behaviour policy is typically non-stationary and non-Markovian, for which it is not practical to compute the expected return using conventional methods such as Dynamic programming. The

alternative of simulating the interaction between agent and environment would require an infinite amount of time as the infinite horizon encompasses an infinite number of actions, transitions and rewards. In the literature, this is usually addressed by measuring the online performance over a *finite horizon* instead. We believe that this approximation is also a suitable option in our case.

As baseline algorithm, we choose Q-learning with ϵ-greedy as Q-learning is arguably the most basic and well-known online model-free value-based algorithm and ϵ-greedy has instilled the initiation of various research endeavours to develop better exploration strategies (Dearden et al., 1998; Kaelbling, 1993; Meuleau and Bourgine, 1999). In addition, it has often been included in related research literature. For completeness, we will also include a number of discussed model-free alternatives in our experiments.

In order to attest the superiority of our algorithm over the baseline algorithm it has to robustly accrue a higher online performance on *all* MDPs. Obviously, it is an improbable quest to check the performance on all possible MDPs and we will therefore consider just four different MDPs, which are described in Subsection 5.1.1. Similarly, we do not have time to perfectly optimise the parameters for each learning algorithm on each MDP. Moreover, we regard fine-tuning the parameters on each MDP to be in conflict with our ambition to discover a robust learning algorithm that is not restrictively dependent on its parameters.

Before we conduct our intended experiment to attest the supposed superiority of Multiple Q-learning, there remain a number of design choices and parameters inherent to Multiple Q-learning that we have to investigate: how many estimates should it use, how should the estimates be updated, which exploration strategy works best, and so on. To this end, we perform two exploratory experiments using only Multiple Q-learning. In Section 5.2, we describe and perform our first experiment on the combination of estimation policy and exploration strategy for Multiple Q-learning. In Section 5.3, we investigate the effect of the number of estimates used by Multiple Q-learning on the online performance. The results of those two experiments are then used to choose appropriate settings for Multiple Q-learning in the final and primary experiment, where we compare Multiple Q-learning to other model-free online alternatives. Figure 5.1 further clarifies the structure of this chapter.

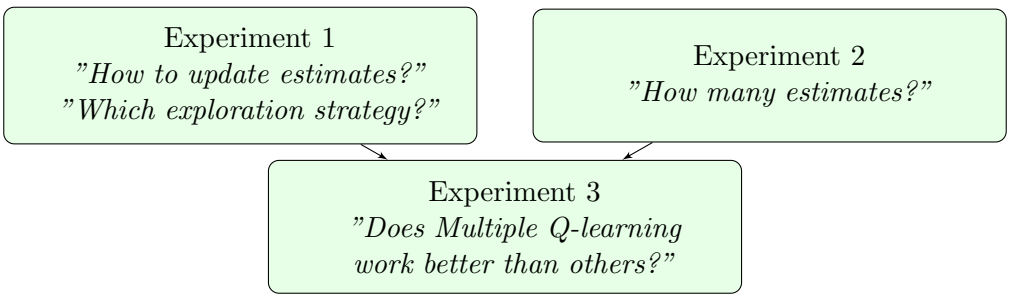


Figure 5.1: Structure of the experiments. Experiment 1 and 2 answer questions regarding the parameters and design choices inherent to Multiple Q-learning. The results of those experiments are then used in experiment 3, which is intended for answering the main research question.

5.1.1 Test Problems

In this subsection we elaborate each MDP that we include in our experiments.

All MDPs in this thesis are finite, stationary and have multiple states and actions. They are chosen such that there is a variety of different state-space and action-space sizes, as well as stochasticity in transitions and rewards.

5.1.1.1 Chain

The chain MDP is a commonly used toy problem for both model-based and model-free reinforcement learning algorithms (Dearden et al., 1998; Poupart et al., 2006).

The purpose of this non-episodic finite stochastic MDP is to test whether an agent quickly learns the optimal policy of always choosing action a rather than being taken in by action b which immediately returns a high reward. Figure 5.2 shows an illustration of the MDP, and Table 5.1 holds the MDP's parameters.

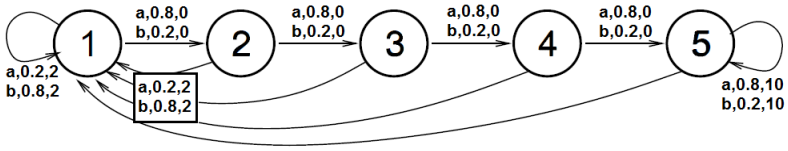


Figure 5.2: The chain MDP (Dearden et al., 1998). Each transition (edge) between two states (nodes) is associated with an action, transition probability and a reward. Action a leads 80% of the time to the next state, ultimately to state 5, and 20% of the time it leads to state 1. Action b works reversely, causing the agent to go to the next state 20% of the time and 80% of the time back to state 1. When the agent transitions from state 5 back to state 5 it gains a reward of 10. When it transitions back to state 1 from any state it gains a reward of 2 and in all other cases it gains a reward of 0.

Table 5.1: The chain MDP

Variable	**Value**
$\mathbb{S}$	$\{1, 2, 3, 4, 5\}$
$\mathbb{A}$	$\{a, b\}$
P	Stochastic, see Fig. 5.2
R	Deterministic, see Fig. 5.2
$[R_{min}, R_{max}]$	$[0, 10]$
S_0	Always state 1
γ	0.9
π_*	Always choose a

5.1.1.2 Cliff Walking

The cliff walking MDP is an episodic finite deterministic MDP where the agent has to reach a goal state (Sutton and Barto, 1998). The shortest route to this goal is to closely walk past a cliff. Although the MDP is completely deterministic, the agent may fall into the cliff due to exploration. In order to perform optimally, the agent will have to be certain about the shortest path to the goal.

When the agent moves to the goal state or enters the cliff area, the current episode ends and the agent's position is changed to the start state from which a new episode can begin. After each action, the agent observes a reward of -1, except for when it falls into the ravine where it observes a reward of -100. The available actions are North, South, West and East. If the agent would move outside the MDP, it remains in its previous state and observes a reward of -1. Figure 5.3 illustrates the cliff walking MDP, and Table 5.2 holds the MDP's parameters, optimal policy and optimal performance.

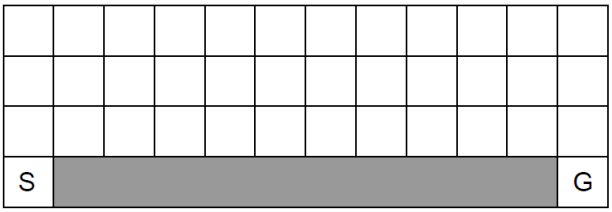


Figure 5.3: The cliff walking MDP (van Seijen et al., 2009). The agent has to move from the start [S] to the goal [G], while avoiding stepping into the cliff (grey area).

Table 5.2: Cliff walking MDP

Variable	**Value**
$\mathbb{S}$	See Fig. 5.3
$\mathbb{A}$	$\{North, South, West, East\}$
P	Deterministic
R	Deterministic: -100 or -1
$[R_{min}, R_{max}]$	$[-100, -1]$
S_0	Always state *start*
γ	0.9
π_*	In order: $N, 11 \times E, S$

5.1.1.3 Six Arms

The six arms MDP has been used in experiments by model-based PACRL and interval estimation (Strehl and Littman, 2004). We believe that it is also a suitable problem for evaluating the online performance of model-free algorithms.

This non-episodic finite MDP has a stochastic transition function and a deterministic reward function, and is presumably called six arms as there are six actions in each state. For each action in the first state, there is a chance that it transitions to another state (mentioned explicitly in Figure 5.4) or remains at the same state and gains zero reward.

The challenge is that the best action often appears as a poor action, as it rarely transitions to a successor state where a lot of reward can be obtained. The agent may thus believe that other actions are better as decent rewards appear more often. Figure 5.4 illustrates the six arms MDP, and Table 5.3 holds its parameters.

5.1.1.4 Random MDP

The random MDP differs from the previous MDPs as it is constructed randomly right before evaluation starts. Random MDPs have been employed before in the literature

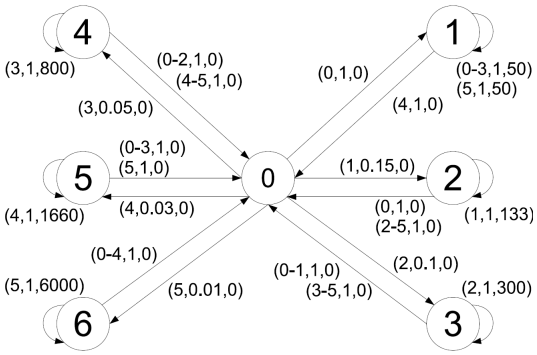


Figure 5.4: The six arms MDP (Strehl and Littman, 2004). Nodes represent states and edges represent transitions with each a tuple (action, transition probability, reward).

Table 5.3: Six arms MDP

Variable	**Value**
$\mathbb{S}$	See Fig. 5.4
$\mathbb{A}$	$\{0, 1, 2, 3, 4, 5\}$
P	Stochastic: see Fig. 5.4
R	Deterministic: see Fig. 5.4
	Unless stated in the figure, the reward is zero.
$[R_{min}, R_{max}]$	$[0, 6000]$
S_0	Always state 0
γ	0.9
π_*	Always choose action 5

(Even-Dar et al., 2003). They are interesting when one wishes to check the robustness of a learning algorithm in a general setting rather than on a specific MDP, which serves one of our ambitions.

Prior to any interaction, we construct a random MDP as follows. We specify the number of states and the number of actions as desired. For each action in each state, the state can transition to up to B different states, where B is a tunable parameter that stands for branching and can cause the state to transition to itself. The probability of a transition is randomly drawn from a continuous uniform distribution on interval $[0, 1]$ and subsequently normalized for each state-action pair. Depending on B, the MDP's states can be sparsely or densely connected.

For each transition, a reward is randomly drawn from a Gaussian distribution with parameters μ and σ. These two parameters are fixed when the MDP is constructed. The mean μ is drawn from a continuous uniform distribution from 0 to μ_{max} which is tunable, and the standard deviation σ is randomly drawn from a continuous uniform distribution from 0 to σ_{max} which is also tunable. In expectation, the reward observed at any timestep is $\frac{\mu_{max}}{2}$.

Thus, the MDP can be small or large, and deterministic or stochastic, depending on how the parameters are set. Table 5.4 describes the random MDP's parameters concisely, which is further specified in the respective experiments that include random MDPs. The random MDPs are fixed during our experiments to ensure a fair comparison.

Table 5.4: Random MDP's parameters

Variable	Value
$\mathbb{S}$	Tunable
$\mathbb{A}$	Tunable
P	For each action in each state, up to B different transitions, Transition probability drawn from $Uniform(0,1)$ and normalized
R	$R_{t+1} \sim Gaussian(\mu, \sigma)$ for each (s_t, a_t, s_{t+1}), $\mu \sim Uniform(0, \mu_{max})$, $\sigma \sim Uniform(0, \sigma_{max})$
$[R_{min}, R_{max}]$	Depends on the randomly sampled parameters
S_0	Always state 0
γ	0.9
π_*	Differs per MDP

5.2 Experiment 1: Estimation Policies and Exploration Strategies

In this experiment we compare a number of combinations of estimation policies and exploration strategies, see respectively Subsections 4.1.2.2 and 4.1.1. We refer to each such combination as a *candidate*. With this experiment, we hope to get insight into two things: which of the discussed exploration strategies works well, and which estimation policy to use for updating estimates.

As discussed in Section 4.1.2.2, the estimates can get stuck if the estimation policy includes actions that the exploration strategy never tries, see Table 4.1. As a result, the learning algorithm does not update its estimates and the exploration strategy may continue to select actions inappropriately.

The structure of this experiment is as follows. First, we describe the dependent variable, independent variable and fixed parameters in the experimental setup. We then report the results in Subsection 5.2.2, where we will also briefly discuss our findings that we will use in subsequent experiments.

5.2.1 Experimental Setup

We compare the candidates on the Cliff walking and Six arms MDPs, see Subsections 5.1.1.2 and 5.1.1.3, measuring their online performance based on their behaviour policy that results from their estimation policy and exploration strategy.

