V-MAX: Tempered Optimism for Better PAC Reinforcement Learning

Karun Rao
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
karunrao97@gmail.com

Shimon Whiteson
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
s.a.whiteson@uva.nl

ABSTRACT

Recent advances in reinforcement learning have yielded several *PAC-MDP* algorithms that, using the principle of optimism in the face of uncertainty, are guaranteed to act near-optimally with high probability on all but a polynomial number of samples. Unfortunately, many of these algorithms, such as *R-MAX*, perform poorly in practice because their initial exploration in each state, before the associated model parameters have been learned with confidence, is random. Others, such as *Model-Based Interval Estimation* (MBIE) have weaker sample complexity bounds and require careful parameter tuning. This paper proposes a new PAC-MDP algorithm called *V-MAX* designed to address these problems. By restricting its optimism to future visits, V-MAX can exploit its experience early in learning and thus obtain more cumulative reward than R-MAX. Furthermore, doing so does not compromise the quality of exploration, as we prove bounds on the sample complexity of V-MAX that are identical to those of R-MAX. Finally, we present empirical results in two domains demonstrating that V-MAX can substantially outperform R-MAX and match or outperform MBIE while being easier to tune, as its performance is invariant to conservative choices of its primary parameter.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—Learning

General Terms

Algorithms

Keywords

Reinforcement learning, Sample complexity

1. INTRODUCTION

In *reinforcement learning* (RL) [17], an agent must learn an optimal *policy* for maximising its expected long-term reward in an initially unknown *Markov decision process* (MDP) [1]. Since a wide range of realistic problems, from game playing to robot control, can be naturally formulated as MDPs, effective reinforcement-learning algorithms are critical to the development of intelligent agents.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

A central challenge in RL is how best to balance *exploration*, in which the agent tries various actions to learn about their effects, and *exploitation*, in which it uses what it has already learned to select actions that maximise expected return. Traditional RL algorithms such as *Q-Learning* [21] rely on ad-hoc exploration mechanisms that ensure each state-action pair is experienced infinitely often. As a result, the convergence of such algorithms to the optimal policy is guaranteed only in the limit.

Fortunately, methods have recently been developed that explore more efficiently and thus obtain guarantees based on the *probably approximately correct* (PAC) [20] framework. Instead of converging to the optimal policy, PAC-MDP algorithms are guaranteed to act near-optimally with high probability on all but a polynomial number of samples.

R-MAX [2], the most well known PAC-MDP method, formalises the principle of *optimism in the face of uncertainty.* If n, the number of times a state-action pair has been visited, is less than a threshold m, it is assumed to have maximal value. Planning on the resulting model yields a policy that either leads the agent to unfamiliar state-action pairs or exploits familiar ones of high value.

Despite its PAC guarantees, R-MAX often performs poorly in practice because its initial exploration is random: until a state-action pair has been visited m times, the agent's experience with it is ignored. *Model-Based Interval Estimation* (MBIE) [16] avoids this problem by computing confidence intervals that quantify the agent's optimism. While MBIE has been shown to outperform R-MAX empirically, the bounds on its sample complexity are not as strong [16].

This paper describes *V-MAX*, a novel PAC-MDP method designed to overcome the weaknesses of both R-MAX and MBIE. Like R-MAX, V-MAX is optimistic about state-action pairs experienced fewer than m times. However, like MBIE, its optimism is tempered by experience. Rather than assuming such state-action pairs have maximal value, it assumes only that the remaining $m - n$ visits will yield maximal return. Thus, its estimate of the state-action pair's value is a weighted average of the expected return based on the n visits and that of the $m - n$ optimistic future visits.

In this way, V-MAX can exploit its experience early in learning and thus obtain more cumulative reward than R-MAX. Furthermore, this additional exploitation does not make exploration less efficient. On the contrary, we prove bounds on the sample complexity of V-MAX that are identical to those of R-MAX. This result shows for the first time that it is possible to employ tempered optimism like that of MBIE but without compromising the resulting PAC bounds.

This paper also presents empirical results comparing the performance of V-MAX to MBIE, R-MAX, and MoR-MAX, another PAC-MDP method, on two tasks: a standard benchmark task from the PAC-MDP literature and a new task containing multiple local optima that is designed to be challenging for PAC-MDP methods. The results on both tasks demonstrate that V-MAX can substantially outperform both R-MAX and MoR-MAX. They also demonstrate that V-MAX, unlike MBIE, performs well in the presence of multiple local optima. Even when such local optima are absent, V-MAX matches the performance of MBIE and in both cases proves substantially easier to tune, as its performance is invariant for conservative choices of m.

The rest of this paper is organized as follows. Section 2 provides background on RL, PAC, and PAC-MDP methods. Section 3 presents V-MAX and a proof of its sample complexity. Section 4 describes our experimental results while Section 5 concludes and suggests directions for future work.

2. BACKGROUND

A Markov decision process can be described as a five-tuple (S, A, R, T, γ), where S is the state space, A is the action space, $R(s, a)$ is the reward function describing the expected reward given state s and action a, $T(s, a, s')$ is the transition function describing the probability of arriving in state s' given s and a, and $0 \leq \gamma < 1$ is the discount factor used when summing an infinite sequence of rewards. An agent's policy $\pi : S \to A$ specifies what action to take in each state. An optimal policy π^* maximises the expected γ-discounted long-term cumulative reward, also known as the *expected discounted return* [17].

If both the reward and transition functions are known, then an agent can compute π^* using a planning method such as *value iteration* (VI) [1]. VI computes the *optimal action-value function* Q^* by iteratively solving the *Bellman optimality equation* (1), until for each state-action pair (s, a), subsequent computations of $Q^*(s, a)$ yield a difference less than some tolerance ϵ. Given Q^*, an optimal policy can be easily derived: $\pi^*(s) = \operatorname{argmax}_{a \in A} Q^*(s, a)$.

$$Q^*(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \max_{a' \in A} Q^*(s', a') \qquad (1)$$

If the reward and/or transition functions are unknown, then the agent can learn Q^* from experience. *Model-based* RL methods do so indirectly by learning $\hat{R}$ and $\hat{T}$, maximum likelihood estimates of the reward and transition functions, and planning on the resulting MDP: $(S, A, \hat{R}, \hat{T}, \gamma)$.

Learning $\hat{R}$ and $\hat{T}$ is complicated by the need to explore the environment to collect samples. If the agent explores too little, it may not learn the dynamics of the MDP accurately. If it explores too much, it may not accumulate enough reward. While it is possible in principle to compute a Bayes-optimal exploration strategy, doing so is typically intractable [3]. As a result, ad-hoc strategies such as ϵ-*greedy* exploration are often used in practice. The probably approximately correct (PAC) learning framework offers a middle ground: tractable algorithms that, while not Bayes-optimal, have upper bounds on their *sample complexity*:

Definition 1. For any fixed $\epsilon > 0$, the *sample complexity of exploration* of an algorithm $\mathcal{A}$ is the number of timesteps t such that the policy at time t, $\mathcal{A}_t$, satisfies $V^{\mathcal{A}_t}(s_t) < V^*(s_t) - \epsilon$ [8].

