
Exploration via Model-based Interval Estimation

Alexander L. Strehl

Department of Computer Science, Rutgers University, Piscataway, NJ USA

STREHL@CS.RUTGERS.EDU

Michael L. Littman

Department of Computer Science, Rutgers University, Piscataway, NJ USA

MLITTMAN@CS.RUTGERS.EDU

Abstract

This paper takes an empirical approach to evaluating three model-based reinforcement-learning methods. All methods intend to speed the learning process by mixing exploitation of learned knowledge with exploration of possibly promising alternatives. We consider ϵ -greedy exploration, which is computationally cheap and popular, but unfocused in its exploration effort; R-Max exploration, a simplification of an exploration scheme that comes with a theoretical guarantee of efficiency; and a well-grounded approach, model-based interval estimation, that better integrates exploration and exploitation and achieves the best performance in our example tasks. Our experiments indicate that effective exploration can result in dramatic improvements in the observed rate of learning.

1. Introduction

One of the distinguishing features of reinforcement learning is the control the learner has over its own inputs. During learning, the decision maker must often choose between decisions that appear to be best at achieving high reward and those that may lead it to parts of the state space where even higher reward can be achieved. Choosing wisely can dramatically decrease the time needed to learn to behave effectively in a new environment.

In this work, we evaluate agents following three exploration schemes, ϵ -greedy, R-Max, and model-based interval exploration (MBIE), and show how each can

trade off exploration to identify promising parts of the state space with exploitation to increase the rate the agent gathers reward. Each algorithm includes a parameter that controls the exploration–exploitation tradeoff and we measure cumulative reward on a set of simple Markov decision processes for a variety of parameter settings. All show a distinctive “inverted u” behavior—for low parameter values, algorithms quickly converge to suboptimal behavior, while for high parameter values, algorithms converge much more slowly, but to more efficient policies. In between, good policies are found relatively quickly, thus maximizing cumulative reward over the period of experimentation.

The idea of model-based interval estimation (MBIE) was first introduced by Wiering (1999). It is inspired by interval estimation (Kaelbling, 1993), shown to produce probably approximately optimal policies in bandit problems (Fong, 1995). In our formulation, it builds a confidence interval on the parameters of a Markov decision process model of the environment, then finds the combination of parameters and agent behavior that leads to an upperbound on the maximum estimated reward, subject to the constraint that parameters lie in the confidence interval. In contrast to the original MBIE approach, our algorithm is simpler, comes with a formal convergence guarantee, and uses a more accurate calculation for the confidence intervals.

Section 2 introduces the exploration–exploitation tradeoff, discusses bandit problems and Markov decision processes, and presents several basic exploration algorithms. Our MBIE approach is presented in Section 4 and experimental comparisons in Section 5.

2. Approaches to Exploration

This section presents several approaches to exploration that have been the target of research in bandit prob-

lems and more complex Markov decision processes.

2.1. Bandit Problems

Bandit problems (Berry & Fristedt, 1985) provide a minimal mathematical model in which to study the exploration–exploitation tradeoff. Here, we imagine we are faced with a slot-machine with a set of k arms, only one of which can be selected on each timestep. Each arm i has a stationary payoff distribution with finite mean μ_i and variance.

The optimal long-term strategy for a k -armed bandit is to always select the arm with the highest mean payoff, $\operatorname{argmax}_i \mu_i$. However, these expected payoffs are not known to the agent, who must expend a number of trials in exploration. By the law of large numbers, increasingly accurate estimates of the mean payoff of each arm can be made by repeated sampling. However, an agent that continues to sample all arms indefinitely will be wasting opportunities to exploit what it has learned to maximize its payoffs.

While a number of positive results are available for optimizing discounted reward (Gittins, 1989), the pursuit of simpler and more general algorithms have led researchers to other objective criteria. One that we would like to highlight is the design of algorithms that are *probably approximately optimal* (PAO). After a relatively small amount of experience (typically polynomial in problem parameters such as k , $1/\epsilon$, $1/\delta$, and the maximum possible reward R_{\max}), a PAO algorithm very likely produces a near-optimal strategy (no more than ϵ less than optimal with probability $1 - \delta$).