In total, there are six candidates, see Table 5.5. Each one consists of a unique combination of estimation policy and exploration strategy, and uses the structure of Algorithm 1. Each candidate uses the stochastic learning rate that is based on online bootstrapping, see Section 4.1.3. All other parameters and number of Q-functions are fixed, see Table 5.6. We will refrain from including other exploration strategies such as ϵ-greedy and Boltzmann action selection on the mean of the Q-functions, because they do not fully exploit the set of Q-functions learned by multiple Q-learning. We will also not combine the exploration strategies with ϵ-greedy as in Section 4.1.1.3, because the purpose of this combination is merely to assure infinite exploration by setting the ϵ parameter at a very low value and hence can not be relied on when we only consider only a finite number of timesteps.

Table 5.5: Candidates for experiment 1

Candidate	**Estimation Policy**	**Exploration Strategy**
MQIA	Individual Max, see Eq. 4.13	ATS, see Eq. 4.7
MQIQ	Individual Max	QTS, see Eq. 4.6
MQIU	Individual Max	UCB $p = 2$, see Eq. 4.5
MQMA	Max Mean, see Eq. 4.14	ATS
MQMQ	Max Mean	QTS
MQMU	Max Mean	UCB $p = 2$

Table 5.6: Fixed parameters of candidates for experiment 1

Variable	**Value**
Learning algorithm structure	Alg. 1
Learning rate scheme	Eq. 4.16 with $w = 0.8$
Number of Q-functions N	20
Initialization	See Subsection 4.1.2.1, with $[Q_{min}, Q_{max}]$ as Cliff walking: $[-100, 0]$, Six arms: $[0, 60000]$
Discount rate γ	0.9

Because Cliff walking is an episodic MDP where the agent has to reach the goal state as soon as possible, we measure the online performance as the average of the expected return obtained per episode over 500 episodes with $\gamma = 1$. An episode ends when the agent observes that the MDP has terminated or when 1000 timesteps have elapsed. The latter is to prevent the experiment from running indefinitely if a candidate gets stuck. Notice that the agent retains its learned knowledge from previous episodes, and that the agent may reach the goal earlier than the episode before.

For the non-episodic Six arms, we measure the online performance as the average reward obtained over 250.000 timesteps, which corresponds to the undiscounted return divided over the number of timesteps. We average in both cases to make results more intuitive, as otherwise they will get into the millions.

For each candidate, we simulate and measure 25 times on the Cliff walking problem, and 20 times on the Six arms problem. This corresponds to the number of repeats, over which we report the mean and the standard deviation in a table. Table 5.7 holds the details of the experiment.

Table 5.7: Details of experiment 1

Variable	**Value**
MDPs	Cliff walking & Six arms, see Subsections 5.1.1.2 & 5.1.1.3
Duration	Cliff walking: 500 episodes, each with max. 1000 timesteps, Six arms: 250.000 timesteps
Independent variable	see Table 5.5
Dependent variable	Cliff walking: average return $\gamma = 1$ over 500 episodes, Six arms: average reward over 250.000 timesteps
Number of Measurements	Cliff walking: 25, Six arms: 20
Reporting results	Tables: mean and standard dev. of the dependent variable

5.2.2 Results

We succinctly show the results for experiment 1 on both MDPs in Table 5.8.

Table 5.8: Results for experiment 1: combinations of estimation policies and exploration strategies on Cliff walking and Six arms. For both, a higher mean indicates better performance and a lower standard deviation corresponds to a more consistent performance.

	Mean ± Std.	
Candidate	Cliff walking	Six arms
MQIA	-95.20 ± 13.58	5481 ± 130.1
MQIQ	-925.4 ± 153.4	5157 ± 1149
MQIU	-411.8 ± 240.1	182.8 ± 76.30
MQMA	-126.9 ± 19.88	5552 ± 65.95
MQMQ	-986.2 ± 5.051	4427 ± 1808
MQMU	-92.74 ± 11.14	5665 ± 247.8

We observe that the candidate MQIU, consisting of the individual max estimation policy and UCB as exploration strategy, performs poorly on both MDPs. Although it does not perform the worst on Cliff walking, it does perform abysmally on the Six arms MDP. On closer inspection, we found that the agent got stuck each time: it repeatedly chose the same actions that did not lead to the estimates to be updated, which is a reason to not consider MQIU any longer. However, the candidate MQMU that uses the max mean estimation policy instead, performed the best on both MDPs consistently.

As for exploration strategy QTS, it appears that it does not work well for either estimation policy. The worst possible average undiscounted return for Cliff walking in our experimental setup is about -1099, which both MQIQ and MQMQ are closer to than to the optimal performance of -13.[29] This can either mean that the agent explores too much or not at all. We found evidence for the latter when we inspected the estimates: Q-functions often shared the same maximal actions, which results in QTS often choosing the same actions. Although the mean of the performance for MQIQ and MQMQ is decent on the Six arms MDP, they also have an alarming standard deviation. Such a combination of a high mean and high standard deviation is possible if QTS did well in some cases, whereas it performed dreadfully in the other cases. This suggests that QTS is not appropriate for balancing exploration with exploitation for MQ.

Candidates that used exploration strategy ATS on the other hand performed really well, only shortly behind MQMU. This hints that the additional Q-functions constructed from the set of learned Q-functions causes the agent to explore more. On the Cliff walking MDP, there is a clear difference between the two estimation policies. However, on the Six arms MDP, no such difference is apparent.

To summarize, three of the evaluated candidates did not work because there was insufficient exploration: both QTS candidates and the combination of the individual max estimation policy and exploration strategy UCB. All other candidates performed decently, with candidate MQMU having the highest performance in both MDPs. In the subsequent experiments, we will include both MQIA and MQMU as they employ different estimation policies and exploration strategies, as they were the strongest contenders and we think it is interesting to see how they compare to other algorithms from the literature.

[29] An undiscounted return of -1099 occurs when the agent walks around for 999 timesteps and walks into the cliff area on the last timestep.

Table 5.9: Details of Experiment 2

Variable	Value
MDPs	Chain & Six arms, see Subsections 5.1.1.1 & 5.1.1.3
Independent variable	Number of Q-functions: $N = \{2, 5, 10, 20, 50\}$
Dependent variable	Average reward accrued
Number of Measurements	Chain walking: 20, Six arms: 20
Reporting results	Table: mean and standard deviation of the average reward over Chain: 25000 timesteps, Six arms: 200000 timesteps for each candidate and number of Q-functions.

5.3 Experiment 2: Number of Estimates

In this experiment we investigate how the performance of multiple Q-learning changes when using different numbers of Q-functions. The purpose of this experiment is to get an idea of how many estimates should be used for experiment 3.

We have two hypotheses with regard to the performance as the number of Q-functions increases. First, the performance might degrade as the number of Q-functions decreases, because exploration ceases when the variability among estimates dissipates. Evidently, more estimates allows for more variability among estimates. Ideally, the computational resources available to the agent allow it to keep track of thousands of estimates, but in reality the agent has only limited memory space and computational power available. Second, the behaviour policy is expected to change quicker with fewer estimates, and as a result it might learn to prefer better actions quicker.

Similar to the previous experiment, we first describe the experimental setup, and then report as well as discuss results in the subsequent subsection. We use the outcome of this experiment for the third experiment in Section 5.4.

5.3.1 Experimental Setup

For this experiment, the independent variable that we vary is the number of Q-functions used by multiple Q-learning: $2, 5, 10, 20$ and 50. Although it is possible to include higher numbers, it defeats the purpose of using a model-free algorithm that considers a scenario where computational resources are scarce. We measure the performance as the average reward accrued, and evaluate on the Chain and Six arms MDPs.

We use the candidates MQIA and MQMU from the first experiment, except that the number of Q-functions is varied for both. In all cases, we use Algorithm 1 and the online bootstrapping learning rate scheme with $w = 0.8$, see Equation 4.16. The estimates were initialized on interval $[0, 100]$ for the Chain MDP and $[0, 60000]$ for the Six arms MDP using Equation 4.12.

The performance of an agent is measured as the average reward accrued over 25000 timesteps on the Chain MDP and 200000 timesteps on the Six Arms MDP. We measure for each combination of a candidate and a number of Q-functions 20 times the performance on each MDP, resulting in a total of 400 measurements. The purpose of averaging is to make the measured performances more intuitive. For candidate MQMU, we set UCB's parameter p at 2 for all numbers of Q-functions. We report the mean and standard deviation of the performance for each candidate and number of Q-functions on both MDPs in a table. Table 5.9 contains the details of this experimental setup.

5.3.2 Results

We show the results of experiment 2 for both candidates with varying numbers of Q-functions on the Chain and Six arms MDPs in Table 5.10.

Table 5.10: Experiment 2, results for different numbers of Q-functions on Chain and Six arms. The mean and standard deviation are over the dependent variable, the average reward. For both, a higher mean indicates better performance and a lower standard deviation corresponds to a more consistent performance.

	Average Reward: Mean ± Std.			
Number of	***MQIA***		***MQMU***	
Q-functions	Chain	Six arms	Chain	Six arms
$N = 2$	2.206 ± 0.138	5038 ± 234.4	3.170 ± 0.097	5503 ± 253.7
$N = 5$	2.343 ± 0.123	5346 ± 142.5	3.355 ± 0.087	5450 ± 296.5
$N = 10$	2.389 ± 0.095	5339 ± 205.2	3.403 ± 0.071	5646 ± 209.4
$N = 20$	2.448 ± 0.113	5283 ± 147.7	3.411 ± 0.051	5667 ± 168.9
$N = 50$	2.446 ± 0.087	5224 ± 186.1	3.420 ± 0.045	5641 ± 242.6

For MQIA, the results suggests that it is better to use more than $N = 2$ estimates as there is more than one standard deviation difference between the means for $N = 2$ and $N \geq 5$ on both MDPs. The performance for other values of N scored quite similarly to each other. One can observe a substantial difference between the performances of MQIA and MQMU on the Chain MDP, but MQIA still performs better than a random policy.

For MQMU on the Chain MDP, the mean average reward between $N = 2$ and $N = 5$ differs more than one standard deviation, and between $N = 2$ and $N \geq 10$ even more than two standard deviations. However on the Six arms problem, no such difference between different numbers of Q-functions is observed.

Despite the clear difference in performance between $N = 2$ and more estimates, the performance of the behaviour policy under $N = 2$ hints that it learns about the optimal policy. For instance, at the Chain problem, if it were to follow any other policy than the optimal or second-best deterministic policy (always choosing action b), then the average reward would be below 2. For the six arms, this is even more evident as the performance level is far higher than the second best deterministic policy (always choosing action 4) that obtains an average reward of 1600. However, we note that one can not tell from the results in Table 5.10 whether any of the candidates converged or will converge to the optimal policy. It is still possible that the candidates converge to the second best policy or not at all.

As according to the results of this experiment, we believe that the best number of Q-functions for both MQIA and MQMU is $N = 10$. First, the performance of $N = 10$ is on each problem and both candidates among the highest with the lowest standard deviation. Second, 10 Q-functions require less space and computation than 20 or 50. Thus, we argue that $N = 10$ gives a good trade-off between computation and performance.

5.4 Experiment 3: Comparison to Other Online Model-free Alternatives

In the final experiment we want to compare the online performance of the two different Multiple Q-learning to other existing online model-free algorithms for MDPs. Evidently, we want to know whether it is better to use MQ than other algorithms, and in special whether it performs better than Q-learning with ϵ-greedy which we regard as the baseline algorithm for model-free online reinforcement learning.