A PAC-MDP method is one that is guaranteed to have, with high probability, a polynomial sample complexity:

Definition 2. An algorithm $\mathcal{A}$ is *PAC-MDP* (Probably Approximately Correct in Markov Decision Processes) if, for any $\epsilon > 0$ and $0 < \delta < 1$, the sample complexity of $\mathcal{A}$ is less than some polynomial in the relevant quantities $(|S|, |A|, R_{\max}, 1/\epsilon, 1/\delta, 1/(1 - \gamma))$ for any MDP M, with probability at least $1 - \delta$, where $R_{\max}$ is an upper bound on the reward function of M [13].[1]

R-MAX [2], described in Algorithm 1, is the simplest and most well known PAC-MDP algorithm. In addition to S, A, γ, and $R_{\max}$, it takes as input two parameters: m, the number of times each state-action pair must be experienced before $\hat{R}$ and $\hat{T}$ are considered near-accurate, and ϵ_1, the accuracy required from VI during the planning step.

Algorithm 1: R-MAX

```
Input: $S$, $A$, $\gamma$, $m$, $\epsilon_1$, $R_{\max}$
1  $\bar{S} \leftarrow S \cup \{z\}$, where $z$ is an arbitrary fictitious state
2  foreach $(s, a) \in \bar{S} \times A$ do
3      $n(s, a) \leftarrow 0$
4      $r(s, a) \leftarrow 0$
5      $\tilde{Q}(s, a) \leftarrow R_{\max}/(1 - \gamma)$
6      $\tilde{R}(s, a) \leftarrow R_{\max}$
7      foreach $s' \in S$ do
8          $n(s, a, s') \leftarrow 0$
9          $\tilde{T}(s, a, s') \leftarrow 0$
10     end
11     $n(s, a, z) \leftarrow 0$
12     $\tilde{T}(s, a, z) \leftarrow 1$
13 end
14 for $t = 1, 2, 3, \ldots$ do
15     Observe current state $s$
16     Execute action $a := \operatorname{argmax}_{a' \in A} \tilde{Q}(s, a')$
17     Observe immediate reward $r$ and next state $s'$
18     if $n(s, a) < m$ then
19         $n(s, a) \leftarrow n(s, a) + 1$
20         $r(s, a) \leftarrow r(s, a) + r$
21         $n(s, a, s') \leftarrow n(s, a, s') + 1$
22         if $n(s, a) = m$ then
23             $\tilde{R}(s, a) \leftarrow r(s, a)/m$
24             foreach $s'' \in \bar{S}$ do $\tilde{T}(s, a, s'') \leftarrow n(s, a, s'')/m$
25             $\tilde{Q} \leftarrow$ Solve $(\bar{S}, A, \tilde{R}, \tilde{T}, \gamma, \epsilon_1)$ using VI
26         end
27     end
28 end
```

R-MAX adds a fictitious maximally rewarding state z to the MDP and initially assumes that all state-action pairs (s, a) (including all (z, a)) yield the maximum reward $R_{\max}$ and transition with probability 1 to z.[2] We use $\tilde{R}$ and $\tilde{T}$ to denote these initially optimistic reward and transition functions. In addition, $\tilde{Q}$, the optimistic value function, is initialized to the maximum possible value, i.e., $\tilde{Q}(s, a) = V_{\max} = R_{\max}/(1 - \gamma)$ for all s and a. Once some (s, a) has been experienced m times, $\tilde{R}(s, a)$ and $\tilde{T}(s, a)$ are set to $\hat{R}(s, a)$ and $\hat{T}(s, a)$, respectively, and the $\tilde{Q}$-values are updated with VI. Thus, R-MAX automatically explores state-action pairs that it is uncertain about and exploits otherwise. Once some (s, a) has been experienced m times, R-MAX assumes the

[1]This definition is slightly modified in that it allows the bounds to also depend on $R_{\max}$ since, unlike [13], we do not assume that $R_{\max} = 1$. In addition, we do not consider the space and computational complexity.

[2]We use z for ease of comparison to V-MAX, but R-MAX can also be implemented by initialising all state-action pairs to have self-transitions with probability 1.

reward and transition functions for (s, a) are near-accurate and stops learning them. The tightest known upper bounds on R-MAX's sample complexity, due to [13], are:[3]

$$O\left(\frac{|S||A|R_{\max}^3}{\epsilon^3(1-\gamma)^6}\left(|S| + \ln\frac{|S||A|}{\delta}\right)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon(1-\gamma)}\right).$$

Modified R-MAX (MoR-MAX) [19] is similar to R-MAX except that, once a state-action pair (s, a) has been experienced m times, it restarts the sample collection for (s, a). For every m such samples that MoR-MAX collects, it creates a trial model consisting of the reward and transition functions based on the m most recent samples of (s, a). It then uses VI to compute $\tilde{Q}'$ based on the trial model and, if $\tilde{Q}'(s, a) \le \tilde{Q}(s, a)$, then it replaces the reward and transition functions for (s, a) with those of the trial model. In this way, MoR-MAX continues learning throughout the agent's lifetime, unlike R-MAX. Note that each time MoR-MAX replaces the reward and transition functions using m new samples, the previous m samples are discarded. With probability $1 - \delta$, the sample complexity of MoR-MAX is:

$$O\left(\frac{|S||A|R_{\max}^2}{\epsilon^2(1-\gamma)^6}\ln\frac{|S||A|R_{\max}}{\delta\epsilon(1-\gamma)}\ln^2\frac{R_{\max}}{\epsilon(1-\gamma)}\right).$$

Ignoring log factors, the sample complexity bounds for MoR-MAX are better than R-MAX in terms of $|S|$, $R_{\max}$, and $1/\epsilon$, and the same in terms of $|A|$ and $1/(1-\gamma)$.

Model-based Interval Estimation (MBIE) [16] improves on the empirical performance of R-MAX by computing confidence intervals on the reward and transition functions.[4] Like R-MAX, MBIE initialises all $\tilde{Q}$-values to $V_{\max}$. Thereafter, at each timestep, it computes the $\tilde{Q}$-values using VI with the following equation in place of the Bellman optimality equation:

$$\tilde{Q}(s,a) \leftarrow \hat{R}(s,a) + \gamma\sum_{s'\in S}\hat{T}(s,a,s')\max_{a'\in A}\tilde{Q}(s',a') + \frac{\beta}{\sqrt{n(s,a)}},$$

where β is an input parameter controlling the balance between exploration and exploitation. Thus, MBIE provides an exploration bonus that drives the agent towards state-action pairs that have been visited fewer times. With probability $1 - \delta$, the sample complexity of MBIE is:

$$O\left(\frac{|S||A|R_{\max}^3}{\epsilon^3(1-\gamma)^6}\left(|S| + \ln\frac{|S||A|R_{\max}}{\delta\epsilon(1-\gamma)}\right)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon(1-\gamma)}\right).$$

The bounds for MBIE are similar to those of R-MAX, except in log factors, where they are worse in terms of $R_{\max}$, $1/\epsilon$, and $1/(1-\gamma)$.

