Reinforcement Learning with a Near Optimal Rate of Convergence

Mohammad Gheshlaghi Azar, Rémi Munos, Mohammad Ghavamzadeh, Hilbert Kappen

Reinforcement Learning with a Near Optimal Rate of Convergence

Mohammad Gheshlaghi Azar M.AZAR@SCIENCE.RU.NL
Department of Biophysics
Radboud University Nijmegen
6525 EZ Nijmegen, The Netherlands
Rémi Munos REMI.MUNOS@INRIA.FR
Mohammad Ghavamzadeh MOHAMMAD.GHAVAMZADEH@INRIA.FR
INRIA Lille, SequeL Project
40 avenue Halley
59650 Villeneuve d'Ascq, France
Hilbert J. Kappen B.KAPPEN@SCIENCE.RU.NL
Department of Biophysics
Radboud University Nijmegen
6525 EZ Nijmegen, The Netherlands

Editor: TBA

Abstract

We consider the problem of model-free reinforcement learning (RL) in the Markovian decision processes (MDP) under the PAC ("probably approximately correct") model. We introduce a new variant of Q-learning, called speedy Q-learning (SQL), to address the problem of the slow convergence in the standard Q-learning algorithm, and prove PAC bounds on the performance of SQL. The bounds show that for any MDP with n state-action pairs and the discount factor $\gamma \in [0, 1)$ a total of $O\big(n \log(n)/((1-\gamma)^4\epsilon^2)\big)$ step suffices for the SQL algorithm to converge to an ϵ-optimal action-value function with high probability. We also establish a lower-bound of $\Omega\big(n \log(n)/((1-\gamma)^2\epsilon^2)\big)$ for all RL algorithms, which matches the upper bound in terms of n and ϵ. Further, our results have better dependencies on ϵ, $1-\gamma$ and the same dependency on n , and thus, are tighter than the best available results for Q-learning. SQL also improves on existing results for the batch Q-value iteration, so far considered to be more efficient than the incremental methods like Q-learning.

1. Introduction

The Markovian decision process (MDP) problem is a classical problem in the fields of operations research and decision theory. When an explicit model of the MDP, transition probabilities and reward function, is known, one can rely on dynamic programming (DP) Bellman (1957) algorithms such as value iteration or policy iteration (see, e.g., Bertsekas, 2007a; Puterman, 1994), to compute the optimal policy. For example, value iteration computes the optimal value function Q^* by successive iterations of the Bellman operator $\mathcal{T}$, which we define later in Section 2. One can show that, in the discounted infinite-horizon setting, the convergence of value iteration is exponentially fast since the Bellman operator $\mathcal{T}$ is a contraction mapping (Bertsekas, 2007b) on the action-value function Q. However, DP relies on an explicit knowledge of the MDP. In many real world problems the transition probabilities are not initially known, but one may observe transition samples using Monte-Carlo sampling, either as a single trajectory (rollout) obtained by following an exploration policy, or by simulating independent transition samples anywhere in the space by resorting to an oracle which draws samples anywhere in the state-action space. The field of reinforcement learning (RL) is concerned with the problem of approximating the optimal policy, or the optimal value function, from the observed reward and transition samples (Szepesvári, 2010; Sutton and Barto, 1998).

One may characterize RL methods as model-based or model-free. In model-based RL we first learn a model of the MDP and then we use this model for computing an approximation of the value functions by dynamic programming techniques. Model-free methods, in contrast, compute directly an approximation of the value function by making use of a sample-based estimate of the Bellman operator, without resorting to learning a model. Q-learning (QL) is a well-known model-free reinforcement learning (RL) algorithm that, incrementally, finds an estimate of the optimal action-value function (Watkins, 1989). The QL algorithm can be seen as a combination of the value iteration algorithm and stochastic approximation, where at each time step k a new estimate of the optimal action-value function is calculated by the following update rule:

$$Q_{k+1} = (1-\alpha_k)Q_k + \alpha_k\left(\mathcal{T}Q_k - \epsilon_k\right),$$

where ϵ_k and α_k denote the estimation error and the learning step at time step k, respectively. One can show, using an induction argument, that for the choice of linear learning step, i.e., $\alpha_k = \frac{1}{k+1}$, QL is taking average over the estimates of Bellman operator throughout the learning process:

$$Q_{k+1} = \frac{1}{k+1}\sum_{j=0}^{k}(\mathcal{T}Q_j - \epsilon_j).$$

It is not then difficult to prove, using a law of large number argument, that the term $1/(k+1)\sum_{j=0}^{k}\epsilon_j$ is asymptotically averaged out and, therefore, for $k \to \infty$ the update rule of Q-learning becomes equivalent to $Q_{k+1} = 1/(k+1)\sum_{j=0}^{k}\mathcal{T}Q_j$. The problem with this result is that the rate of convergence of the recursion $Q_{k+1} = 1/(k+1)\sum_{j=0}^{k}\mathcal{T}Q_j$ to Q^* is significantly slower than the original Bellman recursion $Q_{k+1} = \mathcal{T}Q_k$. In fact, one can show that the asymptotic rate of convergence of Q-learning, with linear learning step, is of order $\tilde{O}(1/k^{1-\gamma})$ (Szepesvári, 1997), which, in the case of γ close to 1, makes the convergence of Q-learning extremely slower than the standard value iteration, which enjoys fast convergence rate of $\tilde{O}(\gamma^k)$. This slow rate of convergence, i.e., high sample complexity, may explain why the practitioners often prefer the batch RL methods such as approximate value iteration (AVI) (Bertsekas, 2007b) to Q-learning despite the fact that Q-learning has an equal, or in some cases even better, computational complexity (per sample) and memory requirements than the batch RL methods.

In this paper, we focus on RL problems that are formulated as finite state-action discounted infinite-horizon MDPs, and propose a new algorithm, called *speedy Q-learning*

(SQL), that addresses the problem of slow convergence of Q-learning. At each time step k, SQL uses two successive estimates of the bellman operator $\mathcal{T}Q_k$ and $\mathcal{T}Q_{k-1}$ to update the action value function Q_k:

$$Q_{k+1} = \alpha_k Q_k + (1-\alpha_k)\left[k\mathcal{T}Q_k - (k-1)\mathcal{T}Q_{k-1} - \epsilon_k\right], \tag{1}$$

that makes its space complexity twice as QL. However, this allows the SQL algorithm to achieve a significantly faster rate of convergence than QL, since it reduces the dependency on the previous Bellman operators from the average $1/(k+1)\sum_{j=0}^{k}\mathcal{T}Q_j$, in the case of QL, to only $\mathcal{T}Q_k + O(1/(k+1))$:

$$\begin{aligned}
Q_{k+1} &= \alpha_k Q_k + (1-\alpha_k)\left[k\mathcal{T}Q_k - (k-1)\mathcal{T}Q_{k-1} - \epsilon_k\right] \\
&= \frac{1}{k+1}\sum_{j=0}^{k}(j\mathcal{T}Q_j - (j-1)\mathcal{T}Q_{j-1} - \epsilon_j) \qquad \text{By induction} \\
&= \frac{1}{k+1}(k\mathcal{T}Q_k - (k-1)\mathcal{T}Q_{k-1} + (k-1)\mathcal{T}Q_{k-1} - (k-2)\mathcal{T}Q_{k-2} + \cdots + \mathcal{T}Q_{-1}) \\
&\quad - \frac{1}{k+1}\sum_{j=0}^{k}\epsilon_j \\
&= \mathcal{T}Q_k + \frac{1}{k+1}(\mathcal{T}Q_{-1} - \mathcal{T}Q_k) - \frac{1}{k+1}\sum_{j=0}^{k}\epsilon_j,
\end{aligned}$$

with the choice of $\alpha_k = 1/(k+1)$. In words, the iterates of SQL, like QL, are expressed in terms of the average estimation error and, therefore, the SQL update rule asymptotically averages out the estimation errors (simulation noise) as well. However, speedy Q-learning has the advantage that at each time step k the iterate Q_{k+1} closely follows (up to a factor of $O(1/(k+1))$ the latest Bellman iterate $\mathcal{T}Q_k$ instead of the average $1/(k+1)\sum_{j=0}^{k}\mathcal{T}Q_j$ in the case of QL. Thus, unlike QL, it does not suffer from the slow convergence due to slow down in the value iteration process (For a detailed comparison of the convergence rate of QL and SQL see Section 3.3).

The idea of using previous estimates of the action-values has already been used in order to improve the performance of QL. A popular algorithm of this kind is Q(λ) (Peng and Williams, 1996; Watkins, 1989), which incorporates the concept of eligibility traces in QL, and has been empirically shown to have a better performance than QL, i.e., Q(0), for suitable values of λ. Another recent work in this direction is *Double Q-learning* (van Hasselt, 2010), which uses two estimators for the action-value function in order to alleviate the over-estimation of action-values in Q-learning. This over-estimation is caused by a positive bias introduced by using the maximum action value as an approximation for the maximum expected action value (van Hasselt, 2010).

The rest of the paper is organized as follows. After introducing the notations used in the paper in Section 2, we present our *Speedy Q-learning* algorithm in Section 3. We first describe the synchronous and asynchronous version of the algorithm in Section 3.1, then state our main theoretical result, i.e., high-probability bounds on the performance of SQL as well as a new lower bound for the sample complexity of reinforcement learning, in

Section 3.2, and finally compare our bound with the previous results on QL and Q-value iteration in Section 3.3. In Section 4, we evaluate the performance of SQL, numerically, on different problem domains. Section 5 contains the detailed proof of the performance bound of the SQL algorithm and a general new lower bound for RL. Finally, we conclude the paper and discuss some future directions in Section 6.

2. Preliminaries

In this section, we introduce some concepts and definitions from the theory of Markov decision processes (MDPs) and stochastic processes that are used throughout the paper. We start by the definition of supremum norm (ℓ_∞ norm). For a real-valued function $g : \mathcal{Y} \mapsto \mathbb{R}$, where $\mathcal{Y}$ is a finite set, the supremum norm of g is defined as $\|g\| \triangleq \max_{y\in\mathcal{Y}} |g(y)|$.

We consider the standard reinforcement learning (RL) framework (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998) in which a learning agent interacts with a stochastic environment and this interaction is modeled as a discrete-time discounted MDP. A discounted MDP is a quintuple $(\mathcal{X}, \mathcal{A}, P, \mathcal{R}, \gamma)$, where $\mathcal{X}$ and $\mathcal{A}$ are the set of states and actions, P is the state transition distribution, $\mathcal{R}$ is the reward function, and $\gamma \in (0, 1)$ is a discount factor. We also denote the horizon of MDP by β defined by $\beta = 1/(1 - \gamma)$. We denote by $P(\cdot|x, a)$ and $r(x, a)$ the probability distribution over the next state and the immediate reward of taking action a at state x, respectively.[1]To keep the representation succinct, we use $\mathcal{Z}$ for the joint state-action space $\mathcal{X} \times \mathcal{A}$.