Our expectation is that MQ will accrue more reward during online deployment than other algorithms, because it directs exploration less conservatively than other directed exploration, such as IEQL and Delayed Q-learning, accounts for non-stationarity of the value samples, unlike Bayesian Q-learning, and actually guides exploration methodically rather than ϵ-greedy and Boltzmann action selection with Q-learning or Expected Sarsa. We formulate two hypotheses:

1. Multiple Q-learning as according to Algorithm 1 with Individual Max as estimation policy and Action-value Thompson Sampling as exploration strategy has a higher online performance than Q-learning with ϵ-greedy:

$$H_0 : \mu_{MQIA} = \mu_{QL}, \;\; H_a : \mu_{MQIA} > \mu_{QL} \;\; ,$$

2. Multiple Q-learning as according to Algorithm 1 with Mean Max as estimation policy and Empirical Upper-confidence bound as exploration strategy has a higher online performance than Q-learning with ϵ-greedy:

$$H_0 : \mu_{MQMU} = \mu_{QL}, \;\; H_a : \mu_{MQMU} > \mu_{QL} \;\; ,$$

where μ_{cand} denotes the mean of the measured performances of a candidate, H_0 denotes the null-hypothesis and H_a denotes the alternative hypothesis. We test for statistical significance using a one-sided Welch's test with a 2.5% significance level (Welch, 1947).

In the following subsection we describe the experimental setup, which includes the environments, candidate algorithms and how we measure online performance. In the subsection that follows we present the results and provide insights to interpreting them. In the next chapter we discuss the results.

5.4.1 Experimental Setup

In this subsection we describe the experimental setup for testing our hypotheses. The general idea is to vary the candidate algorithm, which we describe below, and measure the online performance on a number of environments, which we deem necessary for providing substantial evidence to support, or reject, our hypotheses. We will first discuss the candidates that we include in our experiments and then discuss the environments and the measurement of the online performance.

Candidates

We restrict ourselves to comparing MQ to other model-free online algorithms that we discussed in Chapter 3. However, we will not try every discussed alternative as there are simply too many. We include one algorithm of each approach to uncertainty estimation: Bayesian Q-learning with moment updating (Sec. 3.2.1.2), Global Interval Estimation

Q-learning (IEQL) (Sec. 3.2.2.2) and Delayed Q-learning (Sec. 3.2.3.1). For Bayesian Q-learning, we only consider Thompson sampling for exploration strategy as it assures infinite exploration. For the other two, the greedy action is always selected.

In addition, we will include Q-learning and Expected Sarsa, for which we only consider ϵ-greedy to represent undirected exploration. For each environment, we test the following settings for parameters: $\epsilon_t = \{0.01, 0.03, 0.05, 0.1, 0.15\}$ and a harmonic series based on the number of times the state has been visited: $\epsilon_t = \frac{1}{n(s_t)}$.

For MQ, we test MQIA and MQMU as in the previous experiments with $N = 10$ Q-functions. In the simulation, we include the combination of the exploration strategy used by MQIA and MQMU with ϵ-greedy following a harmonic series with $c = 0.0001$ as described in Section 4.1.1.3. Although we do not expect it to play a major role in the actual behaviour of the agent and hence the online performance during the simulations, it is included for completeness as it assures infinite exploration.[30]

In Table 5.11 we summatively list the candidates that we include in our simulations. In general, we eschew from tuning the parameters' values of the different candidates as this in conflict with our intention to find a robustly performing algortihm rather than fine-tuning the parameter values to perfection and the performance becomes solely based of the tuning. However, as we do not know which values will work best in the general case, we feel obliged to at least try a number of parameters' values. To this end, we have run an extensive number of preliminary simulations.

Table 5.11: Candidates for Experiment 3

Candidate	**Learning Algorithm**	**Exploration Strategy**
MQIA	Multiple Q-learning with Individual Max	ATS (with ϵ-greedy)
MQMU	Multiple Q-learning with Mean Max	EUCB (with ϵ-greedy)
QL	Q-learning (Watkins, 1989)	ϵ-Greedy
ExpS	Expected Sarsa (van Seijen et al., 2009)	ϵ-Greedy
BayQ	Bayesian Q-learning (Dearden et al., 1998)	Thompson Sampling
IEQL	IEQL+ (Meuleau and Bourgine, 1999)	Greedy
DelQ	Delayed Q-learning (Strehl et al., 2006)	Greedy

For all candidates, with the exception of Delayed Q-learning and Bayesian Q-learning, the learning rate scheme used is the hyperharmonic learning rate for which we vary its parameter $w = \{0.5001, 0.6, 0.8, 1.0\}$, see Equation 2.12. For MQMU, we set the parameter value of EUCB at $p = 2$ in all cases for simplicity. For Delayed Q-learning, the number of value samples required before an estimate is updated is varied at $m = \{1, 4, 10, 20, 50\}$. Furthermore, all learning algorithms assume that $\gamma = 0.9$ for all MDPs. The initialization of the candidates depends on the environment and is described below. In general, we found that optimistic initialization worked best for all candidates.

Environments

In order to properly test our hypotheses, we wish to see how the candidates perform in various environments. For this reason, we include four different environments: the Chain MDP, the Cliff MDP, the SixArms MDP, and the RandomMDP (see Section 5.1.1.4).

[30] I.e., the expected number of times a random action is tried by timestep T using ϵ-greedy that follows a harmonic series is upperbounded by $T \cdot c$. E.g., if we set $\epsilon_t = \frac{0.0001}{n(s_t)^w}$ and $T = 20000$, then only at most 2 chosen actions are expected to have been non-greedy actions.

The first three have been used in the previous experiments, and the last one is a Random MDP which dynamics' parameters are randomly initialized when simulation starts.

The reason for considering the Random MDP is that the other problems are specific toy-problems that are relatively small.[31]. Therefore, we include the Random MDP with a large action space and high variance in the rewards. Our adoption of the Random MDP for this experiment is to use 50 randomly generated MDPs with hyperparameters set at $|\mathbb{S}| = 20$, $|\mathbb{A}| = 40$, deterministically transitions to $B = 1$ state, $\mu_{max} = 5$ and $\sigma_{max} = 10$. Each candidate will test on the same randomly generated MDPs exactly once, resulting in 50 measurements of the online performance on the Random MDP.

The initialization of the candidates on each MDP is as follows. We assume that the minimum and maximum observable rewards of each MDP are known: R_{min}, R_{max}. The initialization of Q-learning, Expected Sarsa and Delayed Q-learning are then set at an optimistic value following $Q_0 = \frac{R_{max}}{1-\gamma}$ where $\gamma = 0.9$, thus: Chain $Q_0 = 100$, Cliff $Q_0 = 0$, Six Arms $Q_0 = 60000$ and RandomMDP $Q_0 = \frac{\mu_{max}}{1-\gamma} = 50$. For MQIA and MQMU, we used Equation 4.12 with: Chain $Q_0 = [0, 100]$, Cliff $Q_0 = [-100, 0]$, Six Arms $Q_0 = [0, 60000]$ and RandomMDP $Q_0 = [0, 50]$. For IEQL, the initialization is $Q_0 = 2 \cdot \sigma_{max} = 2 \cdot \frac{R_{max}-R_{min}}{2(1-\gamma)}$ as according to Meuleau and Bourgine (1999), thus: Chain $Q_0 = 100$, Cliff $Q_0 = 500$, Six Arms $Q_0 = 60000$ and RandomMDP $Q_0 = 500$. For Bayesian Q-learning, we performed a number of preliminary experiments as no initialization instructions were given by Dearden (2000), and arrived at the following initializization of the Normal-Gamma distribution's hyperparameters: $\mu_0 = 0.5 \cdot \frac{R_{max}-R_{min}}{1-\gamma}$, $\lambda = 0.005$, $\alpha = 10/9$ and $\beta = 0.005$. This results in the following initial means: $\mu_0 = 50$, Cliff $\mu_0 = -50$, Six Arms $\mu_0 = 30000$ and RandomMDP $\mu_0 = 25$.

Measurements and Statistical Significance

We measure the online performance of the candidates as follows. On the Chain MDP, we measure the average reward over 5000 timesteps in a single episode a total of 50 times. On the Cliff MDP, we measure 25 times the average return with $\gamma = 1$ over 500 episodes, where an episode ends when the terminal state is reached or the maximum number of steps reaches 1000. On the Six Arms MDP, we measure 20 times the average reward over 250000 timesteps. On the Random MDP, we measure on each randomly generated MDP, 50 in total, the average reward over 20000 timesteps. The durations for each MDP were chosen based on the size of the MDP and preliminary experiments where we checked whether the performance of the candidate differed from the very start where it did not yet learn. For Six Arms MDP, most of the actions will rarely result into a transition to another state and therefore we chose a very long duration. Here, the reason for averaging is to show intuitive values for the online performance. We report the results obtained by the candidates on each MDP separately with the mean and the standard deviation of the measured values. Notice that the mean in this case is an approximation of the online performance with infinite horizon as in Equation 3. The details of the measurements are displayed in Table 5.12.

We test statistical significance of one candidate performing better than another candidate using an one-sided Welch's test (Welch, 1947).[32] We chose this test as we consider the possibility that the measured performance of each candidate have unequal variances,

[31] In addition, we were unable to find any specific MDP with a large finite action space in the literature.

[32] The Welch's test assumes the measurements are normally distributed. Based on the central limit theorem, we assume the measured performances of the candidates are approximately normally distributed as they are sums of independent random variables.

and test only one-sided as we are interested in whether one candidate has a higher performance than another. Hence, the alternative hypothesis is that the mean of the measured performances by candidate A is greater than the mean of the measured performances by candidate B. The corresponding null-hypothesis states that the mean of the two candidates' performances are equal. We claim statistical significance when the null hypothesis can be rejected on the 2.5% significance level.

For simplicity, statistical significance is separately tested per MDP. As it is possible that various candidates perform statistically significantly better than a single poorly performing candidate, we mention the statistical significance between candidates that are of interest to us: the best performing candidate compared to the second best, and whether MQIA and MQMU compared to Q-learning.

Table 5.12: Details of experiment 3

Variable	**Value**
MDPs	Chain, Cliff walking, Six arms, and Random MDP
Independent variable	see Table 5.11
Dependent variable	Chain: average reward over 5.000 timesteps Cliff walking: average return $\gamma = 1$ over 500 episodes, Six arms: average reward over 250.000 timesteps Random MDP: average reward over 20.000 timesteps
Number of Measurements	Chain: 50, Cliff: 25, Six arms: 20, RandomMDP: 50
Reporting results	Tables: mean and standard dev. of the dependent variable and Graphs for clarification

5.4.2 Results

In this subsection we report the results for Experiment 3. For each MDP, we separately show a table and a graph of the candidates' performances, and provide a brief discussion and interpretation of the results. The order of discussion of the MDPs follows the order of their introduction in the experimental setup: the Chain MDP, the Cliff MDP, the Six Arms MDP, and the Random MDP. In the next chapter, we provide a more profound discussion of the two Multiple Q-learning candidates and compare their performances to other candidates.

Chain MDP

We present the performance, the average reward over 250.000 timesteps, of the candidates on the Six Arms MDP of Experiment 3 in Table 5.15. In addition, we show in Figure 5.7 the average reward as function of timesteps 0 to t.

We can see that Q-learning performed best on the Chain MDP, and is closely followed by MQMU. The performance of Expected Sarsa and IEQL are not too far off, as the mean of their average reward is about one standard deviation less. Although MQIA performed significantly worse than several other candidates, it still learns about the optimal policy albeit slower. This is because the optimal policy is deterministic with an expected average reward of 3.66, the second-best deterministic policy of always choosing action b has an expected average reward of about 2, and any other policy that is not a mix of the optimal policy and the second-best policy accrues less than 2 reward on

Table 5.13: Experiment 3, best results for the candidates on the Chain MDP. The mean and standard deviation are over the dependent variable, the average reward accrued over 5000 timesteps. A higher mean indicates better performance and a lower standard deviation corresponds to a more consistent performance.

Chain MDP		**Average Reward**
Candidate	Best Parameters	**Mean ± Std.**
MQIA	$Q_0 = 100, w = 0.5001$	2.89 ± 0.149
MQMU	$Q_0 = 100, w = 0.5001$	3.42 ± 0.133
QL	$Q_0 = 100, \epsilon_t = \frac{1}{n(s_t)}, w = 0.5001$	3.46 ± 0.134
ExpS	$Q_0 = 100, \epsilon_t = 0.01, w = 0.5001$	3.38 ± 0.148
BayQ	$\mu_0 = 50, \lambda = 1.0, \alpha = 1.1, \beta = 1.0$	1.69 ± 0.140
IEQL	$Q_0 = 100, w = 0.5001$	3.29 ± 0.143
DelQ	$Q_0 = 100, m = 50$	1.32 ± 0.032

average per timestep. This is also suggested by the graph as the performance of MQIA steadily increases and does not appear to stagnate.