3. METHOD

A critical weakness of R-MAX is that it performs poorly early in learning: since state-action pairs are indistinguishable until they have been experienced m times, R-MAX can only explore randomly during this phase. In essence, this poor performance is due to excessive optimism. To illustrate this point, consider the following example: an MDP with a single state s and two actions a_1 and a_2, which always yield rewards of 0 and $R_{\max}$, respectively. Suppose that at some timestep t, both (s, a_1) and (s, a_2) have been experienced n times each, where $n < m$.[5] Since R-MAX is still uncertain about both state-action pairs, it assumes that $\tilde{R}_t(s, a_1) = \tilde{R}_t(s, a_2) = R_{\max}$ and thus chooses an action randomly. However, given that the n experiences of (s, a_1) produced a reward of 0, R-MAX is clearly overly optimistic about $R(s, a_1)$. Even if a_1 always generates a reward of $R_{\max}$ in the future, $\hat{R}(s, a_1)$ will be at most $(0 * n + (m - n) * R_{\max})/m = (m - n)R_{\max}/m$ by the end of learning, while $\hat{R}(s, a_2)$ can obviously be as high as $R_{\max}$.

The key idea behind V-MAX, our novel PAC-MDP algorithm, is to exploit this insight to temper the excessive optimism of R-MAX. Rather than assuming all state-action pairs visited fewer than m times have maximal value, V-MAX assumes only that the remaining $m-n$ visits will yield maximal return. Hence, V-MAX remains optimistic but can more quickly exploit what it learns. In our example, V-MAX would only need to choose a_1 once to know it is sub-optimal, thereby improving performance early in learning. In the remainder of this section, we formalise the V-MAX algorithm and prove bounds on its sample complexity.

3.1 The V-MAX Algorithm

V-MAX initialises $\tilde{Q}$, $\tilde{R}$, and $\tilde{T}$ exactly as R-MAX does. Unlike R-MAX, however, V-MAX updates $\tilde{R}$ and $\tilde{T}$ at each timestep, using the following equations:

$$\tilde{R}(s,a) = \frac{n(s,a)\hat{R}(s,a) + (m - n(s,a))R_{\max}}{m} \quad (2)$$

$$\tilde{T}(s,a,s') = \begin{cases} \dfrac{n(s,a)\hat{T}(s,a,s')}{m} & \text{if } s' \ne z \\ \dfrac{m - n(s,a)}{m} & \text{if } s' = z. \end{cases} \quad (3)$$

Thus, both the reward and transition function updates mix the observed rewards and transitions with the optimistic assumption that future samples will involve transitions to the maximally rewarding state z. V-MAX also updates $\tilde{Q}$ at each timestep, using VI on the MDP $(S \cup \{z\}, A, \tilde{R}, \tilde{T}, \gamma)$, and then simply follows a greedy policy with respect to $\tilde{Q}$.

A simpler alternative implementation can be derived that uses the empirical reward and transition functions $\hat{R}$ and $\hat{T}$ to compute $\tilde{Q}$ directly, thus eliminating the need to compute $\tilde{R}$ and $\tilde{T}$ and explicitly represent z. Substituting Equations 2 and 3 into the Bellman optimality equation (and omitting (s, a) in the notation for brevity), yields:

$$\begin{aligned}\tilde{Q} &= \frac{n\hat{R} + (m-n)R_{\max}}{m} + \gamma\Big(\frac{m-n}{m}\max_{a'\in A}\tilde{Q}(z,a') \\ &\quad + \sum_{s'\in S}\frac{n\hat{T}(s')}{m}\max_{a'\in A}\tilde{Q}(s',a')\Big) \\ &= \frac{n}{m}\Big(\hat{R} + \gamma\sum_{s'\in S}\hat{T}(s')\max_{a'\in A}\tilde{Q}(s',a')\Big) \\ &\quad + \Big(\frac{m-n}{m}\Big)\Big(R_{\max} + \gamma\max_{a'\in A}\tilde{Q}(z,a')\Big) \\ &= \frac{n}{m}\Big(\hat{R} + \gamma\sum_{s'\in S}\hat{T}(s')\max_{a'\in A}\tilde{Q}(s',a')\Big) + \Big(1 - \frac{n}{m}\Big)V_{\max}.\end{aligned}$$

[3]There are minor differences in the bounds we state, since [13] assume that $R_{\max} = 1$. Also, their bounds include the use of an admissible heuristic for initialising Q-values.

[4]We describe a variant of MBIE that Strehl and Littman call *MBIE with exploration bonus*. We consider only this variant because it is simpler and refer to it as MBIE for conciseness.

[5]We assume the agent does not know that the MDP is deterministic, and thus $m > 1$.

The resulting implementation of V-MAX, described in Algorithm 2, initialises $\tilde{Q}(s,a)$ to $V_{\max}$ for all $(s,a) \in S \times A$. At each timestep, it updates $\tilde{Q}$ using VI on the MDP $(S, A, \hat{R}, \hat{T}, \gamma)$ but with the following equation in place of the Bellman optimality equation:

$$\tilde{Q}(s,a) = \frac{n(s,a)}{m}\left(\hat{R}(s,a) + \gamma \sum_{s' \in S} \hat{T}(s,a,s') \max_{a' \in A} \tilde{Q}(s',a')\right) + \left(1 - \frac{n(s,a)}{m}\right) V_{\max}. \tag{4}$$

Finally, the agent follows a greedy policy with respect to $\tilde{Q}$. The name V-MAX is inspired by this implementation, as it computes an optimistic value function directly from $V_{\max}$.

Algorithm 2: V-MAX

```
Input: $S$, $A$, $\gamma$, $m$, $\epsilon_1$, $R_{\max}$
$V_{\max} \leftarrow R_{\max}/(1-\gamma)$
foreach $(s,a) \in S \times A$ do
    $n(s,a) \leftarrow 0$
    $r(s,a) \leftarrow 0$
    foreach $s' \in S$ do $n(s,a,s') \leftarrow 0$
    $\tilde{Q}(s,a) \leftarrow V_{\max}$
end
for $t = 1, 2, 3, \ldots$ do
    Observe current state $s$
    Execute action $a := \text{argmax}_{a' \in A} \tilde{Q}(s,a')$
    Observe immediate reward $r$ and next state $s'$
    if $n(s,a) < m$ then
        $n(s,a) \leftarrow n(s,a) + 1$
        $r(s,a) \leftarrow r(s,a) + r$
        $n(s,a,s') \leftarrow n(s,a,s') + 1$
        $\hat{R}(s,a) \leftarrow r(s,a)/n(s,a)$
        foreach $s' \in S$ do $\hat{T}(s,a,s') \leftarrow n(s,a,s')/n(s,a)$
        repeat
            $\Delta \leftarrow 0$
            foreach $(s,a) \in S \times A$ do
                if $n(s,a) > 0$ then
                    $q \leftarrow \tilde{Q}(s,a)$
                    $Q \leftarrow \hat{R}(s,a) + \gamma \sum_{s' \in S} \hat{T}(s,a,s') \max_{a' \in A} \tilde{Q}(s',a')$
                    $\tilde{Q}(s,a) \leftarrow (n(s,a)/m)Q + (1 - n(s,a)/m)V_{\max}$
                    $\Delta \leftarrow \max(\Delta, |q - \tilde{Q}(s,a)|)$
                end
            end
        until $\Delta \le \epsilon_1$
    end
end
```

3.2 Sample Complexity of V-MAX

Section 4 will demonstrate that V-MAX can substantially outperform performance R-MAX. Here, we show that these empirical advantages do not come at the expense of its theoretical properties, by proving upper bounds on its sample complexity identical to those of R-MAX. As the example in Section 3 illustrates, V-MAX can explore less than R-MAX. Thus, the bounds we prove here are critical for ensuring that V-MAX's tempered optimism does not increase the chance of converging to a suboptimal model.