Assumption A1 (MDP regularity) *We assume $\mathcal{Z}$ is a finite set with the cardinality n. We also assume that the immediate rewards $r(x, a)$ are in the interval* $[0, 1]$.[2]

A policy kernel $\pi(\cdot|\cdot)$ determines the distribution of the control action given the past observations. The policy is called stationary if the distribution of the control action just depends on the last state x. It is deterministic if it assigns a unique action to all states $x \in \mathcal{X}$. The *value* and the *action-value functions* of a policy π, denoted respectively by $V^\pi : \mathcal{X} \mapsto \mathbb{R}$ and $Q^\pi : \mathcal{Z} \mapsto \mathbb{R}$, are defined as the expected sum of discounted rewards that are encountered when the policy π is executed. Given a MDP, the goal is to find a policy that attains the best possible values, $V^*(x) \triangleq \sup_\pi V^\pi(x)$, $\forall x \in \mathcal{X}$. Function V^* is called the *optimal value function*. Similarly the *optimal action-value function* is defined as $Q^*(x, a) = \sup_\pi Q^\pi(x, a)$, $\forall (x, a) \in \mathcal{Z}$. The optimal action-value function Q^* is the unique fixed-point of the *Bellman optimality operator* $\mathcal{T}$ defined as:

$$(\mathcal{T}Q)(x, a) \triangleq r(x, a) + \gamma \sum_{y\in\mathcal{X}} P(y|x, a)(\mathcal{M}Q)(y), \qquad \forall (x, a) \in \mathcal{Z}.$$

1. In this work, for the sake of simplicity in notation, we assume that the reward $r(x, a)$ is a deterministic function for all $(x, a) \in \mathcal{Z}$. However, it is not difficult to extend our results to the case of stochastic reward under some mild assumptions, e.g., boundedness of the absolute value of reward.
2. Our results also hold if the rewards are taken from some interval $[r_{\min}, r_{\max}]$ instead of $[0, 1]$, in which case the bounds scale with the factor $r_{\max} - r_{\min}$.

where the max operator $\mathcal{M}$ over action-value functions is defined as $(\mathcal{M}Q)(y) = \max_{a\in\mathcal{A}} Q(y,a)$, $\forall y \in \mathcal{X}$.[3] Further, We define the cover time of MDP under the policy π as follows:

Definition 1 (Cover time of MDP) *Let t be an integer. Define $\tau_\pi(x,t)$, the cover time of the MDP under the (non-stationary) policy π, as the minimum number of steps required to visit all state-action pairs $z \in \mathcal{Z}$ starting from state $x \in \mathcal{X}$ at time-step $0 \le t$. Also, the state-action space $\mathcal{Z}$ is* covered *by the policy π if all the state-action pairs are performed at least once under the policy π.*

The following assumption which bounds the expected cover time of the MDP guarantees that, asymptotically, all the state-action pairs are visited infinitely many times under the policy π.

Assumption A2 (Boundedness of the the expected cover time) *Let $0 < L < \infty$ and t be a integer. Then, under the policy π for all $x \in \mathcal{X}$ and $t > 0$ assume that:*

$$\mathbb{E}\left(\tau_\pi(x,t)\right) \le L.$$

3. Speedy Q-Learning

In this section, we introduce a new RL method, called speedy Q-Learning (SQL), derive performance bounds for the asynchronous and synchronous variant of SQL, and compare these bounds with similar results on standard Q-learning.

3.1 Algorithms

In this subsection, we introduce two variants of SQL algorithms, the synchronous SQL and asynchronous SQL. In the asynchronous version, at each time step, the action-value of only one state-action pair, the current observed state-action, is updated, while the action-values of rest of the state-action pairs remain unchanged. For the convergence of this instance of the algorithm, it is required that all the states and actions are visited infinitely many times, which makes the analysis slightly more complicated. On the other hand, having access to an oracle which can generate samples anywhere in the state-action space, the algorithm may be also formulated in a synchronous fashion, in which we first generate a next state $y \sim P(\cdot|x,a)$ for each state-action pair (x,a), and then update the action-values of all the state-action pairs using these samples. The pseudo-code of the synchronous and asynchronous SQL are shown in Algorithm 1 and 2, respectively. One can show that asynchronous SQL is reduced to Algorithm 1 when the cover time $\tau_\pi(x,t) = n$ for all $x \in \mathcal{X}$ and $t \ge 0$, in which case the action-values of all state-action pairs are updated in a row. In other words, the Algorithm 1 can be seen as a special case of Algorithm 2. Therefore, in the sequel we only describe the more general asynchronous SQL algorithm.

As it can be seen from the update rule of Algorithm 2, at each time step, Algorithm 2 keeps track of the action-value functions of the two most recent iterations Q_k and Q_{k+1} ,

3. It is important to note that $\mathcal{T}$ is a γ-contraction mapping w.r.t. to the ℓ_∞-norm, i.e., for any pair of action-value functions Q and Q', we have $\|\mathcal{T}Q - \mathcal{T}Q'\| \le \gamma \|Q - Q'\|$ (Bertsekas, 2007b, Chap. 1).

and its main update rule is of the following form for all $(x,a) \in \mathcal{Z}$ at time step t and the iteration round k:

$$Q_{k+1}(X_t, A_t) = (1-\alpha_k)Q_k(X_t, A_t) + \alpha_k\big(k\mathcal{T}_kQ_k(X_t, A_t) - (k-1)\mathcal{T}_kQ_{k-1}(X_t, A_t)\big), \quad (2)$$

where $\mathcal{T}_kQ(X_t, A_t) = 1/|\mathcal{Y}_k| \sum_{y\in\mathcal{Y}_k}[r(X_t, A_t)+\gamma\mathcal{M}Q(y)]$ is the empirical Bellman optimality operator using the set of next state samples $\mathcal{Y}_k$, where $\mathcal{Y}_k$ is a short-hand notation for $\mathcal{Y}_{k,t}(x,a) = \{y|y \sim P(\cdot|x,a)\}$, the set of all samples generated up to time step t in round k conditioned on the state-action pair $(x,a) \in \mathcal{Z}$. At each time step t, Algorithm 2 works as follows: **(i)** It simulates the MDP for one-step, i.e., it draws the action $A_t \in \mathcal{A}$ from the distribution $\pi(\cdot|x)$ and then make a transition to a new state y_k. **(ii)** it updates the two sample estimates $\mathcal{T}_kQ_{k-1}(X_t, A_t)$ and $\mathcal{T}_kQ_k(X_t, A_t)$ of the Bellman optimality operator (for state-action pair (X_t, A_t) using the next state y_k) applied to the estimates Q_{k-1} and Q_k of the action-value function at the previous and current round $k-1$ and k, **(iii)** it updates the action-value function of (X_t, A_t), generates $Q_{k+1}(X_t, A_t)$, using the update rule of Eq. 2, **(iv)** it checks for the condition that if all $(x,a) \in \mathcal{Z}$ have been visited at least one time at iteration k. If the condition is satisfied then we move to next round $k+1$, otherwise k remains unchanged. Finally, **(v)** we replace X_{t+1} with y_k and repeat the whole process until $t \geq T$. Moreover, we let α_k decays linearly with the number of iterations k, i.e., $\alpha_k = 1/(k+1)$, in Algorithm 2. Also, we notice that the update rule $\mathcal{T}_kQ_k(X_t, A_t) := (1-\eta_N)\mathcal{T}_kQ_k(X_t, A_t) + \eta_N(r(X_t, A_t) + \gamma\mathcal{M}Q_k(y_k))$ is being used to make an unbiased estimate of $\mathcal{T}Q_k$ in an incremental fashion.

Algorithm 1: Synchronous speedy Q-learning

Input: Initial action-values Q_0, discount factor γ and number of steps T
$Q_{-1} := Q_0$; `// Initalization`
$t := k := 0$;
repeat `// Main loop`
  $\alpha_k := \dfrac{1}{k+1}$;
  foreach $(x,a) \in \mathcal{Z}$ **do** `// Update the action-values for all` $(x,a) \in \mathcal{Z}$
    Generate the next state sample $y_k \sim P(\cdot|x,a)$;
    $\mathcal{T}_kQ_{k-1}(x,a) := r(x,a) + \gamma\mathcal{M}Q_{k-1}(y_k)$;
    $\mathcal{T}_kQ_k(x,a) := r(x,a) + \gamma\mathcal{M}Q_k(y_k)$;
    $Q_{k+1}(x,a) := (1-\alpha_k)Q_k(x,a) + \alpha_k\big(k\mathcal{T}_kQ_k(x,a) - (k-1)\mathcal{T}_kQ_{k-1}(x,a)\big)$; `// SQL update rule`
    $t := t+1$;
  end
  $k := k+1$;
until $t \geq T$;
return Q_k

3.2 Main Theoretical Results

The main theoretical results of this paper are expressed as high-probability bounds over the performance of the SQL algorithms for both synchronous and asynchronous cases. We also report a new lower bound on the number of transition samples required, for every reinforcement learning algorithm, to achieve an ϵ-optimal estimate of Q^* with probability

Algorithm 2: Asynchronous speedy Q-learning

Input: Initial action-values Q_0, the policy $\pi(\cdot|\cdot)$, discount factor γ, number of step T and the initial state X_0.

$t := k := 0$; `// Initialization`
$\alpha_0 = 1$;
foreach $(x, a) \in \mathcal{Z}$ **do**
  $Q_{-1}(x, a) := Q_0(x, a)$;
  $N_0(x, a) := 0$;
end
repeat `// Main loop`
  Draw the action $A_t \sim \pi(\cdot|X_t)$;
  Generate the next state sample y_k by simulating $P(\cdot|X_t, A_t)$;
  $\eta_N := \dfrac{1}{N_k(X_t, A_t) + 1}$;
  if $N_k(x, a) > 0$ **then**
    $\mathcal{T}_k Q_{k-1}(X_t, A_t) := (1 - \eta_N)\mathcal{T}_k Q_{k-1}(X_t, A_t) + \eta_N(r(X_t, A_t) + \gamma \mathcal{M} Q_{k-1}(y_k))$;
    $\mathcal{T}_k Q_k(X_t, A_t) := (1 - \eta_N)\mathcal{T}_k Q_k(X_t, A_t) + \eta_N(r(X_t, A_t) + \gamma \mathcal{M} Q_k(y_k))$;
  else
    $\mathcal{T}_k Q_{k-1}(X_t, A_t) := r(X_t, A_t) + \gamma \mathcal{M} Q_{k-1}(y_k)$;
    $\mathcal{T}_k Q_k(X_t, A_t) := r(X_t, A_t) + \gamma \mathcal{M} Q_k(y_k)$;
  end
  $Q_{k+1}(X_t, A_t) := (1 - \alpha_k)Q_k(X_t, A_t) + \alpha_k\big(k\mathcal{T}_k Q_k(X_t, A_t) - (k-1)\mathcal{T}_k Q_{k-1}(X_t, A_t)\big)$; `// SQL update rule`
  $N_k(X_t, A_t) := N_k(X_t, A_t) + 1$;
  $X_{t+1} = y_k$;
  if $\min_{(x,a)\in\mathcal{Z}} N_k(x, a) > 0$ **then** `// Check if all` $(x, a) \in \mathcal{Z}$ `have been visited at round` k
    $k := k + 1$;
    $\alpha_k := \dfrac{\alpha_0}{k + 1}$;
    foreach $(x, a) \in \mathcal{Z}$ **do** $N_k(x, a) := 0$;
  end
  $t := t + 1$;
until $t \geq T$;
return Q_k

(w.p.) $1-\delta$. [4] The derived performance bound shows that SQL has a rate of convergence of $T = O(n\beta^4 \log(n/\delta)/\epsilon^2)$, which matches the proposed lower bound of reinforcement learning in terms of n, ϵ and δ. However, the dependency of our bound on the horizon β is worse than than the lower-bound by a factor of $O(\beta^2)$.