Consider the following simple strategies:

- ϵ -greedy: With probability ϵ , choose an arm at random (explore), otherwise greedily pick the arm with the highest estimated payoff (exploit). To insure that all arms receive at least one pull, arms that have not yet been tried are given an estimate of R_{\max} . A closely related approach, *Boltzmann exploration*, weights the probabilities according to how close actions are to being the best.
- “naïve”: Choose each arm a fixed number of times, c , then choose the action with the highest estimated payoff thereafter. Equivalently, estimate the payoff for any arm that has not yet been chosen c times as R_{\max} and always choose greedily with respect to this estimate.
- IE (interval estimation): Construct a $p\%$ confidence interval for the mean of the payoff distribution for each arm. At each step, choose the action

with the highest upper confidence interval. Equivalently, estimate that each arm has a payoff equal to the estimated mean plus its $p\%$ confidence interval and always choose greedily with respect to this estimate.

Fong (1995) analyzed “naïve” and IE and showed that both are PAO for appropriate choices of the constants c and p . Building on these results, we can show that a choice of ϵ exists so that ϵ -greedy is PAO as well.

Proposition 1 *The ϵ -greedy exploration scheme is PAO in k -armed bandit problems.*

Proof sketch: The idea behind the proof is that there is a value of ϵ , proportional to ϵ , such that the ϵ -greedy policy is sufficiently close to optimal. After a polynomial number of trials, the random exploration is likely to try each arm a sufficient number of times to get an accurate approximation of mean payoffs. \square

Although all three approaches are PAO, in empirical studies, IE is often able to rule out actions more quickly, resulting in faster learning times. In the next section, we transfer these results to a more complex class of learning environments, Markov decision processes.

3. Exploration in MDPs

A Markov Decision Process (MDP) is a four tuple $\langle S, A, T, R \rangle$, where S is the state space, A is the action space, $T : S \times A \times S \rightarrow \mathbb{R}$ is a transition function, and $R : S \times A \rightarrow \mathbb{R}$ is a reward function. The function $T(s, a, s')$ is interpreted as the probability of ending in state s' by taking action a from state s , and $R(s, a)$ is the expected reward for taking action a from state s . Throughout this paper, we assume that the state and action spaces are finite. Furthermore, we assume that a discount factor $0 \leq \gamma < 1$ and a vector of start-state probabilities are associated with each MDP. The discount factor defines the performance metric.

If the dynamics of the MDP are known, a policy can be found mapping states to actions that maximizes the expected discounted reward by solving the Bellman equations,

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q(s', a'), \quad (1)$$

and selecting action $\operatorname{argmax}_a Q(s_t, a)$ at time t . The equations can be solved by *value iteration*, which iteratively applies the equations as an update to an arbitrary initial guess of the values (Puterman, 1994; Sutton & Barto, 1998).

Note that MDPs generalize k -armed bandit problems. Any k -armed bandit problem can be thought of as an MDP with one state and k actions. The rewards for performing each action are the expected value of the rewards for pulling each arm. MDPs can be more complex than k -armed bandit problems because an action not only yields a reward, but also determines the next state (Duff & Barto, 1997).

The model-based reinforcement-learning algorithms we studied include a number of common elements. Let $n(s, a, s')$ be the number of times that the agent has ended in state s' after taking action a from state s during a run, and define $n(s, a) = \sum_{s'} n(s, a, s')$. The maximum likelihood model for the transition function T is $\hat{T}(s, a, s') = n(s, a, s')/n(s, a)$. Similarly, we can estimate $\hat{R}(s, a) = r(s, a)/n(s, a)$, where $r(s, a)$ is the total reward received by the agent from taking action a in state s . In our study, we use “optimistic initialization” in that $\hat{R}(s, a) = R_{\max}$ if $n(s, a) = 0$. We also use $\hat{T}(s, a, s) = 1$ if $n(s, a) = 0$. A greedy policy with respect to the learned model can be computed by solving Equation 1 with $T = \hat{T}$ and $R = \hat{R}$.

3.1. Epsilon-Greedy

The ϵ -greedy approach can be used in MDPs by computing the greedy policy with respect to the maximum likelihood model. When it is time to act, with probability $1 - \epsilon$, the optimal action with respect to the estimated transition model is taken, and with probability ϵ , a random action is chosen.

