Journal of Machine Learning Research ??? (???) ??? Submitted ???; Published ???

Speedy Q-Learning: A Computationally Efficient Reinforcement Learning Algorithm 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 probably approximately correct (PAC) 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 this algorithm. The bounds indicate that for any MDP with n state-action pairs and discount factor $\gamma \in [0,1)$, a total number of $O\big(n\log(n)/((1-\gamma)^4\epsilon^2)\big)$ steps suffices for SQL to converge to an ϵ-optimal action-value function with high probability. We also derive a lower-bound of $\Omega\big(n/((1-\gamma)^2\epsilon^2)\big)$ for all RL algorithms, which matches the upper bound in terms of n (up to a logarithmic factor) and ϵ. Moreover, our results have better dependencies on ϵ and $1-\gamma$ (the same dependency on n), and thus, are tighter than the best available results for Q-learning. The SQL algorithm also improves on existing results for the batch Q-value iteration, in terms of the computational budget required to achieve a near optimal solution.

1. Introduction

Finding an optimal policy for a Markovian decision process (MDP) is a classical problem in the fields of operations research and decision theory. When an explicit model of an MDP

*. An extended abstract of this paper appeared in the Proceedings of Advances in Neural Information Processing Systems (NIPS 24), pp. 2411-2419.

(i.e., transition probability and reward functions) 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 an optimal policy of the MDP. Value iteration algorithm computes the optimal action-value function Q^* by successive iterations of the Bellman operator $\mathcal{T}$ (will be defined 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, value iteration rely 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 obtained by following an exploration policy (a rollout), or by simulating independent transition samples in the state(-action) space using a generative model (simulator) of the dynamical system. The field of reinforcement learning (RL) is concerned with the problem of finding an optimal policy or the optimal value function from the observed reward and transition samples (Sutton and Barto, 1998; Szepesvári, 2010).

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 use it to approximate value functions using DP techniques. In contrary, model-free methods compute an approximation of a value function by making use of a sample-based estimate of the Bellman operator without resorting to learning an explicit model of the dynamical system. Q-learning (QL) is a well-known model-free 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 for all state-action pairs (x, a) is calculated using the following update rule:[1]

$$\begin{aligned} Q_{k+1}(x,a) &= (1-\alpha_k)Q_k(x,a) + \alpha_k \mathcal{T}_k Q_k(x,a) \\ &= (1-\alpha_k)Q_k(x,a) + \alpha_k \left(\mathcal{T}Q_k(x,a) - \epsilon_k(x,a)\right), \end{aligned}$$

where $\epsilon_k(x,a) = \mathcal{T}_k Q_k(x,a) - \mathcal{T}Q_k(x,a)$, $\mathcal{T}_k Q_k(x,a)$ is the empirical estimation of Bellman operator and α_k is the learning step. One may show, using an induction argument, that for the choice of linear learning step, i.e., $\alpha_k = \frac{1}{k+1}$, Q_{k+1} can be seen the average of the estimates of Bellman operator throughout the learning process:

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

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 thus, for $k \to \infty$ the update rule of QL 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 prove that the asymptotic

1. In this section, for the sake of simplicity in the notation, we assume that the action-values of all state-action pairs are updated in parallel. Note that, in general, this assumption is not required for the proof of convergence of Q-learning.

rate of convergence of QL with linear learning step is of order $\tilde{O}(1/k^{1-\gamma})$ (Szepesvári, 1997),[2] which in the case of γ close to 1 makes its convergence extremely slower than the standard value iteration algorithm, which enjoys a fast convergence rate of order $\tilde{O}(\gamma^k)$. 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 QL despite the fact that QL has better 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), to address the problem of slow convergence of Q-learning. The main idea is to modify the update rule of Q-learning such that, at each iteration k, the new estimate of action-value function Q_{k+1} closely follows the Bellman operator $\mathcal{T}Q_k$. This guarantees that the rate of convergence of SQL, unlike QL, is close to the fast rate of convergence of the value iteration algorithm. 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

$$Q_{k+1}(x,a) = \alpha_k Q_k(x,a) + (1-\alpha_k)\big[k\mathcal{T}Q_k(x,a) - (k-1)\mathcal{T}Q_{k-1}(x,a) - \epsilon_k(x,a)\big], \qquad (1)$$

which makes its space complexity twice as QL. However, this allows SQL 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 $\mathcal{T}Q_k + O\big(1/(k+1)\big)$, with the choice of $\alpha_k = 1/(k+1)$:

$$\begin{aligned}
Q_{k+1}(x,a) &= \alpha_k Q_k(x,a) + (1-\alpha_k)\big[k\mathcal{T}Q_k(x,a) - (k-1)\mathcal{T}Q_{k-1}(x,a) - \epsilon_k(x,a)\big] \\
&= \frac{1}{k+1}\sum_{j=0}^{k}\big(j\mathcal{T}Q_j(x,a) - (j-1)\mathcal{T}Q_{j-1}(x,a) - \epsilon_j(x,a)\big) \\
&= \mathcal{T}Q_k(x,a) + \frac{1}{k+1}(\mathcal{T}Q_{-1}(x,a) - \mathcal{T}Q_k(x,a)) - \frac{1}{k+1}\sum_{j=0}^{k}\epsilon_j(x,a),
\end{aligned}$$

where in the second line we rely on an induction argument. This shows that similar to QL, the iterates of SQL are expressed in terms of the average estimation error, and thus, the SQL update rule asymptotically averages out the sampling errors. However, SQL 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. As a result, unlike QL, it does not suffer from the slow convergence due to slow down in the value iteration process (see Section 3.3 for a detailed comparison of QL and SQL's convergence rates).

The idea of using previous estimates of the action-values has already been employed in order to improve the performance of QL. A popular algorithm of this kind is Q(λ) (Watkins, 1989; Peng and Williams, 1996), 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

2. The notation $g = \tilde{O}(f)$ implies that there are constants c_1 and c_2 such that $g \leq c_1 f \log^{c_2}(f)$.

Hasselt, 2010), which uses two estimators for the action-value function in order to alleviate the over-estimation of action-values in QL. 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.

The rest of the paper is organized as follows. After introducing the notation used in the paper in Section 2, we present our *Speedy Q-learning* algorithm in Section 3. We first describe the synchronous and asynchronous versions 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 RL, 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 numerically evaluate the performance of SQL on different problems. Section 5 contains the detailed proofs of the results of Sections 3.2, i.e., performance bounds of SQL 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, definitions, and notation from the Markov decision processes (MDPs) theory 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 denote the *effective* horizon of MDP by β defined as $\beta = 1/(1-\gamma)$. We also 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.[3] 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 that the joint state-action set $\mathcal{Z}$ is finite with cardinality n, and the immediate rewards $r(x, a)$ are in the interval $[0, 1]$.*[4]

A policy π determines the distribution of the control action given the past observations. A policy is called *stationary* if the distribution depends only on the last state x and is *deterministic* if it assigns a unique action to each state $x \in \mathcal{X}$. 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

3. For the sake of simplicity in notation, here we assume that the reward $r(x, a)$ is a deterministic function of state-action pairs (x, a). It is straightforward to extend our results to the case of stochastic rewards under some mild assumptions, e.g., boundedness of the absolute value of the rewards.
4. 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 $r_{\max} - r_{\min}$.