Theorem 2 *Let A1 hold, T be a positive integer and Q_T be the estimate of Q^* by Algorithm 1 at time step T. Then,the uniform approximation error*

$$\|Q^* - Q_T\| \leq \beta^2 \left[\frac{\gamma n}{T} + \sqrt{\frac{2n \log \frac{2n}{\delta}}{T}} \right],$$

w.p. at least $1-\delta$.

Theorem 3 *Let A1 and A2 hold and $T > 0$ be the number of time steps. Then, at step T of Algorithm 2 w.p. at least $1-\delta$*

$$\|Q^* - Q_T\| \leq \beta^2 \left[\frac{\gamma e L \log \frac{2}{\delta}}{T} + \sqrt{\frac{2eL \log\left(\frac{2}{\delta}\right) \log \frac{4n}{\delta}}{T}} \right].$$

These results, combined with Borel-Cantelli lemma (Feller, 1968), guarantee that Q_T converges almost surely to Q^* with the rate $\sqrt{1/T}$ for both Algorithm 1 and 2. Further, the following PAC bounds which quantify the number of steps T required to reach the error $\epsilon > 0$ in estimating the optimal action-value function, w.p. $1-\delta$, are immediate consequences of Theorem 2 and 3, respectively.

Corollary 4 (Finite-time PAC bound of synchronous SQL) *Under A1, after*

$$T = \frac{3.74 n\beta^4 \log \frac{2n}{\delta}}{\epsilon^2}$$

steps (transitions) the uniform approximation error of Algorithm 1 $\|Q^* - Q_T\| \leq \epsilon$, w.p. at least $1-\delta$.

Corollary 5 (Finite-time PAC bound of asynchronous SQL) *Under A1 and A2, after*

$$T = \frac{3.74 eL\beta^4 \log\left(\frac{2}{\delta}\right) \log \frac{4n}{\delta}}{\epsilon^2}$$

steps (transitions) the uniform approximation error of Algorithm 2 $\|Q^* - Q_T\| \leq \epsilon$, w.p. at least $1-\delta$.

The following general result provides a new lower bound on the number of transitions T for every reinforcement learning algorithm to achieve ϵ-optimal performance w.p. $1-\delta$ under the assumption that the reinforcement learning algorithm is (ϵ,δ)-correct on the class of MDPs $\mathbb{M}$.[5]

4. We report the proofs in Section 5.
5. This result improves on the state-of-the-art (Strehl et al., 2009) in terms of dependency on β (by a factor of $O(\beta^2)$). Also, our result is more general than those of Strehl et al. (2009) in the sense that It does not require a sequential update of the value functions or following a deterministic policy.

Definition 6 (**(ϵ, δ)-correct reinforcement learning**) *Define $\mathbb{A}$ as the class of all reinforcement learning algorithms which rely on estimating the action-value function Q^*. Let $Q_T^{\mathfrak{A}}$ be the estimate of Q^* using $T \geq 0$ number of transition samples under the algorithm $\mathfrak{A} \in \mathbb{A}$. We then call $\mathfrak{A} \in \mathbb{A}$ (ϵ, δ)-correct on the class of MDPs $\mathbb{M}$, if there exists some $0 < T^* < +\infty$ such that, for all $M \in \mathbb{M}$, $\left\|Q^* - Q_T^{\mathfrak{A}}\right\| \leq \epsilon$ w.p. at least $1 - \delta$ iff $T > T^*$.*

Theorem 7 (Lower bound on the sample complexity of reinforcement learning) *There exists some ϵ_0, γ_0, n_0, c_1, c_2 and a class of MDPs $\mathbb{M}$ such that for all $\epsilon \in (0, \epsilon_0)$, $\gamma \in (\gamma_0, 1)$, $n > n_0$, $\delta \in (0, 1)$ and every (ϵ, δ)-correct reinforcement learning algorithm $\mathfrak{A} \in \mathbb{A}$ on the class of MDPs $\mathbb{M}$ the number of transitions*

$$T > T^* = \frac{n\beta^2}{c_1\epsilon^2} \log \frac{n}{c_2\delta}.$$

3.3 Relation to Existing Results

In this section, we first compare our results for SQL with the existing results on the convergence of standard QL. The comparison indicates that SQL accelerates the convergence of Q-learning, especially for large values of β and small α's. We then compare SQL with batch Q-value iteration (QVI) in terms of sample and computational complexities, i.e., the number of samples and the computational cost required to achieve an ϵ-optimal solution with high probability, as well as space complexity, i.e., the memory required at each step of the algorithm.

3.3.1 A Comparison with the Convergence Rate of Standard Q-Learning

There are not many studies in the literature concerning the convergence rate of incremental model-free RL algorithms such as QL. Szepesvári (1997) has provided the asymptotic convergence rate for QL under the assumption that all the states have the same next state distribution. This result shows that the asymptotic convergence rate of QL with a linearly decaying learning step has exponential dependency on β, i.e. $T = \tilde{O}(1/\epsilon^\beta)$.

Even-Dar and Mansour (2003) throughly investigated the finite-time behavior of synchronous QL for different time scales. Their main result indicates that by using the polynomial learning step $\alpha_k = 1\big/ (k+1)^\omega$, $0.5 < \omega < 1$, the synchronous variant of Q-learning achieves ϵ-optimal performance w.p. at least $1 - \delta$ after

$$T = O\left(n\left[\left(\frac{\beta^4 \log \frac{n\beta}{\delta\epsilon}}{\epsilon^2}\right)^{\frac{1}{w}} + \left(\beta \log \frac{\beta}{\epsilon}\right)^{\frac{1}{1-\omega}}\right]\right), \tag{3}$$

steps, where the time-scale parameter ω may be tuned to achieve the best performance. When $\gamma \approx 1$, $\beta = 1/(1-\gamma)$ becomes the dominant term in the bound of Eq. 3, and thus, the optimized bound is achieved by finding an ω which minimizes the dependency on β. This leads to the optimized bound of $\tilde{O}\big(\beta^5/\epsilon^{2.5}\big)$ with the choice of $\omega = 4/5$. On the other hand, SQL is guaranteed to achieve the same precision in only $O\big(\beta^4/\epsilon^2\big)$ steps. The difference between these two bounds is substantial for large β^2/ϵ.

Even-Dar and Mansour (2003) also proved bounds for the asynchronous variant of Q-learning in the case that the cover time of MDP can be uniformly bounded from above by some finite constant. The extension of their results to the more realistic case that the expected value of the cover-time is bounded by some $L > 0$ (assumption A2) leads to the following PAC bound:

Proposition 8 (Even-Dar and Mansour, 2003) *Under A1 and A2, for all $\omega \in (0.5, 1)$, after*

$$T = O\left(\left[\frac{\left(L\log\frac{1}{\delta}\right)^{1+3\omega}\beta^4\log\frac{n\beta}{\delta\epsilon}}{\epsilon^2}\right]^{\frac{1}{\omega}} + \left[L\beta\log\frac{1}{\delta}\log\frac{\beta}{\epsilon}\right]^{\frac{1}{1-\omega}}\right)$$

steps (transitions) the uniform approximation error of asynchronous QL $\|Q^* - Q_T\| \leq \epsilon$, w.p. at least $1-\delta$.

The dependence on L in this algorithm is of $O(L^{3+\frac{1}{\omega}} + L^{\frac{1}{1-\omega}})$, which leads, with the choice of $\omega \approx 0.77$, to the optimized dependency of $O(L^{4.34})$, whereas asynchronous SQL achieves the same accuracy after just $O(L)$ steps. This result shows that for MDPS with large expected cover-time, i.e., slow-mixing MDPs, asynchronous SQL may converge substantially faster to a near-optimal solution than its Q-learning counterpart.

3.3.2 SQL vs. Q-Value Iteration

Finite sample bounds for both model-based and model-free (Phased Q-learning) QVI have been derived in Kearns and Singh (1999) and Even-Dar et al. (2002). These algorithms can be considered as the batch version of Q-learning. They show that to quantify ϵ-optimal action-value functions with high probability, we need $\tilde{O}\big(n\beta^5/\epsilon^2\big)$ and $\tilde{O}\big(n\beta^4/\epsilon^2\big)$ samples in model-free and model-based QVI, respectively.[6] A comparison between their results and the main result of this paper suggests that the sample complexity of SQL, which is of order $\tilde{O}\big(n\beta^4/\epsilon^2\big)$,[7] is better than model-free QVI in terms of β. Although the sample complexities of SQL and model-based QVI are of the same order, SQL has a significantly better computational and space complexity than model-based QVI: SQL needs only $2n$ memory space, while the space complexity of model-based QVI is given by $\min(\tilde{O}(n\beta^4/\epsilon^2), n(|\mathcal{X}|+1))$ (see Kearns and Singh, 1999). Also, SQL improves the computational complexity by a factor of $\tilde{O}(\beta)$ compared to both model-free and model-based QVI.[8] Table 1 summarizes the comparisons between SQL and the other RL methods discussed in this section.

4. Experiments

In this section, we analyze empirically the effectiveness of the proposed algorithms on different problem domains. We examine the convergence of synchronous SQL (Algorithm 1) as well as the asynchronous SQL (Algorithm 2) on several discrete state-action problems

6. For the sake of simplicity, in this subsection, we ignore the logarithmic dependencies of the bounds.
7. Note that at each round of SQL n new samples are generated. This combined with the result of Corollary 5 deduces the sample complexity of order $\tilde{O}(n\beta^4/\epsilon^2)$.
8. SQL has the sample and computational complexity of a same order since it performs only one Q-value update per sample, the same argument also applies to the standard Q-learning, whereas, in the case of model-based QVI, the algorithm needs to iterate the action-value function of all state-action pairs at least $\tilde{O}(\beta)$ times using Bellman operator, which leads to a computational complexity of order $\tilde{O}(n\beta^5/\epsilon^2)$ given that only $\tilde{O}(n\beta^4/\epsilon^2)$ entries of the estimated transition matrix are non-zero.

Table 1: Comparison between SQL, Q-learning, model-based and model-free Q-value iteration in terms of sample complexity (SC), computational complexity (CC), and space complexity (SPC).

Method	SQL	Q-learning	Model-based QVI	Model-free QVI
SC	$\tilde{O}\left(\frac{n\beta^4}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^{2.5}}\right)$	$\tilde{O}\left(\frac{n\beta^4}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^2}\right)$
CC	$\tilde{O}\left(\frac{n\beta^4}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^{2.5}}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{n\beta^5}{\epsilon^2}\right)$
SPC	$\Theta(n)$	$\Theta(n)$	$\min(\tilde{O}(n\beta^4/\epsilon^2), n(\lvert\mathcal{X}\rvert+1))$	$\Theta(n)$

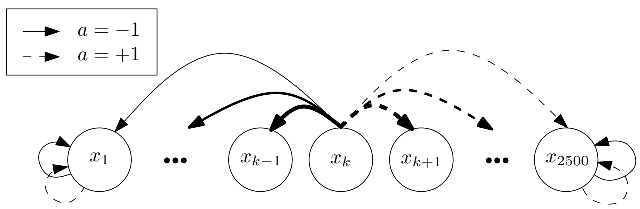

Figure 1: Linear MDP: Illustration of the linear MDP problem. Nodes indicate states. States x_1 and x_{2500} are the two absorbing states and state x_k is an example of interior state. Arrows indicate possible transitions of **these three nodes only**. From x_k any other node is reachable with transition probability (arrow thickness) proportional to the inverse of the distance to x_k (see the text for details).

and compare it with Q-learning (Even-Dar and Mansour, 2003) (QL) and the model-based Q-value iteration (VI) of Kearns and Singh (1999). The source code of all tested algorithms are freely available at `http://www.mbfys.ru.nl/~mazar/Research_Top.html`.