Although ϵ -greedy exploration ensures convergence to an optimal policy in the limit, it displays a number of weaknesses. For one thing, the agent has no control over when or how to explore. A more subtle weakness can be seen in this example. Suppose we have an MDP in which Action 1 in State 1 yields a large reward and returns the agent back to State 1, while Action 2 yields no reward but frequently sends the agent to State 2. From State 1, a reasonable course of action is to take Action 2, giving up the possible reward for Action 1, in hopes that the agent will visit State 2 to commence exploration of that unknown state. This is an example of directed exploration, a behavior not present in the ϵ -greedy algorithm (apart from the brief effect of optimistic initialization). Because of these weaknesses, ϵ -greedy is not PAO—it can require exponentially many steps to find near-optimal behavior (Koenig & Simons, 1993).

3.2. R-Max

Kearns and Singh (2002) showed that a PAO condition for MDPs can be defined and they provided an algo-

rithm that provably achieves this condition. Brafman and Tennenholtz (2002) made use of a similar analysis to show that a simpler algorithm, R-Max, is PAO.

R-Max uses a different estimate of the transition model than does the ϵ -greedy approach. Following the “naïve” algorithm described earlier, R-Max can be parameterized by a count c , with an optimistic estimate used whenever fewer than c trials of experience are available. Specifically, R-Max modifies the Bellman equations to use Equation 1 only in states for which $n(s, a) \geq c$ and for the other states to use $Q(s, a) = R_{\max}/(1 - \gamma)$.

In the R-Max work, the value of the parameter c is set as a function of a number of other parameters, including the number of states, R_{\max} , and the desired certainty of near optimality. In this paper, however, we take c to be a free variable, typically much lower than its formal bound.

Using R-Max, an agent will always choose the action that is optimal given its computed $Q(s, a)$ values. The optimistic values for state–action pairs visited fewer than c times receive a kind of “exploration bonus” and, therefore, when it is time to act, the agent will often be encouraged to visit that state–action pair. Compared to ϵ -greedy exploration, R-Max will explore an unknown action multiple times before using its empirical distribution, and uses no randomness in its action choices.

3.3. Model-based Interval Estimation

R-Max can be viewed as a generalization of the “naïve” algorithm from bandit problems to MDPs. IE was generalized to MDPs by Kaelbling (1993) in the context of Q-learning. During learning, the value of each state–action pair is represented as a distribution, because of noise in the learning process. As in the bandit case, IE uses the upper confidence interval on the mean of this distribution as its value when making updates.

The core type of uncertainty encountered when solving an MDP by reinforcement learning, however, is uncertainty on the model parameters, not the values. Model-based interval-estimation (MBIE) uses formal confidence intervals on the *probability distributions* for each state–action pair. As in the ϵ -greedy and R-Max approaches, the algorithm keeps maximum likelihood estimates \hat{T} and \hat{R} of the transition and reward functions. Like R-Max, it provides an exploration bonus for insufficiently explored state–action pairs. The form of the exploration bonus is different, however, as it is based on the likelihood with which better outcomes than those already experienced are possible. The form

of the algorithm, its motivation, and an efficient implementation are explained next.

4. Computing Policies in MBIE

Interval estimation for bandit problems estimates the mean for each arm. Based on a finite sample, it cannot be sure of the true mean for any arm, so it constructs a confidence interval on the mean. Of all possible means consistent with these confidence intervals, IE assumes that the most optimistic one is true. It then chooses the arm that maximizes its expected payoff under this assumption—it picks the arm with the highest upper confidence interval.

This scheme for optimization under uncertainty has several beneficial properties. First of all, it is easy to compute. Second, by choosing the arm with the highest upper confidence interval, one of two possibilities occurs. Either the estimate is accurate and the decision maker receives a near-optimal reward (in expectation), or the estimate is inaccurate and the decision maker gets new experience that can be used to create a more accurate model.