Thus, we can also see that the behaviour policy under both Bayesian Q-learning and Delayed Q-learning does not learn the optimal policy nor the second-best policy, and hence still extensively switches between actions a and b that accrues a low average reward.[33] Another peculiarity is that Bayesian Q-learning appears to perform worse over time according to the graph, which may indicate that it learns inappropriately.

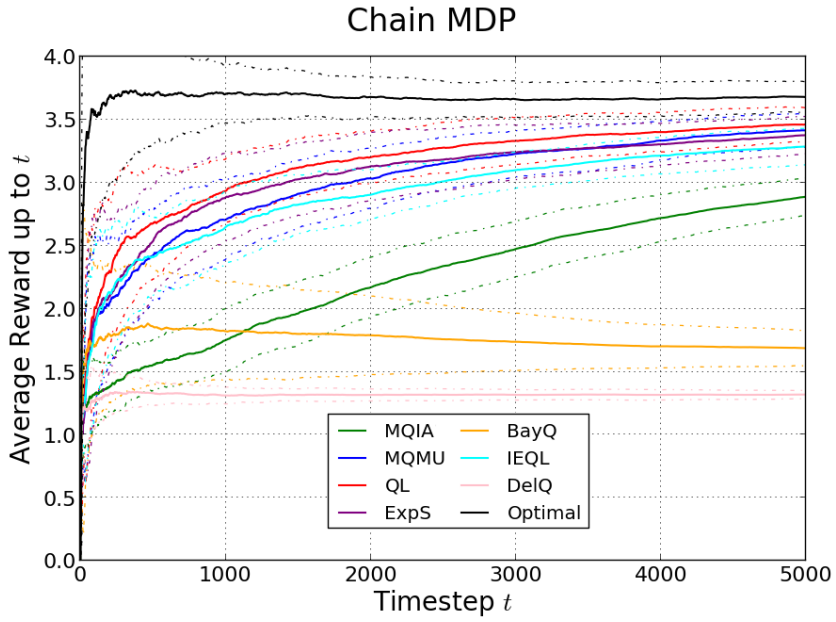


Figure 5.5: Experiment 3, performance of candidates on Chain MDP: average reward accrued as function of timestep 0 to t. The solid lines and dashed lines respectively depict the mean and standard deviation over 50 measurements. The performance of the optimal policy is included as well.

[33]The performance of Bayesian Q-learning reported here is comparable to the one reported in Dearden et al. (1998), where it was not noticed that the performance of Bayesian Q-learning on the Chain MDP is less than the second-best policy.

Furthermore, we can see in Figure 5.5 that the average reward accrued by MQMU, Expected Sarsa and Q-learning are quite close to each other. One might hypothesize that they will converge to the optimal policy as well, but this is unlikely the case of Expected Sarsa as it uses a fixed ϵ-greedy for exploration.

Based on the empirical results, we cannot reject any of the two null hypotheses $H_0 : \mu_{MQIA} = \mu_{QL}$ and $H_0 : \mu_{MQMU} = \mu_{QL}$ as the observed difference in means have a probability much greater than 0.025: respectively $p = 1$ and $p = 0.958$. Incidentally, the Q-learning candidate performs statistically significantly better than all other candidates except MQMU.

Cliff MDP

We present the performance, the average return over 500 episodes, of the candidates on the Cliff MDP of Experiment 3 in Table 5.14. In addition, we show in Figure 5.6 the return accrued per episode of the candidates. This graph shows a perspective on the performance other than the table and allows for a different interpretation of the results.

Table 5.14: Experiment 3, best results for the candidates on the Cliff MDP. The mean and standard deviation are over the dependent variable, the average return accrued over 500 episodes. A higher mean indicates better performance and a lower standard deviation corresponds to a more consistent performance.

Cliff MDP		**Average Return**
Candidate	Best Parameters	**Mean ± Std.**
MQIA	$Q_0 = [-100, 0], w = 0.5001$	-47.1 ± 7.11
MQMU	$Q_0 = [-100, 0], w = 0.5001$	-29.2 ± 1.28
QL	$Q_0 = 0, \epsilon_t = 0.01, w = 0.5001$	-22.6 ± 0.727
ExpS	$Q_0 = 0, \epsilon_t = 0.01, w = 0.5001$	-22.8 ± 0.362
BayQ	$\mu_0 = 495, \lambda = 0.005, \alpha = 1.01, \beta = 0.005$	-108 ± 0.748
IEQL	$Q_0 = 500, \sigma_{max} = 250, z_\theta = 2, w = 0.5001$	-885 ± 3.26
DelQ	$Q_0 = 0, m = 50$	-184 ± 9.76

Unsurprisingly, all candidates performed best when they used the highest learning rate ($w = 0.5001$) as the Cliff MDP is completely deterministic.[34] The best performing candidate was again Q-learning with Expected Sarsa falling just short. In terms of the average return per episode, MQIA and MQMU clearly performed substantially worse. We suspect that MQMU and MQIA initially performed worse than QL and ExpS as multiple estimates have to be updated before the behaviour will change, which requires more samples.

However, we can see in Figure 5.6 that the behaviour policy of MQMU converges to the optimal policy after about 380 episodes. This is an astonishing result as MQMU autonomously improves performance and ceases exploration with minimal parameter tuning. Although the estimate under Q-learning also converged to the optimal Q-function, the respective candidate QL performed less consistent as it regularly fell off the cliff and incurred a large penalty due to using a fixed ϵ-greedy exploration strategy. Based on these two findings, we believe that MQMU would have been the best performing candidate if the performance was measured over more episodes as its initial poor performance is offset by its optimal performance in future episodes.

[34] This is also the reason why MQMU and MQIA performed much better than in Experiment 1.

Nevertheless, based on the empirical results presented in Table 5.14, we cannot reject the two null-hypotheses $H_0 : \mu_{MQIA} = \mu_{QL}$ and $H_0 : \mu_{MQMU} = \mu_{QL}$ as in both cases the observed difference in means have probability $p = 1$. In both cases, Q-learning outperformed MQIA and MQMU in both the statistical and absolute sense.

Delayed Q-learning, IEQL and Bayesian Q-learning were tried with many different initializations and parameters, but all performed on the Cliff MDP unsuccessfully. We suspect that IEQL is too conservative as the δ_t bonuses remain bloated for a very long time, which is inappropriate for a simple deterministic environment. For Bayesian Q-learning and Delayed Q-learning, it is possible that other untried parameters lead to better performance. Although, one would expect that Delayed Q-learning works best with $m = 1$ as it updates the action-value estimates the quickest, which sounds the most suitable for a deterministic environment.

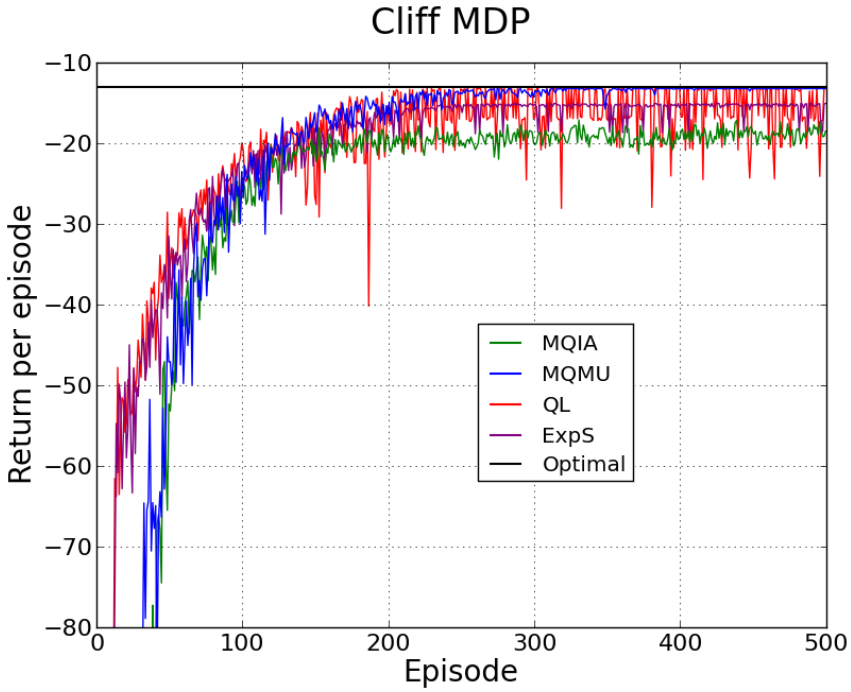


Figure 5.6: Experiment 3, performance of the candidates on Cliff MDP: return as function of the episode. The solid lines depict the mean over 25 measurements. Candidates that performed worse than −80 were omitted. The optimal policy is included, which demonstrates the convergence of some candidates to optimality.

Furthermore, we found that Expected Sarsa learns a safe consistent behaviour policy on the Cliff MDP, which has also been observed in Sutton and Barto (1998) and van Seijen et al. (2009). The reason that MQIA did not yet converge is because it explores a lot more than MQMU initially, which causes the learning rates to deprecate and hence the behaviour policy stabilizes at a suboptimal policy at the later episodes.

Six Arms MDP

We present the performance, the average reward over 250.000 timesteps, of candidates on the Six Arms MDP of Experiment 3 in Table 5.15. In addition, we show the average reward as function of timesteps 0 to t in Figure 5.7.

Table 5.15: Experiment 3, best results for the candidates on the Six Arms MDP. The mean and standard deviation are over the dependent variable, the average reward accrued over 250000 timesteps. A higher mean indicates better performance and a lower standard deviation corresponds to a more consistent performance.

Six Arms MDP		**Average Reward**
Candidate	Best Parameters	**Mean ± Std.**
MQIA	$Q_0 = [0, 60000], w = 0.8$	5477 ± 121
MQMU	$Q_0 = [0, 60000], w = 0.8$	5686 ± 160
QL	$Q_0 = 60000, w = 0.8, \epsilon_t = \frac{1}{n(s_t)}$	3651 ± 1774
ExpS	$Q_0 = 60000, w = 1, \epsilon_t = \frac{1}{n(s_t)}$	5360 ± 212
BayQ	$\mu_0 = 30000, \lambda = 1.0, \alpha = 1.01, \beta = 0.005$	920 ± 2126
IEQL	$Q_0 = 60000, w = 0.8, \sigma_{max} = 30000, z_\theta = 2$	5770 ± 109
DelQ	$Q_0 = 60000, m = 50$	0.214 ± 0.013

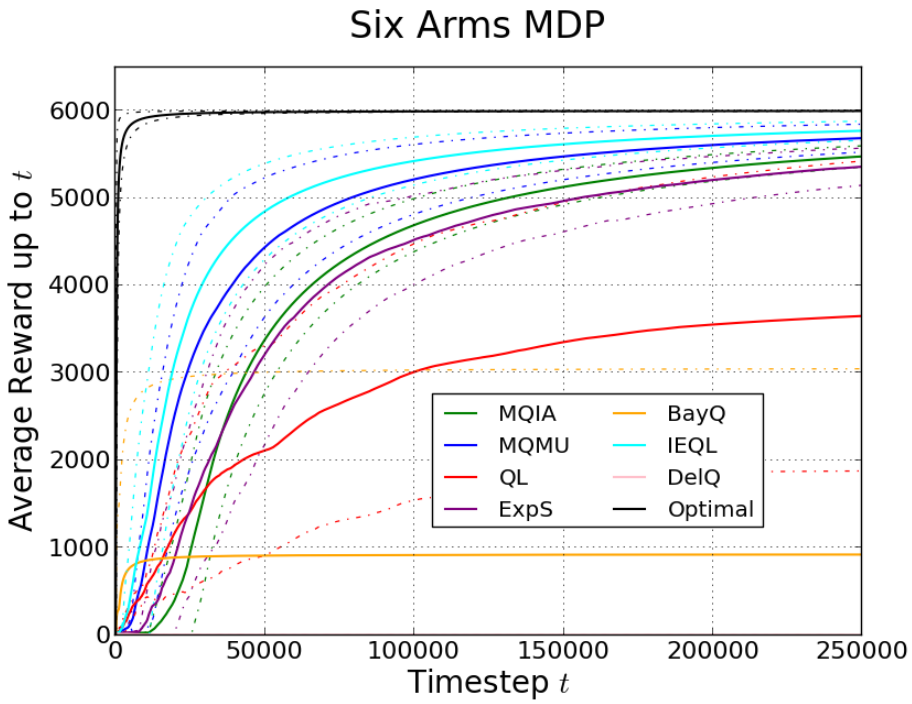


Figure 5.7: Experiment 3, performance of candidates on Six Arms MDP: average reward accrued as function of timestep 0 to t. The solid lines and dashed lines respectively depict the mean and standard deviation over 20 measurements. The performance of the optimal policy is included as well.