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

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

where $\mathcal{M}$ is the max operator over action-value functions and is defined as $(\mathcal{M}Q)(y) = \max_{a \in \mathcal{A}} Q(y,a),\ \forall y \in \mathcal{X}$.[5] We now define the cover time of MDP under the policy π as follows:

Definition 1 (Cover Time) *Let π be a policy over a finite stat-action MDP and $t \geq 0$ be an integer. Define $\tau_\pi(x,t)$ to be the number of time-steps between t and the first future time that all state-action pairs $z \in \mathcal{Z}$ are visited (the MDP is covered) starting from state $x \in \mathcal{X}$ at time-step t and following π. The state-action space $\mathcal{Z}$ is* covered *by the policy π if all the state-action pairs are visited at least once under the policy π.*

The following assumption that 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 expected cover time) *Let $0 < L < \infty$ and t be an integer. We assume that under the policy π, for all $x \in \mathcal{X}$ and $t > 0$, we have*

$$\mathbb{E}\big(\tau_\pi(x,t)\big) \leq L.$$

3. Speedy Q-Learning

In this section, we introduce a new RL algorithm, called *speedy Q-Learning* (SQL), derive performance bounds for its synchronous and asynchronous variants, and compare these bounds with similar results on standard Q-learning (QL).

3.1 Synchronous and Asynchronous SQL Algorithms

In this subsection, we introduce two variants of the SQL algorithm, 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 the 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 a simulator that can generate samples anywhere in the state-action space, the algorithm may be 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 versions of

5. 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 $\|\mathcal{T}Q - \mathcal{T}Q'\| \leq \gamma \|Q - Q'\|$ (Bertsekas, 2007b, Chap. 1).

SQL are shown in Algorithms 1 and 2, respectively. It is possible to show that asynchronous SQL is reduced to synchronous SQL when the cover time $\tau_\pi(x,t) = n$ for all $x \in \mathcal{X}$ and $t \geq 0$. In this case, the action-values of all state-action pairs are updated in a row. In other words, Algorithm 1 may be seen as a special case of Algorithm 2. Therefore, in the sequel we only describe the more general asynchronous SQL algorithm.

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`
  $\alpha_k := \frac{1}{k+1}$;
  foreach $(x,a) \in \mathcal{Z}$ **do** `// Update the action-value function for all` $(x,a) \in \mathcal{Z}$
    Generate the next state sample $y_k \sim P(\cdot|x,a)$;
    $\mathcal{T}_k Q_{k-1}(x,a) := r(x,a) + \gamma \mathcal{M} Q_{k-1}(y_k)$;
    $\mathcal{T}_k Q_k(x,a) := r(x,a) + \gamma \mathcal{M} Q_k(y_k)$;
    $Q_{k+1}(x,a) := (1-\alpha_k) Q_k(x,a) + \alpha_k \big(k \mathcal{T}_k Q_k(x,a) - (k-1) \mathcal{T}_k Q_{k-1}(x,a)\big)$; `// SQL update rule`
    $t := t+1$;
  end
  $k := k+1$;
until $t \geq T$;
return Q_k

As it can be seen from the update rule of Algorithm 2, at each time step, the algorithm keeps track of the action-value functions of the two most recent iterations Q_k and Q_{k-1}, and its main update rule is of the following form at time step t and iteration k:

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

where $\mathcal{T}_k Q(X_t,A_t) = 1/|\mathcal{Y}_k| \sum_{y \in \mathcal{Y}_k} \big[r(X_t,A_t) + \gamma \mathcal{M} Q(y)\big]$ 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 $\mathcal{Y}_{k,t}(x,a)$, the set of all samples generated up to time step t in round k by taking action a in state x. At each time step t, Algorithm 2 works as follows: **(i)** it simulates the MDP for one-step at state X_t, i.e., it first draws the action $A_t \in \mathcal{A}$ from the distribution $\pi(\cdot|X_t)$ and then makes a transition to a new state $y_k \sim P(\cdot|X_t,A_t)$, **(ii)** it updates the two sample estimates $\mathcal{T}_k Q_{k-1}(X_t,A_t)$ and $\mathcal{T}_k Q_k(X_t,A_t)$ of the Bellman optimality operator applied to the estimates Q_{k-1} and Q_k of the action-value function at the previous and current rounds $k-1$ and k, for the state-action pair (X_t,A_t) using the next state y_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 if all $(x,a) \in \mathcal{Z}$ have been visited at least once at iteration k, and if this condition is satisfied, we move to the next round $k+1$, and 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., $\alpha_k = 1/(k+1)$. Note that the update rule $\mathcal{T}_k Q_k(X_t,A_t) := (1-\eta_N)\mathcal{T}_k Q_k(X_t,A_t) + \eta_N (r(X_t,A_t) + \gamma \mathcal{M} Q_k(y_k))$ is used to incrementally generate an unbiased estimate of $\mathcal{T} Q_k$.

Algorithm 2: Asynchronous Speedy Q-learning

Input: initial action-values Q_0, policy π, discount factor γ, number of step T, and 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 taking action A_t in state X_t;
  $\eta_N := \dfrac{1}{N_k(X_t, A_t) + 1}$;
  $\mathcal{T}_k Q_{k-1}(X_t, A_t) := (1 - \eta_N)\mathcal{T}_k Q_{k-1}(X_t, A_t) + \eta_N\big(r(X_t, A_t) + \gamma \mathcal{M} Q_{k-1}(y_k)\big)$;
  $\mathcal{T}_k Q_k(X_t, A_t) := (1 - \eta_N)\mathcal{T}_k Q_k(X_t, A_t) + \eta_N\big(r(X_t, A_t) + \gamma \mathcal{M} Q_k(y_k)\big)$;
  $Q_{k+1}(X_t, A_t) := (1 - \alpha_k)Q_k(X_t, A_t) + \alpha_k\big(k\mathcal{T}_k Q_k(X_t, A_t) - (k-1)\mathcal{T}_k Q_{k-1}(X_t, A_t)\big)$; `// 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
    $k := k + 1$;
    $\alpha_k := \dfrac{1}{k+1}$;
    foreach $(x, a) \in \mathcal{Z}$ **do**
      $N_k(x, a) := 0$;
    end
  end
  $t := t + 1$;
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 for the performance of both synchronous and asynchronous versions of the SQL algorithms. We also report a new lower bound on the number of transitions required for every RL algorithm to achieve an ϵ-optimal estimate of Q^* with high probability (w.p. $1 - \delta$).[6] The derived performance bound shows that SQL has a rate of convergence $T = O\big(n\beta^4 \log(n/\delta)/\epsilon^2\big)$, which matches the proposed lower bound of RL in terms of n (up to a logarithmic factor), ϵ, and δ. However, the dependency of our bound on the horizon β is worse than the lower-bound by a factor of order $O(\beta^2)$.