Translated to the transition probabilities in the learned MDP model, the interval estimation idea suggests that, for each state–action pair, we find the probability distribution $\tilde{T}(s, a, \cdot)$ within a confidence interval of the empirically derived distribution $\hat{T}(s, a, \cdot)$ that leads to the policy with the largest reward. This can be defined with the equations

$$Q(s, a) = \hat{R}(s, a) + \max_{\tilde{T}(s, a, \cdot) \in CI} \gamma \sum_{s'} \tilde{T}(s, a, s') \max_{a'} Q(s', a'), \quad (2)$$

where CI represents the set of probability distributions that are within a $(1 - \delta)$ confidence interval of the observed distribution $\hat{T}(s, a, \cdot)$ given a sample size of $n(s, a)$. The next section provides a method for describing such a set. This is followed by a method for solving Equation 2 efficiently.

4.1. Confidence Intervals

Building on standard results, Weissman et al. (2003) showed that

$$\begin{aligned} \Pr(\|\hat{T}(s, a, \cdot) - \tilde{T}(s, a, \cdot)\|_1 \geq \varepsilon) \\ \leq (n(s, a) + 1)^{k(s, a) - 1} e^{-n(s, a)\varepsilon^2/2}, \end{aligned}$$

where $k(s, a) = |\{s_{\max}\} \cup \{s' | \hat{T}(s, a, s') > 0\}|$, that is, the number of states observed to follow s under action a , along with a hypothetical state s_{\max} , and $\|x(\cdot)\|_1 = \sum_i |x(i)|$ is the L_1 norm. Manipulating this

expression, we get a definition of the confidence interval set $CI = \{\tilde{T}(s, a, \cdot) | \|\tilde{T}(s, a, \cdot) - \hat{T}(s, a, \cdot)\|_1 \leq$

$$\left. \sqrt{\frac{2((k(s, a) - 1) \ln(n(s, a) + 1) - \ln(\delta))}{n(s, a)}} \right\}.$$

In our experiments, however, we used a simplified form of this expression: $CI = \{\tilde{T}(s, a, \cdot) |$

$$\|\tilde{T}(s, a, \cdot) - \hat{T}(s, a, \cdot)\|_1 \leq d\sqrt{k(s, a)/n(s, a)}, \quad (3)$$

with the free parameter d , again controlling the trade-off between exploration (small d) and exploitation (large d).

4.2. Value Iteration with Confidence Intervals

We now argue that Equation 2 can be solved using value iteration, that is, by starting with an arbitrary estimate of $Q(s, a)$ and then repeatedly using Equation 2 as an assignment statement. This follows from the fact that Equation 2 leads to a contraction mapping.

Proposition 2 *Let $Q(s, a)$ and $W(s, a)$ be value functions and $Q'(s, a)$ and $W'(s, a)$ be the result of applying Equation 2 to Q and W , respectively. The update results in a contraction mapping in max-norm, that is, $\max_{s, a} |Q'(s, a) - W'(s, a)| \leq \gamma \max_{s, a} |Q(s, a) - W(s, a)|$.*

Proof sketch: The argument is nearly identical to the standard convergence proof of value iteration (Puterman, 1994), noting that the maximization over the compact set of probability distributions does not interfere with the contraction property. \square

Note that convergence is linear in the discount factor, so a good approximation can be found quickly, or an exact solution in polynomial time assuming a fixed discount factor (Givan et al., 2000).

We seek a probability distribution $\tilde{T}(s, a, \cdot)$ that leads to a high confidence upper bound on the value $Q(s, a)$. To ensure that our estimate is sufficiently large, we assert the existence of a state, s_{\max} , yielding maximum reward and a self loop. Although a transition to s_{\max} is never observed, it is not until the algorithm has seen a sufficient number of transitions from (s, a) that such a transition can be ruled out.

4.3. Solving the CI Constraint

To implement value iteration based on Equation 2, we need to solve the following computational problem. Maximize

$$M_V = \sum_{s'} \tilde{T}(s') V(s'),$$

where $\tilde{T}(\cdot)$ is a probability distribution subject to the constraint $\|\tilde{T}(\cdot) - \hat{T}(s, a, \cdot)\|_1 \leq \varepsilon$, and $\varepsilon = d\sqrt{k(s, a)/n(s, a)}$ as in Equation 3.