Linear MDP: this problem consists of states $x_k \in \mathcal{X}, k = \{1, 2, \ldots, 2500\}$ arranged in a one-dimensional chain (see Figure 1). There are two possible actions $\mathcal{A} = \{-1, +1\}$ (left/right) and every state is accessible from any other state except for the two ends of the chain, which are absorbing states. A state $x_k \in \mathcal{X}$ is called absorbing if $P(x_k|x_k, a) = 1$ for all $a \in \mathcal{A}$ and $P(x_l|x_k, a) = 0, \forall l \neq k$. Any transition to one of these two states has associated reward 1.

The transition probability for an interior state x_k to any other state x_l is inversely proportional to their distance in the direction of the selected action, and zero for all states corresponding to the opposite direction. Formally, consider the following quantity $n(x_l, a, x_k)$ assigned to all non-absorbing states x_k and to every $(x_l, a) \in \mathcal{Z}$:

$$n(x_l, a, x_k) = \begin{cases} \dfrac{1}{|l-k|} & \text{for } (l-k)a > 0 \\ 0 & \text{otherwise} \end{cases}.$$

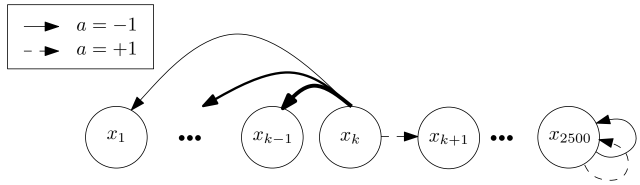

Figure 2: Combination lock: illustration of the combination lock MDP problem. Nodes indicate states. State x_{2500} is the goal (absorbing) state and state x_k is an example of interior state. Arrows indicate possible transitions of **these two nodes only**. From x_k any previous state is reachable with transition probability (arrow thickness) proportional to the inverse of the distance to x_k. Among the future states only x_{k+1} is reachable (arrow dashed).

We can write the transition probabilities as:

$$P(x_l|x_k, a) = \frac{n(x_l, a, x_k)}{\sum\limits_{x_m \in \mathcal{X}} n(x_m, a, x_k)}.$$

Any transition that ends up in one of the interior states has associated reward -1.

The optimal policy corresponding to this problem is to reach the closest absorbing state as soon as possible.

Combination lock: the combination lock problem considered here is a stochastic variant of the reset state space models introduced in Koenig and Simmons (1993), where more than one reset state is possible (see Figure 2).

In our case we consider, as before, a set of states $x_k \in \mathcal{X}, k \in \{1, 2, \ldots, 2500\}$ arranged in a one-dimensional chain and two possible actions $\mathcal{A} = \{-1, +1\}$. In this problem, however, there is only one absorbing state (corresponding to the state *lock-opened*) with associated reward of 1. This state is reached if the all-ones sequence $\{+1, +1, \ldots, +1\}$ is entered correctly. Otherwise, if at some state x_k, $k < 2500$, action -1 is taken, the lock automatically resets to some previous state x_l, $l < k$ randomly (in the original combination lock problem, the reset state is always the initial state x_1).

For every intermediate state, the rewards of actions -1 and $+1$ are set to 0 and -0.01, respectively. The transition probability upon taking the wrong action -1 is, as before, inversely proportional to the distance of the states. That is

$$n(x_k, x_l) = \begin{cases} \dfrac{1}{k-l} & \text{for } l < k \\ 0 & \text{otherwise} \end{cases}, \qquad P(x_l|x_k, 0) = \frac{n(k, l)}{\sum\limits_{x_m \in \mathcal{X}} n(k, m)}.$$

Note that this problem is more difficult than the linear MDP since the goal state is only reachable from one state, x_{2499}.

Grid world: this MDP consists of a grid of 50×50 states. A set of four actions {RIGHT, UP, DOWN, LEFT} is assigned to every state $x \in \mathcal{X}$. The location of each state x of the grid is determined by the coordinates $c_x = (h_x, v_x)$, where h_x and v_x are some integers between 1 and 50. There are 196 absorbing *firewall states* surrounding the grid and another one at the center of grid, for which a reward -1 is assigned. The reward for the firewalls is

$$r(x, a) = -\frac{1}{\|c_x\|_2}, \qquad \forall a \in \mathcal{A}.$$

Also, we assign reward 0 to all of the remaining (non-absorbing) states.

This means that both the top-left absorbing state and the central state have the least possible reward (-1), and that the remaining absorbing states have reward which increases proportionally to the distance to the state in the bottom-right corner (but are always negative).

The transition probabilities are defined in the following way: taking action a from any non-absorbing state x results in a one-step transition in the direction of action a with probability 0.6, and a random move to a state $y \neq x$ with probability inversely proportional to their Euclidean distance $1/\|c_x - c_y\|_2$.

The optimal policy then is to *survive* in the grid as long as possible by avoiding both the absorbing firewalls and the center of the grid. Note that because of the difference between the cost of firewalls, the optimal control prefers the states near the bottom-right corner of the grid, thus avoiding absorbing states with higher cost.

4.0.3 Experimental Setup and Results

We describe now our experimental setting. The convergence properties of SQL are compared with two other algorithms: a Q-learning (Even-Dar and Mansour, 2003) (QL) and the model-based Q-value iteration (QVI) of Kearns and Singh (1999). QVI is a batch reinforcement learning algorithm that first estimates the model using the whole data set and then performs value iteration on the learned model.

All algorithms are evaluated in terms of ℓ_∞-norm performance loss of the action-value function $\|Q^* - Q_T\|$ at time-step T. We choose this performance measure in order to be consistent with the performance measure used in Section 3.2. The optimal action-value function Q^* is computed with high accuracy through value iteration.

We consider QL with polynomial learning step $\alpha_k = 1/(k+1)^\omega$ where $\omega \in \{0.51, 0.6, 0.8\}$ and the linear learning step $\alpha_k = 1/(k+1)$. Note that ω needs to be larger than 0.5, otherwise QL can asymptotically diverge (see Even-Dar and Mansour, 2003, for the proof).

To have a fair comparison of the three algorithms, since each algorithm requires different number of computations per iteration, we fix the total computational budget of the algorithms to the same value for each benchmark. The computation time is constrained to 30 seconds in the case of linear MDP and the combination lock problems. For the grid world, which has twice as many actions as the other benchmarks, the maximum run time is fixed to 60 seconds. We also fix the total number of samples, per state-action, to 1×10^5 samples for all problems and algorithms. Significantly less number of samples leads to a dramatic decrease of the quality of the obtained solutions using all the approaches.

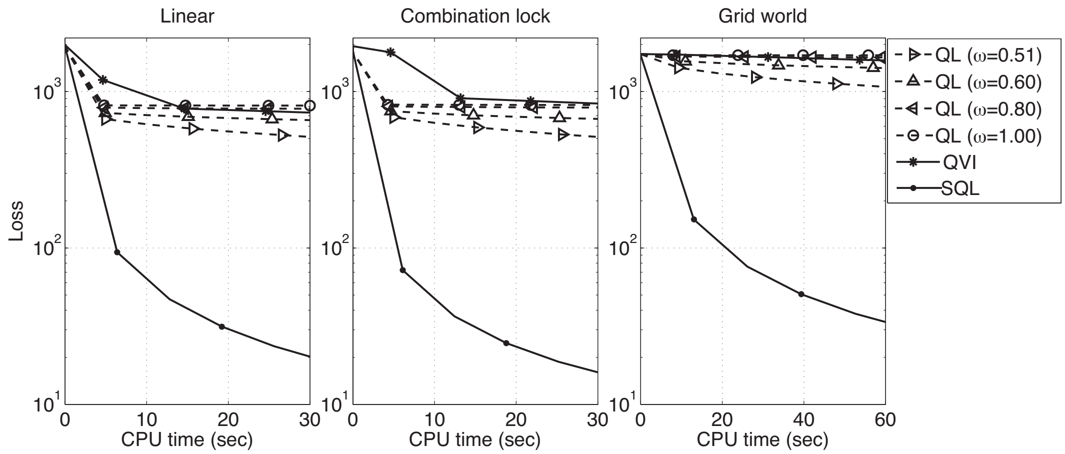

Figure 3: A comparison between SQL, QL and model-based QVI. Each plot compares the performance loss of the policy induced by the algorithms for a different MDP averaged over 50 different runs (see the text for details).

Algorithms were implemented as MEX files (in C++) and ran on a Intel core i5 processor with 8 GB of memory. CPU time was acquired using the system function `times()` which provides process-specific CPU time. Randomization was implemented using `gsl_rng_uniform()` function of the GSL library, which is superior to the standard `rand()`.[9] Sampling time, which is the same for all algorithms, were not included in CPU time. At the beginning of every run **(i)** the action-value functions are randomly initialized in the interval $[-V_{\max}, V_{\max}]$, and **(ii)** a new set of samples is generated from $P(\cdot|x,a)$ for all $(x,a) \in \mathcal{Z}$. The corresponding results are computed after a small fixed amount of iterations.

Figure 3 shows the performance-loss in terms of elapsed CPU time for the three problems and algorithms with the choice of $\beta = 1000$. First, we see that, in all cases, SQL improves the performance very fast, almost by an order of magnitude, in just a few seconds. SQL outperforms both QL and QVI in all the three benchmarks. The minimum and maximum errors are attained for the combination lock problem and the grid world, respectively. We also observe that the difference between the final outcome of SQL and Q-learning (second best method) is very significant, about 30 times in all domains.

Figure 4 and 5 show means and standard deviations of the final performance-loss, respectively, over 50 runs, as a function of the horizon β. We observe that for large values of β, i.e. $\beta \geq 100$, SQL outperforms other methods by more than an order of magnitude in terms of both mean and standard deviation of performance loss. SQL performs slightly worse than QVI for $\beta \leq 10$. However, the loss of QVI scales worse than SQL with β, e.g., for $\beta = 1000$ SQL has an advantage of two order of magnitude over QVI. QL performs for larger values of ω when the horizon β is small, whereas for large values of β smaller ω's are preferable.

9. http://www.gnu.org/s/gsl.

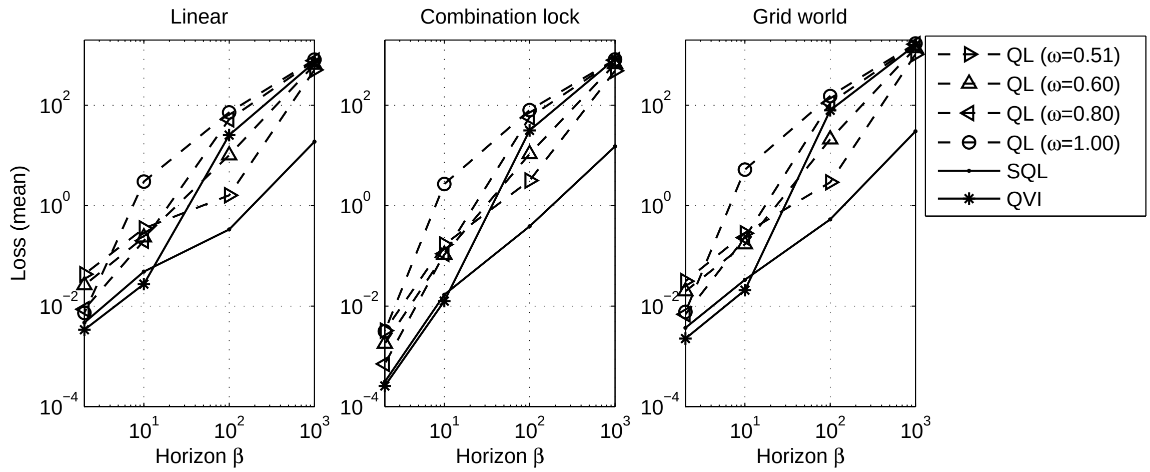

Figure 4: A comparison between SQL, QL and the model-based Q-value iteration (QVI) given a fixed computational and sampling budget. The plot shows means at the end of the simulations for three different algorithm in terms of the horizon of MDP β.