We begin with supporting lemmas showing that the value function used by V-MAX decreases monotonically and is always at most $V_{\max}$. We then prove the main result in Theorem 1, using techniques similar to [13] and [16].

In the following, let the *value iteration step* (vstep) i be the i^{th} value iteration update (Lines 23-24 of Algorithm 2) since $t = 1$. Let $n_{[i]}$, $\tilde{Q}_{[i]}$, $\hat{R}_{[i]}$, and $\hat{T}_{[i]}$ denote the values of n, $\tilde{Q}$, $\hat{R}$, and $\hat{T}$, respectively, at vstep i. Finally, let $(s_{[i]}, a_{[i]})$ be the state-action pair whose $\tilde{Q}$-value is updated at vstep i. As with other PAC-MDP algorithms, we assume that all rewards are non-negative, and are upper bounded by $R_{\max}$.

LEMMA 1. *For all $(s,a) \in S \times A$, and for all policies π and timesteps t:*

$$\tilde{Q}^{\pi}_{t}(s,a) \le V_{\max}.$$

PROOF. Using weak induction on the vsteps i, we prove that for all $(s,a) \in S \times A$ and for all policies π, $\tilde{Q}^{\pi}_{[i]}(s,a) \le V_{\max}$. The base case, that $\tilde{Q}^{\pi}_{[0]}(s,a) = V_{\max}$ for all $(s,a) \in S \times A$, holds because all $\tilde{Q}$-values are initialised to $V_{\max}$. In the inductive step, we assume that, for all $(s,a) \in S \times A$, $\tilde{Q}^{\pi}_{[k]}(s,a) \le V_{\max}$ and must prove that $\tilde{Q}^{\pi}_{[k+1]}(s,a) \le V_{\max}$.

Note that $(s_{[k+1]}, a_{[k+1]})$ is the only state-action pair whose value is updated at vstep $k+1$, so $\tilde{Q}^{\pi}_{[k+1]}(s',a') = \tilde{Q}^{\pi}_{[k]}(s',a')$ for all other $(s',a') \ne (s_{[k+1]}, a_{[k+1]})$. Thus, by the inductive hypothesis, $\tilde{Q}^{\pi}_{[k+1]}(s',a') \le V_{\max}$. Consequently, we only need to prove that:

$$\tilde{Q}^{\pi}_{[k+1]}(s_{[k+1]}, a_{[k+1]}) \le V_{\max}$$

This can be derived from Equation 4 as follows (omitting $(s_{[k+1]}, a_{[k+1]})$ in the notation for brevity).

$$\begin{aligned}
\tilde{Q}^{\pi}_{[k+1]} &= \frac{n_{[k+1]}}{m}\left(\hat{R}_{[k+1]} + \gamma \sum_{s' \in S} \hat{T}_{[k+1]}(s') \tilde{Q}^{\pi}_{[k]}(s', \pi(s'))\right) \\
&\quad + \left(1 - \frac{n_{[k+1]}}{m}\right) V_{\max} \\
&\le \frac{n_{[k+1]}}{m}\left(R_{\max} + \gamma \sum_{s' \in S} \hat{T}_{[k+1]}(s') V_{\max}\right) \\
&\quad + \left(1 - \frac{n_{[k+1]}}{m}\right) V_{\max} \\
&\le \frac{n_{[k+1]}}{m} V_{\max} + \left(1 - \frac{n_{[k+1]}}{m}\right) V_{\max} \\
&\le V_{\max} \quad \square
\end{aligned}$$

LEMMA 2. *For all $(s,a) \in S \times A$, and for all policies π and timesteps $t > 0$:*

$$\tilde{Q}^{\pi}_{t}(s,a) \le \tilde{Q}^{\pi}_{t-1}(s,a).$$

PROOF. Using strong induction on the vsteps i, we prove that, for all $(s,a) \in S \times A$, $\tilde{Q}^{\pi}_{[i]}(s,a) \le \tilde{Q}^{\pi}_{[i-1]}(s,a)$, i.e., $\tilde{Q}^{\pi}(s,a)$ never increases as the number of value iteration updates increases. For the base case, $\tilde{Q}^{\pi}_{[0]}(s,a) = V_{\max}$ for all $(s,a) \in S \times A$ since all $\tilde{Q}$-values are initialised to $V_{\max}$. Also, from Lemma 1, $\tilde{Q}^{\pi}_{[1]}(s,a) \le V_{\max}$. Thus $\tilde{Q}^{\pi}_{[1]}(s,a) \le \tilde{Q}^{\pi}_{[0]}(s,a)$. For the inductive step, we assume that, for all $(s,a) \in S \times A$, and for all j such that $0 < j \le k$, $\tilde{Q}^{\pi}_{[j]}(s,a) \le \tilde{Q}^{\pi}_{[j-1]}(s,a)$. We need to prove that $\tilde{Q}^{\pi}_{[k+1]}(s,a) \le \tilde{Q}^{\pi}_{[k]}(s,a)$.

Note that $(s_{[k+1]}, a_{[k+1]})$ is the only state-action pair whose value is updated at vstep $k+1$, so $\tilde{Q}^{\pi}_{[k+1]}(s',a') = \tilde{Q}^{\pi}_{[k]}(s',a')$ for all other $(s',a') \ne (s_{[k+1]}, a_{[k+1]})$. Now, let τ be the vstep prior to $k+1$ at which $\tilde{Q}(s_{[k+1]}, a_{[k+1]})$ was last updated.[6]

[6]If $\tilde{Q}(s_{[k+1]}, a_{[k+1]})$ was never updated, then $\tilde{Q}^{\pi}_{[k+1]}(s_{[k+1]}, a_{[k+1]}) = \tilde{Q}^{\pi}_{[k]}(s_{[k+1]}, a_{[k+1]})$.