The following procedure can be used to find the maximum value. First, note that if $\max_{s'} \hat{T}(s, a, s') \leq \varepsilon/2$, then $M_V = \max_{s'} V(s')$. This is because the L_1 distance from $\hat{T}(s, a, \cdot)$ to one that puts all its weight on the maximum value of $V(\cdot)$ is less than ε . Otherwise, let $\hat{T}(s') = \hat{T}(s, a, \cdot)$ to start. Let $s^* = \operatorname{argmax}_s V(s)$ and increment $\hat{T}(s^*)$ by $\varepsilon/2$. At this point, $\hat{T}(\cdot)$ is not a probability distribution, since it sums to $1 + \varepsilon/2$; however, its L_1 distance to $\hat{T}(s, a, \cdot)$ is $\varepsilon/2$. We need to remove exactly $\varepsilon/2$ weight from $\hat{T}(\cdot)$. The weight is removed iteratively, taking the maximum amount possible from the state $s^- = \operatorname{argmin}_{s|\hat{T}(s) > 0} V(s)$, since this decreases M_V the least.

Proposition 3 *The procedure just described maximizes M_V .*

Proof sketch: First, the weight on s^* must be $\hat{T}(s, a, s^*) + \varepsilon/2$. If it's larger, $\tilde{T}(\cdot)$ violates either the sum to 1 constraint or the L_1 constraint. Let $\tilde{T}(s') - \hat{T}(s, a, s')$ be the *residual* of state s' . If $\|\tilde{T}(\cdot) - \hat{T}(s, a, \cdot)\|_1 = \varepsilon$, then the sum of the positive residuals is $\varepsilon/2$ and the sum of the negative residuals is $-\varepsilon/2$. For two states s_1 and s_2 , if $V(s_1) > V(s_2)$, then M_V is increased by moving positive residual from s_2 to s_1 or negative residual from s_1 to s_2 . Therefore, putting all positive residual on s^* and moving all the negative residual toward states with smallest $V(s)$ values maximizes M_V among all distributions $\tilde{T}(\cdot)$ with the given L_1 constraint. \square

Givan et al. (2000) derived analogous results for a closely related problem where constraints are expressed in a weighted L_∞ norm.

Combining the various results from this section together, we can efficiently find a policy that maximizes the expected reward in an MDP, assuming that transition probabilities are optimistic but within a fixed confidence interval of the observed transitions. An agent following the MBIE exploration algorithm chooses actions according to this policy to jointly explore the environment and exploit its experience.

5. Experimental Results

To test the effectiveness of the algorithms, we used three simple MDPs, described below. We measured the discounted infinite-horizon performance ($\gamma = .9$) to assess the quality of the learning behavior for each algorithm during the learning process.

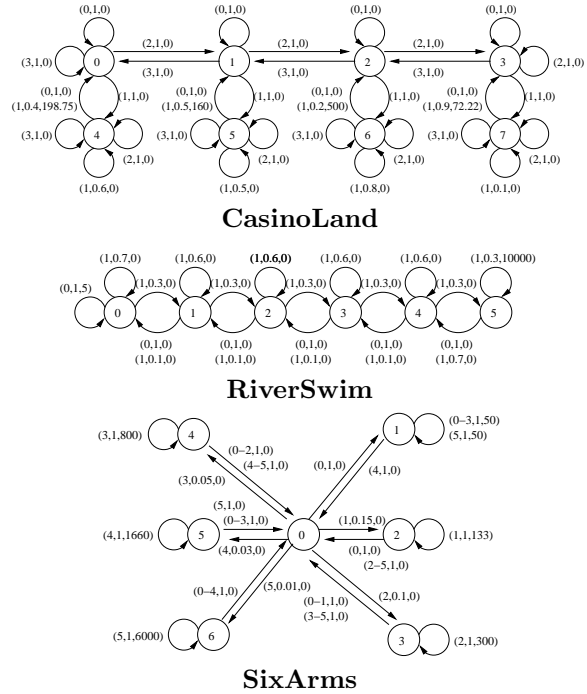


Figure 1. Three test environments: **CasinoLand** (top), **RiverSwim** (middle), and **SixArms** (bottom). Each node in the graph is a state and each edge is labeled by one or more transitions. A transition is of the form (a, p, r) where a is an action, p the probability that action will result in the corresponding transition, and r is the reward for taking the transition. For **CasinoLand** and **SixArms**, the agent starts in State 0. For **RiverSwim** the agent starts in either State 1 or 2 with equal probability.