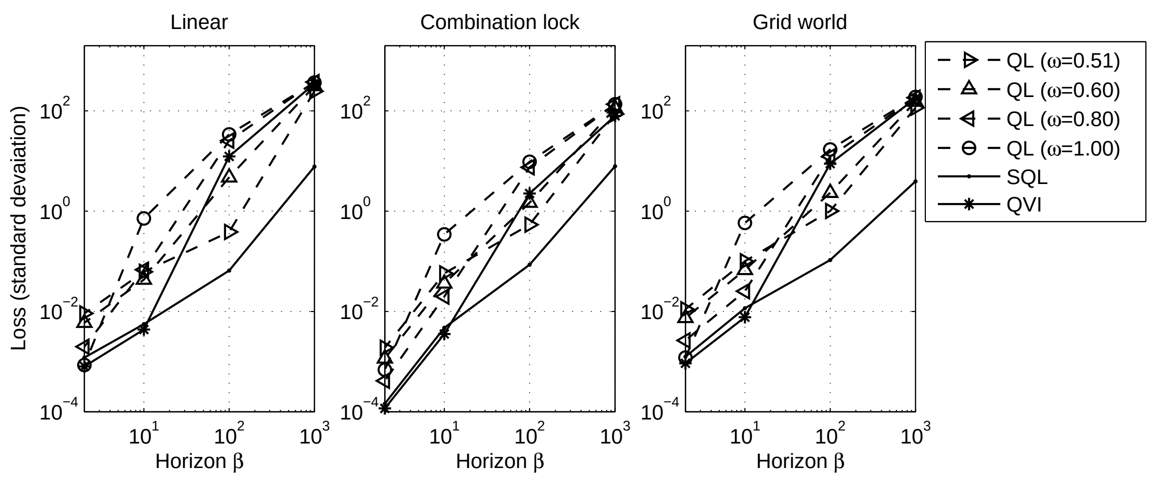

Figure 5: A comparison between SQL, QL and the model-based Q-value iteration (QVI) given a fixed computational and sampling budget. The plot shows variances at the end of the simulations for three different algorithms in terms of the horizon of MDP β.

These results show that, as proved before in Theorem 2, SQL manages to average out the simulation noise caused by sampling and converges, rapidly, to a near optimal solution, which is robust in comparison to other methods. In addition, we can conclude that SQL significantly improves the computational complexity of learning w.r.t. the standard QL and QVI in the three presented benchmarks for our choice of experimental setup.

5. Analysis

In this section, we give some intuition about the convergence of asynchronous variant of SQL and provide the full proof of the finite-time analysis reported in Theorem 2 and Theorem 3. Then, we prove Theorem 7 (RL lower bound) in Subsection 5.1. We start by introducing some notations.

Let $\mathcal{Y}^k$ be the set of all samples drawn at round k of the SQL algorithms and $\mathcal{F}_k$ be the filtration generated by the sequence $\{\mathcal{Y}^0, \mathcal{Y}^1, \mathcal{Y}^2, \dots, \mathcal{Y}^k\}$. We first notice that for all $(x,a) \in \mathcal{Z}$, the update rule of Eq. 2 may be rewritten in the following more compact form:

$$Q_{k+1}(x,a) = (1-\alpha_k)Q_k(x,a) + \alpha_k \mathcal{D}_k[Q_k, Q_{k-1}](x,a),$$

where $\mathcal{D}_k[Q_k, Q_{k-1}](x,a) \triangleq \frac{1}{\alpha_k}[(1-\alpha_k)\mathcal{T}_k Q_k(x,a) - (1-2\alpha_k)\mathcal{T}_k Q_{k-1}(x,a)]$ and $\alpha_k = 1/(k+1)$.

We then define the operator $\mathcal{D}[Q_k, Q_{k-1}]$ as the expected value of the empirical operator $\mathcal{D}_k$ conditioned on $\mathcal{F}_{k-1}$:

$$\begin{aligned}
\mathcal{D}[Q_k, Q_{k-1}](x,a) &\triangleq \mathbb{E}(\, \mathcal{D}_k[Q_k, Q_{k-1}](x,a) |\, \mathcal{F}_{k-1}) \\
&= \mathbb{E}\left(\frac{1-\alpha_k}{\alpha_k} \mathcal{T}_k Q_k(x,a) - \frac{1-2\alpha_k}{\alpha_k} \mathcal{T}_k Q_{k-1}(x,a) \middle|\, \mathcal{F}_{k-1} \right) \\
&= \frac{1-\alpha_k}{\alpha_k} \mathcal{T} Q_k(x,a) - \frac{1-2\alpha_k}{\alpha_k} \mathcal{T} Q_{k-1}(x,a),
\end{aligned}$$

where the last line follows by the fact that, in both Algorithm 1 and 2, $\mathcal{T}_k Q_k(x,a)$ and $\mathcal{T}_k Q_{k-1}(x,a)$ are unbiased empirical estimates of the Bellman optimality operators $\mathcal{T}Q_k(x,a)$ and $\mathcal{T}Q_{k-1}(x,a)$, respectively. Thus, the update rule of SQL can be re-expressed as

$$Q_{k+1}(x,a) = (1-\alpha_k)Q_k(x,a) + \alpha_k \left(\mathcal{D}[Q_k, Q_{k-1}](x,a) - \epsilon_k(x,a) \right), \tag{4}$$

where the estimation error ϵ_k is defined as the difference between the operator $\mathcal{D}[Q_k, Q_{k-1}]$ and its sample estimate $\mathcal{D}_k[Q_k, Q_{k-1}]$ for all $(x,a) \in \mathcal{Z}$:

$$\epsilon_k(x,a) \triangleq \mathcal{D}[Q_k, Q_{k-1}](x,a) - \mathcal{D}_k[Q_k, Q_{k-1}](x,a).$$

We have the property that $\mathbb{E}[\epsilon_k(x,a)|\mathcal{F}_{k-1}] = 0$ which means that for all $(x,a) \in \mathcal{Z}$ the sequence of estimation error $\{\epsilon_1(x,a), \epsilon_2(x,a), \dots, \epsilon_k(x,a)\}$ is a martingale difference sequence w.r.t. the filtration $\mathcal{F}_k$. Finally, we define the martingale $E_k(x,a)$ to be the sum of the estimation errors:

$$E_k(x,a) \triangleq \sum_{j=0}^{k} \epsilon_j(x,a), \qquad \forall (x,a) \in \mathcal{Z}. \tag{5}$$

The following steps lead to the proof of Theorem 2 and Theorem 3 **(i)** Lemma 9 shows the stability of SQL (i.e., the sequence of Q_k stays bounded). **(ii)** Lemma 10 states the key property that the iterate Q_{k+1} is close to the Bellman operator $\mathcal{T}$ applied to the previous iterate Q_k plus an estimation error term of order E_k/k. **(iii)** By induction, Lemma 11 provides a performance bound $\|Q^* - Q_k\|$ in terms of a discounted sum of the cumulative estimation errors $\{E_j\}_{j=0:k-1}$. The above-mentioned results hold for both Algorithm 1 and

Algorithm 2. Subsequently, **(iv)** we concentrate on proving Theorem 2 by making use of a maximal Azuma's inequality, stated in Lemma 13. **(v)** We then extend this result for the case of asynchronous SQL by making use of the result of Lemma 14.

For simplicity of the notations, we remove the dependence on (x,a), e.g., writing Q for $Q(x,a)$ and E_k for $E_k(x,a)$, when there is no possible confusion. Also, we notice that, for all $k \geq 0$, the following relations hold between α_k and α_{k+1} in Algorithm 1 and Algorithm 2:

$$\alpha_{k+1} = \frac{\alpha_k}{\alpha_k + 1} \qquad \text{and} \qquad \alpha_k = \frac{\alpha_{k+1}}{1-\alpha_{k+1}}.$$

Lemma 9 (Stability of SQL) *Let A1 hold and assume that the initial action-value function $Q_0 = Q_{-1}$ is uniformly bounded by $V_{\max} = \beta$, then we have*

$$\|Q_k\| \leq V_{\max}, \quad \|\epsilon_k\| \leq 2V_{\max}, \quad \textit{and} \quad \|\mathcal{D}_k[Q_k, Q_{k-1}]\| \leq V_{\max} \quad \forall k \geq 0.$$

Proof We first prove that $\|\mathcal{D}_k[Q_k, Q_{k-1}]\| \leq V_{\max}$ by induction. For $k=0$ we have:

$$\|\mathcal{D}_0[Q_0, Q_{-1}]\| = \|\mathcal{T}_0 Q_{-1}\| \leq \|r\| + \gamma\|\mathcal{M}Q_{-1}\| \leq R_{\max} + \gamma V_{\max} = V_{\max}.$$

Now for any $k \geq 0$, let us assume that the bound $\|\mathcal{D}_k[Q_k, Q_{k-1}]\| \leq V_{\max}$ holds. Thus

$$\begin{aligned}
\|\mathcal{D}_{k+1}[Q_{k+1}, Q_k]\| &= \left\| \tfrac{1-\alpha_{k+1}}{\alpha_{k+1}} \mathcal{T}_{k+1} Q_{k+1} - \tfrac{1-2\alpha_{k+1}}{\alpha_{k+1}} \mathcal{T}_{k+1} Q_k \right\| \\
&\left\| \left[\tfrac{1-\alpha_{k+1}}{\alpha_{k+1}} - \tfrac{1-2\alpha_{k+1}}{\alpha_{k+1}} \right] r \right\| + \gamma \left\| \tfrac{1-\alpha_{k+1}}{\alpha_{k+1}} \mathcal{M} Q_{k+1} - \tfrac{1-2\alpha_{k+1}}{\alpha_{k+1}} \mathcal{M} Q_k \right\| \\
&\leq \|r\| + \gamma \left\| \tfrac{1-\alpha_{k+1}}{\alpha_{k+1}} \mathcal{M}((1-\alpha_k)Q_k + \alpha_k \mathcal{D}_k[Q_k, Q_{k-1}]) - \tfrac{1-2\alpha_{k+1}}{\alpha_{k+1}} \mathcal{M} Q_k \right\| \\
&= \|r\| + \gamma \left\| \tfrac{1-\frac{\alpha_k}{\alpha_k+1}}{\frac{\alpha_k}{\alpha_k+1}} \mathcal{M}((1-\alpha_k)Q_k + \alpha_k \mathcal{D}_k[Q_k, Q_{k-1}]) - \tfrac{1-2\frac{\alpha_k}{\alpha_k+1}}{\frac{\alpha_k}{\alpha_k+1}} \mathcal{M} Q_k \right\| \\
&= \|r\| + \gamma \left\| \mathcal{M}\Big(\tfrac{1-\alpha_k}{\alpha_k} Q_k + \mathcal{D}_k[Q_k, Q_{k-1}] \Big) - \tfrac{1-\alpha_k}{\alpha_k} \mathcal{M} Q_k \right\| \\
&\leq \|r\| + \gamma \left\| \mathcal{M}\Big(\tfrac{1-\alpha_k}{\alpha_k} Q_k + \mathcal{D}_k[Q_k, Q_{k-1}] - \tfrac{1-\alpha_k}{\alpha_k} Q_k \Big) \right\| \\
&\leq \|r\| + \gamma \|\mathcal{D}_k[Q_k, Q_{k-1}]\| \leq R_{\max} + \gamma V_{\max} = V_{\max},
\end{aligned}$$

and by induction, we deduce that for all $k \geq 0$, $\|\mathcal{D}_k[Q_k, Q_{k-1}]\| \leq V_{\max}$.

