Journal of Machine Learning Research 10 (2009) 2413-2444 Submitted 11/06; Revised 12/08; Published 11/09

Reinforcement Learning in Finite MDPs: PAC Analysis

Alexander L. Strehl[*] ASTREHL@FACEBOOK.COM
Facebook
1601 S California Ave.
Palo Alto, CA 94304

Lihong Li[†] LIHONG@YAHOO-INC.COM
Yahoo! Research
4401 Great America Parkway
Santa Clara, CA 95054

Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University
Piscataway, NJ 08854

Editor: Sridhar Mahadevan

Abstract

We study the problem of learning near-optimal behavior in finite Markov Decision Processes (MDPs) with a polynomial number of samples. These "PAC-MDP" algorithms include the well-known E^3 and R-MAX algorithms as well as the more recent Delayed Q-learning algorithm. We summarize the current state-of-the-art by presenting bounds for the problem in a unified theoretical framework. A more refined analysis for upper and lower bounds is presented to yield insight into the differences between the model-free Delayed Q-learning and the model-based R-MAX.

Keywords: reinforcement learning, Markov decision processes, PAC-MDP, exploration, sample complexity

1. Introduction

In the reinforcement-learning (RL) problem (Sutton and Barto, 1998), an agent acts in an unknown or incompletely known environment with the goal of maximizing an external reward signal. In the most standard mathematical formulation of the problem, the environment is modeled as a finite Markov Decision Process (MDP) where the goal of the agent is to obtain near-optimal discounted return. Recent research has dealt with probabilistic bounds on the number of samples required for near-optimal learning in finite MDPs (Kearns and Singh, 2002; Kakade, 2003; Brafman and Tennenholtz, 2002; Strehl and Littman, 2005; Strehl et al., 2006a,b). The purpose of this paper is to summarize this field of knowledge by presenting the best-known upper and lower bounds for the problem. For the upper bounds, we present constructive proofs using a unified framework in Section 3.1; these tools may be useful for future analysis. While none of the bounds we present are entirely new, the main contribution of this paper is to streamline as well as consolidate their

∗. Some of this work was completed while A. Strehl was at Rutgers University and also while he was at Yahoo! Research.

†. Some of this work was completed while L. Li was at Rutgers University.

analyses. In addition, the bounds we present are stated in terms of an *admissible* heuristic provided to the algorithm (see Section 1.3) and the (unknown) optimal value function. These bounds are more refined than the ones previously presented in the literature and more accurately reflect the performance of the corresponding algorithms. For the lower bound, we provide an improved result that matches the upper bound in terms of the number of states of the MDP.

An outline of the paper is as follows. This introduction section concludes with a formal specification of the problem and related work. In Section 2, R-MAX and Delayed Q-learning are described. Then, we present their analyses and prove PAC-MDP upper bounds in Section 3. A new lower bound is proved in Section 4.

1.1 Main Results

We present two upper bounds and one lower bound on the achievable *sample complexity* of general reinforcement-learning algorithms (see Section 1.5 for a formal definition). The two upper bounds dominate all previously published bounds, but differ from one another. When logarithmic factors are ignored, the first bound, for the R-MAX algorithm, is

$$\tilde{O}(S^2A/(\varepsilon^3(1-\gamma)^6)),$$

while the corresponding second bound, for the Delayed Q-learning algorithm, is

$$\tilde{O}(SA/(\varepsilon^4(1-\gamma)^8)).$$

Here, S and A are the number of states and the number of actions, respectively, of the MDP, ε and δ are accuracy parameters, and γ is a discount factor. R-MAX works by building an approximate MDP model and the S^2A term in its sample complexity follows from requiring accuracy in each of the S^2A parameters of the model. Delayed Q-learning, on the other hand, does not build an explicit model and can be viewed as an approximate version of value iteration. Thus, accuracy only needs to be guaranteed for each of the SA entries in the value function.

While previous bounds are in terms of an upper bound $1/(1-\gamma)$ on the value function, we find that tighter bounds are possible if a more informative value-function upper bound is given. Specifically, we can rewrite the bounds in terms of the initial admissible heuristic values (see Section 1.3) supplied to the algorithms, $U(\cdot,\cdot)$, and the true (unknown) value function $V^*(\cdot)$. Ignoring logarithmic factors, for R-MAX the bound is

$$\tilde{O}\left(\frac{V_{\max}^3 S\,|\{(s,a)\in\mathcal{S}\times\mathsf{A}|U(s,a)\geq V^*(s)-\varepsilon\}|}{\varepsilon^3(1-\gamma)^3}\right), \tag{1}$$

and for Delayed Q-learning

$$\tilde{O}\left(\frac{V_{\max}^3 \sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}[U(s,a)-V^*(s)]_+}{\varepsilon^4(1-\gamma)^4}\right), \tag{2}$$

where $V_{\max}\geq \max_{s,a}U(s,a)$ is an upper bound on the admissible heuristic (and also on the true value function), and $[x]_+$ is defined as $\max(0,x)$ for $x\in\mathbb{R}$. Thus, we observe that for R-MAX one factor of $SA/(1-\gamma)^3$ gets replaced by $|\{(s,a):U(s,a)\geq V^*(s)-\varepsilon\}|V_{\max}^3$,[1] the number of state-action pairs whose heuristic initial value is larger than $V^*-\varepsilon$, while for Delayed Q-learning the

1. This quantity can be as small as $SV_{\max}^3$ and as large as $SAV_{\max}^3$, where $V_{\max}\in[0,\frac{1}{1-\gamma}]$.

factor $SA/(1-\gamma)^4$ is replaced by $V_{\max}^3 \sum_{(s,a)\in S\times A}(U(s,a)-V^*(s))$,[2] $V_{\max}^3$ times the total sum of differences between the heuristic values and the optimal value function. The latter term is better, because it takes more advantage of accurate heuristics. For instance, if $U(s,a)=V^*(s)+\varepsilon$ and $V^*(s)$ is large for all s, then the bound for R-MAX stays essentially the same but the one for Delayed Q-learning is greatly improved. Please see Russell and Norvig (1994) for discussions and references on admissible heuristics. The method of incorporating admissible heuristics into Q-learning (Ng et al., 1999) and R-MAX (Asmuth et al., 2008) are well known, but the bounds given in Equation 1 and Equation 2 are new.

The upper bounds summarized above may be pessimistic and thus may not reflect the worst-case behavior of these algorithms. Developing lower bounds, especially *matching* lower bounds, tells us what can (or cannot) be achieved. Although matching lower bounds are known for deterministic MDPs (Koenig and Simmons, 1996; Kakade, 2003), it remains an open question for general MDPs. The previous best lower bound is due to Kakade (2003), and was developed for the slightly different notion of H-horizon value functions instead of the γ-discounted ones we focus on here. Adapting his analysis to discounted value functions, we get the following lower bound:

$$\Omega\left(\frac{SA}{\varepsilon(1-\gamma)^2}\ln\frac{1}{\delta}\right).$$

Based on the work of Mannor and Tsitsiklis (2004), we provide an improved lower bound

$$\Omega\left(\frac{SA}{\varepsilon^2}\ln\frac{S}{\delta}\right) \tag{3}$$

which simultaneously increases the dependence on both S and $1/\varepsilon$. While we choose to drop dependence on $1/(1-\gamma)$ in our lower bound to facilitate a cleaner analysis, we believe it is possible to force a quadratic dependence by a more careful analysis. This new lower bound (3) has a few important implications. First, it implies that Delayed Q-learning's worst-case sample complexity has the *optimal* dependence on S. Second, it increases the dependence on $1/\varepsilon$ significantly from linear to quadratic. It would be interesting to know whether a cubic dependence on $1/\varepsilon$ is possible, which would match the upper bound for R-MAX (ignoring logarithmic factors).

Our lower bound is tight for the factors S, $1/\varepsilon$, and $1/\delta$, in the weaker *parallel sampling* model (Kearns and Singh, 1999). This finding suggests that a worse dependence on $1/\varepsilon$ is possible only in MDPs with slow *mixing* rates.[3] In both the parallel sampling model and the MDP used to prove the lower bound given by Equation 3 (see Section 4), the distribution of states being sampled/visited mixes extremely fast (in one and two timesteps, respectively). The slower the mixing rate, the more difficult the *temporal credit assignment* problem (Sutton and Barto, 1998). In other words, a worse dependence on $1/\varepsilon$ may require the construction of an MDP where *deep planning* is necessary.

Before finishing the informal introduction, we should point out that the present paper focuses on *worst-case* upper bounds and so the sample complexity of exploration bounds like Equations 1 and 2 can be too conservative for MDPs encountered in practice. However, the algorithms and their analyses have proved useful for guiding development of more practical exploration schemes as well as improved algorithms. First of all, these algorithms formalize the principle of "optimism under the

2. This quantity can be as small as 0 and as large as $SAV_{\max}^4$, where $V_{\max}\in[0,\frac{1}{1-\gamma}]$.

3. There are many ways to define a mixing rate. Roughly speaking, it measures how fast the distribution of states an agent reaches becomes independent of the initial state and the policy being followed.

face of uncertainty" (Brafman and Tennenholtz, 2002) which has been empirically observed to be effective for encouraging active exploration (Sutton and Barto, 1998). Sample complexity analysis not only shows soundness of this principle in a mathematically precise manner, but also motivates novel RL algorithms with efficient exploration (e.g., Nouri and Littman 2009 and Li et al. 2009). Second, there are several places in the proofs where the analysis can be tightened under various assumptions about the MDP. The use of admissible heuristic functions as discussed above is one example; another example is the case where the number of next states reachable from any state-action pair is bounded by a constant, implying the factor S in Equation 1 may be shaved off (cf., Lemma 14). More opportunities lie in MDPs with various structural assumptions. Examples include factored-state MDPs (Kearns and Koller, 1999; Strehl et al., 2007; Diuk et al., 2009), Relocatable Action Models (Leffler et al., 2007), and Object-Oriented MDPs (Walsh et al., 2009), in all of which an exponential reduction in sample complexity can be achieved, as well as in MDPs where prior information about the model is available (Asmuth et al., 2009). Third, the streamlined analysis we present here is very general and applies not only to finite MDPs. Similar proof techniques have found useful in analyzing model-based algorithms for continuous-state MDPs whose dynamics are linear (Strehl and Littman, 2008a) or multivariate normal (Brunskill et al., 2008); see Li (2009) for a survey.

1.2 Markov Decision Processes

This section introduces the Markov Decision Process (MDP) notation used throughout the paper; see Sutton and Barto (1998) for an introduction. Let $\mathcal{P}_X$ denote the set of probability distributions over the set X. A finite MDP M is a five tuple $\langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a finite set called the state space, A is a finite set called the action space, $T : \mathcal{S} \times \mathsf{A} \to \mathcal{P}_{\mathcal{S}}$ is the transition distribution, $\mathcal{R} : \mathcal{S} \times \mathsf{A} \to \mathcal{P}_{\mathbb{R}}$ is the reward distribution, and $0 \leq \gamma < 1$ is a discount factor on the summed sequence of rewards. We call the elements of $\mathcal{S}$ and A states and actions, respectively, and use S and A to denote the number of states and the number of actions, respectively. We let $T(s'|s,a)$ denote the transition probability of state s' of the distribution $T(s,a)$. In addition, $R(s,a)$ denotes the expectation of the distribution $\mathcal{R}(s,a)$.

We assume that the learner (also called the *agent*) receives S, A, and γ as input. The learning problem is defined as follows. The agent always occupies a single state s of the MDP M. The agent is told this state and must choose an action a. It then receives an *immediate reward* $r \sim \mathcal{R}(s,a)$ and is transported to a *next state* $s' \sim T(s,a)$. This procedure then repeats forever. The first state occupied by the agent may be chosen arbitrarily. Intuitively, the solution or goal of the problem is to obtain as large as possible reward in as short as possible time. In Section 1.5, we provide one possible formalization of this objective within the PAC-MDP framework. We define a *timestep* to be a single interaction with the environment, as described above. The t^{th} timestep encompasses the process of choosing the t^{th} action. We also define an *experience* of state-action pair (s,a) to refer to the event of taking action a from state s.