Since $\tau \le k$, and $\tilde{Q}^{\pi}(s_{[k+1]}, a_{[k+1]})$ is unchanged between vsteps τ and $k+1$, we need only prove that:

$$\tilde{Q}^{\pi}_{[k+1]}(s_{[k+1]}, a_{[k+1]}) \le \tilde{Q}^{\pi}_{[\tau]}(s_{[k+1]}, a_{[k+1]}).$$

Let c_n be the number of times $n(s_{[k+1]}, a_{[k+1]})$ is updated between vsteps τ and $k+1$.[7] For the c_n updates, let c_R be the total (undiscounted) reward accumulated and $c_T(s')$ be the number of times that s' is the next state observed. Again omitting $(s_{[k+1]}, a_{[k+1]})$, we can write:

$$n_{[k+1]} = n_{[\tau]} + c_n \tag{5}$$

$$\hat{R}_{[k+1]} = \frac{n_{[\tau]}\hat{R}_{[\tau]} + c_R}{n_{[\tau]} + c_n} \tag{6}$$

$$\hat{T}_{[k+1]}(s') = \frac{n_{[\tau]}\hat{T}_{[\tau]}(s') + c_T(s')}{n_{[\tau]} + c_n} \text{ for all } s' \in S. \tag{7}$$

Letting $\tilde{V}^{\pi}_{[i]}(s')$ denote $\tilde{Q}^{\pi}_{[i]}(s', \pi(s'))$, we use Equations 5–7 to reduce Equation 4:

$$\begin{aligned}
\tilde{Q}^{\pi}_{[k+1]} &= \left(1 - \frac{n_{[k+1]}}{m}\right)V_{\max} + \frac{n_{[k+1]}}{m}\left(\hat{R}_{[k+1]}\right.\\
&\qquad \left. + \gamma \sum_{s' \in S} \hat{T}_{[k+1]}(s')\tilde{V}^{\pi}_{[k]}(s')\right)\\
&= \left(1 - \frac{n_{[\tau]} + c_n}{m}\right)V_{\max} + \frac{n_{[\tau]} + c_n}{m}\left(\frac{n_{[\tau]}\hat{R}_{[\tau]} + c_R}{n_{[\tau]} + c_n}\right.\\
&\qquad \left. + \gamma \sum_{s' \in S} \frac{n_{[\tau]}\hat{T}_{[\tau]}(s') + c_T(s')}{n_{[\tau]} + c_n}\tilde{V}^{\pi}_{[k]}(s')\right)\\
&= \left(1 - \frac{n_{[\tau]}}{m} - \frac{c_n}{m}\right)V_{\max} + \frac{1}{m}\left(n_{[\tau]}\hat{R}_{[\tau]} + c_R\right.\\
&\qquad \left. + \gamma \sum_{s' \in S}\left(n_{[\tau]}\hat{T}_{[\tau]}(s') + c_T(s')\right)\tilde{V}^{\pi}_{[k]}(s')\right)\\
&= \frac{n_{[\tau]}}{m}\left(\hat{R}_{[\tau]} + \gamma \sum_{s' \in S}\hat{T}_{[\tau]}(s')\tilde{V}^{\pi}_{[k]}(s')\right) + \left(1 - \frac{n_{[\tau]}}{m}\right)V_{\max}\\
&\qquad + \frac{1}{m}\left(c_R - c_n V_{\max} + \gamma \sum_{s' \in S} c_T(s')\tilde{V}^{\pi}_{[k]}(s')\right).
\end{aligned}$$

We know that $c_R \le c_n R_{\max}$ and $\sum_{s' \in S} c_T(s') = c_n$. Also, since $\tau - 1 < k$, the inductive hypothesis implies that for all $s' \in S$, $\tilde{V}^{\pi}_{[k]}(s') \le \tilde{V}^{\pi}_{[\tau-1]}(s')$. Lastly, Lemma 1 implies that $\tilde{V}^{\pi}_{[k]}(s') \le V_{\max}$ for all $s' \in S$. We use these facts to further reduce $\tilde{Q}^{\pi}_{[k+1]}$:

$$\begin{aligned}
&\le \frac{n_{[\tau]}}{m}\left(\hat{R}_{[\tau]} + \gamma \sum_{s' \in S}\hat{T}_{[\tau]}(s')\tilde{V}^{\pi}_{[\tau-1]}(s')\right) + \left(1 - \frac{n_{[\tau]}}{m}\right)V_{\max}\\
&\qquad + \frac{1}{m}\left(c_n R_{\max} - c_n V_{\max} + \gamma \sum_{s' \in S} c_T(s')V_{\max}\right)\\
&\le \tilde{Q}^{\pi}_{[\tau]} + \frac{1}{m}\left(c_n R_{\max} - c_n V_{\max} + \gamma \sum_{s' \in S} c_T(s')V_{\max}\right)\\
&\le \tilde{Q}^{\pi}_{[\tau]} \quad \square
\end{aligned}$$

[7]While $c_n \in \{0, 1\}$ for V-MAX, it could be larger if full VI was not performed at each timestep.

THEOREM 1. *Suppose that $0 < \epsilon < V_{\max}$ and $0 < \delta < 1$ are two real numbers and $M = (S, A, R, T, \gamma)$ is any MDP. There exist $m = O\left(\frac{(|S| + \ln(|S||A|/\delta))R^2_{\max}}{\epsilon^2(1-\gamma)^4}\right)$ and $\epsilon_1 = O(\epsilon)$ such that if V-MAX is executed on M with inputs m and ϵ_1, then the following holds. Let $\mathcal{A}_t$ denote V-MAX's policy at time t, and let s_t denote the state at time t. With probability at least $1 - \delta$, $V^{\mathcal{A}_t}_M(s_t) \ge V^*_M(s_t) - \epsilon$ is true for all but*

$$O\left(\frac{|S||A|R^3_{\max}}{\epsilon^3(1-\gamma)^6}\left(|S| + \ln\frac{|S||A|}{\delta}\right)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon(1-\gamma)}\right)$$

timesteps t.

PROOF. Let $m = O\left(\frac{(|S| + \ln(|S||A|/\delta))R^2_{\max}}{\epsilon_1^2(1-\gamma)^4}\right)$ and let K_t be the set of state-action pairs experienced at least m times by timestep t. Let $H = \lceil \frac{1}{1-\gamma} \ln \frac{R_{\max}}{\epsilon_1(1-\gamma)} \rceil$ and E_M be the event that a state-action pair not in K_t is encountered in a trial generated by starting from s_t and following $\mathcal{A}_t$ for H timesteps in M. We consider two mutually exclusive cases.

In the first case, $\Pr(E_M) \ge \epsilon_1/V_{\max}$. We treat H timesteps in M as one toss of a weighted coin, and E_M as the event that it shows heads. If we see $m|S||A|$ heads, then all $(s, a) \in S \times A$ are in K. As a result of the Chernoff-Hoeffding bound, with probability $1-\delta$, after j tosses, $m|S||A|$ or more heads are seen, for some $j = O(\frac{m|S||A|V_{\max}}{\epsilon_1} \ln \frac{1}{\delta})$. Since each toss is equivalent to H timesteps in M, with probability $1-\delta$, all state-action pairs are in K after $O(\frac{m|S||A|HR_{\max}}{\epsilon_1(1-\gamma)} \ln \frac{1}{\delta})$ timesteps. Also, $\Pr(E_M) = 0$ once all state-action pairs are in K. Thus, given m and H, $\Pr(E_M) < \epsilon_1/V_{\max}$ after

$$O\left(\frac{|S||A|R^3_{\max}}{\epsilon_1^3(1-\gamma)^6}\left(|S| + \ln\frac{|S||A|}{\delta}\right)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon_1(1-\gamma)}\right)$$

timesteps.