Now the bound on ϵ_k follows from $\|\epsilon_k\| = \|\mathbb{E}(\mathcal{D}_k[Q_k, Q_{k-1}]|\mathcal{F}_{k-1}) - \mathcal{D}_k[Q_k, Q_{k-1}]\| \leq 2V_{\max}$, and the bound $\|Q_k\| \leq V_{\max}$ is deduced by noticing that $Q_k = \frac{1}{k}\sum_{j=0}^{k-1} \mathcal{D}_j[Q_j, Q_{j-1}]$.
■

The next lemma shows that Q_k is close to $\mathcal{T}Q_{k-1}$, up to a $O(\frac{1}{k})$ term minus the cumulative estimation error $\frac{1}{k}E_{k-1}$.

Lemma 10 *Under A1, for any $k \geq 1$:*

$$Q_k = \mathcal{T}Q_{k-1} + \frac{1}{k}\left(\mathcal{T}Q_0 - \mathcal{T}Q_{k-1} - E_{k-1}\right). \tag{6}$$

Proof We prove this result by induction. The result holds for $k = 1$, where (6) reduces to (4). We now show that if the property (6) holds for $k \geq 1$ then it also holds for $k + 1$. Assume that (6) holds for k. Then, from (4) we have:

$$
\begin{aligned}
Q_{k+1} &= (1-\alpha_k)Q_k + \alpha_k \left[\frac{1-\alpha_k}{\alpha_k}\mathcal{T}Q_k - \frac{1-2\alpha_k}{\alpha_k}\mathcal{T}Q_{k-1} - \epsilon_k\right] \\
&= (1-\alpha_k)\left(\mathcal{T}Q_{k-1} + \alpha_{k-1}\left(\mathcal{T}Q_0 - \mathcal{T}Q_{k-1} - E_{k-1}\right)\right) \\
&\quad + \alpha_k \left[\frac{1-\alpha_k}{\alpha_k}\mathcal{T}Q_k - \frac{1-2\alpha_k}{\alpha_k}\mathcal{T}Q_{k-1} - \epsilon_k\right] \\
&= (1-\alpha_k)\left[\mathcal{T}Q_{k-1} + \frac{\alpha_k}{1-\alpha_k}\left(\mathcal{T}Q_0 - \mathcal{T}Q_{k-1} - E_{k-1}\right)\right] \\
&\quad + (1-\alpha_k)\mathcal{T}Q_k - (1-2\alpha_k)\mathcal{T}Q_{k-1} - \alpha_k\epsilon_k \\
&= (1-2\alpha_k)\mathcal{T}Q_{k-1} + \alpha_k\left(\mathcal{T}Q_0 - E_{k-1}\right) + (1-\alpha_k)\mathcal{T}Q_k - (1-2\alpha_k)\mathcal{T}Q_{k-1} - \alpha_k\epsilon_k \\
&= (1-\alpha_k)\mathcal{T}Q_k + \alpha_k\left(\mathcal{T}Q_0 - E_{k-1} - \epsilon_k\right) = \mathcal{T}Q_k + \alpha_k\left(\mathcal{T}Q_0 - \mathcal{T}Q_k - E_k\right) \\
&= \mathcal{T}Q_k + \frac{1}{k+1}\left(\mathcal{T}Q_0 - \mathcal{T}Q_k - E_k\right).
\end{aligned}
$$

Thus (6) holds for $k + 1$, and is thus true for all $k \geq 1$. ∎

Now we bound the difference between Q^* and Q_k in terms of the discounted sum of the cumulative estimation errors $\{E_0, E_1, \ldots, E_{k-1}\}$.

Lemma 11 (Error Propagation of SQL) *Let A1 hold and assume that the initial action-value function $Q_0 = Q_{-1}$ is uniformly bounded by $V_{\max} = \beta$, then for all $k \geq 1$, we have:*

$$
\|Q^* - Q_k\| \leq \frac{1}{k}\left[2\gamma\beta^2 + \sum_{j=1}^{k}\gamma^{k-j}\|E_{j-1}\|\right] \tag{7}
$$

$$
\leq \frac{\beta}{k}\left[2\gamma\beta + \max_{j=1:k}\|E_{j-1}\|\right]. \tag{8}
$$

Proof

We first notice that $\sum_{j=1}^{k}\gamma^{k-j}\|E_{j-1}\| \leq \beta \max_{j=1:k}\|E_{j-1}\|$ for any sequence of cumulative errors errors $\{E_0, E_1, E_2, \ldots, E_{k-1}\}$. Therefore, we only need to prove (7) and (8) follows.

Again we prove this lemma by induction. The result holds for $k = 1$ as:

$$
\|Q^* - Q_1\| = \|\mathcal{T}Q^* - \mathcal{T}Q_0 - \epsilon_0\| \leq \|Q^* - Q_0\| + \|\epsilon_0\| \leq 2\gamma V_{\max} + \|\epsilon_0\| \leq 2\gamma\beta^2 + \|E_0\|.
$$

We now show that if the bound holds for k, then it also holds for $k + 1$. Thus, assume that (7) holds for k. By using Lemma 10:

$$
\begin{aligned}
\|Q^* - Q_{k+1}\| &= \left\|Q^* - \mathcal{T}Q_k - \frac{1}{k+1}\left(\mathcal{T}Q_0 - \mathcal{T}Q_k - E_k\right)\right\| \\
&\leq \|\alpha_k(\mathcal{T}Q^* - \mathcal{T}Q_0) + (1-\alpha_k)(\mathcal{T}Q^* - \mathcal{T}Q_k)\| + \alpha_k\|E_k\| \\
&\leq \alpha_k\|\mathcal{T}Q^* - \mathcal{T}Q_0\| + (1-\alpha_k)\|\mathcal{T}Q^* - \mathcal{T}Q_k\| + \alpha_k\|E_k\| \\
&\leq 2\gamma\alpha_k V_{\max} + \gamma(1-\alpha_k)\|Q^* - Q_k\| + \alpha_k\|E_k\|.
\end{aligned}
$$

By plugging (7) into (9) we then deduce:

$$\begin{aligned}
\left\|Q^* - Q_{k+1}\right\| &\leq 2\alpha_k\gamma V_{\max} + \gamma(1-\alpha_k)\alpha_{k-1}\left[2\gamma\beta^2 + \sum_{j=1}^{k}\gamma^{k-j}\left\|E_{j-1}\right\|\right] + \alpha_k\left\|E_k\right\| \\
&\leq 2\alpha_k\gamma\beta + \gamma(1-\alpha_k)\frac{\alpha_k}{1-\alpha_k}\left[2\gamma\beta^2 + \sum_{j=1}^{k}\gamma^{k-j}\left\|E_{j-1}\right\|\right] + \alpha_k\left\|E_k\right\| \\
&\leq 2\alpha_k\gamma\beta + \gamma\alpha_k 2\gamma\beta^2 + \gamma\alpha_k\sum_{j=1}^{k}\gamma^{k-j}\left\|E_{j-1}\right\| + \alpha_k\left\|E_k\right\| \\
&= \frac{1}{k+1}\left[2\gamma\beta^2 + \sum_{j=1}^{k+1}\gamma^{k+1-j}\left\|E_{j-1}\right\|\right].
\end{aligned}$$

Thus, the result holds for $k+1$ thus for all $k \geq 1$ by induction. ■

In the next lemma, we prove a high probability bound over the estimation error term of Lemma 11

Lemma 12 *Let A1 hold and assume that the initial action-value function $Q_0 = Q_{-1}$ is uniformly bounded by $V_{\max} = \beta$ then for all $0 < \delta < 1$ and $k \geq 1$, we have:*

$$\max_{j=1:k}\left\|E_{j-1}\right\| \leq \beta\sqrt{8k\log\frac{2n}{\delta}}, \qquad \textit{w.p. } 1-\delta. \tag{9}$$

Proof We start by providing a high probability bound for $\max_{1\leq j\leq k}|E_{j-1}(x,a)|$ for a given (x,a). First notice that:

$$\begin{aligned}
\mathbb{P}\left(\max_{1\leq j\leq k}|E_{j-1}(x,a)| > \epsilon\right) &= \mathbb{P}\left(\max\left[\max_{1\leq j\leq k}(E_{j-1}(x,a)), \max_{1\leq j\leq k}(-E_{j-1}(x,a))\right] > \epsilon\right) \\
&= \mathbb{P}\left(\left\{\max_{1\leq j\leq k}(E_{j-1}(x,a)) > \epsilon\right\}\bigcup\left\{\max_{1\leq j\leq k}(-E_{j-1}(x,a)) > \epsilon\right\}\right) \\
&\leq \mathbb{P}\left(\max_{1\leq j\leq k}(E_{j-1}(x,a)) > \epsilon\right) + \mathbb{P}\left(\max_{1\leq j\leq k}(-E_{j-1}(x,a)) > \epsilon\right),
\end{aligned} \tag{10}$$

and each term is now bounded by using a maximal Azuma-Hoeffding's inequality, reminded now (see e.g., Cesa-Bianchi and Lugosi (2006)).