We can tell from the table that IEQL performed best with MQMU not far behind. MQIA performed a whole standard deviation less than MQMU, and Expected Sarsa performed another standard deviation worse than MQMU. Still, the graph suggests that IEQL, MQMU, MQIA and Expected Sarsa converge to the optimal policy as the average reward accumulates towards the optimal 6000.

The performance under Q-learning, with high mean and high variance, indicates that the learned behaviour policy sometimes follows an optimal policy and sometimes a

suboptimal policy, because the optimal policy accrues an expected reward of 6000 per timestep and policies that prefer any another arm accrue an expected reward of 1600 or lower per timestep. We observe the same for Bayesian Q-learning, except that it performs worse on average due to converging to a poorer suboptimal policy.

For Delayed Q-learning we have checked the performance for numerous parameters on the Six Arms MDP, but none of them obtained a better result than the one reported in the table. This may suggest that Delayed Q-learning is not appropriate for this MDP, or is simply too parameter dependent in order to obtain adequate performance. In either case, we can regard Delayed Q-learning as not suited for that want a robust 'off-the-shelf' algorithm for online reinforcement learning.

Using the one-sided right-tailed Welch's test with a 2.5% significance level, we can reject both null-hypotheses $H_0 : \mu_{MQIA} = \mu_{QL}$ and $H_0 : \mu_{MQMU} = \mu_{QL}$ based on the empirical results as the observed differences respectively have a probability of $1.3 \cdot 10^{-4}$ and $4.0 \cdot 10^{-5}$ which are far below 0.025. Therefore, we accept in both cases the alternative hypothesis. Using the same test to attest the statistical significance of the best-performing IEQL has a higher mean than the second-best MQMU, we cannot reject the null-hypothesis $H_0 : \mu_{MQMU} = \mu_{IEQL}$ on a 2.5% significance level ($p = 0.034$). In other words, the difference in performance between IEQL and MQMU is statistically not significant.

Random MDP

We present the performance, the average reward over 20.000 timesteps, of the candidates over 50 randomly generated Random MDPs of Experiment 3 in Table 5.16. In Figure 5.8, we show the average reward as a function from timestep 0 to t.

In Figure 5.8, we can clearly see that both Q-learning and Expected Sarsa perform much better than the other candidates. Although MQMU and MQIA accrue better performance than the other three candidates, the standard deviation of MQMU is twice as great as those of Q-learning and Expected Sarsa. As evident from both Table 5.16 and Figure 5.8, both Q-learning and Expected Sarsa outperform MQIA and MQMU (as well other candidates) in the statistical and absolute sense. Hence, we cannot reject the null-hypotheses $H_0 : \mu_{MQIA} = \mu_{QL}$ and $H_0 : \mu_{MQMU} = \mu_{QL}$ based on a right-tailed Welch's test on a 2.5% significance level (both $p = 1$).

Oddly, both Delayed Q-learning and Bayesian Q-learning did not seem to improve the behaviour of the agent as the average reward remains rather constant for all 20.000 timesteps. In fact, Bayesian Q-learning even accrued slightly less reward per timestep over time. We tried various parameters settings, but were unable to find better results. It is possible that the tried parameters were simply inappropriate for the general case, and specific parameters have to be tuned per MDP.

Table 5.16: Experiment 3, best results for the candidates on the Random MDP. The mean and standard deviation are over the dependent variable, the average reward accrued over 5000 timesteps. A higher mean indicates better performance and a lower standard deviation corresponds to a more consistent performance.

Random MDP		**Average Reward**
Candidate	Best Parameters	**Mean ± Std.**
MQIA	$Q_0 = [0, 50], w = 0.5001$	3.73 ± 0.170
MQMU	$Q_0 = [0, 50], w = 1$	3.95 ± 0.322
QL	$Q_0 = 100, \epsilon_t = \frac{1}{n(s_t)}, w = 0.8$	4.49 ± 0.151
ExpS	$Q_0 = 50, \epsilon_t = 0.01, w = 0.8$	4.45 ± 0.129
BayQ	$\mu_0 = 25, \lambda = 0.005, \alpha = 1.01, \beta = 0.005$	2.42 ± 0.170
IEQL	$Q_0 = 50, \sigma_{max} = 25, z_\theta = 2, w = 1$	3.38 ± 0.110
DelQ	$Q_0 = 50, m = 50$	2.52 ± 0.080

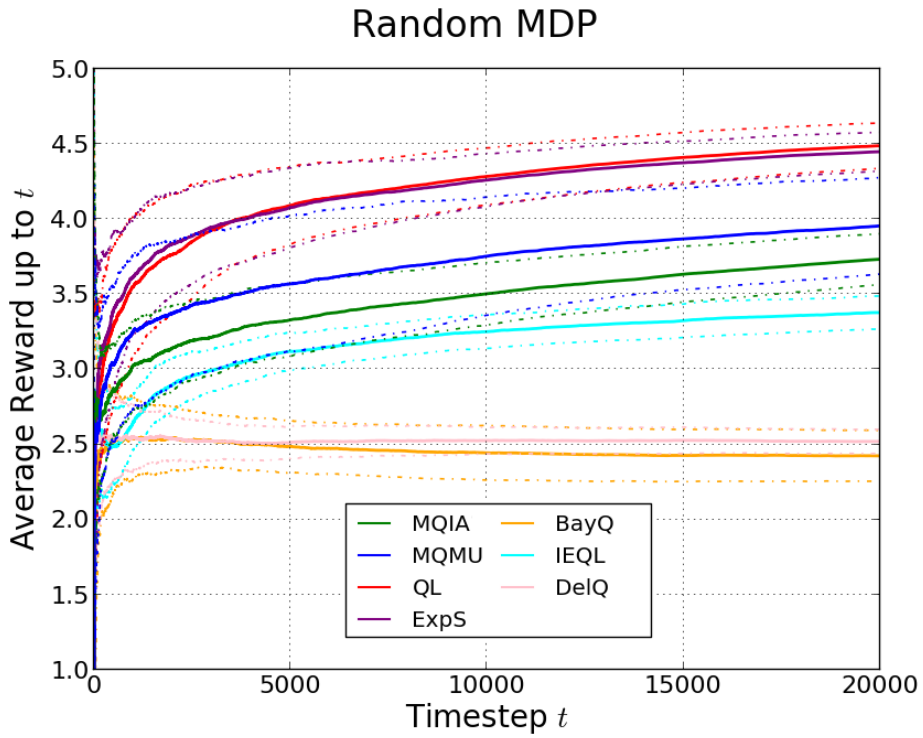


Figure 5.8: Experiment 3, performance of candidates on Random MDP: average reward accrued as function of timestep 0 to t. The solid lines and dashed lines respectively depict the mean and standard deviation over 50 measurements that correspond to 50 randomly generated MDPs. We did not include the performance under an optimal policy on each MDP for simplicity.

Chapter 6

Discussion

In this chapter we discuss Multiple Q-learning and the experiments, and mention some suggestions for future work. The next chapter concludes this thesis.

6.1 Discussion

In the first experiment, we found that the exploration strategy QTS did not work well with Multiple Q-learning as it prematurely ceased exploration. The other two proposed exploration strategies, ATS and EUCB, obtained better results. As we foresaw in Section 4.1.2.2, EUCB did not mix well with Individual Max as the exploration got stuck with exploiting the same action regardless of how the parameter would be tuned. Here, we identified MQIA and MQMU as the best performing candidates of the tried Multiple Q-learning variants and decided to include both of them in the other two experiments.

Prior to the second experiment we had conflicting ideas about whether the performance would increase or decrease as the number of estimates increases. We suspected that fewer estimates would allow the behaviour policy to change quicker and thus better performance, but that it would also come with a larger risk of getting stuck and thus worse performance. For more estimates, we suspected that it was less likely to get stuck and obtain a higher performance in the end, but with a slower start. In the results, we observed that for both MQIA and MQMU the performance was lowest when only 2 estimates were used. Nevertheless, 2 estimates still posed to be adequate for the behaviour policy to converge, albeit slower than with more estimates, towards the optimal policy in both tested MDPs. It might be interesting for future research to investigate the sufficient conditions regarding the number of estimates such that the estimates will still converge, because fewer estimates require less memory space and computation. In the end, we decided to only include candidates with 10 estimates for the third and final experiment, because 5 estimates performed poorer and more than 10 did not seem to further improve performance.

The final experiment was the most intriguing as it would not only be of use to answer our main research question, but also give us insight to the performance of other online model-free alternatives from the literature. To our surprise, most of them performed substantially worse than Q-learning and Expected Sarsa on all MDPs. This is fascinating as Q-learning was commonly included in the respective literature of those algorithms and obtained poorer performance than shown in our experiments. The apparent difference between our experimental setup and the ones from the literature is that we opted for an optimistic initialization which ϵ-greedy greatly benefited from.

Unfortunately, we were unable to let Bayesian Q-learning and Delayed Q-learning shine on any of the MDPs. However, Bayesian Q-learning performed similarly to what has been reported in its originating work (Dearden et al., 1998). As for Delayed Q-learning, we believe that tuning its parameters for ORL is too complicated.

Regarding our main research question, the empirical results only prompted us to reject the null hypotheses for both MQIA and MQMU on the Six Arms MDP. On the other MDPs, Q-learning posed to be the better performing candidate, and sometimes by a landslide. However, on the Cliff MDP, we also observed that MQMU accrues lower returns initially, but performed consistently optimal much later on. We think it is likely that if we were to consider the average return over more episodes (1000 or more), then MQMU might prove to be best.

Furthermore, we only tuned the learning rate scheme's parameter for MQIA and MQMU in the final experiments, whereas for Q-learning with ϵ-greedy we also tried 6 different values for ϵ. One might argue that the performance of ϵ-greedy was mostly due to fine-tuning its parameters. For instance, we found during preliminary experiments that on the Six Arms MDP all fixed values for ϵ result in very poor performance. In conjunction, we found that MQMU obtained performance on par with model-based state-of-the-art algorithms on the Chain problem when the EUCB parameter was set to zero, which corresponds to always acting greedy with respect to the mean of the action-value estimates. We did not include this result as this would be reporting the best results after trying many settings on something that is supposedly unknown. Our perspective is that finding a proper balance between the amount of tuning parameters and assuring unbiased performances for different algorithms is not trivially done, and in certain sense more labor-intensive than one wishes to devote.

Another point to discuss is that we answer our research questions using the average reward per timestep (and average return per episode for the Cliff MDP) over a finite horizon as performance metric. Other performance metrics might prove to be better for evaluating MQ, because an initial abysmal performance will largely effect the average reward despite that it may perform optimally later on. For example, we could have considered the number of timesteps required for the learned estimation policy to achieve a performance threshold or the offline performance: the expected return obtained by a greedy policy on the learned value estimates after a number of timesteps (van Hasselt, 2011). Also, the candidates themselves assume that interaction proceeds over an infinite horizon and learn the action-values in terms of an infinite number of rewards. Hence, one can speak of a mismatch between what the algorithms optimise and what is measured in our experiments. We believe that measuring over a finite horizon was the only logical choice, because it is usually a very complicated task to compute the expected return of a non-Markovian non-stationary stochastic policy over an infinite horizon.