In the second case, $\Pr(E_M) < \epsilon_1/V_{\max}$. Recall that for some policy $\mathcal{A}_t$ on the MDP M, $V^{\mathcal{A}_t}_M$ denotes the true value function. Let $V(s, T)$ denote the T-step value of s. Given H and Lemma 2 of [9]:

$$V^{\mathcal{A}_t}_M(s_t) \ge V^{\mathcal{A}_t}_M(s_t, H).$$

Let z be a fictitious state, and let $\bar{S} = S \cup \{z\}$. Let $\bar{M} = (\bar{S}, A, \bar{R}, \bar{T}, \gamma)$ be an MDP, where for all $(s, a, s') \in S \times A \times S$, $\bar{R}(s, a) = R(s, a)$, $\bar{T}(s, a, s') = T(s, a, s')$, and $\bar{T}(s, a, z) = 0$. $\bar{R}(z, .)$ and $\bar{T}(z, ., .)$ are set arbitrarily. Since $s_t \ne z$ and the reward and transition functions in M and $\bar{M}$ are equal on $S \times A \times S$, $V^{\mathcal{A}_t}_M(s_t, H) = V^{\mathcal{A}_t}_{\bar{M}}(s_t, H)$. Thus:

$$V^{\mathcal{A}_t}_M(s_t) \ge V^{\mathcal{A}_t}_{\bar{M}}(s_t, H).$$

Let $M_K = (\bar{S}, A, R_K, T_K, \gamma)$ be an MDP with $R_K = \bar{R}$ and $T_K(.) = \bar{T}(.)$ for state-action pairs in K_t, while $R_K = \tilde{R}_t$ and $T_K(.) = \tilde{T}_t(.)$ for state-action pairs not in K_t, where $\tilde{R}_t$ and $\tilde{T}_t$ are defined with Equations 2 and 3. Let $E_{\bar{M}}$ be the event that a state-action pair not in K_t is encountered in a trial generated by starting from s_t and following $\mathcal{A}_t$ for H timesteps in $\bar{M}$. Lemma 8 of [15] implies that $V^{\mathcal{A}_t}_{\bar{M}}(s_t, H) \ge V^{\mathcal{A}_t}_{M_K}(s_t, H) - V_{\max}\Pr(E_{\bar{M}})$. However, since M and $\bar{M}$ differ only on z, which is never encountered, $\Pr(E_{\bar{M}}) = \Pr(E_M) < \epsilon_1/V_{\max}$. It follows that:

$$V^{\mathcal{A}_t}_M(s_t) \ge V^{\mathcal{A}_t}_{M_K}(s_t, H) - \epsilon_1.$$

From Lemma 2 of [9], $V^{\mathcal{A}_t}_{M_K}(s_t, H) \ge V^{\mathcal{A}_t}_{M_K}(s_t) - \epsilon_1$. Thus:

$$V^{\mathcal{A}_t}_M(s_t) \ge V^{\mathcal{A}_t}_{M_K}(s_t) - 2\epsilon_1.$$

Let $\hat{M}_K = (\bar{S}, A, \hat{R}_K, \hat{T}_K, \gamma)$ be an MDP with $\hat{R}_K = \hat{R}_t$ and $\hat{T}_K(.) = \hat{T}_t(.)$ for state-action pairs in K_t, while $\hat{R}_K = \tilde{R}_t$ and $\hat{T}_K(.) = \tilde{T}_t(.)$ for state-action pairs not in K_t. Note that $\hat{M}_K$ and M_K have equal reward and transition functions for all state-action pairs not in K_t. Thus, given m, Lemma 15 of [13] implies[8] that with probability at least $1-\delta$, $V^{\mathcal{A}_t}_{M_K}(s) \geq V^{\mathcal{A}_t}_{\hat{M}_K}(s) - \epsilon_1$ for all $s \in S$. Consequently, with probability at least $1-\delta$:

$$V^{\mathcal{A}_t}_{M}(s_t) \geq V^{\mathcal{A}_t}_{\hat{M}_K}(s_t) - 3\epsilon_1.$$

Now, let $\hat{M}_t = (S, A, \hat{R}_t, \hat{T}_t, \gamma)$ be the MDP that is learned by V-MAX using maximum likelihood estimation. We know that $\hat{R}_t = \tilde{R}_K$ and $\hat{T}_t(.) = \tilde{T}_K(.)$ for state-action pairs not in K_t. For these pairs, the derivation in Section 3.1 implies that computing $\tilde{Q}$-values using Equation 4 for VI in $\hat{M}_t$ is equivalent to computing Q-values using the Bellman optimality equation for VI in $\hat{M}_K$. Furthermore, for state-action pairs in K_t, Equation 4 reduces to the Bellman optimality equation. Thus, for all $s \in S$, $V^{\mathcal{A}_t}_{\hat{M}_K}(s) = \tilde{V}^{\mathcal{A}_t}_{\hat{M}_t}(s)$. Hence, with probability at least $1-\delta$:

$$V^{\mathcal{A}_t}_{M}(s_t) \geq \tilde{V}^{\mathcal{A}_t}_{\hat{M}_t}(s_t) - 3\epsilon_1.$$

Since $\mathcal{A}_t$ is greedy with respect to $\tilde{Q}_{\hat{M}_t}$, for all $s \in S$, and all policies π, $\tilde{V}^{\mathcal{A}_t}_{\hat{M}_t}(s) \geq \tilde{V}^{\pi}_{\hat{M}_t}(s)$. Let π^* denote the optimal policy for M. Thus, with probability at least $1-\delta$:

$$V^{\mathcal{A}_t}_{M}(s_t) \geq \tilde{V}^{\pi^*}_{\hat{M}_t}(s_t) - 3\epsilon_1.$$

Let $\hat{M}_x = (S, A, \hat{R}_x, \hat{T}_x, \gamma)$ be an MDP where $\hat{R}_x$ and $\hat{T}_x$ are maximum likelihood estimates of R and T at some timestep $x \geq t$, such that all $(s,a) \in S \times A$ are in K_x. Then, Lemma 2 implies that for all $s \in S$ and all policies π, $\tilde{V}^{\pi}_{\hat{M}_t}(s) \geq \tilde{V}^{\pi}_{\hat{M}_x}(s)$. Equation 4 reduces to the Bellman optimality equation for all $(s,a) \in K_x$, implying that for all $s \in S$, and for all policies π, $\tilde{V}^{\pi}_{\hat{M}_x}(s) = V^{\pi}_{\hat{M}_x}(s)$. Consequently, with probability at least $1-\delta$:

$$V^{\mathcal{A}_t}_{M}(s_t) \geq V^{*}_{\hat{M}_x}(s_t) - 3\epsilon_1.$$

Given our choice of m, an application of Lemma 15 of [13] yields that with probability at least $1-\delta$, $V^*_{\hat{M}_x}(s) \geq V^*_M(s) - \epsilon_1$, for all $s \in S$. Thus, with probability at least $1-\delta$:

$$V^{\mathcal{A}_t}_{M}(s_t) \geq V^{*}_{M}(s_t) - 4\epsilon_1.$$

Setting $\epsilon_1 = \epsilon/4$ yields the desired result. $\square$

4. EXPERIMENTS

We evaluate the empirical performance of V-MAX on two tasks. As comparisons, we use R-MAX and MoR-MAX, because of their close relationship to V-MAX, and MBIE, because of its strong empirical track record: no other PAC-MDP method has been shown to substantially outperform it. On the contrary, it has greatly outperformed E^3 [9], R-MAX, and Delayed Q-Learning [13] and matched the performance of Optimistic Initial Model (OIM) [18].[9]