Lemma 13 (Maximal Hoeffding-Azuma's Inequality) *Let $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ be a martingale difference sequence w.r.t. a sequence of random variables $\{X_1, X_2, \ldots, X_k\}$ (i.e., $\mathbb{E}(V_{j+1}|X_1, \ldots X_j) = 0$ for all $0 < j \leq k$) such that $\mathcal{V}$ is uniformly bounded by $L > 0$. If we define $S_k = \sum_{i=1}^{k} V_i$, then for any $\epsilon > 0$, we have:*

$$\mathbb{P}\left(\max_{1\leq j\leq k} S_j > \epsilon\right) \leq \exp\left(\frac{-\epsilon^2}{2kL^2}\right).$$

As mentioned earlier, the sequence of random variables $\{\epsilon_0(x,a), \epsilon_1(x,a), \cdots, \epsilon_k(x,a)\}$ is a martingale difference sequence w.r.t. the filtration $\mathcal{F}_k$ (generated by the random samples $\{y_0, y_1, \ldots, y_k\}(x,a)$ for all (x,a)), i.e., $\mathbb{E}[\epsilon_k(x,a)|\mathcal{F}_{k-1}] = 0$. It follows from Lemma 13 that for any $\epsilon > 0$ we have:

$$\begin{aligned} \mathbb{P}\left(\max_{1\le j\le k}(E_{j-1}(x,a)) > \epsilon\right) &\le \exp\left(\frac{-\epsilon^2}{8kV_{\max}^2}\right) \\ \mathbb{P}\left(\max_{1\le j\le k}(-E_{j-1}(x,a)) > \epsilon\right) &\le \exp\left(\frac{-\epsilon^2}{8kV_{\max}^2}\right). \end{aligned} \tag{11}$$

By combining (11) with (10) we deduce that $\mathbb{P}\left(\max_{1\le j\le k}|E_{j-1}(x,a)| > \epsilon\right) \le 2\exp\left(\frac{-\epsilon^2}{8kV_{\max}^2}\right)$, and by a union bound over the state-action space, we deduce that

$$\mathbb{P}\left(\max_{1\le j\le k}\|E_{j-1}\| > \epsilon\right) \le 2n\exp\left(\frac{-\epsilon^2}{8kV_{\max}^2}\right). \tag{12}$$

This bound can be rewritten, for any $\delta > 0$, as:

$$\mathbb{P}\left(\max_{1\le k\le T}\|E_{j-1}\| \le V_{\max}\sqrt{8k\log\frac{2n}{\delta}}\right) \ge 1-\delta,$$

which completes the proof. ∎

The result of Theorem 2 then follows by plugging (9) into (8) and taking into account that after T steps of Algorithm 1 we have $T \le n(k+1)$. For the proof of Theorem 3 we rely on the following result which bounds the number of steps required to visit all states-actions k times with high probability :

Lemma 14 *Under A2, from any initial state x_0 and for any integer $k > 0$, after a run of $T = keL\log\frac{1}{\delta}$ steps the state-action space $\mathcal{Z}$ is covered at least k times under the policy π w.p. at least $1-\delta$.*

Proof

We begin by defining the random event $\mathcal{Q}_k$ as the number of steps required to cover the whole MDP for k times starting from any state $x_0 \in \mathcal{X}$ at any time $t > 0$. We then bound $\mathcal{Q}_k$ for any x_0 in high probability using Markov inequality (Feller, 1968):

$$\mathbb{P}(\mathcal{Q}_k > ekL) \le \frac{\mathbb{E}(\mathcal{Q}_k)}{ekL} \le \frac{k\sup_{t>0}\max_{x\in\mathcal{X}}\mathbb{E}(\tau_\pi(x,t))}{ekL} \le \frac{kL}{ekL} = \frac{1}{e}.$$

In words after a run of length ekL the probability that the entire state-action space is not covered for at least k times is less than $\frac{1}{e}$. The fact that the bound holds for any initial state and time implies that after $m > 0$ intervals of length ekL the chance of not covering $\mathcal{Z}$ for k times is less than $\frac{1}{e^m}$:

$$\mathbb{P}(\mathcal{Q}_k > mekL) \le \frac{1}{e^m}.$$

By the choice of $m = \log \frac{1}{\delta}$ we deduce:

$$\mathbb{P}\left(\mathcal{Q}_k > keL \log \frac{1}{\delta}\right) \leq \delta.$$

The bound can be then rewritten as follows:

$$\mathbb{P}\left(\mathcal{Q}_k \leq keL \log \frac{1}{\delta}\right) \geq 1 - \delta,$$

which concludes the proof.

■

Plugging the results of Lemma 14 and Lemma 12 into (8) concludes the proof of theorem 3.

5.1 Proof of Theorem 7

In this subsection, we provide the full proof of Theorem 7. In our analysis, we rely on the likelihood-ratio method, which has been perviously used to prove a lower-bound for multi-armed bandits, (see Mannor and Tsitsiklis, 2004), and extend this approach for reinforcement learning in Markovian decision problems. We also make use of some technical lemmas in (Strehl et al., 2009). We begin by defining a class of MDPs for which the proposed lower-bound holds. The class of MDPs $\mathbb{M}$ (see Figure 6) is defined as a set of all MDPs with $n = 3n_1 \geq 3$ state-action pairs, in which the set of state-action pairs $\mathcal{Z}$ is made up of three smaller sets $\mathcal{Z}_0$, $\mathcal{Z}_1$ and $\mathcal{Z}_2$ each of size n_1. We make the assumption that, for all $M \in \mathbb{M}$, both $\mathcal{Z}_1$ and $\mathcal{Z}_2$ consist of only absorbing states, see Section 4 for the definition of absorbing states. We also assume that for all $M \in \mathbb{M}$, every entry z_0^l, $1 \leq l \leq n_1$, is connected to only two other state-action pairs $z_1^l \in \mathcal{Z}_1$ and $z_2^l \in \mathcal{Z}_2$, such that the probability of moving from z_0^l to z_1^l is given by $p_M(z_0^l)$, for all $1 \leq l \leq n_1$. Also, we assume that the instant reward $r(z)$ is set to 1 for all $z \in \mathcal{Z}_1$ and 0 for the rest of the state-actions. One can then solve the Bellman equation, For all $z \in \mathcal{Z}$, in close-form and compute the optimal action-value function Q^* as follows:

$$Q^*(z) = \begin{cases} \gamma\beta p_M(z) & z \in \mathcal{Z}_0 \\ \beta & z \in \mathcal{Z}_1 \\ 0 & z \in \mathcal{Z}_2 \end{cases}.$$

Here, we only consider a subset of 2 MDPs in the class $\mathbb{M}$ denoted by $\mathbb{M}^* = \{M_0, M_1\}$. For all $M_m \in \mathbb{M}^*$ and $z \in \mathcal{Z}_1$, the state transition probability $p_M(z)$ is given by:

$$p_M(z) = \begin{cases} \dfrac{1 + 8\epsilon/\beta}{2} & M = M_0 \\ \dfrac{1}{2} & M = M_1 \end{cases}.$$

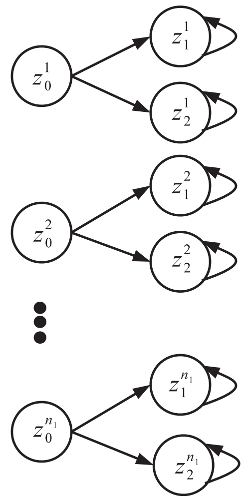

Figure 6: Illustration of the class of MDP considered for the lower-bounds. Nodes indicate state-(action) pairs. Arrows indicate possible transitions of these nodes. (see the text for details).

In the sequel, we restrict ourselves to the case that the only unknown factor is the state-transition probability p_M. Therefore, the RL task is to learn only those value functions which depend on p_M, i.e., $Q^*(z)$ for all $z \in \mathcal{Z}_0$. We also denote by $\mathbb{E}_m$ ad $\mathbb{P}_m$ the expectation and the probability under the model M_m in the rest of this section.

We follow the following steps in the proof: **(i)** we prove a lower-bound for the sample-complexity of each state-action pair $z \in \mathcal{Z}_0$ on the class of MDP $\mathbb{M}^*$. **(ii)** we then make use of the fact that the estimates of $Q^*(z)$ for different $z \in \mathcal{Z}_0$ are independent of each others to combine these bounds and prove the result of Theorem 7.

We begin by proving that for all RL algorithms $\mathfrak{A} \in \mathbb{A}$, there exists an MDP $M_m \in \mathbb{M}^*$ and constants $c_0 > 0$ and $c > 0$ such that for all $\epsilon < \epsilon_0 = 1/8$:

$$\mathbb{P}_m(|Q^*(z) - Q^{\mathfrak{A}}_{t_z}(z)|) > \epsilon) > \theta_z/c_0, \tag{13}$$

where $\theta_z \triangleq \exp\left(-ct_z\epsilon^2/\beta^2\right)$ and $Q^{\mathfrak{A}}_{t_z}(z)$ is an empirical estimate of the action-value function $Q^*(z)$ by the RL algorithm $\mathfrak{A}$ using $t_z > 0$ transition samples from the state-action pair $z \in \mathcal{Z}_0$. To prove this lower bound we make use of a contradiction argument, i.e., we assume that there exists an algorithm $\mathfrak{A} \in \mathbb{A}$ for which:

$$\mathbb{P}_m(|Q^*(z)-Q^{\mathfrak{A}}_{t_z}(z)|) > \epsilon) \leq \theta_z/c_0, \quad \text{or equivalently} \quad \mathbb{P}_m(|Q^*(z)-Q^{\mathfrak{A}}_{t_z}(z)|) \leq \epsilon) \geq 1-\theta_z/c_0,$$

for all $M_m \in \mathbb{M}^*$ and show that this assumption leads to a contradiction. To prove this we need to, first, introduce some notations: We define the event $\Omega_1(z) \triangleq \{|Q^*_0 - Q^{\mathfrak{A}}_t(z)| \leq \epsilon\}$ for all $z \in \mathcal{Z}$, where $Q^*_0 \triangleq \gamma/(2(1-\gamma))$ is the optimal action-value function for all $z \in \mathcal{Z}_0$ under the MDP M_0. We then define $K_t \triangleq r_1 + r_2 + \cdots + r_t$ as the sum of rewards of trying $z \in \mathcal{Z}_0$ for t times. We also introduce the event $\Omega_2(z)$, for all $z \in \mathcal{Z}_0$:

$$\Omega_2(z) \triangleq \left\{ \max_{1\leq t\leq t_z} \left(\frac{1}{2}t - K_t\right) \leq \sqrt{\frac{t_z \log \frac{1}{\theta_z}}{2}} \right\}.$$

We also define $\mathcal{S}(z) \triangleq \Omega_1(z) \cap \Omega_2(z)$. The following lemmas are also required for our analysis.[10]

Lemma 15 (Mannor and Tsitsiklis, 2004) *For all $\epsilon < \epsilon_0 = 1/8$ and $\gamma > \gamma_0 = 1/2$ we have:*

$$\mathbb{P}_0(\Omega_2(z)) > \frac{3}{4}, \qquad \forall z \in \mathcal{Z}.$$

Lemma 16 *if $0 \leq x \leq \sqrt{2}/2$ and $y \geq 0$ then*

$$(1-x)^y \geq \exp(-dxy),$$

where $d = 1.78$.

Now, we proceed with the proof of the lower-bound. By the assumption that $\mathbb{P}_m(|Q^*(z) - Q^{\mathfrak{A}}_{t_z}(z)|) > \epsilon) \leq \theta_z/c_0$ for all $M_m \in \mathbb{M}^*$, we have $\mathbb{P}_0(\Omega_1(z)) \geq 1 - \theta_z/c_0 \geq 1 - 1/c_0 = 3/4$ with the choice of $c_0 = 4$. This combined with Lemma 15 implies that $\mathbb{P}_0(\mathcal{S}(z)) > 1/2$, for all $z \in \mathcal{Z}$. Based on this result, we prove the following lemma:

Lemma 17 *For all $z \in \mathcal{Z}_0$: $\mathbb{P}_1(\Omega(z)) > \theta_z/c_0$.*

Proof We define W as the history of all the outcomes of trying z for t_z times and the likelihood function $L_m(w)$ for all $M_m \in \mathbb{M}^*$ as:

$$L_m(w) \triangleq \mathbb{P}_m(W = w),$$

for every possible history w and $M_m \in \mathbb{M}^*$. This function can be used to define a random variable $L(W_m)$. The likelihood ratio of the event w between two MDPs M_1 and M_0 can be written as:

$$\begin{aligned} \frac{L_1(W)}{L_0(W)} &= \frac{(\frac{1}{2}+4\epsilon/\beta)^{k_z}(\frac{1}{2}-4\epsilon/\beta)^{t_z}}{\frac{1}{2}^{t_z+k_z}} \\ &= (1+8\epsilon/\beta)^{k_z}(1-8\epsilon/\beta)^{k_z}(1-8\epsilon/\beta)^{t_z-2k_z} \\ &= (1-64\epsilon^2/\beta^2)^{k_z}(1-8\epsilon/\beta)^{t_z-2k_z}, \end{aligned} \tag{14}$$

where k_z is a short-hand notation for K_{t_z}.