Theorem 2 (Performance Bound of Synchronous SQL) *Let A1 hold and Q_T be the estimate of Q^* generated by Algorithm 1 after T steps. Then, with probability at least $1-\delta$, we have*

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

6. We report the detailed proofs in Section 5.

Theorem 3 (Performance Bound of Asynchronous SQL) *Let A1 and A2 hold and Q_T be the estimate of Q^* generated by Algorithm 2 after T steps. Then, with probability at least $1-\delta$, we have*

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

These results, combined with the Borel-Cantelli lemma (Feller, 1968), guarantee that Q_T converges almost surely to Q^* with the rate $\sqrt{1/T}$ for both Algorithms 1 and 2. Moreover, the PAC bounds of Corollaries 4 and 5, 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 Theorems 2 and 3, respectively.

Corollary 4 (Finite-time PAC Bound of Synchronous SQL) *Under A1, after*

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

steps (transitions), the uniform approximation error of Algorithm 1 is small, i.e., $\|Q^* - Q_T\| \leq \epsilon$, with probability at least $1-\delta$.[7]

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

$$T = \lceil \frac{4\ e\ L\ \beta^4\ \log \frac{2}{\delta}\ \log \frac{4n}{\delta}}{\epsilon^2} \rceil$$

steps (transitions), the uniform approximation error of Algorithm 2 is small, i.e., $\|Q^* - Q_T\| \leq \epsilon$, with probability at least $1-\delta$.

The following general result provides a new lower bound on the number of transitions T for every RL algorithm to achieve an ϵ-optimal performance w.p. $1-\delta$, under the assumption that the RL algorithm is (ϵ, δ)-correct.[8]

Definition 6 ((ϵ, δ, T^*)-correct RL algorithm) *Let $\mathbb{A}$ be the class of all RL algorithms that rely on estimating the action-value function Q^* and $Q_T^{\mathfrak{A}}$ be the estimate of Q^* by the algorithm $\mathfrak{A} \in \mathbb{A}$ after $T \geq 0$ transitions. The RL algorithm $\mathfrak{A} \in \mathbb{A}$ is called (ϵ, δ, T^*)-correct on the class of MDPs $\mathbb{M}$, if for all $T > T^*$ and for all $M \in \mathbb{M}$, we have $\left\|Q^* - Q_T^{\mathfrak{A}}\right\| \leq \epsilon$ with probability at least $1-\delta$.*

Theorem 7 (Lower bound on the sample complexity of RL) *There exists some ϵ_0, δ_0, γ_0, c_1, c_2, and a class of MDPs $\mathbb{M}$, such that for all $\epsilon \in (0, \epsilon_0)$, $\gamma \in (\gamma_0, 1)$, $\delta \in (0, \delta_0)$,*

7. For every real number u, $\lceil u \rceil$ is defined as the smallest integer number not less than u.
8. This result improves on the state-of-the-art (Strehl et al., 2009) in terms of the dependency on β, by a factor of $O(\beta^2)$. Our result is also more general than the one by Strehl et al. (2009) in the sense that it does not require a sequential update of the value functions or following a deterministic policy. On the other hand, our result is slightly worse than the previous lower bound in terms of dependency on n.

and every (ϵ, δ, T^*)*-correct RL algorithm* $\mathfrak{A} \in \mathbb{A}$ *on the class of MDPs* $\mathbb{M}$ *the number of transitions*

$$T > T^* = \lceil \frac{n\beta^2}{c_1\epsilon^2} \log \frac{1}{c_2\delta} \rceil.$$

3.3 Relation to the Existing Results

In this section, we first compare our results for the SQL algorithm with the existing results on the convergence of the standard Q-learning. The comparison indicates that SQL accelerates the convergence of QL, especially for large values of β and small values of α. We then compare SQL with batch Q-value iteration (QVI) in terms of the sample and computational complexities, i.e., the number of samples and the number of time units [9] 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 the 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) 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) investigated the finite-time behavior of synchronous QL for different time scales. Their main result indicates that by using the polynomial learning step $\alpha_k = 1/(k+1)^{\omega}$, $0.5 < \omega < 1$, synchronous QL 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 $\gamma \approx 1$, the horizon $\beta = 1/(1-\gamma)$ becomes the dominant term in the bound of Eq. 3, and thus, the bound is optimized by finding an ω that minimizes the dependency on β. This leads to the optimized bound of order $\tilde{O}\big(\beta^5/\epsilon^{2.5}\big)$ with the choice of $\omega = 0.8$. On the other hand, SQL is guaranteed to achieve the same precision with 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:

9. In the sequel, we consider the CPU time required to compute a single-sample estimate of the Bellman optimality operator as the time unit.

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

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

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 order $O(L^{3+\frac{1}{\omega}} + L^{\frac{1}{1-\omega}})$, which with the choice of $\omega \approx 0.77$ leads to the optimized dependency of order $O(L^{4.34})$, whereas asynchronous SQL achieves the same accuracy after just $O(L)$ steps. This result indicates 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 QL 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; Even-Dar et al., 2002; Kakade, 2004, chap. 9.1). 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.[10] A comparison between their results and the main result of this paper suggests that the sample complexity of SQL, which is of order $\tilde{O}\big(n\beta^4/\epsilon^2\big)$, 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 $\min\big(\tilde{O}(n\beta^4/\epsilon^2), n(|\mathcal{X}|+1)\big)$ (Kearns and Singh, 1999). SQL also improves the computational complexity by a factor of $\tilde{O}(\beta)$ compared to both model-free and model-based QVI.[11] Table 1 summarizes the comparisons between SQL and the RL methods discussed in this section.

4. Experiments

In this section, we empirically evaluate the performance of the synchronous SQL (Algorithm 1) on several discrete state-action problems. We also examine the convergence of these algorithms and compare it with Q-learning and model-based Q-value iteration (QVI) (Kearns and Singh, 1999). The source code of all the algorithms is available at `http://www.mbfys.ru.nl/~mazar/Research_Top.html`.

10. For the sake of simplicity, here we ignore the logarithmic dependencies of the bounds.
11. Since SQL performs only one Q-value update per sample, its sample and computational complexities of are of the same order. The same argument also applies to the standard Q-learning. On the other hand, in the case of model-based QVI, the algorithm needs to iterate the action-value function of all the state-action pairs at least $\tilde{O}(\beta)$ times. This 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.

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	$\Theta(n)$	$\Theta(n)$	$\min\left(\tilde{O}(n\beta^4/\epsilon^2), n(\lvert\mathcal{X}\rvert+1)\right)$	$\Theta(n)$

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

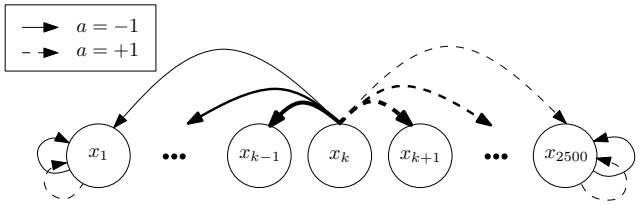

Figure 1: The Linear MDP problem. States x_1 and x_{2500} are the two absorbing states and state x_k is an example of the interior states. The absorbing states have reward 1 and the interior states have reward -1. From an interior state x_k, all the other states in the direction of the selected action are reachable with a probability proportional to their inverse distance to x_k (see Eq. 4).