Each learning experiment took place over 6000 steps. Every 30 steps, we began accumulating discounted reward, resulting in a sequence of 200 samples of the quality of the policy as it evolved. Each experiment was repeated 100 times and the results were averaged.

Each of the three algorithms we studied has one principle parameter (ϵ , c , and d), which, coarsely, trades off between exploration and exploitation. To better quantify the affect of the parameters, we repeated experiments with a range of parameter settings. Results indicate that the values of the parameters had a very significant effect on the performance.

The three MDPs we used in our experiments are illustrated in Figure 1. They were chosen to provide challenging examples of environments in which to explore. The first MDP, **CasinoLand**, consists of six rooms in a two-by-three grid (two rows of three rooms each). While in the top row, the agent can move in one of four directions (up, down, left, right), which allows the agent to completely traverse the top row

(bumping into walls has no effect). When in a room on the bottom row, the agent can no longer move left or right, but can either move back up or pull a lever. Each time a lever is pulled, there is a small probability of large reward. The MDP is designed so that the lever with the smallest probability of generating reward is actually the best lever to pull.

The second MDP, **RiverSwim**, also consists of six states, here all in one row. The agent starts on one of the states near the beginning of the row. The two actions available to the agent are to swim left or right. Swimming to the right (against the current of the river) will more often than not leave the agent in the same state, but will sometimes transition the agent to the right (and with a much smaller probability to the left). Swimming to the left (with the current) always succeeds in moving the agent to the left, until the leftmost state is reached in which case swimming to the left yields a small reward of five units. The agent receives a much larger reward, of ten thousand units, for swimming upstream to the rightmost state.

The final MDP, **SixArms**, consists of seven states, one of which is the initial state. The agent must choose from six different actions. Each action pulls a different arm, with a different payoff probability, on a k -armed bandit. When the arm pays off, the agent is sent to another state. Within that new state, large rewards can be obtained. The higher the payoff probability for an arm, the smaller the reward received (from the room the agent is sent to by the arm).

5.1. Results

Figure 2 shows the algorithms’ performances on the three different domains at different parameter settings. Each of the graphs plots the cumulative discounted reward, averaging over 100 runs for each algorithm over the sequence of 200 measurements of the discounted reward. Since we are concerned with efficient exploration, cumulative reward is an appropriate measure of performance—algorithms must quickly discover and exploit the environment in which they are learning. On the graph, changes in slopes reflect changes in the current behavior of the agent.

Each graph displays the performance of one algorithm on one problem over a sampling of parameter settings. The first column of graphs provides the results for ϵ -greedy, the second for R-Max, and the third for MBIE. The first row is **CasinoLand**, then **RiverSwim**, then **SixArms**. The line that reaches the maximum y-axis value on each graph corresponds to the best parameter setting among those tests.

In general, it can be seen that the best parameter setting for the MBIE algorithm outperforms those of the other two (the mean cumulative reward is larger for all problems, $p < .05$). In all cases, the lower the parameter setting, the less exploration the algorithm will exhibit. This is most evident in the case of R-Max. Notice that when c is set to 1 in the **CasinoLand** experiment, the algorithm outperforms all other settings for the first 25 or so samples, due to the early exploitation of knowledge, but ultimately it learns a suboptimal policy (small slope) and is overtaken.

In fact, for each parameter setting of R-Max, an abrupt shift is visible in the cumulative reward plot. This is the point at which the algorithm starts exploiting its knowledge about the domain. For a higher value of c , this transition occurs later in time, but the resulting increase in cumulative reward is steeper—an indication that the extra exploration paid off, allowing the algorithm to adopt a better policy. In some cases ($c = 13$ and $c = 27$ in the center graph, for example), the additional exploration afforded by the larger parameter value does not result in a better policy—here, we see the cumulative reward of the learned policy is the same in both cases and the $c = 27$ run simply waits longer to begin to exploit.