6.1.1 Remarks on Multiple Q-learning

In this section we briefly state several remarks on the Multiple Q-learning algorithm regarding how the estimates are updated and possible points for improvement.

The online bootstrapping-based learning rate scheme introduces diversity among the multiple estimates, but may also introduce a structural bias. This is because the empirical distribution of the observed samples may disagree with the true distribution. As a result, the multiple estimates could be far off. Although, we believe that this is partially accounted for by initially spreading the estimates over a large interval, which

results in diversity due to using different value samples as well.

As Multiple Q-learning closely resembles Q-learning, it also suffers some of the same disadvantages such as the overestimation bias. For Multiple Q-learning, both proposed estimation policies Individual Max and Mean Max are not robust to outliers: a single estimate can severely affect which actions are considered and how the implicit policy improves. In order to ameliorate the overestimation bias, one may consider adopting ideas from Double Q-learning. E.g., randomly assigning pairs of the multiple estimates to use each others' action-values. Another alternative that we looked into is to use a trimmed mean over the different action-value estimates.

Another point of improvement is that the suggested stochastic learning rate scheme for Multiple Q-learning allows none of the estimates to be updated. This occurs when for all of the estimates the sampled $m \sim Poisson(1)$ equals zero. The probability of this happening for N estimates is:

$$\binom{N}{0}(1 - e^{-1})^N \ ,$$

which is derived from the binomial distribution and uses the fact that $Pr(Poisson(1) = 0) = e^{-1} = 0.3679$. Fortunately, this probability becomes smaller as the number of estimates is increased.

6.2 Future Work

In this section we mention ideas for future work.

1. *Algorithmic Improvement*: an obvious path for future research is to incorporate ideas from other algorithms that can learn action-values with fewer samples in exchange for more computation. For example, we can update estimates of various state-action values simultaneously per timestep.

   We believe that the most improvement can be found here as updating estimates intrinsically determines how quickly the agent changes its behaviour and how appropriately it improves its performance. The essence is that the faster the estimates quickly and accurately converge to the optimal policy, the quicker the agent will start performing optimally. We think that it is relatively straightforward to combine the idea of multiple estimates with other algorithms as they revolve around distributing samples among different estimates and initialization. For example, we can profit from the sample efficiency of model-based reinforcement learning and keep track of a number of model estimates. Other options are eligibility traces (Sutton and Barto, 1998), just-in-time algorithms and best-match equations for reinforcement learning (van Seijen et al., 2011).

2. *Multi-armed Bandits*: we believe that MQ can be directly applied to address the exploration-vs-exploitation dilemma in multi-armed bandits with minor adjustments. Here, the value samples are only single sample rewards and hence unbiased. Using MQ's stochastic learning rate scheme that is based on online bootstrapping and a (large) number of estimates one may be able to approximate confidence intervals in an online fashion, which can be used for ORL, see Appendix B.

3. *Other Multiple Q-learning Options*: there remain numerous different options that one can consider for MQ: other learning rate schemes, ways of initializing, policy improvement and exploration strategies. During our research, we have attempted

many different approaches, but there is no doubt that there are more possibilities that we did not yet thought of.

4. ϵ-*greedy*: it was interesting to discover that in the past many researches have been conducted to guide exploration in a directed manner with the ambition of improved online (or offline) performance, but failed to appropriately tune Q-learning. Many of those researches never considered to initialize the Q-function at an optimistic value, and nonchalantly concluded that their proposed algorithm works better. During our research, we found that those algorithms were not easily employed, whereas ϵ-greedy was tuned and running with the least effort. Thus, an intriguing research perspective is to thoroughly examine and understand when and why ϵ-greedy, as well as Boltzmann action selection for that matter, perform well or better than other options on MDPs. The foremost question being: is it possible to perform better than a properly tuned ϵ-greedy?

Chapter 7

Conclusion

In this thesis we took on the ambitious task of performing robustly in an online reinforcement learning scenario where one is limited to online model-free value-based learning algorithms. In the literature, we found a variety of algorithms that are intended, or could be used, for this purpose. Many of them conjectured that one should direct exploration rather than undirectedly, such as ϵ-greedy, and provided various arguments.

However, we came to the realization that most of them only performed well on specific MDPs as they were dependent on fine-tuned parameters or restrictive assumptions. Furthermore, we discovered that the elementary Q-learning with ϵ-greedy significantly outperformed all of them when optimistically initialized, which requires little effort.

This sparked our interest in researching and cultivating an even better algorithm that followed the philosophy of directed exploration, but benefits from the same advantages as Q-learning. In this endeavour, we arrived at the idea of using multiple estimates based on Q-learning for action selection: Multiple Q-learning.

We investigated numerous ways of updating and using the multiple estimates, and in the end decided upon two variants based on empirical results: MQIA and MQMU. Unfortunately, these variants also were unsuccessful in empirically outperforming Q-learning with ϵ-greedy on several environments. MQMU only managed to obtain significantly better results on the Six Arms MDP, but was outclassed by IEQL instead.

However, we believe our effort was not in vain as MQMU performed well overall with even less effort required to tune the exploration parameters than Q-learning. We are certain that it would prove to be best on the Cliff MDP if we measured the performance over more episodes. In addition, it severely outperformed IEQL in all other environments than the Six Arms MDP.

We conclude that the answer to our main research question, whether multiple estimates is beneficial for model-free online reinforcement learning, is two-fold. We can answer *"no"* when considering that its performance was not statistically significantly better, and was surpassed by a single-estimate based algorithm (Q-learning) on most of the tested MDPs. We can answer *"yes"* when considering that its performance is better on at least one of the MDPs, performed as one of the best on all MDPs and required the least effort for tuning parameters.

Therefore, we believe that MQ has introduced an interesting idea for balancing exploration with exploitation, but requires more research to fully answer our research question satisfactorily. We expect that MQ will improve the most by further investigating how the estimates can be updated more quickly and accurately.

Appendix A

Nomenclature

General Notation

Symbol	Description
$lhs = rhs$	Left-hand side (lhs) is equal in value to right-hand side (rhs)
$lhs \approx rhs$	Lhs is approximately equal in value to rhs
$lhs >, < rhs$	Lhs is higher than / lower than rhs
$lhs \geq, \leq rhs$	Lhs is equal or higher than / equal or lower than rhs
$lhs \leftarrow rhs$	Lhs becomes rhs
$lhs \equiv rhs$	Lhs is congruent to rhs
$lhs \iff rhs$	Lhs holds if and only if rhs
$0 \in \mathbb{N}, \mathbb{R}$	Set of natural numbers (incl. 0), Set of real numbers
$\{elements\}$	Set of elements
$\{elements \mid expression\}$	Set of elements derived by expression
$element \in set$	Element is a member of set
$set \times otherset$	Cartesian product of set and otherset
$\|set\|$	Cardinality of set
$[a, b]$	Closed interval containing values: $\{x \in \mathbb{R} \mid a \leq x \leq b\}$
$(a, b]$	Interval closed from above and open from below: $\{x \in \mathbb{R} \mid a < x \leq b\}$
$Function : domain \to range$	Function declaration
$Function(a)$	Function call with argument a
$\max_{args} expression(arg)$	The maximum value of the expression given the arguments
$\arg\max_{args} expression(arg)$	The set of elements of the given argument for which the expression attains its maximum value
$\|\|expression\|\|$	Maximum norm for expression
$\|expression\|$	Absolute value for expression
$Pr(X = x)$	Marginal probability that random variable X equals random variate x
$Pr(X \mid Y)$	Conditional probability of X given Y
$\mathbb{E}\{X\}$	Expected value of random variable X
$\mathbb{E}\{X \mid Y\}$	Conditional expected value of random variable X given Y
$X \sim D$	Random variable X has probability distribution D

Table A.1: Notations used in this thesis.

Notations for Modelling Problems and Solutions

Symbol	Description	Formula
Modelling		
$M \in \mathfrak{M}$	Markov decision process (MDP), Space of MDPs	$M = \langle \mathbb{S}, \mathbb{A}, P, R, I, \gamma \rangle$
$t \in \{0, \dots, T\}$	Timestep, Horizon	$T \subseteq \mathbb{N}$
$s, s_t \in \mathbb{S}$	State, State at timestep t, State space	$\lvert\mathbb{S}\rvert \subseteq \mathbb{N}$
$a, a_t \in \mathbb{A}$	Action, Action at t, Action space	$\lvert\mathbb{A}\rvert \subseteq \mathbb{N}$
(s, a)	State-action pair	
S_t, A_t, R_t	Random variables for State, Action, Reward at t	
$S_{t+1} \sim P(S_t, A_t)$	Transition, Transition function	$P : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$
$R_{t+1} \sim R(S_t, A_t, S_{t+1})$	Reward, Reward function	$R : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \times \mathbb{R} \to [0, 1]$
$S_0 \sim I$	Initial state probability distribution	$I : \mathbb{S} \to [0, 1]$
γ	Discount rate parameter	$\gamma \in [0, 1]$
$H_{0:t}$	Random History from timestep 0 to t	Eq. 2.1
Solutions: Policies		
$\pi, \pi_t \in \Pi$	Policy function (at t), Space of policies	See Sec. 2.1.1
$\pi(s, a)$	Action-probability for action a in state s	$\pi(s, a) \in [0, 1]$
$J(\pi, M)$	Value of a policy on an MDP	Eq. 2.2
π_*	Optimal policy	Eq. 2.3
Value Functions		
V	Value function	$V : \mathbb{S} \to \mathbb{R}$
$V(s)$	State-value of state s	$\forall_{s \in \mathbb{S}} : V(s) \in \mathbb{R}$
V^*	Optimal value function	$\forall_{\pi \in \Pi}, \forall_{s \in \mathbb{S}} : V^*(s) \geq V^\pi(s)$
$Q \in \mathbb{Q}$	Q-function, Space of Q-functions	$Q : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$
$Q^\pi(s, a)$	Action-value of (s, a) under π	Eq. 2.4
Q^*	Optimal Q-function for an MDP	Eq. 2.8
$Q^*(s, a)$	Optimal action-value of (s, a)	$\forall_{\pi \in \Pi} : Q^*(s, a) \geq Q^\pi(s, a)$
Learning		
π_t^b	Behaviour policy of the agent at t	$\pi_t^b \in \Pi$
Q_t	Q-function estimate at t	$Q_t \in \mathbb{Q}$
$Q_t(s, a)$	Action-value estimate at t	$\forall_{(s,a)} : Q_t(s, a) \in \mathbb{R}$
π_t^e	Estimation policy at t	$\pi_t^e \in \Pi$
$\mathbb{M}$	Set of actions that maximise some expression	$\arg\max_{a \in \mathbb{A}} expression$
X_t	Value sample at t	$X_t \in \mathbb{R}$
α_t	Learning rate at t	$\alpha_t \in [0, 1]$
$n(s, a)$	Counter function per state-action pair	$n : \mathbb{S} \times \mathbb{A} \to \mathbb{N}$
Multiple Q-learning		
Q_t^i	i-th Q-function estimate at t	$Q_t^i \in \mathbb{Q}$
π_t^i	Estimation policy of i-th Q-function at t	$\pi_t^i \in \Pi$
X_t^i	Value sample for i-th Q-function at t	$X_t^i \in \mathbb{R}$
α_t^i	Learning rate for i-th Q-function at t	$\alpha_t \in [0, 1]$
$n_t^i(s, a)$	Counter function for i-th Q-function at t	$n : \mathbb{S} \times \mathbb{A} \to \mathbb{N}$

Table A.2: Notation used for reinforcement learning.

Appendix B

Online Bootstrapping

In this appendix we show an online approximation of the sampling distribution of the sample mean using *online bootstrapping* (Oza and Russell, 2001). This can be used for online model-free action-value estimation, as an action-value estimate conceptually corresponds to a sample mean. Notice that temporal-difference learning is essentially an incremental method to estimating sample means, which can be easily adopted for online scenarios where data arrives as a stream or can not be memorized.