Linear MDP: This problem consists of 2500 states $\mathcal{X} = \{x_1, \ldots, x_{2500}\}$ 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 every 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 thus, $P(x_l|x_k, a) = 0$, $\forall l \neq k$. Any transition to one of the two absorbing states has reward 1 and to any other state (interior states) has reward -1. The transition probability from 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 the states in the opposite direction. This can be written formally as (x_k is an interior state, i.e., $x_k \neq x_1, x_{2500}$)

$$P(x_l|x_k, a) \propto \begin{cases} \dfrac{1}{|l-k|} & \text{if } (l-k)a > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

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

Combination Lock: This problem is a stochastic variant of the reset state space model introduced in Koenig and Simmons (1993), where more than one reset state is possible (see Figure 2). Similar to the Linear MDP problem, here we consider 2500 states $\mathcal{X} = \{x_1, \ldots, x_{2500}\}$ arranged in a one-dimensional chain and two possible actions $\mathcal{A} =$

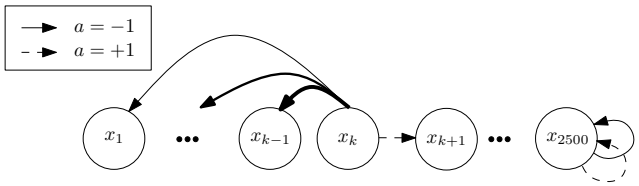

Figure 2: The Combination Lock problem. State x_{2500} is the absorbing goal state and state x_k is an example of the interior states. The goal state has reward 1 and is reached only of an all-ones $\{+1, \ldots, +1\}$ sequence of actions is taken. For every interior state, the rewards of actions -1 and $+1$ are set to 0 and -0.01, respectively. The transition probability upon taking the wrong action -1 is inversely proportional to the distance of the states.

$\{-1, +1\}$. However, there is only one absorbing state (corresponding to the state *lock-opened*) with associated reward of 1 in this problem. This state is reached if an all-ones sequence $\{+1, \ldots, +1\}$ of actions is taken. Otherwise, if at some state $x_k,\ k < 2500$, action -1 is taken, the lock randomly resets to a state $x_l,\ l < k$ (in the original combination lock problem, the lock always resets to the initial state x_1). For every interior 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 inversely proportional to the distance of the states (similar to the Linear MDP problem), i.e.,

$$P(x_l|x_k, -1) \propto \begin{cases} \dfrac{1}{k-l} & \text{if } l < k, \\ 0 & \text{otherwise.} \end{cases}$$

Note that this problem is more difficult than Linear MDP since the goal state is only reachable from one single 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. The location of each state x on the grid is determined by its coordinates $c_x = (h_x, v_x)$, where h_x and v_x are integers from 1 to 50. There are 196 absorbing *firewall states* at the edges of the grid and one at the center of the grid, i.e., $c_x = (25, 25)$. The reward of the central firewall state is -1, other firewall states is defined as $r(x, a) = -1/\left\|c_x\right\|_2,\ \forall a \in \mathcal{A}$, and all the other states (non-absorbing states) is 0. This means that both the top-left and central absorbing (firewall) states have the minimum reward -1, and the remaining absorbing states have rewards that increase proportionally to their distance to the state in the bottom-right corner. The transition probabilities are defined in the following way: taking an action a at any non-absorbing state x results in a one-step transition in the direction of a with probability 0.6, and a random move to a state $y \neq x$ with probability inversely proportional to their Euclidean distance $1/\left\|c_x - c_y\right\|_2$.

The optimal policy here is to *survive* as long as possible in the grid by avoiding the absorbing firewall states at the center and edges of the grid. Note that since the costs of the firewall states are different, the optimal policy would prefer the states near the bottom-right corner of the grid in order to avoid high-cost absorbing states.

4.1 Experimental Setup and Results

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

All the 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 using 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 may 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 for the linear MDP and combination lock problems. For the grid world, which has twice as many actions as the other benchmarks, the maximum running time is fixed to 60 seconds. We also fix the total number of samples, per state-action, to 10^5 samples for all problems and algorithms. Smaller number of samples leads to a dramatic decrease in the quality of the solutions of all the three algorithms. Algorithms were implemented as MEX files (in C++) and ran on a Intel core i5 processor with 8 GB of memory. CPU time was computed using the system function `times()` that provides process-specific CPU time. Randomization was implemented using `gsl_rng_uniform()` function of the GSL library, which is superior to the standard `rand()`.[12] Sampling time, which is the same for all the algorithms, were not included in the 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. All the results are averaged over 50 different runs.

Figure 3 shows the performance-loss in terms of the elapsed CPU time for the three problems and algorithms with the choice of $\beta = 1000$. We observe that SQL outperforms QL and QVI in all the three problems. It achieves a reasonable performance very rapidly, just in a few seconds. The minimum and maximum errors are attained for the combination lock and grid world problems, respectively. We also observe that the difference between the final outcome of SQL and QL (the second best method) is significant, about 30 times, in all domains.

Figures 4 and 5 show the means and standard deviations of the final performance-loss 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 about

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

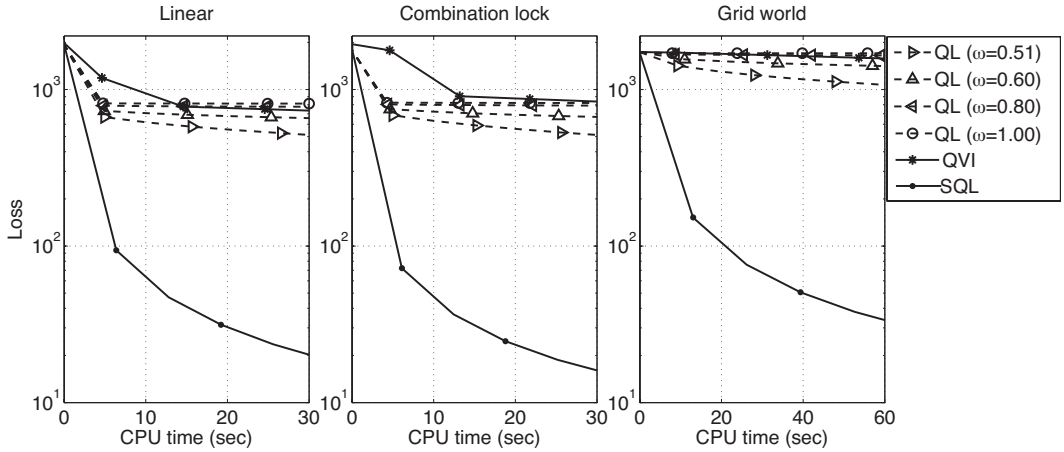

Figure 3: A comparison between SQL, QL, and QVI. Each plot compares the performance loss of the policies induced by the algorithms at one of the three problems considered in this section. All the results are averaged over 50 different runs.

two order of magnitude advantage over QVI. QL performs better for larger values of ω when the horizon β is small, whereas for large values of β smaller ω's are more preferable.