A *policy* is any strategy for choosing actions. A stationary policy is one that produces an action based on only the current state, ignoring the rest of the agent's history. We assume (unless noted otherwise) that rewards[4] all lie in the interval $[0,1]$. For any policy π, let $V_M^\pi(s) = \mathbf{E}[\sum_{j=1}^{\infty} \gamma^{j-1} r_j | s]$ ($Q_M^\pi(s,a) = \mathbf{E}[\sum_{j=1}^{\infty} \gamma^{j-1} r_j | s,a]$) denote the discounted, infinite-horizon value (action-value) func-

4. It is easy to generalize, by linear transformations (Ng et al., 1999), to the case where the rewards are bounded above and below by known but arbitrary constants without changing the optimal policy.

tion for π in M (which may be omitted from the notation) from state s. If H is a positive integer, let $V_M^\pi(s,H)$ denote the H-step value of policy π from s. If π is non-stationary, then s is replaced by a *partial path* $c_t = (s_1, a_1, r_1, \ldots, s_t)$, in the previous definitions. Specifically, let s_t and r_t be the t^{th} encountered state and received reward, respectively, resulting from execution of policy π in some MDP M. Then, $V_M^\pi(c_t) = \mathbf{E}[\sum_{j=0}^{\infty} \gamma^j r_{t+j} | c_t]$ and $V_M^\pi(c_t, H) = \mathbf{E}[\sum_{j=0}^{H-1} \gamma^j r_{t+j} | c_t]$. These expectations are taken over all possible infinite paths the agent might follow in the future. The optimal policy is denoted π^* and has value functions $V_M^*(s)$ and $Q_M^*(s,a)$. Note that a policy cannot have a value greater than $1/(1-\gamma)$ by the assumption that the maximum reward is 1.[5]

1.3 Admissible Heuristics

We also assume that the algorithms are given an admissible heuristic for the problem before learning occurs. An **admissible heuristic** is a function $U : \mathcal{S} \times \mathsf{A} \to \mathbb{R}$ that satisfies $U(s,a) \geq Q^*(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathsf{A}$. We also assume that $U(s,a) \leq V_{\max}$ for all $(s,a) \in \mathcal{S} \times \mathsf{A}$ and some quantity $V_{\max}$. Prior information about the problem at hand can be encoded into the admissible heuristic and its upper bound $V_{\max}$. With no prior information, we can always set $U(s,a) = V_{\max} = 1/(1-\gamma)$ since $V^*(s) = \max_{a \in \mathsf{A}} Q^*(s,a)$ is at most $1/(1-\gamma)$. Therefore, without loss of generality, we assume $0 \leq U(s,a) \leq V_{\max} \leq 1/(1-\gamma)$ for all $(s,a) \in \mathcal{S} \times \mathsf{A}$.

1.4 A Note on the Use of Subscripts

Each algorithm that we consider maintains several variables. For instance, an *action value* or *action-value estimate*, $Q(s,a)$, sometimes called a *Q-value*, where (s,a) is any state-action pair, is maintained. We will often discuss a particular instance or time t during the execution of the algorithm. In this case, when we refer to $Q(s,a)$ we mean the value of that variable at the current moment. To be more explicit, we may write $Q_t(s,a)$, which refers to the value of $Q(s,a)$ immediately preceding the t^{th} action of the agent. Thus, $Q_1(s,a)$ is the initial value of $Q(s,a)$.

1.5 PAC-MDP Model

There are three essential ways to quantify the performance of a reinforcement-learning algorithm. They are *computational complexity*, the amount of per-timestep computation the algorithm uses during learning; *space complexity*, the amount of memory used by the algorithm; and *learning complexity*, a measure of how much experience the algorithm needs to learn in a given task. The last of these is difficult to define and several different ideas have been discussed in the literature. On the one hand, requiring an algorithm to "optimally explore"—meaning to obtain maximum expected discounted reward ($\mathbf{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$) over a known prior of MDPs—is an extremely difficult task tractable only in highly specialized cases (Gittins, 1989). Thus, we consider the relaxed but still challenging and useful goal of acting near-optimally on all but a polynomial number of steps (Kakade, 2003; Strehl and Littman, 2008b).

To formalize the notion of "efficient learning", we allow the learning algorithm to receive two additional inputs, ε and δ, both positive real numbers. The first parameter, ε, controls the quality of behavior we require of the algorithm (how close to optimality do we want the algorithm to be) and the second parameter, δ, is a measure of confidence (how certain do we want to be of the algorithm's

5. Thus, when comparing our results to the original R-MAX paper Brafman and Tennenholtz (2002), note that 1 takes the place of the quantity *Rmax*.

performance). As these parameters approach zero, greater exploration and learning is necessary, as higher quality is demanded of the algorithms.

In the following definition, we view an algorithm as a non-stationary (in terms of the current state) policy that, on each timestep, takes as input an entire history or trajectory through the MDP (its actual history) and outputs an action (which the agent then executes). Formally, we define the policy of any algorithm $\mathcal{A}$ at a fixed instance in time t to be a function $\mathcal{A}_t : \{\mathcal{S} \times \mathsf{A} \times [0,1]\}^* \times \mathcal{S} \to \mathsf{A}$, that maps future paths to future actions.[6]

Definition 1 *(Kakade 2003) Let $c = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots)$ be a random path generated by executing an algorithm $\mathcal{A}$ in an MDP M. For any fixed $\varepsilon > 0$, the* **sample complexity of exploration** *(**sample complexity**, for short) of $\mathcal{A}$ is the number of timesteps t such that the policy at time t, $\mathcal{A}_t$, satisfies $V^{\mathcal{A}_t}(s_t) < V^*(s_t) - \varepsilon$.*

Note that the sample complexity of an algorithm is dependent on some infinite-length path through the MDP. We believe this definition captures the essence of measuring learning. It directly measures the number of times the agent acts poorly (quantified by ε) and we view "fast" learners as those that act poorly as few times as possible. Based on this intuition, we define what it means to be an "efficient" learning algorithm.

Definition 2 *An algorithm $\mathcal{A}$ is said to be an* **efficient PAC-MDP** *(Probably Approximately Correct in Markov Decision Processes) algorithm if, for any $\varepsilon > 0$ and $0 < \delta < 1$, the per-timestep computational complexity, space complexity, and the sample complexity of $\mathcal{A}$ are less than some polynomial in the relevant quantities $(S, A, 1/\varepsilon, 1/\delta, 1/(1-\gamma))$, with probability at least $1-\delta$. It is simply* **PAC-MDP** *if we relax the definition to have no computational complexity requirement.*

The terminology, PAC, in this definition is borrowed from Angluin (1988) for the distribution-free supervised-learning model of Valiant (1984). One thing to note is that we only require a PAC-MDP algorithm to behave poorly (non-ε-optimally) on no more than a small (polynomially) number of timesteps. We do not place any limitations on when the algorithm acts poorly or how poorly it acts on those timesteps. This definition is in contrast to Valiant's PAC notion, which is more "off-line" in that it requires the algorithm to make all of its mistakes ahead of time (during the learning phase) before identifying a near-optimal policy. The notion of PAC-MDP is also closely related to the Mistake Bound (MB) model of Littlestone (1988) where the goal of a learner that predicts sequentially must make a small (polynomial) number of mistakes during a whole run. Indeed, if we count every timestep in which an algorithm behaves non-ε-optimally as a mistake, then a PAC-MDP algorithm makes only a polynomial number of mistakes during a whole run with high probability, similar to an MB algorithm. However, a mistake in a PAC-MDP algorithm refers to the quality of a policy rather than prediction errors as in MB.

Efficient learnability in the sample-complexity framework from above implies efficient learnability in a more realistic framework called *Average Loss* that measures the actual return (sum of rewards) achieved by the agent against the expected return of the optimal policy (Strehl and Littman, 2008b). The analysis of R-MAX by Kakade (2003) and of MBIE by Strehl and Littman (2005) use the same definition as above. The analysis of R-MAX by Brafman and Tennenholtz (2002) and of

6. The action of an agent on timestep t in state s_t is given by the function evaluated at the empty history, $\mathcal{A}_t(\emptyset, s_t)$.

E^3 by Kearns and Singh (2002) use slightly different definitions of efficient learning.[7] Our analyses are essentially equivalent, but simpler in the sense that mixing-time arguments are avoided. Compared with recently published regret bounds (Auer et al., 2009), our sample complexity bounds are easier to obtain and do not depend on quantities like mixing time or diameter that may be hard to determine *a priori*.

1.6 Related Work

There has been some theoretical work analyzing RL algorithms. In a Bayesian setting, with a known prior over possible MDPs, we could ask for the policy that maximizes expected reward. This problem has been solved (Gittins, 1989) for a specialized class of MDPs, called *K*-armed bandits. However, a solution to the more general problem seems unlikely to be tractable, although progress has been made (Duff and Barto, 1997; Poupart et al., 2006).

Early results include proving that under certain conditions various algorithms can, in the limit, compute the optimal value function from which the optimal policy can be extracted (Watkins and Dayan, 1992). These convergence results make no performance guarantee after only a finite amount of experience, although more recent work has looked at convergence rates (Szepesvári, 1998; Kearns and Singh, 1999; Even-Dar and Mansour, 2003). These types of analyses make assumptions that simplify the exploration issue.

The work by Fiechter (1994) was the first to prove that efficient (polynomial) approximate learning is achievable, via a model-based algorithm, when the agent has an action that *resets* it to a distinguished start state. Other recent work has shown that various model-based algorithms, including E^3 (Kearns and Singh, 2002), R-MAX (Brafman and Tennenholtz, 2002), and MBIE (Strehl and Littman, 2005) can achieve polynomial learning guarantees without the necessity of resets.

2. Algorithms

The total number of RL algorithms introduced in the literature is huge, so we limit the study to those with the best formal PAC-MDP learning-time guarantees. The two algorithms we study are R-MAX and Delayed Q-learning, because the best sample complexity bounds known for any PAC-MDP algorithm are dominated by the bound for one of these two algorithms. However, the bounds for R-MAX and Delayed Q-learning are incomparable—the bound for R-MAX is better in terms of $1/\varepsilon$ and $1/(1-\gamma)$, while the bound for Delayed Q-learning is better in terms of S. In fact, in Section 4 we will show that the sample complexity of Delayed Q-learning is optimal in terms of S via a matching lower bound.

2.1 R-MAX

Suppose that the agent has acted for some number of timesteps and consider its experience with respect to some fixed state-action pair (s,a). Let $n(s,a)$ denote the number of timesteps in which the agent has taken action a from state s. Suppose the agent has observed the following $n(s,a)$ immediate rewards for taking action a from state s: $r[1],r[2],\ldots,r[n(s,a)]$. Then, the empirical

7. Kearns and Singh (2002) dealt with discounted and undiscounted MDPs differently. In the discounted case the agent is required to halt after a polynomial amount of time and output a near-optimal policy from the current state, with high probability.

mean reward is

$$\hat{R}(s,a) := \frac{1}{n(s,a)} \sum_{i=1}^{n(s,a)} r[i].$$

Let $n(s,a,s')$ denote the number of times the agent has taken action a from state s and immediately transitioned to the state s'. Then, the *empirical transition distribution* is the distribution $\hat{T}(s,a)$ satisfying

$$\hat{T}(s'|s,a) := \frac{n(s,a,s')}{n(s,a)} \text{ for each } s' \in S.$$

In the R-MAX algorithm, the action-selection step is always to choose the action that maximizes the current action value, $Q(s,\cdot)$. The update step is to solve the following set of Bellman equations:

$$\begin{aligned} Q(s,a) &= \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s'|s,a) \max_{a'} Q(s',a'), && \text{if } n(s,a) \geq m, \\ Q(s,a) &= U(s,a), && \text{otherwise,} \end{aligned} \tag{4}$$

where $\hat{R}(s,a)$ and $\hat{T}(\cdot|s,a)$ are the empirical (maximum-likelihood) estimates for the reward and transition distribution of state-action pair (s,a) using only data from the first m observations of (s,a). Solving this set of equations is equivalent to computing the optimal action-value function of an MDP, which we call *Model(R-MAX)*. This MDP uses the empirical transition and reward distributions for those state-action pairs that have been experienced by the agent at least m times. Rather than attempt to model the other state-action pairs, we assert their value to be $U(s,a)$, which is guaranteed to be an upper bound on the true value function. An important point is that R-MAX uses *only the first m samples* in the empirical model. To avoid complicated notation, we redefine $n(s,a)$ to be the minimum of m and the number of times state-action pair (s,a) has been experienced. This usage is consistent with the pseudo-code provided in Algorithm 1. That is, the computation of $\hat{R}(s,a)$ and $\hat{T}(s'|s,a)$ in Equation 4, uses only the first $n(s,a) = m$ samples.