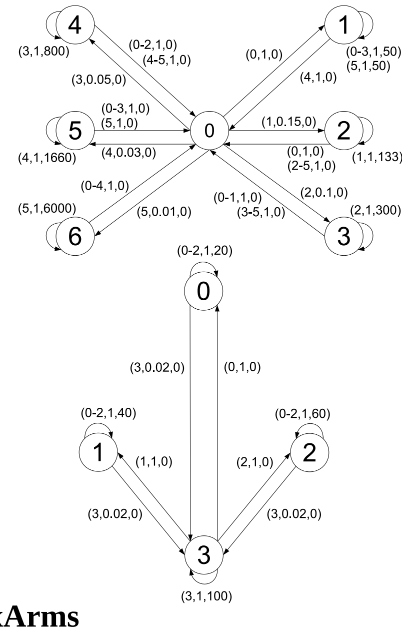
Since our goal is to show that V-MAX advances the PAC-MDP state of the art, we restrict our analysis to PAC-MDP algorithms. Comparisons to efficient-exploration algorithms developed under other frameworks, such as *regret* [5] and *knows what it knows* (KWIK) [11], are left to future work.

Following [14] and [16], we use cumulative (undiscounted) reward as our evaluation metric. Even though maximising the cumulative reward is not the goal of PAC-MDP methods, it is a suitable metric for our experiments because it shows that V-MAX can accrue more reward while matching the sample complexity of R-MAX.

For each algorithm, performance is affected by the value of its critical parameters: ϵ_1 and m for R-MAX, MoR-MAX, and V-MAX and ϵ_1 and C (where $\beta = CV_{\max}$) for MBIE. For all algorithms, we set ϵ_1 to 0.01, at which VI finds the optimal policy. For each MDP, we first measure the average cumulative reward after 25,000 timesteps over 100 runs on each MDP across a large range of values for m and C. Using the results, we select a smaller range of parameter values likely to perform well. For our final results, we measure the average cumulative reward over 1000 runs across this selected range. For more timesteps, our results are qualitatively similar. For significantly fewer timesteps, none of the algorithms learn the optimal policy, thus biasing the results.

For each algorithm, if two or more actions have the same value, the agent takes the action that has been tried fewer times. Remaining ties are broken by choosing actions in ascending order, e.g., action 0 is chosen before action 1. This action-selection method is a simple way to ensure all algorithms try lower-valued actions before discovering the optimal action in the highest-valued state, without giving any algorithm an advantage.

Figure 1: SixArms and Anchor MDPs. Each vertex denotes a state and each edge a state transition. A tuple (a, p, r) indicates that action a causes the given transition with probability p and reward r and causes a self-transition with probability $1-p$ and reward 0.



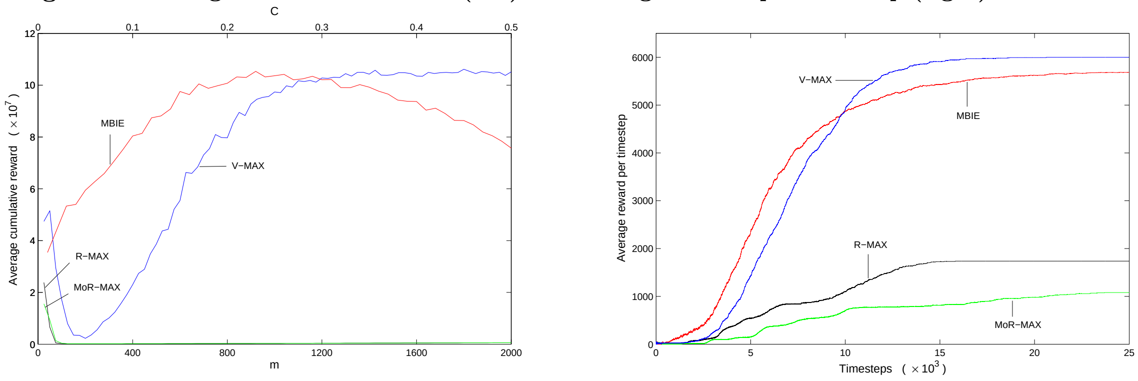
4.1 SixArms

The SixArms MDP in Figure 1 (top) was used to show that MBIE can outperform R-MAX, which in turn outperforms both E^3 and ϵ-greedy exploration [14, 16]. Thus, it is an important benchmark test for V-MAX. The agent starts in state 0 and chooses between 6 actions, each of which may

[8]While lemma 15 of [13] is stated for R-MAX, the same result holds for V-MAX, from an application of Lemmas 13 and 14 of [13] in lemma 2 of [16].

[9]OIM also has substantially weaker sample complexity bounds than R-MAX, V-MAX, and MBIE.

Figure 2: Average cumulative reward (left) and average reward per timestep (right) on SixArms.



lead to states 1-6. Once there, the agent can remain and receive a reward at each timestep or return to state 0. The optimal policy is to try to reach state 6 and then repeatedly choose action 5. Since the probability of reaching state 6 is small, the agent typically must try each action several times to discover the optimal policy. The discount factor is 0.95.

Figure 2 (left) shows the average cumulative reward over a range of parameter values for each algorithm (C for MBIE, and m for the other algorithms). V-MAX vastly outperforms R-MAX and MoR-MAX and matches the performance of MBIE. In addition, Figure 2 (right), which shows the average reward per timestep for each algorithm at optimal parameter values, demonstrates that V-MAX is the only algorithm able to consistently achieve the maximum reward per timestep. MBIE obtains similar cumulative reward by gaining higher rewards in the initial phases of the experiment.

Note that MoR-MAX, which has the strongest sample complexity bounds, has the worst empirical performance. To see why, consider state 0. A problematic scenario for R-MAX, MoR-MAX, and V-MAX is where the agent never transitions from state 0 to any of the states 2 and above. Then, it will forever assume that, in state 0, all actions other than 0 cause a self-transition. However, to avoid this scenario, R-MAX and V-MAX need only ensure with high probability that this does not happen in the *first* m samples for each state-action pair. By contrast, MoR-MAX repeatedly collects batches of m samples per state-action pair and discards them when new samples lead to a lower value for that pair. Consequently, MoR-MAX must ensure with high probability that this scenario does not occur on *any* batch of m samples. In general, this requires higher values of m, yielding too much initial exploration.

Overall, results on SixArms show that V-MAX can match the empirical performance of MBIE while maintaining the stronger theoretical guarantees of R-MAX. Furthermore, Figure 2 (left) shows that, as long as m is set conservatively, V-MAX's performance is invariant to it, essentially eliminating the need for parameter tuning. In contrast, MBIE requires careful tuning of C to match V-MAX's performance.

4.2 Anchor

In SixArms, MBIE accrues more reward early in learning by more often choosing relatively good, but suboptimal, actions in states 1-5. However, in this MDP, exploiting such local optima has little cost, since the agent can at any time deterministically return to state 0. Therefore, we hypothesize that MBIE will perform poorly in MDPs where choosing locally optimal actions significantly impedes the agent's progress towards better state-action pairs.