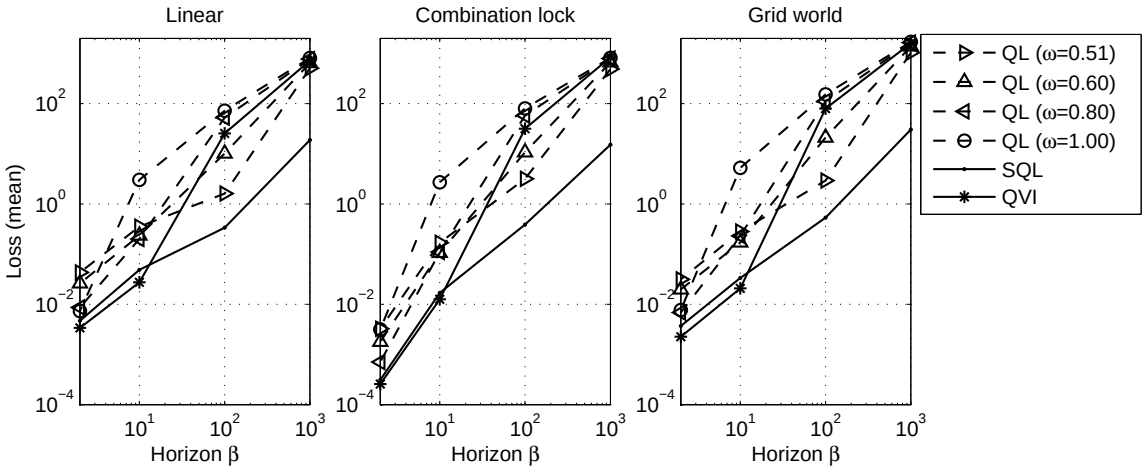

Figure 4: A comparison between SQL, QL, and QVI given a fixed computational and sampling budget. The plot shows the means of the final performance of the algorithms in terms of the horizon β. All the results are averaged over 50 different runs.

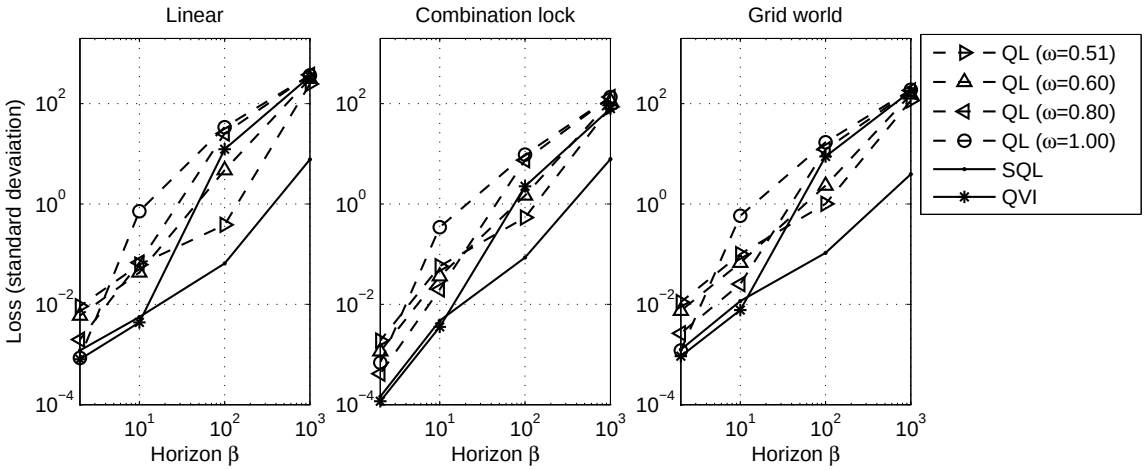

Figure 5: A comparison between SQL, QL, and QVI given a fixed computational and sampling budget. The plot shows the the standard deviations of the final performance of the algorithms in terms of the horizon β. All the results are averaged over 50 different runs.

These results are consistent with the performance bound of Theorem 2 and indicate that the SQL algorithm manages to average out the simulation noise caused by sampling, and converges rapidly to a near optimal solution, which is robust in comparison to the other algorithms. Moreover, we may conclude that SQL significantly improves the computational complexity of learning w.r.t. the standard QL and QVI in the three problems studied in this section.

5. Analysis

In this section, we first give some intuition about the convergence of asynchronous variant of SQL and provide the full proof of the finite-time analysis reported in Theorems 2 and 3. We then prove Theorem 7, the RL lower bound, in Section 5.1. We start by introducing some notation.

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 $\{\mathcal{Y}^0, \mathcal{Y}^1, \ldots, \mathcal{Y}^k\}$. Note that for all $(x,a) \in \mathcal{Z}$, the update rule of Equation 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 $\mathcal{D}_k[Q_k, Q_{k-1}](x,a) \triangleq \frac{1}{\alpha_k}\big[(1-\alpha_k)\mathcal{T}_k Q_k(x,a) - (1-2\alpha_k)\mathcal{T}_k Q_{k-1}(x,a)\big]$ and $\alpha_k = 1/(k+1)$. We now define the operator $\mathcal{D}[Q_k, Q_{k-1}]$ as the expected value of the empirical operator $\mathcal{D}_k$ conditioned on the filtration $\mathcal{F}_{k-1}$, i.e.,

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

where the last equality follows from the fact that in both Algorithms 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 rewritten as

$$Q_{k+1}(x,a) = (1-\alpha_k)Q_k(x,a) + \alpha_k\big(\mathcal{D}[Q_k,Q_{k-1}](x,a) - \epsilon_k(x,a)\big), \tag{5}$$

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 $\mathcal{D}_k[Q_k,Q_{k-1}]$, i.e.,

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

We have the property that $\mathbb{E}\big[\epsilon_k(x,a)|\mathcal{F}_{k-1}\big] = 0$, which means that for all $(x,a) \in \mathcal{Z}$, the sequence of estimation errors $\big\{\epsilon_1(x,a), \epsilon_2(x,a), \ldots, \epsilon_k(x,a)\big\}$ 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, i.e.,

$$E_k(x,a) \triangleq \sum_{j=0}^{k} \epsilon_j(x,a), \qquad \forall (x,a) \in \mathcal{Z}. \tag{6}$$

The following steps lead to the proof of Theorems 2 and 3: **(i)** Lemma 9 shows the stability of SQL, i.e., the sequence of Q_k's stays bounded in SQL. **(ii)** Lemma 10 states the key property that each iterate Q_{k+1} in SQL is close to the Bellman operator applied to the previous iterate Q_k, i.e., $\mathcal{T}Q_k$. More precisely, in this lemma we show that Q_{k+1} is equal to $\mathcal{T}Q_k$ plus an estimation error term of order E_k/k. **(iii)** Lemma 11 provides a performance bound on $\|Q^* - Q_k\|$ in terms of a discounted sum of the cumulative estimation errors $\{E_j\}_{j=0}^{k-1}$. Lemma 9 to 11 hold for both Algorithms 1 and 2. **(iv)** Given these results, we prove Theorem 2 using a maximal Azuma's inequality stated in Lemma 12. **(v)** Finally, we extend this proof to asynchronous SQL, and prove Therorem 3, using the result of Lemma 14.