Any implementation of R-MAX must choose a technique for solving the set of Equations 4 such as dynamic programming and linear programming approaches (Puterman, 1994), and this choice will affect the computational complexity of the algorithm. However, for concreteness we choose *value iteration* (Puterman, 1994), a relatively simple and fast MDP solving routine that is widely used in practice. Rather than require exact solution of Equations 4, a more practical approach is to only guarantee a near-optimal greedy policy. The following two classic results are useful in quantifying the number of iterations needed.

Proposition 3 *(Corollary 2 from Singh and Yee 1994) Let $Q'(\cdot,\cdot)$ and $Q^*(\cdot,\cdot)$ be two action-value functions over the same state and action spaces. Suppose that Q^* is the optimal value function of some MDP M. Let π be the greedy policy with respect to Q' and π^* be the greedy policy with respect to Q^*, which is the optimal policy for M. For any $\alpha > 0$ and discount factor $\gamma < 1$, if $\max_{s,a} \{|Q'(s,a) - Q^*(s,a)|\} \leq \alpha(1-\gamma)/2$, then $\max_s \{V^{\pi^*}(s) - V^{\pi}(s)\} \leq \alpha$.*

Proposition 4 *Let $\beta > 0$ be any real number satisfying $\beta < 1/(1-\gamma)$ where $\gamma < 1$ is the discount factor. Suppose that value iteration is run for $\left\lceil \frac{\ln(1/(\beta(1-\gamma)))}{1-\gamma} \right\rceil$ iterations where each initial action-value estimate, $Q(\cdot,\cdot)$, is initialized to some value between 0 and $1/(1-\gamma)$. Let $Q'(\cdot,\cdot)$ be the resulting action-value estimates. Then, we have that $\max_{s,a} \{|Q'(s,a) - Q^*(s,a)|\} \leq \beta$.*

Proof Let $Q_i(s,a)$ denote the action-value estimates after the i^{th} iteration of value iteration.[8] Let $\Delta_i := \max_{(s,a)} |Q^*(s,a) - Q_i(s,a)|$. Now, we have that

$$\begin{aligned}
\Delta_i &= \max_{(s,a)} |(R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*(s')) - (R(s,a) + \gamma \sum_{s'} T(s,a,s') V_{i-1}(s'))| \\
&= \max_{(s,a)} |\gamma \sum_{s'} T(s,a,s')(V^*(s') - V_{i-1}(s'))| \\
&\leq \gamma \Delta_{i-1}.
\end{aligned}$$

Using this bound along with the fact that $\Delta_0 \leq 1/(1-\gamma)$ shows that $\Delta_i \leq \gamma^i/(1-\gamma)$. Setting this value to be at most β and solving for i yields $i \geq \frac{\ln(\beta(1-\gamma))}{\ln \gamma}$. We claim that

$$\frac{\ln \frac{1}{\beta(1-\gamma)}}{1-\gamma} \geq \frac{\ln(\beta(1-\gamma))}{\ln(\gamma)}. \tag{5}$$

Note that Equation 5 is equivalent to the statement $1-\gamma \leq -\ln \gamma$, which follows from the identity $e^x \geq 1+x$. ■

The previous two propositions imply that if we require value iteration to produce an α-optimal policy it is sufficient to run it for $O\left(\frac{\ln(1/(\alpha(1-\gamma)))}{1-\gamma}\right)$ iterations. The resulting pseudo-code for R-MAX is given in Algorithm 1. We have added a real-valued parameter, ε_1, that specifies the desired closeness to optimality of the policies produced by value iteration. In Section 3.2.2, we show that both m and ε_1 can be set as functions of the other input parameters, ε, δ, S, A, and γ, in order to make theoretical guarantees about the learning efficiency of R-MAX.

2.2 Delayed Q-learning

The *Delayed Q-learning* algorithm was introduced by Strehl et al. (2006b) as the first algorithm that is known to be PAC-MDP and its per-timestep computational demands are minimal (roughly equivalent to those of Q-learning). Due to its low memory requirements, it can also be viewed as a *model-free* algorithm and the first to be provably PAC-MDP. Its analysis is also noteworthy because the polynomial upper bound on its sample complexity is a significant improvement, asymptotically, over the best previously known upper bound for any algorithm, when only the dependence on S and A is considered.

The algorithm is called "delayed" because it waits until a state-action pair has been experienced m times before updating that state-action pair's associated action value, where m is a parameter provided as input. When it does update an action value, the update can be viewed as an average of the target values for the m most recently missed update opportunities. An important observation is that, when m is large enough, a Delayed Q-learning update will be sufficiently close to a true Bellman update (Lemma 22). In this sense, this algorithm is similar to Real-Time Dynamic Programming (Barto et al., 1995), but uses online transitions to dynamically form an approximate Bellman backup.

To encourage exploration, Delayed Q-learning uses the "optimism in the face of uncertainty" principle as in R-MAX. Specifically, its initial action-value function is an over-estimate of the true

8. The initial values are therefore denoted by $Q_0(\cdot,\cdot)$.

Algorithm 1 R-MAX

Inputs: S, A, γ, m, ε_1, and $U(\cdot,\cdot)$
for all (s,a) **do**
  $Q(s,a) \leftarrow U(s,a)$ // *action-value estimates*
  $r(s,a) \leftarrow 0$
  $n(s,a) \leftarrow 0$
  for all $s' \in S$ **do**
    $n(s,a,s') \leftarrow 0$
  end for
end for
for $t = 1,2,3,\cdots$ **do**
  Let s denote the state at time t.
  Choose action $a := \text{argmax}_{a' \in A} Q(s,a')$.
  Let r be the immediate reward and s' the next state after executing action a from state s.
  if $n(s,a) < m$ **then**
    $n(s,a) \leftarrow n(s,a)+1$
    $r(s,a) \leftarrow r(s,a)+r$ // *Record immediate reward*
    $n(s,a,s') \leftarrow n(s,a,s')+1$ // *Record immediate next-state*
    if $n(s,a) = m$ **then**
      for $i = 1,2,3,\cdots,\left\lceil \frac{\ln(1/(\varepsilon_1(1-\gamma)))}{1-\gamma} \right\rceil$ **do**
        for all $(\bar{s},\bar{a})$ **do**
          if $n(\bar{s},\bar{a}) \geq m$ **then**
            $Q(\bar{s},\bar{a}) \leftarrow \hat{R}(\bar{s},\bar{a}) + \gamma \sum_{s'} \hat{T}(s'|\bar{s},\bar{a}) \max_{a'} Q(s',a')$.
          end if
        end for
      end for
    end if
  end if
end for

function; during execution, the successive value function estimates remain over-estimates with high probability, thanks to the delayed update rule (Lemma 23).

Like R-MAX, Delayed Q-learning performs a finite number of action-value updates. Due to the strict restrictions on the computational demands used by Delayed Q-learning, slightly more sophisticated internal logic is needed to guarantee this property. Pseudo-code[9] for Delayed Q-learning is provided in Algorithm 2. More details are provided in the following subsections.

In addition to the standard inputs, the algorithm also relies on two free parameters,

- $\varepsilon_1 \in (0,1)$: Used to provide a constant "exploration bonus" that is added to each action-value estimate when it is updated.

9. Compared to the implementation provided by Strehl et al. (2006b), we have modified the algorithm to keep track of $b(s,a)$, the "beginning" timestep for the current attempted update for (s,a). The original pseudo-code kept track of $t(s,a)$, the time of the last attempted update for (s,a). The original implementation is less efficient and adds a factor of 2 to the computational bounds. The analysis of Strehl et al. (2006b) also applies to the pseudo-code presented here, however.

Algorithm 2 Delayed Q-learning

Inputs: S, A, γ, m, ε_1, and $U(\cdot,\cdot)$
for all (s,a) **do**
  $Q(s,a) \leftarrow U(s,a)$ *// action-value estimates*
  $AU(s,a) \leftarrow 0$ *// used for attempted updates*
  $l(s,a) \leftarrow 0$ *// counters*
  $b(s,a) \leftarrow 0$ *// beginning timestep of attempted update*
  $LEARN(s,a) \leftarrow true$ *// the LEARN flags*
end for
$t^* \leftarrow 0$ *// time of most recent action value change*
for $t = 1,2,3,\cdots$ **do**
  Let s denote the state at time t.
  Choose action $a := \text{argmax}_{a' \in A} Q(s,a')$.
  Let r be the immediate reward and s' the next state after executing action a from state s.
  if $b(s,a) \leq t^*$ **then**
    $LEARN(s,a) \leftarrow true$
  end if
  if $LEARN(s,a) = true$ **then**
    if $l(s,a) = 0$ **then**
      $b(s,a) \leftarrow t$
    end if
    $l(s,a) \leftarrow l(s,a) + 1$
    $AU(s,a) \leftarrow AU(s,a) + r + \gamma \max_{a'} Q(s',a')$
    if $l(s,a) = m$ **then**
      if $Q(s,a) - AU(s,a)/m \geq 2\varepsilon_1$ **then**
        $Q(s,a) \leftarrow AU(s,a)/m + \varepsilon_1$
        $t^* \leftarrow t$
      else if $b(s,a) > t^*$ **then**
        $LEARN(s,a) \leftarrow false$
      end if
      $AU(s,a) \leftarrow 0$
      $l(s,a) \leftarrow 0$
    end if
  end if
end for

- A positive integer m: Represents the number of experiences of a state-action pair before an update is allowed.

In the analysis of Section 3.3, we provide precise values for m and ε_1 in terms of the other inputs (S, A, ε, δ, and γ) that guarantee the resulting algorithm is PAC-MDP. In addition to its action-value estimates, $Q(s,a)$, the algorithm also maintains the following internal variables,

- $l(s,a)$ for each (s,a): The number of samples (or target values) gathered for (s,a).

- $AU(s,a)$ for each (s,a): Stores the running sum of target values used to update $Q(s,a)$ once enough samples have been gathered.
- $b(s,a)$ for each (s,a): The timestep for which the first experience of (s,a) was obtained for the most recent or ongoing attempted update.
- $LEARN(s,a) \in \{true, false\}$ for each (s,a): A Boolean flag that indicates whether or not, samples are being gathered for (s,a).

2.2.1 THE UPDATE RULE

Suppose that, at time $t \geq 1$, action a is performed from state s, resulting in an *attempted update*, according to the rules to be defined in Section 2.2.2. Let $s_{k_1}, s_{k_2}, \ldots, s_{k_m}$ be the m most recent next-states observed from executing (s,a) at times $k_1 < k_2 < \cdots < k_m$, respectively ($k_m = t$). For the remainder of the paper, we also let r_i denote the i^{th} reward received during the execution of Delayed Q-learning.

Thus, at time k_i, action a was taken from state s, resulting in a transition to state s_{k_i} and an immediate reward r_{k_i}. After the t^{th} action, the following update occurs:

$$Q_{t+1}(s,a) = \frac{1}{m}\sum_{i=1}^{m}(r_{k_i} + \gamma V_{k_i}(s_{k_i})) + \varepsilon_1 \tag{6}$$

as long as performing the update would result in a new action-value estimate that is at least ε_1 smaller than the previous estimate. In other words, the following equation must be satisfied for an update to occur:

$$Q_t(s,a) - \left(\frac{1}{m}\sum_{i=1}^{m}(r_{k_i} + \gamma V_{k_i}(s_{k_i}))\right) \geq 2\varepsilon_1. \tag{7}$$

If this condition does not hold, then no update is performed, and so $Q_{t+1}(s,a) = Q_t(s,a)$.

2.2.2 MAINTENANCE OF THE *LEARN* FLAGS

We provide an intuition behind the behavior of the *LEARN* flags. Please see Algorithm 2 for a formal description of the update rules. The main computation of the algorithm is that every time a state-action pair (s,a) is experienced m times, an update of $Q(s,a)$ is attempted as in Section 2.2.1. For our analysis to hold, however, we cannot allow an infinite number of attempted updates. Therefore, attempted updates are only allowed for (s,a) when $LEARN(s,a)$ is *true*. Besides being set to *true* initially, $LEARN(s,a)$ is also set to *true* when any state-action pair is updated (because our estimate $Q(s,a)$ may need to reflect this change). $LEARN(s,a)$ can only change from *true* to *false* when no updates are made during a length of time for which (s,a) is experienced m times and the next attempted update of (s,a) fails. In this case, no more attempted updates of (s,a) are allowed until another action-value estimate is updated.

2.2.3 DELAYED Q-LEARNING'S MODEL

Delayed Q-learning was introduced as a *model-free* algorithm. This terminology was justified by noting that the space complexity of Delayed Q-learning, which is $O(SA)$, is much less than what is needed in the worst case to completely represent an MDP's transition probabilities ($O(S^2A)$).