We first discuss online bootstrapping and then discuss how it can be used with incremental estimation of the sample mean.

B.1 Online Bootstrapping

Online bootstrapping is an online version of non-parametric bootstrapping, which is a statistical resampling method to approximate the sampling distribution of a statistic. To this end, one uses the empirical data as a surrogate distribution of the true distribution, which allows for several statistical inferences (Efron, 1979).

Suppose we have a original set of N samples: $X_1, X_2, \ldots, X_N$, then 'offline bootstrapping' proceeds by performing sampling N times with replacement from this set and compute a statistic using the drawn samples. This is done many times to get a variety of estimates, which in turn can be used to estimate the standard error.

In Oza and Russell (2001), they notice that each sample in the set can be replicated zero, one, two or more times in each *bootstrap set*. Mathematically, the number of copies K of a sample in a bootstrap set follows a binomial distribution:

$$Pr(K = k) = \binom{N}{k} N^{-k} \Big(1 - \frac{1}{N}\Big)^{N-k} \quad .$$

Hence, they argue that one can consider each sample in the original one at a time and replicate K copies using the above equation, instead of drawing samples with replacement. Notice that it is possible that one can obtain a bootstrap set with more or less samples.

In an online scenario, samples usually arrive continuously and we may not know N the total number of samples. In Oza and Russell (2001), they consider the case where N goes to infinity, which results in the distribution of K following a $Poisson(1)$ distribution:

$$Pr(K = k) = \frac{exp(-1)}{k!} \quad . \tag{B.1}$$

B.2 Incremental Estimation of the Sample Mean

We consider the estimation of the population mean μ of an unknown distribution. We assume that we are in an online scenario where we can not retain prior samples and can only update estimates in an online fashion.

Assume that $X_1, X_2, \ldots$ are independent random variables. Let θ denote an estimate for μ. The conventional sample mean of N samples is:

$$\theta = N^{-1} \sum_{n=1}^{N} X_n \ . \tag{B.2}$$

Suppose that we are at the n-th sample. In our scenario we can not keep track of the samples X_1 to X_{n-1} and must therefore estimate the sample mean in an incremental fashion. Let θ_n be the estimate after including the n-th sample. The incremental mean estimate is then:

$$\theta_n = \theta_{n-1} + \frac{1}{n}(X_n - \theta_{n-1}) \ , \tag{B.3}$$

and let θ_0 be initialized arbitrarily.

In temporal-difference learning, one identifies $Q_t(s,a)$ as θ_{n-1} and $Q_{t+1}(s,a)$ as θ_n. The learning rate α takes the place of $\frac{1}{n}$ and enables one to give more weight to recent samples using learning rate schemes.

B.3 Online Approximation of the Sampling Distribution of the Sample Mean

Here, we can combine the previous two ideas to arrive at an approximate sampling distribution of the sample mean.

We proceed by using multiple estimates that are updated individually as shown in Algorithm 2. Let θ_n^i denote the i-th bootstrap estimate for the sample mean after the first n samples. And let n^i denote the number of times the i-th estimate was updated, which is initialized at zero. Each θ_n^i is updated using Equation B.3.

Algorithm 2: Online approximation of the sampling distribution of the sample mean

input: Number of estimates I

for $n \leftarrow 1, 2, \ldots$ **do**
  Observe X_n
  for $i \leftarrow [1, I]$ **do**
    $k \leftarrow Poisson(1)$ `See Equation B.1`
    while $k > 0$ **do**
      $n^i \leftarrow n^i + 1$
      $\theta_n^i \leftarrow \theta_{n-1}^i + \frac{1}{n^i}(X_n - \theta_{n-1}^i)$ `See Equation B.3`
      $k \leftarrow k - 1$
    end
  end
end

We perform a small experiment to demonstrate the working of online estimation of the sampling distribution of the sample mean. We compare its performance to offline bootstrapping that uses the conventional sample mean, see Equation B.2. We suppose that samples are drawn from a Normal distribution with mean $\mu = 1$ and $\sigma = 10$. We report two results:

1. Figure B.1 shows the mean plus/minus the standard deviation of the estimates when using *online bootstrapping with incremental mean* and *offline bootstrapping with the conventional sample mean*. We consider an online scenario where the original set grows over time: from 1 to $N = 2500$ samples. Hence, the n-th original set contains all previous samples plus n-th sample. In both cases we use 50 bootstrap estimates. The estimates are initially set at $Normal(0, \sigma)$.[35]
2. Figure B.2 shows the empirical standard error of the sample mean for both online and offline bootstrapping. To demonstrate its accuracy, we also include the theoretical standard error of the sample mean: $\frac{\sigma}{\sqrt{n}}$. We use similar settings as the previous one.

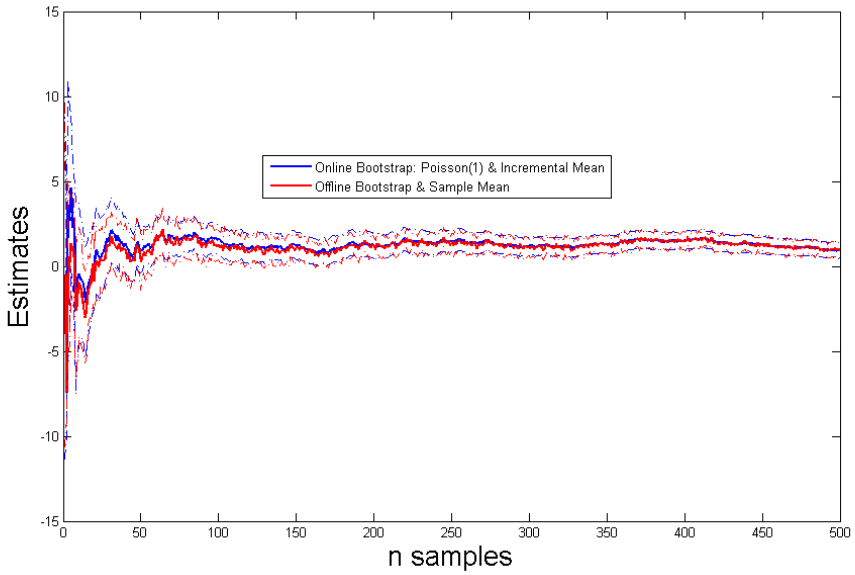


Figure B.1: Graph showing the mean plus/minus the standard deviation of the estimates for both online and offline bootstrapping.

[35] Notice that after the first update, the initial estimate is effectively forgotten.

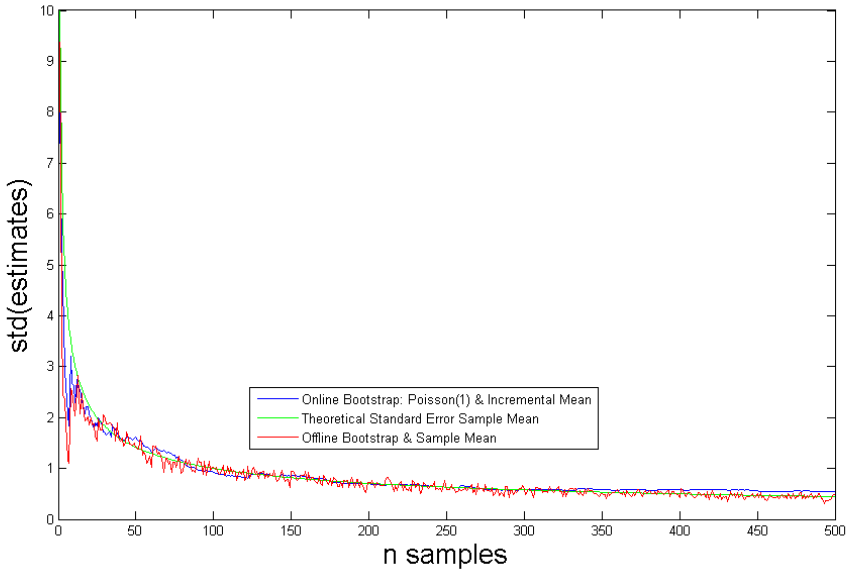


Figure B.2: Graph showing the standard error of the estimates for both online and offline bootstrapping, and the theoretical standard error.

Appendix C

Proofs

In this part we will note the lemmas and proofs used and referred to within this thesis.

C.1 Lemmas

Lemma 1. *(Singh et al., 2000) (Asynchronous Single-Step Updates) Consider a stochastic process* (ζ_t, Δ_t, F_t)*, where* $\zeta_t, \Delta_t, F_t : X \to \mathbb{R}$ *satisfy the equations*

$$\Delta_{t+1}(x_t) = (1 - \zeta_t(x_t))\Delta_t(x_t) + \zeta_t(x_t)F_t(x_t) \ ,$$

where $x_t \in X$ *and* $t = 0, 1, 2, \ldots$*. Let* P_t *be a sequence of increasing* σ*-fields such that* ζ_0 *and* Δ_0 *are* P_0*-measurable and* ζ_t*,* Δ_t *and* F_{t-1} *are* P_t*-measurable,* $t \geq 1$*. Then* Δ_t *converges to zero with probability one if the following are assumed to hold:*

1. *the set* X *is finite,*
2. $\zeta_t(x_t) \in [0, 1]$*,* $\sum_{t=1}^{\infty} \zeta_t(x_t) = \infty$*,* $\sum_{t=1}^{\infty} [\zeta_t(x_t)]^2 < \infty$ *w.p.1 and* $\forall x \neq x_t : \zeta_t(x) = 0$*,*
3. $||E\{F_t \mid P_t\}|| \leq \kappa||\Delta_t|| + c_t$*, where* $\kappa \in [0, 1)$ *and* c_t *converges to zero w.p.1,*
4. $Var\{F_t(x_t) \mid P_t\} \leq K(1 + \kappa||\Delta_t||)^2$*, where* K *is some constant,*

where $||\cdot||$ *denotes a maximum norm.*

The general idea is to apply this lemma to temporal-difference learning algorithms by identifying, e.g., $X = S \times A$, $P_t = \{Q_0, s_0, a_0, \alpha_0, r_1, s_1, a_1, \ldots, s_t, a_t\}$, $v_t = (s_t, a_t)$, $\zeta_t(v_t) = \alpha_t(s_t, a_t)$ and $\Delta_t(v_t) = Q_t(s_t, a_t) - Q^*(s_t, a_t)$ (van Hasselt, 2011). If one can prove that Δ_t converges to zero with probability one, then convergence of the action-value estimates to the optimal action-values is assured.

C.2 General Q-learning

In this section we describe the proof of convergence for General Q-learning, which subsumes Q-learning and (Expected) Sarsa (van Hasselt, 2011). The proof is largely adopted from van Hasselt (2011), with minor alterations that allow one to easily fill in proofs for convergence to the optimal Q-function or the Q-function of another arbitrary policy. By doing so, we can conveniently use this in subsequent proofs for MQ.

Theorem 2. *(van Hasselt, 2011) (Convergence of General Q-learning)*
General Q-learning as defined by

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)\big(r_{t+1} + \gamma \sum_a \pi_t^e(s_{t+1}, a) Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\big) \ ,$$

converges to the optimal Q-function whenever the following assumptions hold:

1. $\mathbb{S}$ *and* $\mathbb{A}$ *are finite,*
2. $\alpha_t(s_t, a_t) \in [0, 1]$, $\sum_{t=1}^{\infty} \alpha_t(s_t, a_t) = \infty$,
   $\sum_{t=1}^{\infty} [\alpha_t(s_t, a_t)]^2 < \infty$ *w.p.1, and* $\forall (s, a) \neq (s_t, a_t) : \alpha_t(s, a) = 0$,
3. *The estimation policy* π_t^e *is greedy in the limit, and the behaviour policy* π_t^b *ensures infinite exploration,*
4. *The reward function of the MDP is bounded;* $Var\{r_{t+1} \mid P_t\} < \infty$.