For simplicity of the notation, we often remove the dependence on (x,a), e.g., writing Q for $Q(x,a)$ and E_k for $E_k(x,a)$. Also note that for all $k \geq 0$, the following relations hold between α_k and α_{k+1} in Algorithms 1 and 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*

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

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 let us assume that for any $k \geq 0$, $\|\mathcal{D}_k[Q_k, Q_{k-1}]\| \leq V_{\max}$. Then we obtain

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

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

The bound on ϵ_k follows from $\|\epsilon_k\| = \|\mathbb{E}\big(\mathcal{D}_k[Q_k, Q_{k-1}] | \mathcal{F}_{k-1}\big) - \mathcal{D}_k[Q_k, Q_{k-1}]\| \leq V_{\max}$, and the bound $\|Q_k\| \leq V_{\max}$ is deduced by the fact 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 $k \geq 1$ we have*

$$Q_k = \mathcal{T} Q_{k-1} + \frac{1}{k} \left(\mathcal{T} Q_0 - \mathcal{T} Q_{k-1} - E_{k-1} \right). \tag{7}$$

Proof We prove this result by induction. The result holds for $k = 1$, where Equation 7 reduces to Equation 5. We now show that if Equation 7 holds for $k \geq 1$ then it also holds for $k+1$. Assume that Equation 7 holds for k, then from Equation 5 we have

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

Thus Equation 7 holds for $k+1$, and as a result, holds 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 in 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*

$$\|Q^* - Q_k\| \leq \frac{1}{k}\left[\gamma\beta^2 + \sum_{j=1}^{k} \gamma^{k-j} \|E_{j-1}\|\right] \tag{8}$$

$$\leq \frac{\beta}{k}\left[\gamma\beta + \max_{j=1:k} \|E_{j-1}\|\right]. \tag{9}$$

Proof For any sequence of cumulative errors $\{E_0, E_1, \ldots, E_{k-1}\}$, we have $\sum_{j=1}^{k} \gamma^{k-j} \|E_{j-1}\| \leq \beta \max_{j=1:k} \|E_{j-1}\|$. Thus, we only need to prove (8), and (9) will automatically follow. We again prove this lemma by induction. The result holds for $k = 1$ since we have

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

Note that the first equality follows from Lemma 10. We now show that if the bound holds for k, then it should also hold for $k+1$. If (8) holds for k, then using Lemma 10 we have

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

Since we assumed that (8) holds for k, we may write

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

Thus, Equation 8 holds for $k+1$, and as a result by induction, it holds for all $k \geq 1$. ■

Before stating the next lemma, we report the maximal Azuma-Hoeffding's inequality (see e.g., Cesa-Bianchi and Lugosi 2006), which is used in the proof of this lemma.

Proposition 12 (Maximal Azuma-Hoeffding'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*

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

We now state Lemma 13 in which we prove a high probability bound on the estimation error term in Lemma 11, i.e., $\max_{j=1:k} \|E_{j-1}\|$.

Lemma 13 *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 $k \geq 1$, with probability at least $1-\delta$, we have*

$$\max_{j=1:k} \|E_{j-1}\| \leq \beta \sqrt{2k \log \frac{2n}{\delta}}. \tag{10}$$

Proof We begin by providing a high probability bound on $\max_{1\leq j\leq k} |E_{j-1}(x,a)|$ for a given state-action pair (x,a). Note that

$$\begin{aligned}
\mathbb{P}\left(\max_{j=1:k} |E_{j-1}(x,a)| > \epsilon\right) &= \mathbb{P}\left(\max\left[\max_{j=1:k} E_{j-1}(x,a)\ ,\ \max_{j=1:k}\big(-E_{j-1}(x,a)\big)\right] > \epsilon\right) \\
&= \mathbb{P}\left(\left\{\max_{j=1:k} E_{j-1}(x,a) > \epsilon\right\} \bigcup \left\{\max_{j=1:k}\big(-E_{j-1}(x,a)\big) > \epsilon\right\}\right) \\
&\leq \mathbb{P}\left(\max_{j=1:k} E_{j-1}(x,a) > \epsilon\right) + \mathbb{P}\left(\max_{j=1:k}\big(-E_{j-1}(x,a)\big) > \epsilon\right),
\end{aligned} \tag{11}$$

We can now bound both terms in (11) using the maximal Azuma-Hoeffding's inequality stated in Proposition 12. As mentioned earlier, the sequence of random variables $\big\{\epsilon_0(x,a), \epsilon_1(x,a), \ldots, \epsilon_k(x,a)\big\}$ is martingale difference w.r.t. the filtration $\mathcal{F}_k$ generated by random samples $\{y_0, y_1, \ldots, y_k\}(x,a)$ for all (x,a), i.e., $\mathbb{E}[\epsilon_k(x,a)|\mathcal{F}_{k-1}] = 0$. So, we have

$$\begin{aligned}
\mathbb{P}\left(\max_{j=1:k} E_{j-1}(x,a) > \epsilon\right) &\leq \exp\left(\frac{-\epsilon^2}{2kV_{\max}^2}\right), \\
\mathbb{P}\left(\max_{j=1:k}\big(-E_{j-1}(x,a)\big) > \epsilon\right) &\leq \exp\left(\frac{-\epsilon^2}{2kV_{\max}^2}\right).
\end{aligned} \tag{12}$$

Combining (12) with (11), we deduce

$$\mathbb{P}\left(\max_{j=1:k} |E_{j-1}(x,a)| > \epsilon\right) \leq 2\exp\left(\frac{-\epsilon^2}{2kV_{\max}^2}\right),$$

and then by a union bound over the state-action space, we obtain

$$\mathbb{P}\left(\max_{j=1:k} \|E_{j-1}\| > \epsilon\right) \leq 2n\exp\left(\frac{-\epsilon^2}{2kV_{\max}^2}\right). \tag{13}$$

Equation 13 may be rewritten for any $\delta > 0$ as

$$\mathbb{P}\left(\max_{j=1:k} \|E_{j-1}\| \leq V_{\max}\sqrt{2k \log \frac{2n}{\delta}}\right) \geq 1-\delta,$$

which concludes the proof. ■

Proof of Theorem 2 The result of the theorem follows by plugging Equation 10 into Equation 9, and taking into account that if $n(k-1) < T \leq nk$ then Algorithm 1 stops after k iterations and returns Q_k. ■

For the proof of Theorem 3, we rely on the following lemma which bounds the number of steps required to visit all state-action pairs k times with high probability.