However, there is a sense in which Delayed Q-learning can be thought of as using a model. This interpretation follows from the fact that Delayed Q-learning's update (Equation 6) is identical to ε_1 plus the result of a full Bellman backup using the empirical (maximum likelihood) model derived from the m most recent experiences of the state-action pair being updated. Since m is much less than what is needed to accurately model the true transition probability (in the $L1$ distance metric), we say that Delayed Q-learning uses a *sparse model* (Kearns and Singh, 1999). In fact, Delayed Q-learning uses this sparse model precisely once, throws it away, and then proceeds to gather experience for another sparse model. When $m = 1$, this process may occur on every timestep and the algorithm behaves very similarly to a version of Q-learning that uses a unit learning rate.

3. PAC-MDP Analysis

First, we present a general framework that allows us to prove the bounds for both algorithms. We then proceed to analyze R-MAX and Delayed Q-learning.

3.1 General Framework

We now develop some theoretical machinery to prove PAC-MDP statements about various algorithms. Our theory will be focused on algorithms that maintain a table of action values, $Q(s,a)$, for each state-action pair (denoted $Q_t(s,a)$ at time t).[10] We also assume an algorithm always chooses actions greedily with respect to the action values. This constraint is not really a restriction, since we could define an algorithm's action values as 1 for the action it chooses and 0 for all other actions. However, the general framework is understood and developed more easily under the above assumptions. For convenience, we also introduce the notation $V(s)$ to denote $\max_a Q(s,a)$ and $V_t(s)$ to denote $V(s)$ at time t.

Definition 5 *Suppose an RL algorithm $\mathcal{A}$ maintains a value, denoted $Q(s,a)$, for each state-action pair $(s,a) \in \mathcal{S} \times \mathsf{A}$. Let $Q_t(s,a)$ denote the estimate for (s,a) immediately before the t^{th} action of the agent. We say that $\mathcal{A}$ is a* **greedy algorithm** *if the t^{th} action of $\mathcal{A}$, a_t, is $a_t := \text{argmax}_{a \in \mathsf{A}} Q_t(s_t,a)$, where s_t is the t^{th} state reached by the agent.*

For all algorithms, the action values $Q(\cdot,\cdot)$ are implicitly maintained in separate max-priority queues (implemented with max-heaps, say) for each state. Specifically, if $\mathsf{A} = \{a_1,\ldots,a_k\}$ is the set of actions, then for each state s, the values $Q(s,a_1),\ldots,Q(s,a_k)$ are stored in a single priority queue. Therefore, the operations $\max_{a' \in \mathsf{A}} Q(s,a)$ and $\text{argmax}_{a' \in \mathsf{A}} Q(s,a)$, which appear in almost every algorithm, takes constant time, but the operation $Q(s,a) \leftarrow V$ for any value V takes $O(\ln(A))$ time (Cormen et al., 1990). It is possible that other data structures may result in faster algorithms.

The following is a definition of a new MDP that will be useful in our analysis.

Definition 6 *Let $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ be an MDP with a given set of action values, $Q(s,a)$, for each state-action pair (s,a), and a set K of state-action pairs, called the* **known state-action pairs**. *We define the* **known state-action MDP** *$M_K = \langle \mathcal{S} \cup \{z_{s,a} | (s,a) \notin K\}, \mathsf{A}, T_K, R_K, \gamma \rangle$ as follows. For each unknown state-action pair, $(s,a) \notin K$, we add a new state $z_{s,a}$ to M_K, which has self-loops for each*

10. However, the main result in this subsection (Theorem 10) does not rely on the algorithm having an explicit representation of each action value. For example, they could be implicitly held inside of a function approximator (e.g., Brunskill et al. 2008).

action ($T_K(z_{s,a}|z_{s,a},\cdot) = 1$). For all $(s,a) \in K$, $R_K(s,a) = R(s,a)$ and $T_K(\cdot|s,a) = T(\cdot|s,a)$. For all $(s,a) \notin K$, $R_K(s,a) = Q(s,a)(1-\gamma)$ and $T_K(z_{s,a}|s,a) = 1$. For the new states, the reward is $R_K(z_{s,a},\cdot) = Q(s,a)(1-\gamma)$.

The known state-action MDP is a generalization of the standard notions of a "known state MDP" of Kearns and Singh (2002) and Kakade (2003). It is an MDP whose dynamics (reward and transition functions) are equal to the true dynamics of M for a subset of the state-action pairs (specifically those in K). For all other state-action pairs, the value of taking those state-action pairs in M_K (and following any policy from that point on) is equal to the current action-value estimates $Q(s,a)$. We intuitively view K as a set of state-action pairs for which the agent has sufficiently accurate estimates of their dynamics.

Definition 7 *For algorithm $\mathcal{A}$, for each timestep t, let K_t (we drop the subscript t if t is clear from context) be a set of state-action pairs defined arbitrarily in a way that depends only on the history of the agent up to timestep t (before the $(t)^{\text{th}}$ action). We define A_K to be the event, called the* **escape event**, *that some state-action pair $(s,a) \notin K_t$ is experienced by the agent at time t.*

The following is a well-known result of the Chernoff-Hoeffding Bound and will be needed later; see Li (2009, Lemma 56) for a slightly improved result.

Lemma 8 *Suppose a weighted coin, when flipped, has probability $p > 0$ of landing with heads up. Then, for any positive integer k and real number $\delta \in (0,1)$, there exists a number $m = O((k/p)\ln(1/\delta))$, such that after m tosses, with probability at least $1-\delta$, we will observe k or more heads.*

One more technical lemma is needed before presenting the main result in this section. Note that even if we assume $V^*_M(s) \le V_{\max}$ and $Q(s,a) \le V_{\max}$ for all $s \in \mathcal{S}$ and $a \in \mathsf{A}$, it may not be true that $V^*_{M_K}(s) \le V_{\max}$. However, the following lemma shows we may instead use $2V_{\max}$ as an upper bound.

Lemma 9 *Let $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ be an MDP whose optimal value function is upper bounded by $V_{\max}$. Furthermore, let M_K be a known state-action MDP for some $K \subseteq \mathcal{S} \times \mathsf{A}$ defined using value function $Q(s,a)$. Then, $V^*_{M_K}(s) \le V_{\max} + \max_{s',a'} Q(s',a')$ for all $s \in \mathcal{S}$.*

Proof For any policy π and any state $s \in \mathcal{S}$, let $(s_1,a_1,r_1,s_2,a_2,r_2,s_3,a_3,r_3,\ldots)$ be a path generated by starting in state $s = s_1$ and following π in the known state-action MDP, M_K, where s_t and r_t are the state and reward at timestep t, and $a_t = \pi(s_t)$ for all t. The value function, $V^\pi_{M_K}(s)$, can be written as (see, e.g., Sutton and Barto 1998)

$$V^\pi_{M_K}(s) = \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid s_1 = s, \pi\right],$$

which says the quantity $V^\pi_{M_K}(s)$ is the expected discounted total reward accumulated on this random path. Here, we use $\mathbf{E}_{M_K}$ to denote the expectation with respect to randomness in the MDP M_K.

Denote by τ be the *first* timestep in which $(s_\tau, a_\tau) \notin K$; note $\tau = \infty$ if all visited state-actions are in K. Due to construction of M_K, if τ is finite, then

$$\begin{aligned} s_\tau &= s_{\tau+1} = s_{\tau+2} = \cdots \\ a_\tau &= a_{\tau+1} = a_{\tau+2} = \cdots = \pi(s_\tau) \\ r_\tau &= r_{\tau+1} = r_{\tau+2} = \cdots = (1-\gamma)Q(s_\tau, a_\tau). \end{aligned}$$

Thus, for any fixed $\tau \geq 1$, the discounted total reward

$$\begin{aligned} & r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \\ &= \quad r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} + \gamma^{\tau-1} Q(s_\tau, a_\tau) \\ &\leq \quad r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} + \max_{s',a'} Q(s',a'), \end{aligned}$$

where the first step is due to the way we define transition/reward functions in M_K for state-actions outside K. The above upper bound holds for all fixed value of τ (finite or infinite), and so

$$\begin{aligned} & \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid s_1 = s, \pi\right] \\ &\leq \quad \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} \mid s_1 = s, \pi\right] + \max_{s',a'} Q(s',a'). \end{aligned}$$

Finally, since the transition and reward functions of M and M_K are identical for state-actions in K, we have

$$\begin{aligned} & \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} \mid s_1 = s, \pi\right] \\ &= \quad \mathbf{E}_{M}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1} \mid s_1 = s, \pi\right], \end{aligned}$$

which implies

$$\begin{aligned} & \mathbf{E}_{M_K}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid s_1 = s, \pi\right] \\ &\leq \quad \mathbf{E}_{M}\left[r_1 + \gamma r_2 + \cdots + \gamma^{\tau-2} r_{\tau-1}\right] + \max_{s',a'} Q(s',a') \\ &\leq \quad V_M^\pi(s) + \max_{s',a'} Q(s',a') \\ &\leq \quad V_{\max} + \max_{s',a'} Q(s',a'). \end{aligned}$$

■

Note that all learning algorithms we consider take ε and δ as input. We let $\mathcal{A}(\varepsilon,\delta)$ denote the version of algorithm $\mathcal{A}$ parameterized with ε and δ. The proof of Theorem 10 follows the structure of the work of Kakade (2003), but generalizes several key steps. The theorem also generalizes a previous result by Strehl et al. (2006a) by taking the admissible heuristic into account.

Theorem 10 *Let $\mathcal{A}(\varepsilon,\delta)$ be any greedy learning algorithm such that, for every timestep t, there exists a set K_t of state-action pairs that depends only on the agent's history up to timestep t. We assume that $K_t = K_{t+1}$ unless, during timestep t, an update to some state-action value occurs or the escape event A_K happens. Let M_{K_t} be the known state-action MDP and π_t be the current greedy policy, that is, for all states s, $\pi_t(s) = \operatorname{argmax}_a Q_t(s,a)$. Furthermore, assume $Q_t(s,a) \leq V_{\max}$ for all t and (s,a). Suppose that for any inputs ε and δ, with probability at least $1-\delta$, the following conditions hold for all states s, actions a, and timesteps t: (1) $V_t(s) \geq V^*(s) - \varepsilon$ (optimism), (2) $V_t(s) - V_{M_{K_t}}^{\pi_t}(s) \leq \varepsilon$ (accuracy), and (3) the total number of updates of action-value estimates plus the number of times the escape event from K_t, A_K, can occur is bounded by $\zeta(\varepsilon,\delta)$ (learning complexity). Then, when $\mathcal{A}(\varepsilon,\delta)$ is executed on any MDP M, it will follow a 4ε-optimal policy from its current state on all but*

$$O\left(\frac{V_{\max}\zeta(\varepsilon,\delta)}{\varepsilon(1-\gamma)} \ln\frac{1}{\delta} \ln\frac{1}{\varepsilon(1-\gamma)}\right)$$

timesteps, with probability at least $1-2\delta$.

Proof Suppose that the learning algorithm $\mathcal{A}(\varepsilon,\delta)$ is executed on MDP M. Fix the history of the agent up to the t^{th} timestep and let s_t be the t^{th} state reached. Let $\mathcal{A}_t$ denote the current (non-stationary) policy of the agent. Let $H = \frac{1}{1-\gamma}\ln\frac{1}{\varepsilon(1-\gamma)}$. From Lemma 2 of Kearns and Singh (2002), we have that $|V^{\pi}_{M_{K_t}}(s,H) - V^{\pi}_{M_{K_t}}(s)| \leq \varepsilon$, for any state s and policy π. Let W denote the event that, after executing policy $\mathcal{A}_t$ from state s_t in M for H timesteps, one of the two following events occur: (a) the algorithm performs a successful update (a change to any of its action values) of some state-action pair (s,a), or (b) some state-action pair $(s,a) \notin K_t$ is experienced (escape event A_K). Assuming the three conditions in the theorem statement hold, we have the following:

$$\begin{aligned}
& V^{\mathcal{A}_t}_M(s_t,H) \\
&\quad \geq \quad V^{\pi_t}_{M_{K_t}}(s_t,H) - 2V_{\max}\Pr(W) \\
&\quad \geq \quad V^{\pi_t}_{M_{K_t}}(s_t) - \varepsilon - 2V_{\max}\Pr(W) \\
&\quad \geq \quad V(s_t) - 2\varepsilon - 2V_{\max}\Pr(W) \\
&\quad \geq \quad V^*(s_t) - 3\varepsilon - 2V_{\max}\Pr(W).
\end{aligned}$$

The first step above follows from the fact that following $\mathcal{A}_t$ in MDP M results in behavior identical to that of following π_t in M_{K_t} unless event W occurs, in which case a loss of at most $2V_{\max}$ can occur (Lemma 9). The second step follows from the definition of H above. The third and final steps follow from Conditions 2 and 1, respectively, of the proposition.