As a result of this effect, all the algorithms exhibit an “inverted u” behavior—for small parameter values, cumulative reward is low because the policy learned is suboptimal due to inadequate exploration. For large parameter values, cumulative reward is low because the algorithm waits a long time before exploiting. Each algorithm performs best with an intermediate setting of its parameter.¹

MBIE does not display a sharp transition—its exploration is more smoothly integrated with exploitation. This tends to allow it to accumulate more reward in the early stages, while still exploring enough to eventually discover the optimal policy. This advantage of MBIE is most evident in the **SixArms** domain. For high enough values of c in R-Max, the discovery of the optimal policy is assured. However, its choice of which part of the domain to explore first is arbitrary and thus it tends to explore non-optimal regions first. The **SixArms** domain was sufficiently tough that R-Max had not completed its exploration phase by the end of the 6000 steps. On the other hand, MBIE explored a little, then choose to continue to explore in the most promising direction (which in many cases was to exhibit a near-optimal policy). So, while it is the case that both algorithms would tend to ade-

¹The sole exception is that ϵ -greedy performed best in **CasinoLand** when ϵ was set to 0.

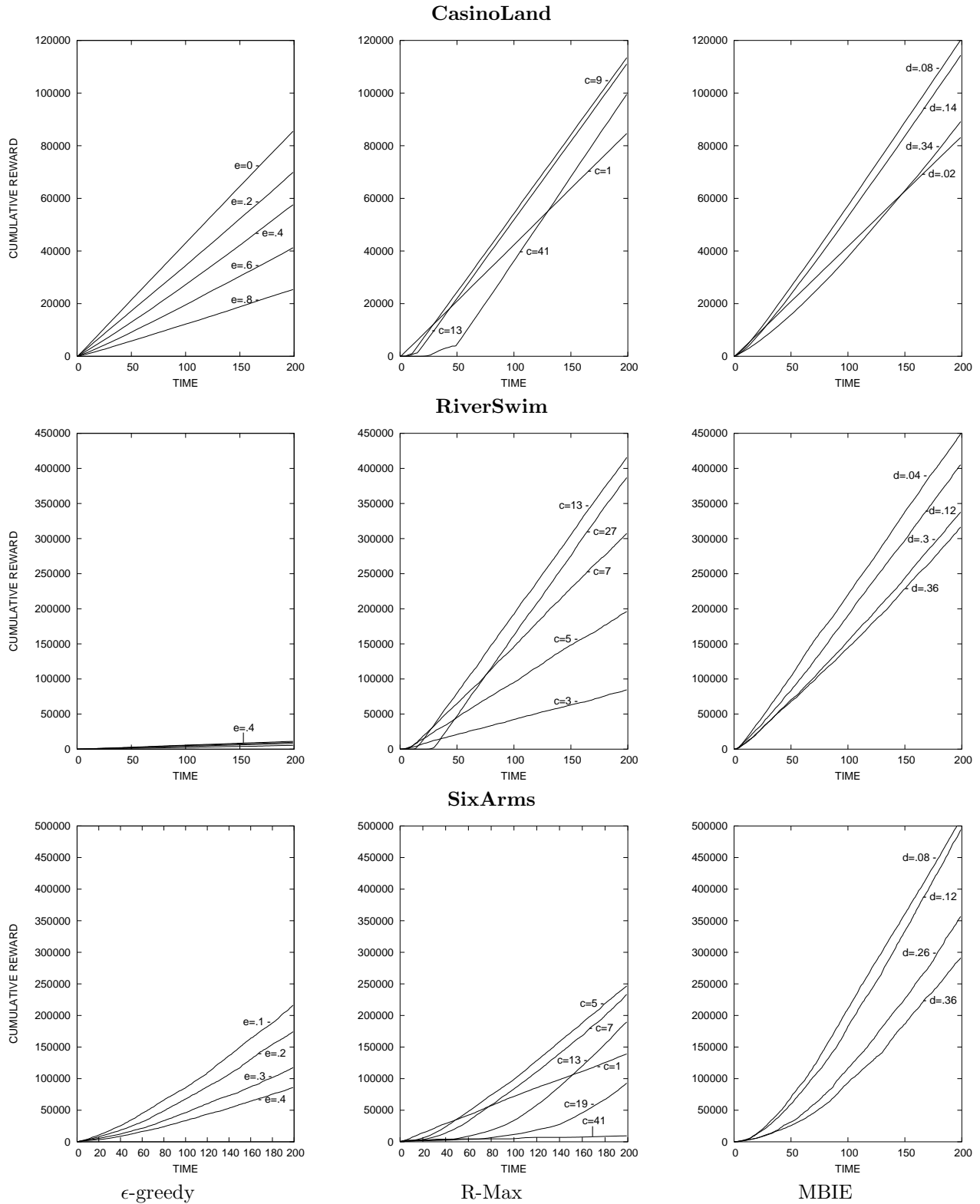


Figure 2. A comparison of three exploration schemes— ϵ -greedy (left), R-Max (middle), and MBIE (right)—on three small MDPs—CasinoLand (top), RiverSwim (middle), and SixArms (bottom)—shows the tradeoffs between exploration and exploitation on cumulative reward over an initial learning period.

quately explore the entire environment given enough time, MBIE's more focused approach to exploration allowed it to accumulate much greater reward over the finite time used to test the algorithms.

6. Conclusion

The confidence-interval approach we described is efficient and theoretically motivated. It also performed quite well in the small-scale tests we reported. The basic idea of the approach is (1) use experienced transitions to create a model of the environment, (2) assign values to all states in a way that assigns state-action pairs initially optimistic values but more realistic values with increasing experience, (3) choose actions greedily with respect to these values. Our MBIE scheme uses a particular approach to assigning values, but many other options are also available. In Laplace smoothing, a prior is placed on each state-action pair that assumes it has experienced z transitions to an optimistically initialized state. We have experimented with this approach in bandit problems and found it to match or outperform MBIE. However, we also expect that theoretical results will be easier to obtain for MBIE.