Proof. One can prove theorem 2 by showing that the conditions of Lemma 1 hold. We identify $X = S \times A$, $P_t = \{Q_0, s_0, a_0, \alpha_0, r_1, s_1, a_1, \ldots, s_t, a_t\}$, $v_t = (s_t, a_t)$ and $\zeta_t(v_t) = \alpha_t(s_t, a_t)$. The first and second assumptions of the theorem correspond with the first and second assumptions of the lemma, and the fourth assumption of the theorem assures that $Var\{F_t(x_t) \mid P_t\} \leq K(1 + \kappa ||\Delta_t||)^2 < \infty$ (is bounded) and hence the fourth condition of the lemma is also satisfied. Thus, one only have to prove that the third assumption of the lemma holds as well. We derive

$$\begin{aligned} \Delta_t(s_t, a_t) &= Q_t(s_t, a_t) - Q^*(s_t, a_t) \\ F_t(s_t, a_t) &= \frac{1}{\alpha_t(s_t, a_t)} (\Delta_{t+1} - (1 - \alpha_t(s_t, a_t))\Delta_t(s_t, a_t)) \\ &= r_t + \gamma \sum_a \pi_t^e(s_{t+1}, a) Q_t(s_{t+1}, a) - Q^*(s_t, a_t)) \ . \end{aligned}$$

We can assume that $\forall (s, a) \neq (s_t, a_t) : F_t(s, a) = 0$. Notice that for Q-learning $\sum_a \pi_t^e(s_{t+1}, a)$ can be substituted by $\max_a$, for Expected Sarsa the update remains the same, and for Sarsa the estimation policy collapses to a single action having an action-probability of one and others zero.

If we can prove that $||\mathbb{E}\{F_t \mid P_t\}|| \leq \kappa ||\Delta_t|| + c_t$, where $\kappa \in [0, 1)$ and c_t converges to zero, then all the conditions of the lemma hold and Δ_t converges to zero. This implies that the Q-function estimate converges to the optimal Q-function. The proof continues on the next page.

The following is taken from van Hasselt (2011) with some minor corrections applied:

$$
\begin{aligned}
&||\mathbb{E}\left\{F_t \,|\, P_t\right\}|| \\
&= \left|\left|\mathbb{E}\left\{ r_t + \gamma \sum_{a'} \pi_t^e(s_{t+1}, a') Q_t(s_{t+1}, a') - Q^*(s_t, a_t) \,\middle|\, P_t \right\}\right|\right| \\
&= \left|\left|\mathbb{E}\left\{ r_t + \gamma \sum_{a'} \pi_t^e(s_{t+1}, a') Q_t(s_{t+1}, a') \,\middle|\, P_t \right\} - \sum_{s'} P(s_t, a_t, s') \big(R(s_t, a_t, s') + \gamma max_{a'} Q^*(s', a')\big)\right|\right| \\
&= \left|\left|\sum_{s'} P(s_t, a_t, s') \Big(\gamma \sum_{a'} \pi_t^e(s', a') Q_t(s', a') - \gamma max_{a'} Q^*(s', a')\Big)\right|\right| \\
&\leq \gamma \max_s \left| \sum_a \pi_t^e(s, a) Q_t(s, a) - \max_a Q^*(s, a) \right| \\
&\leq \gamma \max_s \left| \max_a Q_t(s, a) - \max_a Q^*(s, a) \right| + \gamma \max_s \left| \sum_a \pi_t^e(s, a) Q_t(s, a) - \max_a Q_t(s, a) \right| \\
&\leq \gamma \,||\Delta_t\,|| + \gamma \max_s \left| \sum_a \pi_t^e(s, a) Q_t(s, a) - max_a Q_t(s, a) \right| \quad , \qquad \text{(C.1)}
\end{aligned}
$$

where $R(s, a, s')$ denotes the expected immediate reward after transitioning to successor state s' after taking action a in state s, and $P(s, a, s')$ is the probability associated with the transition. We can identify $c_t = \gamma \max_s |\sum_a \pi_t^e(s, a) Q_t(s, a) - \max_a Q_t(s, a)|$ and $\kappa = \gamma$. This means that c_t will converge to zero for estimation policies that are greedy (in the limit). Therefore, as long as $\gamma < 1$, all conditions of Lemma 1 hold and convergence of Q_t to Q^* is ensured. Notice that this theorem and proof can be applied to Q-learning, Expected Sarsa and more temporal-difference learning algorithms. □

C.3 Proofs for Multiple Q-learning

In this section we show proof for Theorems 1 and 1. The general approach that we take is to show that the way estimates are updated closely resembles Q-learning, for which it is known to converge to the optimal Q-function given certain conditions hold.

Multiple Q-learning with Individual Max

Proof. (Theorem 1, Multiple Q-learning with Individual Max, Eq. 4.13)
The update for the i-th Q-function under Multiple Q-learning with Individual Max is defined by (shortened)

$$Q_{t+1}^i(s_t, a_t) = Q_t^i(s_t, a_t) + \alpha_t^i(s_t, a_t)\big(r_{t+1} + \gamma \max_a Q_t^i(s_{t+1}, a) - Q_t^i(s_t, a_t)\big) \ .$$

Because every i-th Q-function is updated in a similar fashion, it suffices to prove for one Q-function under Multiple Q-learning with Individual Max converges to the optimal Q-function. We can do this by showing that the update is equivalent to Q-learning, for which it is known to converge to the optimal Q-function under the same assumptions as for Theorem 1.

That is, we can follow the structure of the proof for Theorem 2 by identifying $Q_t = Q_t^i$, $\alpha_t = \alpha_t^i$ and substitute $\sum_a \pi_t^e(s_{t+1}, a)$ by $\max_a$. As the implicit estimation policy under

Individual Max is a greedy policy on Q_t^i, see Equation 4.13, the estimation policy is obviously greedy (in the limit). General Q-learning and Theorem 1 have the other assumptions in common. Therefore, provided that the assumptions hold as for General Q-learning to satisfy the conditions of Lemma 1, every Q_t^i converges to Q^* under Multiple Q-learning with Individual Max. □

Multiple Q-learning with Mean Max

Proof Sketch. (Conjecture 1, Multiple Q-learning with Mean Max, Eq. 4.14)
Unlike Theorem 1, the estimation policy Mean Max for MQ makes the estimates dependent on the other estimates. Adopting Lemma 1 for proving convergence is not trivial in this case as the different estimates are intrinsically connected with one another. Notice that one can just show convergence of one Q-function estimate, as the other estimates are updated identically.

A plausible step towards proving convergence to Q^* is that if, for each i-th Q-function estimate, the estimation policy π_t^i converges to some π_t^e:

$$\forall_{i\in[1,N]} : \lim_{t\to\infty} \pi_t^i \to \pi_t^e \ .$$

Notice that for Mean Max, the estimation policies for the Q-function estimates are identical at each timestep:

$$\forall_{i,j\in[1,N]}, \forall_{0\leq t\leq\infty} : \pi_t^i = \pi_t^j \ .$$

Using Lemma 1, one can then show that all Q-function estimates will converge to the same $Q^{\pi_t^e}$ as the updates will correspond to Expected Sarsa. Given that all Q-function estimates are identical, the estimation policy under Max Mean will act identically to a greedy policy with respect to a single Q-function like Individual Max:

$$\forall_{i,j} : Q_t^i = Q_t^j \implies \forall_{i,j} : \max_x Q_t^i(x) = \max_x \sum_k Q_t^k(x) = \max_x [n \cdot Q_t^i(x)] = \max_x Q_t^j(x) \ ,$$

where n is the number of Q-functions. We believe this will be an essential step for showing that the estimation policy will be greedy in the limit, which is required for convergence to Q^*.

However, convergence of the estimation policy to some π_t^e relies on the evolution of multiple Q-function estimates. We found in the literature only convergence proofs for estimation policies dependent on a single estimate, or where it was simply assumed to converge (Singh et al., 2000). The dilemma here is that it is not easy to prove that the estimation policy will converge at all: it might constantly shift from one target to another target estimation policy. We believe that Lemma 1 could to be extended to accommodate for multiple estimates.

Bibliography

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

D.P. Bertsekas and S.E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Athena Scientific, 2007.

S.J. Bradtke. *Incremental dynamic programming for on-line adaptive optimal control*. PhD thesis, University of Massachusetts at Amherst, 1994.

L. Buşoniu, R. Babuška, B. de Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC Press, 2010.

O. Chapelle and L. Li. An empirical evaluation of Thompson Sampling. In *Neural Information Processing Systems)*, 2011.

T. Cohen and Knight W.J. Convergence and divergence of $\sum_{n=1}^{\infty} 1/n^p$. 52(3):178, 1979.

R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 761–768. John Wiley & Sons LTD., 1998.

R.W. Dearden. Learning and Planning in Structured Worlds, 2000.

M.O.G. Duff. *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 201–208. ACM, 2005.

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. *The Journal of Machine Learning Research*, 5:1–25, 2003.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning*, volume 20, pages 162–169, 2003.

R.A. Howard. *Dynamic programming and Markov processes*. MIT Press, 1960.

L.P. Kaelbling. *Learning in Embedded Systems*. PhD thesis, Stanford University, Stanford, California, 1993.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

S. Koenig and R.G. Simmons. Complexity analysis of real-time reinforcement learning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, page 99–105. Menlo Park, CA: AAAI Press/MIT Press , 1993.

V. Kuleshov and D. Precup. Algorithms for the multi-armed bandit problem. *Journal of Machine Learning Research*, 2010.

S. Lange, Th. Gabel, and M. Riedmiller. Batch reinforcement learning. In *Reinforcement Learning: State-Of-the-Art*, pages 45–73. Springer, 2012.

L. Li. Sample Complexity Bounds of Exploration. In *Reinforcement Learning: State-Of-the-Art*, pages 175–204. Springer, 2012.

J.J. Martin. *Bayesian decision problems and Markov chains*. Wiley, 1967.

N. Meuleau and P. Bourgine. Exploration of Multi-State Environments: Local Measures and Back-Propagation of Uncertainty. *Machine Learning*, 35(2):117–154, May 1999.

N.C. Oza and S. Russell. *Online ensemble learning*. PhD thesis, University of California, Berkeley, 2001.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 697–704. ACM, 2006.

M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. new York, NY, USA, 1994.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, University of Cambridge, Department of Engineering, 1994.

S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M.L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, pages 881–888. ACM, 2006.

A.L. Strehl and Michael L. Littman. An empirical evaluation of interval estimation for markov decision processes. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 128–135. IEEE, 2004.

A.L. Strehl, L. Li, and M.L. Littman. Reinforcement learning in finite MDPs: PAC analysis. *The Journal of Machine Learning Research*, 10:2413–2444, 2009.

M. Strens. A Bayesian Framework for Reinforcement Learning. In *Proc. 17th International Conf. on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000.

R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2010.

W.R. Thompson. On the Likelihood that one Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25:285–294, 1933.

S.B. Thrun. Efficient exploration in reinforcement learning. Technical report, Carnegie-Mellon University, 1992.

J.N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

H. van Hasselt. Double Q-Learning. In *Advances in Neural Information Processing Systems*, volume 23. The MIT Press, 2010.

H. van Hasselt. *Insights in Reinforcement Learning*. PhD thesis, Utrecht University, 2011.

H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering. Exploiting Best-Match Equations for Efficient Reinforcement Learning. *Journal of Machine Learning Research*, 12:2045–2094, 2011.

H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering. A theoretical and empirical analysis of Expected Sarsa. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pages 177–184. IEEE, 2009.

N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart. Bayesian Reinforcement Learning. In *Reinforcement Learning: State-Of-the-Art*, pages 359–386. Springer, 2012.

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 956–963. ACM, 2005.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.

C. J. C. H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 8:279–292, 1992.

Bernard L Welch. The generalization of "student's" problem when several different population variances are involved. *Biometrika*, pages 28–35, 1947.

S.D. Whitehead and D.H. Ballard. *A study of cooperative mechanisms for faster reinforcement learning.* University of Rochester, Department of Computer Science, 1991.

M.A. Wiering. *Explorations in efficient reinforcement learning.* PhD thesis, University of Amsterdam, 1999.

M.A. Wiering and M. van Otterlo. *Reinforcement Learning: State-Of-the-Art.* Adaptation, Learning, and Optimization. Springer, 2012.