Now, suppose that $\Pr(W) < \frac{\varepsilon}{2V_{\max}}$. Then, we have that the agent's policy on timestep t is 4ε-optimal:

$$V^{\mathcal{A}_t}_M(s_t) \geq V^{\mathcal{A}_t}_M(s_t,H) \geq V^*_M(s_t) - 4\varepsilon.$$

Otherwise, we have that $\Pr(W) \geq \frac{\varepsilon}{2V_{\max}}$, which implies that an agent following $\mathcal{A}_t$ will either perform a successful update in H timesteps, or encounter some $(s,a) \notin K_t$ in H timesteps, with probability at least $\frac{\varepsilon}{2V_{\max}}$. Call such an event a "success". Then, by Lemma 8, after $O(\frac{\zeta(\varepsilon,\delta)HV_{\max}}{\varepsilon}\ln 1/\delta)$ timesteps t where $\Pr(W) \geq \frac{\varepsilon}{2V_{\max}}$, $\zeta(\varepsilon,\delta)$ successes will occur, with probability at least $1-\delta$. Here, we have identified the event that a success occurs after following the agent's policy for H steps with the event that a coin lands with heads facing up. However, by Condition 3 of the proposition, with probability at least $1-\delta$, $\zeta(\varepsilon,\delta)$ is the maximum number of successes that will occur throughout the execution of the algorithm.

To summarize, we have shown that with probability $1-2\delta$, the agent will execute a 4ε-optimal policy on all but $O(\frac{\zeta(\varepsilon,\delta)HV_{\max}}{\varepsilon}\ln\frac{1}{\delta}) = O(\frac{\zeta(\varepsilon,\delta)V_{\max}}{\varepsilon(1-\gamma)}\ln\frac{1}{\delta}\ln\frac{1}{\varepsilon(1-\gamma)})$ timesteps. ■

3.2 Analysis of R-MAX

We will analyze R-MAX using the tools from Section 3.1.

3.2.1 COMPUTATIONAL COMPLEXITY

When the initial value function is $U(s,a) = 1/(1-\gamma)$ for all (s,a), there is a simple way to change the R-MAX algorithm that has a minimal affect on its behavior and saves greatly on computation. The important observation is that for a fixed state s, the maximum action-value estimate, $\max_a Q(s,a)$ will be $1/(1-\gamma)$ until all actions have been tried m times. Thus, there is no need to

run value iteration (lines 17 to 25 in Algorithm 1) until each action has been tried exactly m times. In addition, if there are some actions that have been tried m times and others that have not, the algorithm should choose one of the latter. One method to accomplish this balance is to order each action and try one after another until all are chosen m times. Kearns and Singh (2002) called this behavior "balanced wandering". However, it is not necessary to use balanced wandering; for example, it would be perfectly fine to try the first action m times, the second action m times, and so on. Any deterministic method for breaking ties in line 11 of Algorithm 1 is valid as long as mA experiences of a state-action pair results in all action being chosen m times.

On most timesteps, the R-MAX algorithm performs a constant amount of computation to choose its next action. Only when a state's last action has been tried m times does it solve its internal model. Our version of R-MAX uses value iteration to solve its model. Therefore, the per-timestep computational complexity of R-MAX is

$$\Theta\left(SA(S+\ln(A))\left(\frac{1}{1-\gamma}\right)\ln\frac{1}{\varepsilon_1(1-\gamma)}\right).$$

This expression is derived using the fact that value iteration performs $\left\lceil \frac{1}{1-\gamma}\ln\frac{1}{\varepsilon_1(1-\gamma)} \right\rceil$ iterations, where each iteration involves SA full Bellman backups (one for each state-action pair). A Bellman backup requires examining all possible $O(S)$ successor states and the update to the priority queue takes time $O(\ln(A))$. Note that R-MAX updates its model at most S times. From this observation we see that the total computation time of R-MAX is $O\left(B+\frac{S^2A(S+\ln(A))}{1-\gamma}\ln\frac{1}{\varepsilon_1(1-\gamma)}\right)$, where B is the number of timesteps for which R-MAX is executed.

When a general admissible initial value function U is used, we need to run value iteration whenever some $n(s,a)$ reaches the threshold m. In this case, a similar analysis shows that the total computation time of R-MAX is $O\left(B+\frac{S^2A^2(S+\ln(A))}{1-\gamma}\ln\frac{1}{\varepsilon_1(1-\gamma)}\right)$.

3.2.2 SAMPLE COMPLEXITY

The main result of this section is the following theorem.

Theorem 11 *Suppose that* $0 \le \varepsilon < \frac{1}{1-\gamma}$ *and* $0 \le \delta < 1$ *are two real numbers and* $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ *is any MDP. There exists inputs* $m = m(\frac{1}{\varepsilon}, \frac{1}{\delta})$ *and* ε_1*, satisfying* $m(\frac{1}{\varepsilon}, \frac{1}{\delta}) = O\left(\frac{(S+\ln(SA/\delta))V_{\max}^2}{\varepsilon^2(1-\gamma)^2}\right)$ *and* $\frac{1}{\varepsilon_1} = O(\frac{1}{\varepsilon})$*, such that if R-MAX is executed on* M *with inputs* m *and* ε_1*, then the following holds. Let* $\mathcal{A}_t$ *denote R-MAX's policy at time* t *and* s_t *denote the state at time* t*. With probability at least* $1-\delta$*,* $V_M^{\mathcal{A}_t}(s_t) \ge V_M^*(s_t) - \varepsilon$ *is true for all but*

$$O\left(\frac{\left|\{(s,a) \in \mathcal{S} \times \mathsf{A} | U(s,a) \ge V^*(s) - \varepsilon\}\right|}{\varepsilon^3(1-\gamma)^3}\left(S+\ln\frac{SA}{\delta}\right)V_{\max}^3\ln\frac{1}{\delta}\ln\frac{1}{\varepsilon(1-\gamma)}\right)$$

timesteps t*.*

First, we discuss the accuracy of the model maintained by R-MAX. The following lemma shows that two MDPs with similar transition and reward functions have similar value functions. Thus, an agent need only ensure accuracy in the transitions and rewards of its model to guarantee near-optimal behavior.

Lemma 12 *(Strehl and Littman, 2005) Let $M_1 = \langle \mathcal{S}, \mathsf{A}, T_1, R_1, \gamma \rangle$ and $M_2 = \langle \mathcal{S}, \mathsf{A}, T_2, R_2, \gamma \rangle$ be two MDPs with non-negative rewards bounded by 1 and optimal value functions bounded by $V_{\max}$. Suppose that $|R_1(s,a) - R_2(s,a)| \leq \alpha$ and $\|T_1(s,a,\cdot) - T_2(s,a,\cdot)\|_1 \leq 2\beta$ for all states s and actions a. There exists a constant $C > 0$ such that for any $0 \leq \varepsilon \leq 1/(1-\gamma)$ and stationary policy π, if $\alpha = 2\beta = C\varepsilon(1-\gamma)/V_{\max}$, then*

$$|Q_1^\pi(s,a) - Q_2^\pi(s,a)| \leq \varepsilon.$$

Let $n_t(s,a)$ denote the value of $n(s,a)$ at time t during execution of the algorithm. For R-MAX, let the "known" state-action pairs K_t, at time t (See Definition 6), to be

$$K_t := \{(s,a) \in \mathcal{S} \times \mathsf{A} | n_t(s,a) \geq m\},$$

which is dependent on the parameter m that is provided as input to the algorithm. In other words, K_t is the set of state-action pairs that have been experienced by the agent at least m times. We will show that for large enough m, the dynamics, transition and reward, associated with these pairs can be accurately approximated by the agent.

The following event will be used in our proof that R-MAX is PAC-MDP. We will provide a sufficient condition (specifically, L_1-accurate transition and reward functions) to guarantee that the event occurs, with high probability. In words, the condition says that the value of any state s, under any policy, in the empirical known state-action MDP ($\hat{M}_{K_t}$) is ε_1-close to its value in the true known state-action MDP (M_{K_t}).

Event A1 *For all stationary policies π, timesteps t and states s during execution of the R-MAX algorithm on some MDP M, $|V_{M_{K_t}}^\pi(s) - V_{\hat{M}_{K_t}}^\pi(s)| \leq \varepsilon_1$.*

Next, we quantify the number of samples needed from both the transition and reward distributions for a state-action pair to compute accurate approximations.

Lemma 13 *Suppose that $r[1], r[2], \ldots, r[m]$ are m rewards drawn independently from the reward distribution, $\mathcal{R}(s,a)$, for state-action pair (s,a). Let $\hat{R}(s,a)$ be the empirical (maximum-likelihood) estimate of $\mathcal{R}(s,a)$. Let δ_R be any positive real number less than 1. Then, with probability at least $1 - \delta_R$, we have that $|\hat{R}(s,a) - \mathcal{R}(s,a)| \leq \varepsilon^R_{n(s,a)}$, where*

$$\varepsilon^R_m := \sqrt{\frac{\ln(2/\delta_R)}{2m}}.$$

Proof This result follows directly from Hoeffding's bound (Hoeffding, 1963). ■

Lemma 14 *Suppose that $\hat{T}(s,a)$ is the empirical transition distribution for state-action pair (s,a) using m samples of next states drawn independently from the true transition distribution $T(s,a)$. Let δ_T be any positive real number less than 1. Then, with probability at least $1 - \delta_T$, we have that $\|T(s,a) - \hat{T}(s,a)\|_1 \leq \varepsilon^T_{n(s,a)}$ where*

$$\varepsilon^T_m = \sqrt{\frac{2[\ln(2^S - 2) - \ln(\delta_T)]}{m}}.$$

Proof The result follows immediately from an application of Theorem 2.1 of Weissman et al. (2003).[11] ■

Lemma 15 *There exists a constant C such that if R-MAX with parameters m and ε_1 is executed on any MDP $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ and m satisfies*

$$m \geq C V_{\max}^2 \left(\frac{S + \ln(SA/\delta)}{{\varepsilon_1}^2 (1-\gamma)^2} \right) = \tilde{O}\left(\frac{S V_{\max}^2}{{\varepsilon_1}^2 (1-\gamma)^2} \right),$$

then Event A1 will occur with probability at least $1-\delta$.

Proof Event A1 occurs if R-MAX maintains a close approximation of its known state-action MDP. By Lemmas 9 and 12, it is sufficient to obtain $(C\varepsilon_1(1-\gamma)/V_{\max})$-approximate transition and reward functions (where C is a constant), for those state-action pairs in K_t. The transition and reward functions that R-MAX uses are the maximum-likelihood estimates, using only the first m samples (of immediate reward and next-state pairs) for each $(s,a) \in K$. Intuitively, as long as m is large enough, the empirical estimates for these state-action pairs will be accurate, with high probability.[12] Consider a fixed state-action pair (s,a). From Lemma 13, we can guarantee the empirical reward distribution is accurate enough, with probability at least $1-\delta'$, as long as $\sqrt{\frac{\ln(2/\delta')}{2m}} \leq C\varepsilon_1(1-\gamma)/V_{\max}$. From Lemma 14, we can guarantee the empirical transition distribution is accurate enough, with probability at least $1-\delta'$, as long as $\sqrt{\frac{2[\ln(2^S-2)-\ln(\delta')]}{m}} \leq C\varepsilon_1(1-\gamma)/V_{\max}$. It is possible to choose m, as a function of the parameters of the MDP M, large enough so that both these expressions are satisfied but small enough so that

$$m \propto \frac{S + \ln(1/\delta')}{\varepsilon_1^2 (1-\gamma)^2} V_{\max}^2.$$

With this choice, we guarantee that the empirical reward and empirical distribution for a single state-action pair will be sufficiently accurate, with high probability. However, to apply the simulation bounds of Lemma 12, we require accuracy for all state-action pairs. To ensure a total failure probability of δ, we set $\delta' = \delta/(2SA)$ in the above equations and apply the union bound over all state-action pairs. ■

Proof (of Theorem 11). We apply Theorem 10. Let $\varepsilon_1 = \varepsilon/2$. Assume that Event A1 occurs. Consider some fixed time t. First, we verify Condition 1 of the theorem. We have that $V_t(s) \geq V^*_{\hat{M}_{K_t}}(s) - \varepsilon_1 \geq V^*_{M_{K_t}}(s) - 2\varepsilon_1 \geq V^*(s) - 2\varepsilon_1$. The first inequality follows from the fact that

11. The result of Weissman et al. (2003) is established using an information-theoretic argument. A similar result can be obtained (Kakade, 2003) by the multiplicative form of Chernoff's bounds.
12. There is a minor technicality here. The samples, in the form of immediate rewards and next states, experienced by an online agent in an MDP are not necessarily independent samples. The reason is that the learning environment or the agent could prevent future experiences of state-action pairs based on previously observed outcomes. Nevertheless, all the tail inequality bounds, including the Chernoff and Hoeffding Bounds, that hold for independent samples also hold for online samples in MDPs that can be viewed as martingales, a fact that follows from the Markov property. There is an extended discussion and formal proof of this fact elsewhere (Strehl and Littman, 2008b). An excellent review (with proofs) of the tail inequalities for martingales that we use in the present paper is by McDiarmid (1989).