We have also considered Good-Turing smoothing (McAllester & Schapire, 2000), since another way of viewing the problem of exploration is as one of estimating the probability of an unseen event (a better transition than any of those already witnessed).

We believe exploration is the key to efficient reinforcement learning. Future work needs to focus on automatically identifying effective parameter values and on extending exploration approaches to larger and more complex state spaces in which generalization is necessary.

Acknowledgments

Thanks to the National Science Foundation and to DARPA IPTO for some early support. We also thank colleagues Sanjoy Dasgupta, David McAllester, Sergio Verdu and Tsachy Weissman for pointers to relevant literature.

References

Berry, D. A., & Fristedt, B. (1985). *Bandit problems: Sequential allocation of experiments*. London, UK: Chapman and Hall.

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX—a general polynomial time algorithm for near-optimal

reinforcement learning. *Journal of Machine Learning Research*, 3, 213–231.

- Duff, M. O., & Barto, A. G. (1997). Local bandit approximation for optimal learning problems. *Advances in Neural Information Processing Systems* (pp. 1019–1025). The MIT Press.
- Fong, P. W. L. (1995). A quantitative study of hypothesis selection. *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)* (pp. 226–234).
- Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. Wiley-Interscience series in systems and optimization. Chichester, NY: Wiley.
- Givan, R., Leach, S., & Dean, T. (2000). Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122, 71–109.
- Kaelbling, L. P. (1993). *Learning in embedded systems*. Cambridge, MA: The MIT Press.
- Kearns, M. J., & Singh, S. P. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49, 209–232.
- Koenig, S., & Simmons, R. G. (1993). Complexity analysis of real-time reinforcement learning. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 99–105). Menlo Park, CA: AAAI Press/MIT Press.
- McAllester, D., & Schapire, R. E. (2000). On the convergence rate of Good-Turing estimators. *Proceedings of the 13th Annual Conference on Computational Learning Theory* (pp. 1–6).
- Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons, Inc.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. The MIT Press.
- Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., & Weinberger, M. J. (2003). *Inequalities for the L1 deviation of the empirical distribution* (Technical Report HPL-2003-97R1). Hewlett-Packard Labs.
- Wiering, M. (1999). *Explorations in efficient reinforcement learning*. Doctoral dissertation, University of Amsterdam.