Lemma 14 *Under A2, from any initial state x_0 and for any integer $k > 0$, after running Algorithm 2 for $T = ekL \log \frac{1}{\delta}$ steps, the state-action space $\mathcal{Z}$ is covered at least k times under the policy π with probability at least $1 - \delta$.*

Proof For any state $x_0 \in \mathcal{X}$ and time $t > 0$, we define a random variable $\mathcal{Q}_k$ as the number of steps required to cover the MDP k times starting from x_0 at time t. Using Markov inequality (Feller, 1968), for any x_0 and t, we can bound $\mathcal{Q}_k$ with high probability as

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

This mean that after a run of length ekL, the probability that the entire state-action space is not covered 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 the MDP k times is less than $\frac{1}{e^m}$, i.e., $\mathbb{P}(\mathcal{Q}_k > mekL) \leq \frac{1}{e^m}$. With the choice of $m = \log \frac{1}{\delta}$, we obtain $\mathbb{P}(\mathcal{Q}_k > ekL \log \frac{1}{\delta}) \leq \delta$. The bound then can be rewritten as

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

which concludes the proof. ■

Proof of Theorem 3 Plugging the results of Lemmas 14 and 13 into Equation 9 (each holds with probability at least $1 - \delta'$, with $\delta' = \delta/2$) concludes the proof of the theorem. ■

5.1 Proof of Theorem 7 - The Lower Bound

In this section, we provide the 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 (Mannor and Tsitsiklis, 2004), and extend this approach to RL and MDPs. We also make use of some technical lemmas in Strehl et al. (2009).

Let us consider the class of MDPs $\mathbb{M}$ as the set of all MDPs with $n = 3n_1 \geq 3$ state-action pairs in which the state-action space $\mathcal{Z}$ is made of three smaller sets $\mathcal{Z}_0$, $\mathcal{Z}_1$, and $\mathcal{Z}_2$ each of size n_1. 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 an absorbing state). For all $M \in \mathbb{M}$, every state-action pair $z_0^l \in \mathcal{Z}_0$ is connected to only two other state-action pairs $z_1^l \in \mathcal{Z}_1$ and $z_2^l \in \mathcal{Z}_2$ with probabilities $p_M(z_0^l)$ and $1 - p_M(z_0^l)$, respectively. The instant reward $r(z)$ is set to 1 for every state-action pair in $\mathcal{Z}_1$ and 0 elsewhere. One may solve the Bellman equation for M and compute its optimal action-value function Q^* in closed-form as follows:

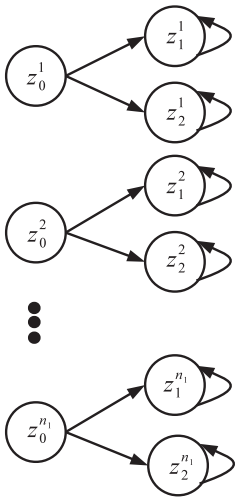

Figure 6: The class of MDPs considered in the proof of Theorem 7. Nodes represent state-action pairs and arrows show transitions between the nodes (see the text for details).

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

We consider a set of $n_1 + 1$ MDPs in $\mathbb{M}$, $\mathbb{M}^* = \{M_0, M_1, \ldots, M_{n_1}\}$ with transition probabilities $p_{M_0}(z_0^i) = 1/2,\ i = 1, \ldots, n_1$, and for $l = 1, \ldots, n_1,\ p_{M_l}(z_0^l) = 1/2 + 2\epsilon/\gamma\beta$ and $p_{M_l}(z_0^i) = 1/2,\ i = 1, \ldots, l-1, l+1, \ldots, n_1$. Note that here we make the assumption that

$$\frac{2\epsilon}{\gamma\beta} \le \frac{1}{2}. \tag{14}$$

For the MDPs in $\mathbb{M}^*$, we have

$$Q^*_{M_0}(z) = \begin{cases} \gamma\beta/2 & z \in \mathcal{Z}_0, \\ \beta & z \in \mathcal{Z}_1, \\ 0 & z \in \mathcal{Z}_2. \end{cases} \qquad Q^*_{M_l}(z) = \begin{cases} \gamma\beta/2 + 2\epsilon & z = z_0^l, \\ \gamma\beta/2 & z \in \mathcal{Z}_0,\ z \neq z_0^l, \\ \beta & z \in \mathcal{Z}_1, \\ 0 & z \in \mathcal{Z}_2. \end{cases}$$

We define $\epsilon_0 = 1/8$, $\delta_0 = e^{-4}/2$, and $\gamma_0 = 0.41$. From now on, we fix some $\epsilon \in (0, \epsilon_0)$, $\delta \in (0, \delta_0)$, $\gamma \in (\gamma_0, 1)$, and an algorithm $\mathfrak{A} \in \mathbb{A}$, which we assume to be (ϵ, δ, T^*)-correct. We denote by $\mathbb{E}_{M_l}$ and $\mathbb{P}_{M_l}$ the expectation and probability for MDP M_l. We now define

$$t^* = \frac{\beta^2}{c\epsilon^2} \log \frac{1}{2\delta},$$

where c is a constant whose value will be determined later. We denote by T_l the number of times that the state-action pair z_0^l is tried during the execution of the algorithm $\mathfrak{A}$. We

assume that for $l \neq 0$, we have $T_l \leq t^*$. We will show that under this assumption, the algorithm $\mathfrak{A}$ violates (ϵ, δ, T^*)-correctness. This will then follow that we must have $T_l > t^*$ for $l \neq 0$. Without loss of generality we assume that the above condition holds for $l = 1$, so we have $T_1 \leq t^*$. We define $K_t = X_1^1 + \ldots + X_t^1$, where $X_i^1 = 1$ if after the i'th execution of state-action pair z_0^1, the algorithm ends up in a state-action pair in $\mathcal{Z}_1$, and $X_i^1 = 0$ if it ends up in a state-action pair in $\mathcal{Z}_2$. We let $\mathcal{E}_1$ be the event defined by

$$\mathcal{E}_1 = \left\{ \max_{1 \leq t \leq t^*} |2K_t - t| < \sqrt{t^* \log(\frac{1}{2\delta})} \right\}.$$

Similar to Lemma 2 in Mannor and Tsitsiklis (2004), we can show that $\mathbb{P}_{M_0}(\mathcal{E}_1) > 3/4$, in which we have used the fact that $\delta < e^{-4}/2$. We now define $\mathcal{E}_2$ as the event that[13]

$$\forall T > T^*, \qquad \frac{\gamma\beta}{2} - \epsilon < Q_T^{\mathfrak{A}}(z_0^1) < \frac{\gamma\beta}{2} + \epsilon. \tag{15}$$

Since under event $\mathcal{E}_2$, we have $|Q_{M_0}^*(z_0^1) - Q_T^{\mathfrak{A}}(z_0^1)| \leq \epsilon$, we may write $\mathbb{P}_{M_0}(\mathcal{E}_2) \geq (1-\delta)$, and since $\delta < e^{-4}/2$ then $\mathbb{P}_{M_0}(\mathcal{E}_2) \geq 3/4$. Let $\mathcal{E}$ be the event that $\mathcal{E}_1$ and $\mathcal{E}_2$ occur, i.e., $\mathcal{E} = \mathcal{E}_1 \cap \mathcal{E}_2$. We then have $\mathbb{P}_{M_0}(\mathcal{E}) > 1/2$.