R-MAX computes its action values by computing an ε_1-approximate solution of its internal model ($\hat{M}_{K_t}$) (using Proposition 4). The second inequality follows from Event A1 and the third from the fact that M_{K_t} can be obtained from M by removing certain states and replacing them with a maximally rewarding state whose actions are self-loops, an operation that only increases the value of any state. Next, we note that Condition 2 of the theorem follows from Event A1. Finally, observe that the learning complexity, $\zeta(\varepsilon,\delta) \leq |\{(s,a)|U(s,a) \geq V^*(s)-\varepsilon\}|m$. To see this fact, first note that state-action pair (s,a) with $U(s,a) < V^*(s)-\varepsilon$ will never be experienced, with high probability, because initially the agent chooses actions greedily with respect to $U(s,a)$ and there always exists another action a' such that $Q_t(s,a') > V^*(s)-\varepsilon$. Next, note that each time an escape occurs, some $(s,a) \notin K$ is experienced. However, once (s,a) is experienced m times, it becomes part of and never leaves the set K. To guarantee that Event A1 occurs with probability at least $1-\delta$, we use Lemma 15 to set m. ■

3.3 Analysis of Delayed Q-learning

In this section, we analyze the computational and sample complexity of Delayed Q-learning.

3.3.1 COMPUTATIONAL COMPLEXITY

On most timesteps, Delayed Q-learning performs only a constant amount of computation. Its worst-case computational complexity per timestep is

$$\Theta(\ln(A)),$$

where the logarithmic term is due to updating the priority queue that holds the action-value estimates for the current state. Since Delayed Q-learning performs at most $SA\left(1+\frac{SA}{(1-\gamma)\varepsilon_1}\right)$ attempted updates (see Lemma 19), each update involves m transitions, and each transition requires computing the greedy action whose computation complexity is $O(\ln(A))$, the total computation time of Delayed Q-learning is

$$O\left(B+\frac{mS^2A^2\ln(A)}{\varepsilon_1(1-\gamma)}\right),$$

where B is the number of timesteps for which Delayed Q-learning is executed. Since the number of attempted updates is bounded by a constant, the amortized computation time per step is $O(1)$ as B approaches ∞.

3.3.2 SAMPLE COMPLEXITY

In this section, we show that Delayed Q-learning is PAC-MDP.

Theorem 16 *(Strehl et al., 2006b) Suppose that $0 \leq \varepsilon < \frac{1}{1-\gamma}$ and $0 \leq \delta < 1$ are two real numbers and $M = \langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$ is any MDP. There exists inputs $m = m(\frac{1}{\varepsilon}, \frac{1}{\delta})$ and ε_1, satisfying $m(\frac{1}{\varepsilon}, \frac{1}{\delta}) = O\left(\frac{(1+\gamma V_{\max})^2}{\varepsilon_1^2}\ln\frac{SA}{\varepsilon_1\delta(1-\gamma)}\right)$ and $\frac{1}{\varepsilon_1} = O(\frac{1}{\varepsilon(1-\gamma)})$, such that if Delayed Q-learning is executed on M, then the following holds. Let $\mathcal{A}_t$ denote Delayed Q-learning's policy at time t and s_t denote the state at*

time t. With probability at least $1-\delta$, $V_M^{\mathcal{A}_t}(s_t) \geq V_M^*(s_t) - \varepsilon$ is true for all but

$$O\left(\frac{V_{\max}(1+\gamma V_{\max})^2 \sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}[U(s,a)-V^*(s)]_+}{\varepsilon^4(1-\gamma)^4} \ln\frac{1}{\delta}\ln\frac{1}{\varepsilon(1-\gamma)}\ln\frac{SA}{\delta\varepsilon(1-\gamma)}\right)$$

timesteps t.

Definition 17 *An* **update** *(or* **successful update***) of state-action pair (s,a) is a timestep t for which a change to the action-value estimate $Q(s,a)$ occurs. An* **attempted update** *of state-action pair (s,a) is a timestep t for which (s,a) is experienced, $LEARN(s,a) = true$ and $l(s,a) = m$. An attempted update that is not successful is an* **unsuccessful update**.

To prove the main theorem we need some additional results. The following lemmas are modified slightly from Strehl et al. (2006b). For convenience, define

$$\kappa := \frac{SA}{(1-\gamma)\varepsilon_1}.$$

Lemma 18 *The total number of updates during any execution of Delayed Q-learning is at most κ.*

Proof Consider a fixed state-action pair (s,a). Its associated action-value estimate $Q(s,a)$ is initialized to $U(s,a) \leq 1/(1-\gamma)$ before any updates occur. Each time $Q(s,a)$ is updated it decreases by at least ε_1. Since all rewards encountered are non-negative, the quantities involved in any update (see Equation 6) are non-negative. Thus, $Q(s,a)$ cannot fall below 0. It follows that $Q(s,a)$ cannot be updated more than $1/(\varepsilon(1-\gamma))$ times. Since there are SA state-action pairs, we have that there are at most $SA/(\varepsilon(1-\gamma))$ total updates. ■

Lemma 19 *The total number of attempted updates during any execution of Delayed Q-learning is at most $SA(1+\kappa)$.*

Proof Consider a fixed state-action pair (s,a). Once (s,a) is experienced for the m^{th} time, an attempted update will occur. Suppose that an attempted update of (s,a) occurs during timestep t. Afterwards, for another attempted update to occur during some later timestep t', it must be the case that a successful update of some state-action pair (not necessarily (s,a)) has occurred on or after timestep t and before timestep t'. From Lemma 18, there can be at most κ total successful updates. Therefore, there are at most $1+\kappa$ attempted updates of (s,a). Since there are SA state-action pairs, there can be at most $SA(1+\kappa)$ total attempted updates. ■

Definition 20 *During timestep t of the execution of Delayed Q-learning, we define K_t to be the set*

$$K_t := \left\{(s,a)\in\mathcal{S}\times\mathsf{A} \mid Q_t(s,a) - \left(R(s,a)+\gamma\sum_{s'}T(s'|s,a)V_t(s')\right) \leq 3\varepsilon_1\right\}.$$

The set K_t consists of the state-action pairs with low Bellman residual. The state-action pairs not in K_t are the ones whose action-value estimates are overly optimistic in the sense that they would decrease significantly if subjected to a Bellman backup (as in value iteration). Intuitively, if $(s,a) \notin K_t$, then it is very likely that (s,a) will be updated successfully by Delayed Q-learning if visited m times. This intuition is formalized by the following definition and lemma.

Definition 21 *Suppose we execute Delayed Q-learning in an MDP M. Define* **Event A2** *to be the event that for all timesteps t, if $(s,a) \notin K_{k_1}$ and an attempted update of (s,a) occurs during timestep t, then the update will be successful, where $k_1 < k_2 < \cdots < k_m = t$ are m last timesteps during which (s,a) is experienced consecutively by the agent.*

Lemma 22 *Suppose we execute Delayed Q-learning with parameter m satisfying*

$$m \geq \frac{(1+\gamma V_{\max})^2}{2{\varepsilon_1}^2} \ln\left(\frac{3SA}{\delta}\left(1+\frac{SA}{\varepsilon_1(1-\gamma)}\right)\right) \tag{8}$$

in an MDP M. The probability that Event A2 occurs is greater than or equal to $1-\delta/3$.

Proof Fix any timestep k_1 (and the complete history of the agent up to k_1) satisfying: the agent is in state s and about to take action a, where $(s,a) \notin K_{k_1}$ on timestep k_1, $LEARN(s,a) = true$, and $l(s,a) = 0$ at time k_1. In other words, if (s,a) is experienced $m-1$ more times after timestep k_1, then an attempted update will result. Let $Q = [(s[1],r[1]),\ldots,(s[m],r[m])] \in (S \times \mathbb{R})^m$ be any sequence of m next-state and immediate reward tuples. Due to the Markov assumption, whenever the agent is in state s and chooses action a, the resulting next-state and immediate reward are chosen independently of the history of the agent. Thus, the probability of the joint event

1. (s,a) is experienced $m-1$ more times, and
2. the resulting next-state and immediate reward sequence equals Q

is at most the probability that Q is obtained by m independent draws from the transition and reward distributions (for (s,a)). Therefore, it suffices to prove this lemma by showing that the probability that a random sequence Q could cause an unsuccessful update of (s,a) is at most $\delta/3$. We prove this statement next.

Suppose m rewards, $r[1],\ldots,r[m]$, and m next states, $s[1],\ldots,s[m]$, are drawn independently from the reward and transition distributions, respectively, for (s,a). By a straightforward application of the Hoeffding bound (with random variables $X_i := r[i] + \gamma V_{k_1}(s[i])$ so that $0 \leq X_i \leq (1+\gamma V_{\max})$), it can be shown that our choice of m guarantees that

$$\frac{1}{m}\sum_{i=1}^{m}(r[i]+\gamma V_{k_1}(s[i])) - \mathbf{E}[X_1] < \varepsilon_1$$

holds with probability at least $1-\delta/(3SA(1+\kappa))$. If it does hold and an attempted update is performed for (s,a) using these m samples, then the resulting update will succeed. To see the claim's validity, suppose that (s,a) is experienced at times $k_1 < k_2 < \cdots < k_m = t$ and at time k_i the agent is transitioned to state $s[i]$ and receives reward $r[i]$ (causing an attempted update at time t). Then, we have that

$$Q_t(s,a) - \left(\frac{1}{m}\sum_{i=1}^{m}(r[i]+\gamma V_{k_i}(s[i]))\right) \;>\; Q_t(s,a) - \mathbf{E}[X_1] - \varepsilon_1 \;>\; 2\varepsilon_1.$$

We have used the fact that $V_{k_i}(s') \leq V_{k_1}(s')$ for all s' and $i = 1, \ldots, m$. Therefore, with high probability, Equation 7 will be true and the attempted update of $Q(s,a)$ at time k_m will succeed.

Finally, we extend our argument, using the union bound, to all possible timesteps k_1 satisfying the condition above. The number of such timesteps is bounded by the same bound we showed for the number of attempted updates (that is, $SA(1+\kappa)$). ■

The next lemma states that, with high probability, Delayed Q-learning will maintain optimistic action values.