We then determine the lower bounds of the terms in the RHS of (14), when the event $\mathcal{S} = \Omega_1 \cap \Omega_2$ occurs:

If $\mathcal{S}(z)$ occurs, then $\Omega_1(z)$ occurs, and we have $k \leq t$. Therefore, we deduce:

$$(1-64\epsilon^2/\beta^2)^{k_z} \geq (1-64\epsilon^2/\beta^2)^{t_z} \geq e^{-64t_z d\epsilon^2/\beta^2} = e^{-64\frac{d}{c}\log(\frac{1}{\theta_z})} = \theta_z^{64\frac{d}{c}}. \tag{15}$$

10. For the proofs see Mannor and Tsitsiklis (2004).

Likewise, if the event $\mathcal{S}(z)$ occurs, then $\Omega_2(z)$ occurs which leads to:

$$t_z - 2k_z \leq 2\sqrt{\frac{t_z \log(1/\theta_z)}{2}},$$

which implies:

$$(1-8\epsilon/\beta)^{t_z-2k_z} \geq (1-8\epsilon/\beta)^{2\sqrt{\frac{t_z\log(1/\theta_z)}{2}}} \geq e^{-16d\epsilon/\beta\sqrt{\frac{t_z\log(1/\theta_z)}{2}}} = e^{\frac{-16d}{\sqrt{2c}}\log(\frac{1}{\theta_z})} = \theta_z^{\frac{16d}{\sqrt{2c}}}. \tag{16}$$

By plugging (15) and (16) into (14), we deduce:

$$\frac{L_1(W)}{L_0(W)} \geq \theta^{16d(\frac{1}{\sqrt{2c}}+\frac{4}{c})}.$$

Then, by the choice of $c = 700$, we obtain:

$$\frac{L_1(W)}{L_0(W)}\mathbf{1}_{\mathcal{S}} \geq 2\theta_z/c_0\mathbf{1}_{\mathcal{S}},$$

where $\mathbf{1}_{\mathcal{S}}$ is the indicator function of the event $\mathcal{S}(z)$. Then by a change of measure we deduce:

$$\mathbb{P}_1(\Omega_1(z)) \geq \mathbb{P}_1(\mathcal{S}(z)) = \mathbb{E}_1[\mathbf{1}_{\mathcal{S}}] = \mathbb{E}_0\left(\frac{L_1(W)}{L_0(W)}\mathbf{1}_{\mathcal{S}}\right) \geq \mathbb{E}_0\left[2\theta_z/c_0\mathbf{1}_{\mathcal{S}}\right] = 2\theta_z/c_0\mathbb{P}_0(\mathcal{S}) > \theta_z/c_0,$$

where we make use of the fact that $\mathbb{P}_0(\mathcal{S}(z)) > \frac{1}{2}$.

■

Lemma 17 states that, for all $z \in \mathcal{Z}_0$, $\mathbb{P}_1(\Omega_1(z)) > \theta_z/c$, which violates the assumption that, for all $z \in \mathcal{Z}_0$ and $M_m \in \mathbb{M}^*$, $\mathbb{P}_m(|Q^*(z) - Q^{\mathfrak{A}}_{t_z}(z)|) \leq \epsilon) \geq 1 - \theta_z/c_0$ under Algorithm $\mathfrak{A} \in \mathbb{A}$, due to the fact $\Omega_1(z)$ and $\{|Q^*(z) - Q^{\mathfrak{A}}_{t_z}(z)| \leq \epsilon\}$ are two separate events. The contradiction between the result of Lemma 17 and the assumption which leads to this result proves the lower bound of Eq. (13).

Based on the lower-bound of Eq. (13), we can prove the lower-bound on the total number of samples $T = \sum_{z\in\mathcal{Z}_0} t_z$. To prove the lower bound, we rely on the following lemmas from (Strehl et al., 2009):

Lemma 18 (Strehl et al., 2009) *Let c and Δ be constants in $(0,1)$, $\mathcal{Y}$ be a finite set with the cardinality N and $t_y > 0$ for all $y \in \mathcal{Y}$. Define $T = \sum_{y\in\mathcal{Y}} t_y$. We then have :*

$$\prod_{y\in\mathcal{Y}}\left(1 - c\Delta^{t_y}\right) \leq \left(1 - c\Delta^{\frac{T}{N}}\right)^N.$$

Lemma 19 *If there exist some positive constants c_1 and c_2 such that*

$$\left[1 - \frac{1}{c_0}\exp\left(-\frac{c\xi T}{\psi}\right)\right]^{\psi} \leq 1 - \delta,$$

for some positive quantities T, ξ and $\delta \leq 0.397\psi$, then:

$$T \leq \frac{\psi}{c_1 \eta} \log\left(\frac{\psi}{c_2 \delta}\right),$$

with $c_1 = c$ and $c_2 = c_0/1.78$.

Proof

In order to prove this lemma we make use of the inequality $\exp(x) \geq 1 + x$:

$$1 - \frac{1}{c_0} \exp\left(-\frac{c\xi T}{\psi}\right) \leq (1-\delta)^{\frac{1}{\psi}} \leq \exp\left(\frac{-\delta}{\psi}\right) \leq 1 - \frac{\delta}{1.78\psi},$$

where in the last step we rely on Lemma 16, which imposes the condition $\delta \leq 0.397\psi$. We then deuce by collecting terms:

$$\frac{\delta}{1.78\psi} \leq \frac{1}{c_0} \exp\left(-\frac{c\xi T}{\psi}\right). \tag{17}$$

The result then follows by solving (17) w.r.t. T.

■

We now make use of Eq. (13), Lemma 18 and Lemma 19, also the fact that for every $z_1, z_2 \in \mathcal{Z}_0$ the random events $\{|Q^*(z_1) - Q^{\mathfrak{A}}_{t_{z_1}}(z_1)| \leq \epsilon\}$ and $\{|Q^*(z_2) - Q^{\mathfrak{A}}_{t_{z_2}}(z_2)| \leq \epsilon\}$ are independent of each others,[11] to prove that there exists an MDP in the class $\mathbb{M}^*$ such that:

$$\begin{aligned}
\mathbb{P}(\|Q^* - Q^{\mathfrak{A}}_T\| \leq \epsilon) \leq \mathbb{P}\left[\bigcap_{z \in \mathcal{Z}_0} |Q^*(z) - Q^{\mathfrak{A}}_{t_z}(z)| \leq \epsilon\right] &= \prod_{z \in \mathcal{Z}_0} \mathbb{P}(|Q^*(z) - Q^{\mathfrak{A}}_{t_z}(z)| \leq \epsilon) \\
&= \prod_{z \in \mathcal{Z}_0} \left[1 - \frac{1}{c_0} \exp\left(\frac{-c t_z \epsilon^2}{\beta^2}\right)\right] \\
&\leq \left[1 - \frac{1}{c_0} \exp\left(\frac{-cT\epsilon^2}{\beta^2 n_1}\right)\right]^{n_1} \\
&\leq \left[1 - \frac{1}{c_0} \exp\left(\frac{-cT\epsilon^2}{\beta^2 n/3}\right)\right]^{n/3} \\
&\leq 1 - \delta,
\end{aligned}$$

with $T \leq \beta^2 n/(c_1 \epsilon^2) \log(n/(c_2\delta))$, $\delta \leq 0.0662n$, $c_1 = 3c = 2100$ and $c_2 = c_0/1.78 = 2.24$.[12] In words, if the total number of samples T is less than $\beta^2 n/(c_1\epsilon^2) \log(n/(c_2\delta))$ then the probability of $\|Q^* - Q^{\mathfrak{A}}_T\| \leq \epsilon$ is at maximum $1 - \delta$ on either M_0 or M_1. This is equivalent to the statement that for every RL algorithm $\mathfrak{A} \in \mathbb{A}$ to be (ϵ, δ)-correct on the set $\mathbb{M}^*$, and subsequently on the class of MDPs $\mathbb{M}$, the total number of transitions T needs to satisfy $T > \beta^2 n/(c_1\epsilon^2) \log(n/(c_2\delta))$, which concludes the proof of Theorem 7.

11. This is due to the disjoint structure of the class $\mathbb{M}$.
12. The result holds for all $\delta \in (0, 1)$ with the choice of $n \geq n_0 = 16$.

6. Conclusions and Future Work

In this paper, we introduced a new reinforcement learning algorithm, called speedy Q-learning (SQL). We analyzed the finite-time behavior of this algorithm as well as its asymptotic convergence to the optimal action-value function. Our result is in the form of high probability bound on the performance loss of SQL, which suggests that the algorithm converges to the optimal action-value function in a faster rate than the standard Q-learning. The numerical experiments in Section 4 confirm our theoretical results showing that for large value of β SQL outperforms the other RL methods by a wide margin. Overall, SQL is a simple, efficient and theoretically well-founded reinforcement learning algorithm, which improves on existing RL algorithms such as Q-learning and the sample-based value iteration.

In this work, we are only interested in the estimation of the optimal action-value function and not the problem of exploration. Therefore, we did not compare our result to the PAC-MDP methods (Szita and Szepesvári, 2010; Strehl et al., 2009) and the upper-confidence bound based algorithms (Jaksch et al., 2010; Bartlett and Tewari, 2009), in which the choice of the exploration policy impacts the behavior of the learning algorithms. However, we believe that it would be possible to gain w.r.t. the state of the art in PAC-MDPs, by combining the asynchronous version of SQL with a smart exploration strategy. This is mainly due to the fact that the bound for SQL has been proved to be tighter than the RL algorithms that have been used for estimating the value function in PAC-MDP methods, especially in the model-free case. Also, SQL has a better computational requirement in comparison to the standard RL methods. We consider this as a subject for future research.

Another possible direction for future work is to scale up SQL to large (possibly continuous) state and action spaces where function approximation is needed. We believe that it would be possible to extend our current SQL analysis to the continuous case along the same path as in the fitted value iteration analysis by Munos and Szepesvári (2008) and Antos et al. (2007). This would require extending the error propagation result of Lemma 11 to a ℓ_2-norm analysis and combining it with the standard regression bounds.

References

A. Antos, R. Munos, and Cs. Szepesvári. Fitted Q-iteration in continuous action-space MDPs. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, 2007.

P. L. Bartlett and A. Tewari. REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

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

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, Belmount, Massachusetts, third edition, 2007a.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume II. Athena Scientific, Belmount, Massachusetts, third edition, 2007b.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, Massachusetts, 1996.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, New York, NY, USA, 2006.

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

E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *15th Annual Conference on Computational Learning Theory*, pages 255–270, 2002.

W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968.

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

M. Kearns and S. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems 12*, pages 996–1002. MIT Press, 1999.

S. Koenig and R. G. Simmons. Complexity analysis of real-time reinforcement learning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*. AAAI Press, 1993.

S. Mannor and J. N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004.

R. Munos and Cs. Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

J. Peng and R. J. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22(1-3): 283–290, 1996.

M. L. Puterman. *Markov Decision Processes, Discrete Stochastic Dynamic Programming.* A Wiley-Interscience Publication, 1994.

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

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, Massachusetts, 1998.

Cs. Szepesvári. The asymptotic convergence-rate of Q-learning. In *Advances in Neural Information Processing Systems 10, Denver, Colorado, USA, 1997*, 1997.

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

I. Szita and Cs. Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1031–1038. Omnipress, 2010.

H. van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems 23*, pages 2613–2621, 2010.

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