We now show that if $T_1 \leq t^*$ and $c \geq 100$, then $\mathbb{P}_{M_1}(\mathcal{E}_2) > \delta$. Let W be the history of all the outcomes of trying state-action pair z_0^1 for T_1 times. We define the likelihood function $L_{M_l}(W) = \mathbb{P}_{M_l}(W = w)$ for every possible history w. Given the definition of the likelihood function the likelihood ratio $L_{M_1}(W)/L_{M_0}(W)$ may be written as

$$\frac{L_{M_1}(W)}{L_{M_0}(W)} = \frac{(\frac{1}{2} + \frac{2\epsilon}{\gamma\beta})^{K_1}(\frac{1}{2} - \frac{2\epsilon}{\gamma\beta})^{T_1 - K_1}}{(\frac{1}{2})^{T_1}} = (1 - \frac{16\epsilon^2}{\gamma^2\beta^2})^{K_1}(1 - \frac{4\epsilon}{\gamma\beta})^{T_1 - 2K_1}, \tag{16}$$

where we used K_1 as a shorthand notation for K_{T_1}. We now proceed to lower-bound the terms on the right hand side of Eq. 16 when event $\mathcal{E}$ occurs. Since $K_1 \leq T_1 \leq t^*$, we have

$$\begin{aligned}(1 - \frac{16\epsilon^2}{\gamma^2\beta^2})^{K_1} &\geq (1 - \frac{16\epsilon^2}{\gamma^2\beta^2})^{t^*} = (1 - \frac{16\epsilon^2}{\gamma^2\beta^2})^{\left(\beta^2/(c\epsilon^2)\right)\log(1/2\delta)} \\ &\overset{(a)}{\geq} e^{-\left((16d)/(c\gamma^2)\right)\log(1/2\delta)} = (2\delta)^{(16d)/(c\gamma^2)},\end{aligned} \tag{17}$$

where in **(a)** we used Lemma 3 in Mannor and Tsitsiklis (2004), and thus, $d = 1.78$ and we should assume that

$$\frac{16\epsilon^2}{\gamma^2\beta^2} \leq \frac{1}{\sqrt{2}}. \tag{18}$$

Similarly, if event $\mathcal{E}$ has occurred, then event $\mathcal{E}_1$ has occurred, which implies

$$T_1 - 2K_1 \leq \sqrt{t^* \log(\frac{1}{2\delta})} = \frac{\beta \log(\frac{1}{2\delta})}{\epsilon\sqrt{c}}.$$

13. Note that here the scenario is that the algorithm $\mathfrak{A}$ has been executed for the total number of steps $T > T^*$, while the total number of times that the state-action pair z_0^1 has been executed is $T_1 \leq t^*$.

Therefore, we have

$$(1-\frac{4\epsilon}{\gamma\beta})^{T_1-2K_1} \geq (1-\frac{4\epsilon}{\gamma\beta})^{\left(\beta/(\epsilon\sqrt{c})\right)\log(1/2\delta)} \overset{(b)}{\geq} e^{-\left((4d)/(\gamma\sqrt{c})\right)\log(1/2\delta)} = (2\delta)^{(4d)/(\sqrt{c}\gamma)}, \quad (19)$$

where in **(b)** we assumed that

$$\frac{4\epsilon}{\gamma\beta} \leq \frac{1}{\sqrt{2}}. \quad (20)$$

Note that the assumption in Eq. 20 implies the other two assumptions in Eqs. 14 and 18. Thus, we only need to guarantee that the assumption of Eq. 20 holds (selected $\epsilon_0 = 1/8$ and $\gamma_0 = 0.41$ guarantee that). Now substituting Eqs. 17 and 19 in Eq. 16, we obtain

$$\frac{L_{M_1}(W)}{L_{M_0}(W)} \geq (2\delta)^{\left((16d)/(c\gamma^2)\right)+\left((4d)/(\sqrt{c}\gamma)\right)}.$$

By selecting the constant c large enough ($c\gamma^2 = 100$ suffices), we obtain that $L_{M_1}(W)/L_{M_0}(W)$ is larger than 2δ, whenever event $\mathcal{E}$ occurs. So, we may write

$$\frac{L_{M_1}(W)}{L_{M_0}(W)}\mathbf{1}_{\mathcal{E}} \geq 2\delta\mathbf{1}_{\mathcal{E}},$$

where $\mathbf{1}_{\mathcal{E}}$ is the indicator function of event $\mathcal{E}$. Then, we have

$$\mathbb{P}_{M_1}(\mathcal{E}_2) \geq \mathbb{P}_{M_1}(\mathcal{E}) = \mathbb{E}_{M_1}[\mathbf{1}_{\mathcal{E}}] = \mathbb{E}_{M_0}\left[\frac{L_{M_1}(W)}{L_{M_0}(W)}\mathbf{1}_{\mathcal{E}}\right] \geq \mathbb{E}_{M_0}[2\delta\mathbf{1}_{\mathcal{E}}] = 2\delta\mathbb{P}_{M_0}(\mathcal{E}) > \delta,$$

where we used the fact that $\mathbb{P}_{M_0}(\mathcal{E}) > 1/2$. Since the event $\mathcal{E}_2$, i.e., $\forall T > T^*,\ \frac{\gamma\beta}{2} - \epsilon < Q_T^{\mathfrak{A}}(z_0^1) < \frac{\gamma\beta}{2} + \epsilon$ implies that $\forall T > T^*,\ |Q^*_{M_1}(z_0^1) - Q_T^{\mathfrak{A}}(z_0^1)| > \epsilon$, the fact that $\mathbb{P}_{M_1}(\mathcal{E}_2) > \delta$ implies that if $T_1 \leq t^*$ then $\mathbb{P}_{M_1}\left(\{\forall T > T^*,\ |Q^*_{M_1}(z_0^1) - Q_T^{\mathfrak{A}}(z_0^1)| \leq \epsilon\}\right) < 1-\delta$ which contradicts the assumption that the algorithm $\mathfrak{A}$ is (ϵ, δ, T^*) correct. Therefore, the algorithm $\mathfrak{A}$ is (ϵ, δ, T^*) correct if $T_1 > t^*$. By repeating this process for $l = 1, \ldots, n_1$, we obtain

$$\text{if} \quad \mathbb{P}_{M_l}\left(\{\forall T > T^*,\ |Q^*_{M_l}(z_0^l) - Q_T^{\mathfrak{A}}(z_0^l)| \leq \epsilon\}\right) \geq 1-\delta, \text{then} \quad T_l > t^* \qquad l = 1, \ldots, n_1.$$

The result then follows by summing up T_l's for all state-action pairs $z_0^l \in \mathcal{Z}_0$.

6. Conclusions and Future Work

In this paper, we presented a new reinforcement learning (RL) 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 results are in the form of high probability bounds on the performance loss of SQL, which suggest 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 $\beta = 1/(1-\gamma)$, SQL outperforms the standard Q-learning by a wide margin. Overall, SQL is a simple, efficient, and theoretically well-founded RL algorithm that improves on the existing similar methods such as Q-learning and 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 results to PAC-MDP methods (Strehl et al., 2009; Szita and Szepesvári, 2010) and upper-confidence bound based algorithms (Bartlett and Tewari, 2009; Jaksch et al., 2010), in which the choice of the exploration policy has an influence on the behavior of the learning algorithm. 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 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 leave this as a subject for future work.

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 the current SQL analysis to the continuous case along the same line as in the fitted value iteration analysis by Antos et al. (2007) and Munos and Szepesvári (2008).

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.

S. M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, 2004.

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.