Lemma 23 *During execution of Delayed Q-learning, if m satisfies Equation 8, then $Q_t(s,a) \geq Q^*(s,a)$ holds for all timesteps t and state-action pairs (s,a), with probability at least $1-\delta/3$.*

Proof It can be shown, by a similar argument as in the proof of Lemma 22, that $(1/m)\sum_{i=1}^{m}(r_{k_i} + \gamma V^*(s_{k_i})) > Q^*(s,a) - \varepsilon_1$ holds, for all attempted updates, with probability at least $1-\delta/3$. Assuming this equation does hold, the proof is by induction on the timestep t. For the base case, note that $Q_1(s,a) = U(s,a) \geq Q^*(s,a)$ for all (s,a). Now, suppose the claim holds for all timesteps less than or equal to t. Thus, we have that $Q_t(s,a) \geq Q^*(s,a)$, and $V_t(s) \geq V^*(s)$ for all (s,a). Suppose s is the t^{th} state reached and a is the action taken at time t. If it does not result in an attempted update or it results in an unsuccessful update, then no action-value estimates change, and we are done. Otherwise, by Equation 6, we have that $Q_{t+1}(s,a) = (1/m)\sum_{i=1}^{m}(r_{k_i} + \gamma V_{k_i}(s_{k_i})) + \varepsilon_1 \geq (1/m)\sum_{i=1}^{m}(r_{k_i} + \gamma V^*(s_{k_i})) + \varepsilon_1 \geq Q^*(s,a)$, by the induction hypothesis and an application of the equation from above. ■

Lemma 24 *(Strehl et al., 2006b) If Event A2 occurs, then the following statement holds: If an unsuccessful update occurs at time t and $LEARN_{t+1}(s,a) = false$, then $(s,a) \in K_{t+1}$.*

Proof (By contradiction) Suppose an unsuccessful update occurs at timestep t, $LEARN_{t+1}(s,a) = false$, and $(s,a) \notin K_{t+1}$. Let $k_1 < k_2 < \cdots < k_m$ be the most recent m timesteps in which a is taken in state s. Clearly, $k_m = t$. Because of Event A2, we have $(s,a) \in K_{k_1}$. Since no update occurred on timestep t, we have that $K_t = K_{t+1}$. It follows from $K_t = K_{t+1}$ that $(s,a) \notin K_t$, implying that there must exist some timestep $t' > k_1$ in which a successful update occurs. Thus, by the rules of Section 2.2.2, $LEARN_{t+1}(s,a)$ remains $true$, which contradicts our assumption. ■

The following lemma bounds the number of timesteps t in which a state-action pair $(s,a) \notin K_t$ is experienced.

Lemma 25 *If Event A2 occurs and $Q_t(s,a) \geq Q^*(s,a)$ holds for all timesteps t and state-action pairs (s,a), then the number of changes to the Q-function is at most $\sum_{(s,a)\in S\times A} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$, and the number of timesteps t such that a state-action pair $(s_t,a_t) \notin K_t$ is at most $2m\sum_{(s,a)\in S\times A} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$.*

Proof We claim that $Q(s,a)$ cannot be changed more than $\frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$ times. First, note that $Q(s,a)$ is initialized to $U(s,a)$ and each successful update decreases its value by at least ε_1. Now, let $a^* = \text{argmax}_a Q^*(s,a)$. By assumption $Q(s,a^*) \geq Q^*(s,a^*) = V^*(s)$. Thus, we conclude that once

$Q(s,a)$ falls below $V^*(s)$, action a will never again be chosen in state s, since actions are chosen greedily with respect to $Q(\cdot,\cdot)$. Updates to (s,a) only occur after (s,a) has been experienced. Thus, at most $\frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$ changes to $Q(s,a)$ can occur, and the total number of changes to the Q-function is at most $\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$.

Suppose $(s,a)\notin K_t$ is experienced at time t and $LEARN_t(s,a) = \mathit{false}$ (implying the last attempted update was unsuccessful). By Lemma 24, we have that $(s,a)\in K_{t'+1}$ where t' was the time of the last attempted update of (s,a). Thus, some successful update has occurred since time $t'+1$. By the rules of Section 2.2.2, we have that $LEARN(s,a)$ will be set to *true* and by Event A2, the next attempted update will succeed.

Now, suppose that $(s,a)\notin K_t$ is experienced at time t and $LEARN_t(s,a) = \mathit{true}$. Within at most m more experiences of (s,a), an attempted update of (s,a) will occur. Suppose this attempted update takes place at time q and that the m most recent experiences of (s,a) happened at times $k_1 < k_2 < \cdots < k_m = q$. By Event A2, if $(s,a)\notin K_{k_1}$, the update will be successful. Otherwise, since $(s,a)\in K_{k_1}$, some successful update must have occurred between times k_1 and t (since $K_{k_1}\neq K_t$). Hence, even if the update is unsuccessful, $LEARN(s,a)$ will remain *true*, $(s,a)\notin K_{q+1}$ will hold, and the next attempted update of (s,a) will be successful.

In either case, if $(s,a)\notin K_t$, then within at most $2m$ more experiences of (s,a), a successful update of $Q(s,a)$ will occur. Thus, reaching a state-action pair not in K_t at time t will happen at most $2m\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}} \frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}$ times. ■

Using these Lemmas we can prove the main result.

Proof (of Theorem 16) We apply Theorem 10. Set m as in Lemma 22 and let $\varepsilon_1 = \varepsilon(1-\gamma)/3$. First, note that K_t is defined with respect to the agent's action-value estimates $Q(\cdot,\cdot)$ and other quantities that don't change during learning. Thus, we have that $K_t = K_{t+1}$ unless an update to some action-value estimate takes place. We now assume that Event A2 occurs, an assumption that holds with probability at least $1-\delta/3$, by Lemma 22. By Lemma 23, we have that Condition 1 of Theorem 10 holds, namely that $V_t(s)\geq V^*(s)-\varepsilon$ for all timesteps t. Next, we claim that Condition 2, $V_t(s)-V^{\pi_t}_{M_{K_t}}(s)\leq \frac{3\varepsilon_1}{1-\gamma}=\varepsilon$ also holds. For convenience let M' denote M_{K_t}. Recall that for all (s,a), either $Q_t(s,a)=Q^{\pi_t}_{M'}(s,a)$ when $(s,a)\notin K_t$, or $Q_t(s,a)-(R(s,a)+\gamma\sum_{s'}T(s'|s,a)V_t(s'))\leq 3\varepsilon_1$ when $(s,a)\in K_t$ (by definition of K_t). Note that $V^{\pi_t}_{M'}$ is the solution to the following set of Bellman equations:

$$\begin{aligned} V^{\pi_t}_{M'}(s) &= R(s,\pi_t(s))+\gamma\sum_{s'\in S}T(s'|s,\pi_t(s))V^{\pi_t}_{M'}(s') && \text{if } (s,\pi_t(s))\in K_t,\\ V^{\pi_t}_{M'}(s) &= Q_t(s,\pi_t(s)), && \text{if } (s,\pi_t(s))\notin K_t. \end{aligned}$$

The vector V_t is the solution to a similar set of equations except with some additional positive reward terms on the right-hand side for the case $(s,\pi_t(s))\in K_t$, each bounded by $3\varepsilon_1$, due to our definition of the set K_t. This fact implies that $V_t(s)-V^{\pi_t}_{M_{K_t}}(s)\leq\frac{3\varepsilon_1}{1-\gamma}$, as desired; see, e.g., Munos and Moore (2000) for a proof. Finally, for Condition 3 of Theorem 10, we note that by Lemma 25, $\zeta(\varepsilon,\delta) = O\left(2m\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}\frac{[U(s,a)-V^*(s)]_+}{\varepsilon_1}\right) = O\left(\frac{(1+\gamma V_{\max})^2\sum_{(s,a)\in\mathcal{S}\times\mathsf{A}}[U(s,a)-V^*(s)]_+}{\varepsilon^3(1-\gamma)^3}\ln\frac{SA}{\varepsilon\delta(1-\gamma)}\right)$, where $\zeta(\varepsilon,\delta)$ is the number of updates and escape events that occur during execution of Delayed Q-learning with inputs ε and δ (equivalently, with inputs ε_1 and m, which are derived from ε and δ). ■

We've proven upper bounds on the learning complexity of Delayed Q-learning and R-MAX. The analysis techniques are general and have proven useful in analyzing other related algorithms (Asmuth et al., 2008; Brunskill et al., 2008; Leffler et al., 2007; Strehl et al., 2007; Strehl and Littman, 2008a).

4. A New Lower Bound

The main result of this section (Theorem 26) is an improvement on published lower bounds for learning in MDPs. Existing results (Kakade, 2003) show a linear dependence on S and ε, but we find that a linearithmic on S and a quadratic dependence on ε are necessary for any reinforcement-learning algorithm $\mathcal{A}$ that satisfies the following assumptions:

- $\mathcal{A}_t$ is a deterministic policy at all timesteps t, and
- $\mathcal{A}_t$ and $\mathcal{A}_{t+1}$ can differ only in s_t; namely, the action-selection policy of the algorithm may change only in the most recently visited state.

Both assumptions are introduced to simplify our analysis. We anticipate the same lower bound to hold without these assumptions as they do not appear to restrict the power of an algorithm in the family of difficult-to-learn MDPs that we will describe soon. Also, while we choose to drop dependence on $1/(1-\gamma)$ in our new lower bound to facilitate a cleaner analysis, we believe it is possible to force a quadratic dependence by a more careful analysis. Finally, we note that the analysis bears some similarity to the lower bound analysis of Leffler et al. (2005) although their result is different and is for a different learning model.

Theorem 26 *For any reinforcement-learning algorithm $\mathcal{A}$ that satisfies the two assumptions above, there exists an MDP M such that the sample complexity of $\mathcal{A}$ in M is*

$$\Omega\left(\frac{SA}{\varepsilon^2}\ln\frac{S}{\delta}\right).$$

To prove this theorem, consider the family of MDPs depicted in Figure 1. The MDPs have $S = N+2$ states: $\mathcal{S} = \{1,2,\ldots,N,+,-\}$, and A actions. For convenience, denote by $[N]$ the set $\{1,2,\ldots,N\}$. Transitions from each state $i \in [N]$ are the same, so only the transitions from state 1 are depicted. One of the actions (the solid one) deterministically transports the agent to state $+$ with reward $0.5+\varepsilon$. Let a be any of the other $A-1$ actions (the dashed ones). From any state $i \in [N]$, taking a will transition to $+$ with reward 1 and probability p_{ia}, and to $-$ with reward 0 otherwise, where $p_{ia} \in \{0.5, 0.5+2\varepsilon\}$ are numbers very close to $0.5+\varepsilon$. Furthermore, for each i, there is at most one a such that $p_{ia} = 0.5+2\varepsilon$. Transitions from states $+$ and $-$ are identical: they simply reset the agent to one of the states in $[N]$ uniformly at random.

In fact, the MDP defined above can be viewed as N copies of a multi-armed bandit problem where the states $+$ and $-$ are dummy states for randomly resetting the agent to the next "real" state. Therefore, the optimal action in a state i is independent of the optimal action in any other state: it is the solid action if $p_{ia} = 0.5$ for all dashed actions a; otherwise, it is the dashed action a for which $p_{ia} = 0.5+2\varepsilon$. Intuitively, this MDP is hard to learn for exactly the same reason that a biased coin is hard to learn if the bias (that is, the probability of head after a coin toss) is close to 0.5.

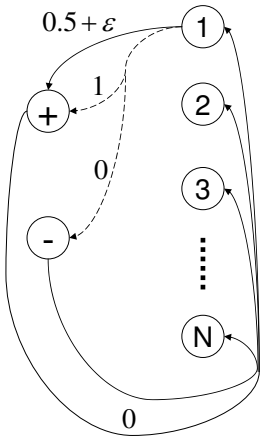

Figure 1: The difficult-to-learn MDPs for an improved sample complexity lower bound.

Lemma 27 *There exist constants $c_1, c_2 \in (0,1)$ such that during a whole run of the algorithm $\mathcal{A}$, for any state $i \in [N]$, the probability that $\mathcal{A}$ takes sub-optimal actions in i more than m_i times is at least $p(m_i)$, where*

$$p(m_i) := c_2 \exp\left(-\frac{m_i \varepsilon^2}{c_1 A}\right).$$

The following result is useful for proving Lemma 27.