To test this hypothesis, we use the Anchor MDP shown in Figure 1 (bottom). The agent starts in state 0 and can choose, in states 0-2, to remain and receive a sub-optimal reward or try to reach state 3, where maximal reward is available. Since the agent initially chooses actions in ascending order, it must try each action in each states 0-2 at least once before discovering that action 3 in state 3 yields reward $R_{\max} = 100$. The discount factor is 0.99.

The optimal policy is to choose action 3 in each state. However, in each state 0-2, action 3 is the only one that produces no reward. Furthermore, all actions in these states usually produce self-transitions, with only action 3 transitioning to state 3 with low probability. Consequently, to determine the optimal policy efficiently, an agent must avoid being distracted by the sub-optimal rewards offered by actions 0-2 and continue to attempt action 3. Unlike in SixArms, the agent cannot deterministically escape the local optima (in states 0-2), making efficient exploration crucial.

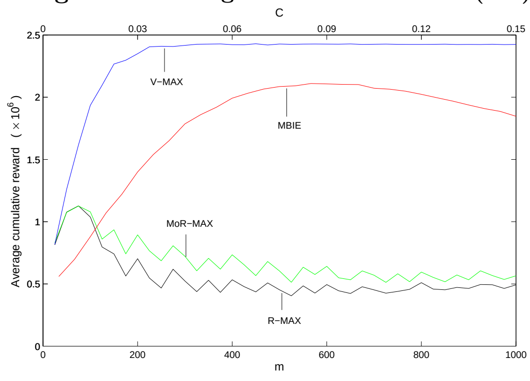
Figure 3 (left) shows the average cumulative reward for each algorithm on Anchor. As in SixArms, V-MAX substantially outperforms both R-MAX and MoR-MAX. However, it also substantially outperforms MBIE, supporting the hypothesis that MBIE is vulnerable to distraction by local optima. In addition, as in SixArms, the performance of V-MAX is invariant for large choices of m, whereas MBIE again requires careful tuning of C.

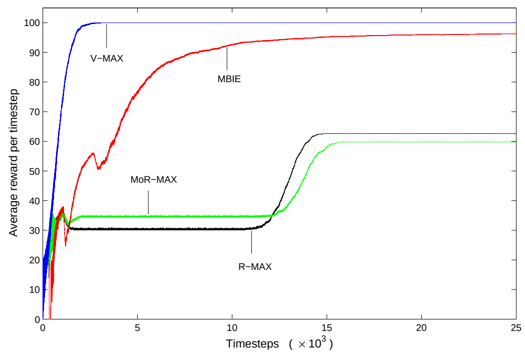
In Figure 3 (left), the performance of both R-MAX and MoR-MAX oscillates as m changes. We suspect this is a result of R-MAX and MoR-MAX periodically cycling between the states 0 to 2 during exploration. Since m affects how many timesteps they spend in each of these states, performance can increase for values of m in which time expires just before the agent cycles to the low reward states 0 and/or 1. However, since increasing m increases exploration, the performance of R-MAX and MoR-MAX decreases overall, despite the oscillations, as m is increased beyond 100.

Figure 3 (right) shows the average reward per timestep in Anchor. Only V-MAX consistently accrues maximal reward per timestep. Although MBIE gets close, it takes significantly longer than V-MAX to do so. In addition, both R-MAX and MoR-MAX remain in states 1 and 2 for a long time, constantly receiving the sub-optimal rewards of 40 and 60, respectively. Though the effect is less extreme for MBIE, its performance also dips in the same places.

Overall, the SixArms and Anchor experiments demonstrate that 1) V-MAX can greatly outperform both R-MAX and MoR-MAX, 2) V-MAX can match MBIE and, on problems with multiple local optima, substantially outperform it, and 3) V-MAX, whose performance is invariant for large

Figure 3: Average cumulative reward (left) and average reward per timestep (right) on Anchor.





m, essentially obviates the need for parameter tuning while MBIE's performance is sensitive to the choice of C.

5. FUTURE WORK AND CONCLUSIONS

The development of V-MAX creates several opportunities for future work. We hope to extend our theoretical results by proving that, for the same m, V-MAX strictly dominates R-MAX on all MDPs in that it never makes an exploratory mistake that R-MAX does not make, given the same $\hat{R}$ and $\hat{T}$. In addition to sample complexity, per-timestep computational complexity may also be important in large MDPs. In such cases, techniques for reducing the computational cost of PAC-MDP algorithms [4, 12] may benefit V-MAX as well. Also, extensions to R-MAX that exploit problem structure, such as Fitted R-MAX [6], R-MAXQ [7] and RAM-Rmax [10], could be adapted to use V-MAX instead. Lastly, since we suspect that the core idea of tempered optimism is actually orthogonal to the quality of the bounds, it could be used to derive other PAC-MDP methods, e.g., MoV-MAX, with strong empirical performance and even tighter bounds.

Overall, the theoretical and empirical results presented in this paper demonstrate that V-MAX is a significant step forward for PAC RL, since it can outperform state-of-the-art PAC-MDP algorithms while maintaining attractive theoretical guarantees. In addition, these guarantees show for the first time that tempered optimism is possible without compromising sample complexity bounds. Finally, our experiments suggest that V-MAX requires little or no parameter tuning. Consequently, we hope that it will help make PAC-MDP algorithms more accessible for general usage.

6. ACKNOWLEDGMENTS

Thanks Lihong Li, Alexander Strehl, Michael Littman, and Harm van Seijen for their helpful comments.

7. REFERENCES

[1] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.
[2] R. I. Brafman and M. Tennenholtz. R-MAX – a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.
[3] M. O. Duff. *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes.* PhD thesis, U. of Mass.-Amherst, 2002.
[4] M. Grzés and J. Hoey. Efficient planning in R-MAX. In *Proc. of AAMAS*, 2011.
[5] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.
[6] N. K. Jong and P. Stone. Model-based exploration in continuous state spaces. In *The Symposium on Abstraction, Reformulation, and Approximation*, 2007.
[7] N. K. Jong and P. Stone. Hierarchical model-based reinforcement learning: Rmax + MAXQ. In *Proc. of ICML*, 2008.
[8] S. M. Kakade. *On the Sample Complexity of Reinforcement Learning.* PhD thesis, University College London, 2003.
[9] M. J. Kearns and S. P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.
[10] B. R. Leffler, M. L. Littman, and T. Edmunds. Efficient reinforcement learning with relocatable action models. In *Proc. of AAAI*, pages 572–577, 2007.
[11] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Know what it knows: A framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.
[12] A. L. Strehl, L. Li, and M. L. Littman. Incremental model-based learners with formal learning-time guarantees. In *Proc. of UAI*, pages 485–493, 2006.
[13] A. L. Strehl, L. Li, and M. L. Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.
[14] A. L. Strehl and M. L. Littman. An empirical evaluation of interval estimation for Markov decision processes. In *Proc. of ICTAI*, pages 129–135, 2004.
[15] A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *Proc. of ICML*, pages 857–864, 2005.
[16] A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.
[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, MA, 1998.
[18] I. Szita and A. Lörincz. The many faces of optimism: A unifying approach. In *Proc. of ICML*, 2008.
[19] I. Szita and C. Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proc. of ICML*, pages 1031–1038, 2010.
[20] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
[21] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.