Lemma 28 *(Mannor and Tsitsiklis, 2004, Theorem 1) Consider the K-armed bandit problem and let $\varepsilon, \delta \in (0,1)$. We call an algorithm $\mathcal{A}_B$ (ε,δ)-correct if it always terminates after a finite number T of trials and outputs an ε-optimal arm with probability at least $1-\delta$. Here, the sample complexity T is a random variable, and we let $\mathbf{E}$ be the expectation with respect to randomness in the bandit's rewards and $\mathcal{A}_B$ (if the algorithm is stochastic). Then there exist constants $c_1, c_2, \varepsilon_0, \delta_0 \in (0,1)$, such that for every $K \geq 2$, $\varepsilon \in (0,\varepsilon_0)$, and $\delta \in (0,\delta_0)$, and for every (ε,δ)-correct algorithm $\mathcal{A}_B$, there is a K-armed bandit problem such that*

$$\mathbf{E}[T] \geq \frac{c_1 K}{\varepsilon^2} \ln \frac{c_2}{\delta}.$$

Proof (of Lemma 27) If we treat decision making in each state as an A-arm bandit problem, finding the optimal action for that state becomes one of finding an ε-optimal arm (action) in the bandit problem. This bandit problem is the one used by Mannor and Tsitsiklis (2004) to establish the sample complexity lower bound in Lemma 28.[13]

By construction of the MDP in Figure 1, there is at most one optimal action in each state $i \in [N]$. Thus, if any RL algorithm $\mathcal{A}$ can guarantee, with probability at least $1-\delta_i$, that at most m_i sub-optimal actions are taken in state i during a whole run, then we can turn it into a bandit algorithm $\mathcal{A}_B$ with a sample complexity of $2m_i+1$ in the following way: we simply run $\mathcal{A}$ for $2m_i+1$ steps and the majority action must be ε-optimal with probability at least $1-\delta_i$. In other words, Lemma 28

13. The lower bound of Mannor and Tsitsiklis (2004) is for *expected* sample complexity. But, this result automatically applies to *worst-case* sample complexity, which is what we consider in the present paper.

for sample complexity in K-armed bandits results immediately in a lower bound for the total number of sub-optimal actions taken by $\mathcal{A}$, yielding

$$m_i \geq \frac{c_1 A}{\varepsilon^2} \ln \frac{c_2}{\delta_i}$$

for appropriately chosen constants c_1 and c_2. Reorganizing terms gives the desired result. ■

We will need two technical lemma to prove the lower bound. Their proofs are given after the proof of the main theorem.

Lemma 29 *Let c and Δ be constants in $(0,1)$. Under the constraints $\sum_i m_i \leq \zeta$ and $m_i > 0$ for all i, the function*

$$f(m_1, m_2, \ldots, m_N) = 1 - \prod_{i=1}^{N} (1 - c\Delta^{m_i})$$

is minimized when $m_1 = m_2 = \cdots = m_N = \frac{\zeta}{N}$. Therefore,

$$f(m_1, m_2, \ldots, m_N) \geq 1 - \left(1 - c\Delta^{\frac{\zeta}{N}}\right)^N.$$

Lemma 30 *If there exist some constants $c_1, c_2 > 0$ such that*

$$\delta \geq 1 - \left(1 - c_2 \exp\left(-\frac{\zeta\eta}{c_1\Psi}\right)\right)^{\Psi},$$

for some positive quantities η, ζ, Ψ, and δ, then

$$\zeta = \Omega\left(\frac{\Psi}{\eta} \ln \frac{\Psi}{\delta}\right).$$

Proof (of Theorem 26) Let $\zeta(\varepsilon, \delta)$ be an upper bound of the sample complexity of any PAC-MDP algorithm $\mathcal{A}$ with probability at least $1 - \delta$. Let sub-optimal actions be taken m_i times in state $i \in [N]$ during a whole run of $\mathcal{A}$. Consequently,

$$\delta \geq \Pr\left(\sum_{i=1}^{N} m_i > \zeta(\varepsilon, \delta)\right) = 1 - \Pr\left(\sum_{i=1}^{N} m_i \leq \zeta(\varepsilon, \delta)\right),$$

where the first step is because the actual sample complexity of $\mathcal{A}$ is at least $\sum_i m_i$.

We wish to find a lower bound for the last expression above by optimizing the values of m_i's subject to the constraint, $\sum_i m_i \leq \zeta(\varepsilon, \delta)$. Due to the statistical independence of what states $i \in [N]$ are visited by the algorithm,[14] we can factor the probability above to obtain

$$\delta \geq 1 - \max_{m_1, \ldots, m_N; \sum_i m_i \leq \zeta(\varepsilon,\delta)} \prod_{i=1}^{N} (1 - p(m_i)).$$

14. It does not help for the algorithm to base its policy in one state on samples collected in other states, due to the independence of states in this MDP. If an algorithm attempts to do so, an adversary can make use of this fact to assign p_{ia} to even *increase* the failure probability of the algorithm.

where Lemma 27 is applied.

We now use Lemma 29 to obtain a lower bound of the last expression above, which in turn lower-bounds δ. Applying this lemma with $c = c_2$ and $\Delta = \exp(-\frac{\varepsilon^2}{c_1 A})$ gives

$$\delta \geq 1 - \left(1 - c_2 \exp\left(-\frac{\zeta(\varepsilon,\delta)\varepsilon^2}{c_1 NA}\right)\right)^N. \tag{9}$$

The theorem then follows immediately from Lemma 30 using $\Psi = N$ and $\eta = \varepsilon^2/A$. ■

Proof (of Lemma 29) Since $f(m_1,\ldots,m_N) \in (0,1)$, finding the *minimum* of f is equivalent to finding the *maximum* of the following function:

$$g(m_1,m_2,\ldots,m_N) = \ln(1 - f(m_1,m_2,\ldots,m_N)) = \sum_{i=1}^{N} \ln(1 - c\Delta^{m_i}),$$

under the same constraints. Due to the concavity of $\ln(\cdot)$, we have

$$g(m_1,m_2,\ldots,m_N) \leq N \ln\left(\frac{1}{N}\sum_{i=1}^{N}(1 - c\Delta^{m_i})\right) = N \ln\left(1 - \frac{c}{N}\sum_{i=1}^{N}\Delta^{m_i}\right).$$

Finally, we use the fact that the arithmetic mean is no less than the geometric mean to further simplify the upper bound of g:

$$g(m_1,m_2,\ldots,m_N) \leq N \ln\left(1 - c\Delta^{\frac{1}{N}\sum_{i=1}^{N} m_i}\right) \leq N \ln\left(1 - c\Delta^{\frac{\zeta}{N}}\right).$$

Equality holds in all inequalities above when $m_1 = m_2 = \cdots = m_N = \frac{\zeta}{N}$. ■

Proof (of Lemma 30) Reorganizing terms in Equation (9) gives

$$1 - c_2 \exp\left(-\frac{\zeta\eta}{c_1\Psi}\right) \geq (1-\delta)^{\frac{1}{\Psi}}.$$

The function $(1-\delta)^{1/\delta}$ is a decreasing function of δ for $0 < \delta < 1$, and $\lim_{\delta\to 0^+}(1-\delta)^{1/\delta} = 1/e$. Therefore, as long as δ is less than some constant $c_3 \in (0,1)$, we will have

$$(1-\delta)^{\frac{1}{\Psi}} = \left((1-\delta)^{\frac{1}{\delta}}\right)^{\frac{\delta}{\Psi}} \geq (c_4)^{\frac{\delta}{\Psi}} = \exp\left(-\frac{c_5\delta}{\Psi}\right),$$

where $c_4 = (1-c_3)^{1/c_3} \in \left(0,\frac{1}{e}\right)$ and $c_5 = \ln\frac{1}{c_4} \in (1,\infty)$ are two constants. It is important to note that c_3 (and thus c_4 and c_5) does not depend on η or Ψ. Now, apply the inequality $e^x \geq 1 + x$ for $x = -c_5\delta/\Psi$ to get $\exp(-c_5\delta/\Psi) \geq 1 - c_5\delta/\Psi$. The above chain of inequalities results in:

$$1 - c_2 \exp\left(-\frac{\zeta\eta}{c_1\Psi}\right) \geq 1 - \frac{c_5\delta}{\Psi}.$$

Solving this inequality for ζ gives the desired lower bound for ζ. ■

We have shown a new sample complexity lower bound that has a linearithmic dependence on S in the worst case. Thus, Delayed Q-learning is optimal in the sense of minimizing the dependence (of sample complexity of exploration) on the number of states.

5. Conclusion

We have presented and improved PAC-MDP upper and lower bounds reported in the literature. We studied two algorithms, R-MAX (which is model *based*) and Delayed Q-learning (which is model *free*) that are able to make use of non-trivial admissible heuristic functions. Comparing the relative strengths of model-based and model-free algorithms has been an important problem in the reinforcement-learning community (see, e.g., Atkeson and Gordon 1997 and Kearns and Singh 1999). Our analysis indicates that both can learn efficiently in finite MDPs in the PAC-MDP framework. The bounds suggest that a model-free method can be less sensitive on the size of the state space (linearithmic vs. quadratic dependence in the bound, matching the lower bound) whereas a model-based method can be less sensitive to the effective horizon, $1/(1-\gamma)$. Future work should focus on tightening bounds further and expanding analyses to state spaces in which generalization is necessary.

Acknowledgments

The authors appreciate supports from the National Science Foundation (IIS-0325281) and Rutgers University (Bevier fellowship). We thank John Langford, Eric Wiewiora, Sham Kakade, and Csaba Szepesvári for helpful discussions, especially on the analysis of Delayed Q-learning. We also thank the anonymous reviewers for their insightful comments that have significantly improved the quality of the paper.

References

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

John Asmuth, Michael L. Littman, and Robert Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 604–609. AAAI Press, 2008.

John Asmuth, Lihong Li, Michael L. Littman, Ali Nouri, and David Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Christopher G. Atkeson and Geoffrey J. Gordon, editors. *Proceedings of the ICML-97 Workshop on Modelling in Reinforcement Learning*, 1997. URL `http://www.cs.cmu.edu/~ggordon/ml97ws/`.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *Advances in Neural Information Processing Systems 21*, pages 89–96, 2009.

Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

Ronen I. Brafman and Moshe Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Emma Brunskill, Bethany R. Leffler, Lihong Li, Michael L. Littman, and Nicholas Roy. CORL: A continuous-state offset-dynamics reinforcement learner. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 53–61, 2008. An extended version appears in the Journal of Machine Learning Research, volume 10, pages 1955–1988, 2009.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

Carlos Diuk, Lihong Li, and Bethany R. Leffler. The adaptive *k*-meteorologists problem and its application to structure discovery and feature selection in reinforcement learning. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pages 249–256, 2009.

Michael O. Duff and Andrew G. Barto. Local bandit approximation for optimal learning problems. In *Advances in Neural Information Processing Systems*, volume 9, pages 1019–1025. The MIT Press, 1997.

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

Claude-Nicolas Fiechter. Efficient reinforcement learning. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 88–97. Association of Computing Machinery, 1994.

John C. Gittins. *Multi-armed Bandit Allocation Indices*. Wiley Interscience Series in Systems and Optimization. John Wiley & Sons Inc, Chichester, NY, 1989.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Michael J. Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 740–747, 1999.

Michael J. Kearns and Satinder P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems 11*, pages 996–1002. The MIT Press, 1999.

Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002.

Sven Koenig and Reid G. Simmons. The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22(1–3):227–250, 1996.

Bethany R. Leffler, Michael L. Littman, Alexander L. Strehl, and Thomas J. Walsh. Efficient exploration with latent structure. In *Robotics: Science and Systems*, pages 81–88, 2005.

Bethany R. Leffler, Michael L. Littman, and Timothy Edmunds. Efficient reinforcement learning with relocatable action models. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 572–577, 2007.

Lihong Li. *A Unifying Framework for Computational Reinforcement Learning Theory*. PhD thesis, Rutgers University, New Brunswick, NJ, 2009.

Lihong Li, Michael L. Littman, and Christopher R. Mansley. Online exploration in least-squares policy iteration. In *Proceedings of the Eighteenth International Conference on Agents and Multiagent Systems*, pages 733–739, 2009.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithms. *Machine Learning*, 2(4):285–318, 1988.

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

Colin McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141 of *London Mathematical Society Lecture Notes*, pages 148–188. Cambridge University Press, 1989.

Rémi Munos and Andrew W. Moore. Rates of convergence for variable resolution schemes in optimal control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 647–654, 2000.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999.

Ali Nouri and Michael L. Littman. Multi-resolution exploration in continuous spaces. In *Advances in Neural Information Processing Systems 21*, pages 1209–1216, 2009.

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, pages 697–704, 2006.

Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1994. ISBN 0-13-103805-2.

Satinder P. Singh and Richard C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

Alexander L. Strehl and Michael L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 857–864, 2005.

Alexander L. Strehl and Michael L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Advances in Neural Information Processing Systems 20*, pages 1417–1424, 2008a.

Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008b.

Alexander L. Strehl, Lihong Li, and Michael L. Littman. Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 485–493, 2006a.

Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *Proceedings of the Twenty-Third International Conference on Machine learning*, pages 881–888, 2006b.

Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. Efficient structure learning in factored-state MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 645–650, 2007.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Csaba Szepesvári. The asymptotic convergence-rate of Q-learning. In *Advances in Neural Information Processing Systems 10*, pages 1064–1070, 1998.

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

Thomas J. Walsh, István Szita, Carlos Diuk, and Michael L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J. Weinberger. Inequalities for the L1 deviation of the empirical distribution. Technical Report HPL-2003-97R1, Hewlett-Packard Labs, 2003.